A matheuristic approach to the shift minimisation personnel task scheduling problem

Pieter Smet · Greet Vanden Berghe

Received: date / Accepted: date

Abstract In the context of personnel rostering, several levels of granularity have been discussed. Typically, these levels range from very coarse grained (e.g. days-off scheduling) to more finely granulated (e.g. tour scheduling). However, in some cases a more detailed type of assignment is required. Not only shifts need to be assigned to personnel, but also the allocation of tasks is incorporated in the optimisation of the rosters. The present paper introduces a matheuristic approach based on local search for the subproblem of assigning tasks to a set of multi-skilled employees whose working times are already determined. Experimental results show that the presented algorithm is capable of finding new best solutions for the benchmark instances.

Keywords Integrated personnel rostering · Personnel task scheduling · Matheuristic · Local search · Constructive heuristic

1 Introduction

After several years of research, personnel rostering remains a relevant academic timetabling subject. As personnel costs have become the major part of operational expenses, it is ever so important to try and organise a given workforce as efficiently as possible in order to reduce the associated costs and to increase employee satisfaction.

In the personnel scheduling literature, assigning shifts to personnel is often the most fine-grained level at which the allocation is being discussed, even

Pieter Smet
CODeS, KAHO Sint-Lieven
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
E-mail: pieter.smet@kahosl.be

Greet Vanden Berghe
CODeS, KAHO Sint-Lieven
Department of Computer Science, KU Leuven
E-mail: greet.vandenberghe@kahosl.be

though some authors do incorporate elements at a more detailed level (Ernst et al 2004). For example, Beer et al (2008) discuss the problem of assigning breaks in shifts. The shifts are already assigned, and breaks need to be planned such that various restrictions and requirements are met.

The assignment of particular tasks to employees during a shift is often not incorporated in the construction of rosters. In some cases, employees know automatically which tasks to perform during working hours. This is often the case in hospitals, where nurses know exactly what they are supposed to do in each shift (Burke et al 2004). However, in other cases tasks are assigned to employees in an ad hoc manner, often resulting in unnecessary usage of resources. Therefore it is recommendable to incorporate the assignment of tasks in the construction of rosters for employees, in order to reduce operational expenses while maintaining a high quality of service.

The practical relevance of this problem is apparent in various contexts. In the food production industry, due to the short batches being produced, employees have to perform tasks on more than one machine during one shift. In order to create efficient solutions, it is necessary to integrate the assignment of shifts and tasks resulting in an structured problem combining personnel rostering and task allocation.

The aforementioned problem has only been addressed by a few authors. Meisels and Schaerf (2003) discuss a general class of employee timetabling problems in which, during each shift, tasks need to be assigned to employees. No temporal details concerning the tasks are incorporated in the assignment, only the required number of employees for each task in each shift is given. Detienne et al (2009) present two cut generation based approaches for another employee timetabling problem. Their problem contains two decision stages. First the working times of employees are determined. Second, for each employee and for each working period, the used qualification of the employee is decided on. Guyon et al (2010) include scheduling decisions concerning specific activities. The integrated employee timetabling and production scheduling problem is solved using both exact and heuristic approaches including Benders decomposition and a cut generation approach based on the work of Detienne et al (2009). The system presented by Dowling et al (1997) is developed for rostering employees and assigning tasks to employees at an international airport. First, the personnel rostering problem is solved heuristically for a long scheduling period (35 days). Afterwards individual tasks are allocated to the available employees on a day-to-day basis.

Burke et al (2006) discuss a related personnel rostering problem in which the coverage requirements are not specified per shift type, but in terms of time intervals. The time interval coverage requirements are translated to shift type coverage constraints. A metaheuristic is then used to solve the resulting personnel rostering problem in an efficient way.

In the present paper we discuss a solution approach to the shift minimisation personnel task scheduling problem (SMPTSP), originally introduced by Krishnamoorthy and Ernst (2001). The problem considers assigning tasks to multi-skilled employees, while minimising the number of employees used.

Figure 1 shows an example of a solution for an SMPTSP instance, in which 60 tasks are assigned to 25 employees over a one day period. Krishnamoorthy et al (2011) present a Lagrangean relaxation based approach that combines two problem specific heuristics. They have found feasible solutions for 135 out of 137 instances. For a large number of these instances their algorithm is capable of finding the optimal solution. Furthermore, they discuss some interesting properties of the problem and present algorithms that can be used to solve particular subproblems of the SMPTSP.

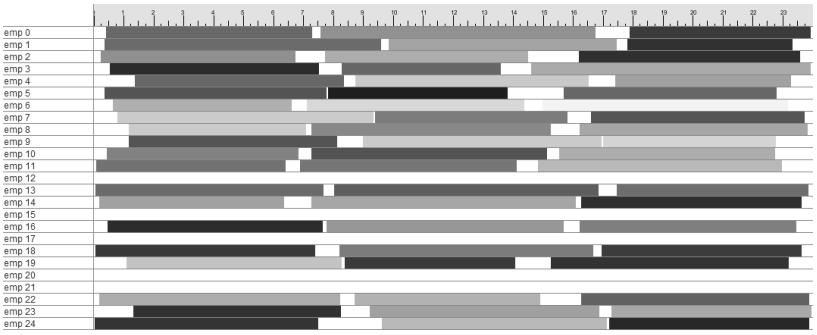


Fig. 1: Optimal solution for an SMPTSP instance.

The approach presented in this paper is a very large-scale neighbourhood search algorithm in which neighbouring solutions are reached by solving a heuristically selected subproblem to optimality. This solution approach is based on the principles of matheuristics (Maniezzo et al 2009), in which (meta)heuristics and exact approaches are combined to exploit the strengths of both solution techniques.

Della Croce and Salassa (2010) describe a matheuristic based on a variable neighbourhood search for a real world nurse rostering problem. Different neigbourhoods are searched by including additional constraints. These constraints fix particular variables which are selected heuristically. Computational results show that this matheuristic approach significantly outperforms exact commercial general purpose solvers. Matheuristic approaches have been applied in various other contexts such as vehicle routing (Doerner and Schmid 2010), permutation flow shop scheduling (Della Croce et al 2011) and the multidimensional knapsack problem (Hanafi et al 2010).

The rest of the paper is organised as follows. The problem definition is provided in Section 2. In Section 3 the solution approach is presented. Details are provided for both a constructive heuristic and the hybrid improvement heuristic. The experimental setup and results are discussed in Section 4. Section 5 concludes the paper and presents future work.

2 Problem definition

Let $J = \{1, ..., n\}$ be the set of tasks to be assigned to employees and $W = \{1, ..., m\}$ the set of employees. Each task $j \in J$ has a duration d_j, a start time s_j and a finish time $f_j = s_j + d_j$. Each employee $w \in W$ has a set of tasks $T_w \subseteq J$ which he/she can perform. Similarly, there exists a set $P_j \subseteq W$ for each task $j \in J$ which contains all employees that can perform task j. Both T_w and P_j are defined based on required qualifications and time windows.

An interval graph $G = (J, A)$ can be defined with J the set of nodes and A the set of arcs. Two nodes i and j are connected when their respective time intervals, $[s_i,f_i]$ and $[s_j,f_j]$, overlap. The set of maximal cliques in the interval graph is defined as C. For interval graphs, this set can be found in polynomial time by first sorting the nodes based on start time and then applying a forward pass algorithm. The set $C = \{K_1, ..., K_t\}$ consists of sets $K_t \subseteq J$ such that any pair of tasks in K_t overlaps in time and K_t is maximal. There are no tasks in $J \setminus K_t$ which overlap with any of the tasks in K_t. In terms of the SMPTSP, it is obvious that overlapping tasks, represented by nodes in K_t, should be assigned to different employees. For each employee $w \in W$, the set of maximal cliques $C^w = \{K_1^w, ..., K_t^w\}$ is constructed in the same way as C, but for C^w, only the set of tasks for which the employee is qualified is considered. An employee w can only be assigned to one of the tasks from each set $K_t^w \in C^w$. This ensures that there are no overlapping assignments in a solution.

To solve the SMPTSP, a feasible solution has to be found in which all tasks in J are assigned to qualified employees from W in a non-preemptive manner, while minimising the number of workers used.

Two sets of decision variables are defined for the mathematical model:

$$x_{jw} = \begin{cases} 1 & \text{if task } j \in J \text{ is assigned to employee } w \in W \\ 0 & \text{otherwise} \end{cases}$$

$$y_w = \begin{cases} 1 & \text{if employee } w \in W \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

The SMPTSP can now be defined as follows (Krishnamoorthy et al 2011):

$$min \sum_{w \in W} y_w \tag{1}$$

$$s.t. \sum_{w \in P_j} x_{jw} = 1 \qquad \forall\ j \in J \tag{2}$$

$$\sum_{j \in K_t^w} x_{jw} \leq y_w \qquad \forall\ w \in W,\ K_t^w \in C^w \tag{3}$$

$$0 \leq y_w \leq 1 \qquad \forall\ w \in W \tag{4}$$

$$x_{jw} \in \{0, 1\} \qquad \forall\ j \in J,\ w \in W \tag{5}$$

The objective function (1) states that the number of used employees should be minimised. Constraints (2) ensure that each task is only performed by one employee, and that no infeasible assignments in terms of qualifications are

made. Constraints (3) make sure that tasks are only assigned to employees who are active in the final solution. Furthermore, these constraints do not allow for overlapping tasks to be assigned to an employee. Finally, constraints (4) and (5) set bounds for the decision variables.

The SMPTSP can be seen as an application of list colouring on interval graphs, which is NP-complete. Colours correspond to employees and vertices correspond to tasks. The qualifications of the employees are represented by the list of feasible colours on each vertex. Other applications of list colouring on interval graphs include room allocation (Carter and Tovey 1992) and register assignment (Zeitlhofer and Wess 2003).

A class of problems similar to the SMPTSP are the interval scheduling problems (Kolen et al 2007). Here, a set of jobs with fixed start and end times are given as well as a set of machines that can process the jobs. The goal is to decide which jobs to assign and to which machines, while e.g. maximising the value of the assigned jobs. The difference between the basic interval scheduling problem and the SMPTSP lies in the fact that in the SMPTSP not all machines (employees) are qualified for all jobs, and that machines (employees) are not always available. Furthermore, all tasks should be assigned in a feasible solution for the SMPTSP.

3 A hybrid heuristic approach

We present a hybrid heuristic local search algorithm for the SMPTSP, based on the principles of matheuristics. The solution approach is hybrid in the sense that neighbouring solutions are reached by solving a randomly selected part of the problem to optimality using a general purpose solver. Details on the local search procedure and the explored neighbourhood are given in Section 3.2. To ensure feasibilty of the final solution, the algorithm remains in the feasible search space during the length of the search. A constructive heuristic, described in Section 3.1, is designed to provide a feasible initial solution.

3.1 Constructive heuristic

Krishnamoorthy et al (2011) state that when the qualification constraints are relaxed, i.e. when all employees are qualified to perform all tasks, the resulting problem can be solved in polynomial time. For this purpose, they describe a forward pass maximal clique algorithm on an interval graph (Gupta et al 1979). This algorithm assigns all tasks in order of increasing starting time, using, if possible, an employee who already has tasks assigned. This characteristic of the SMPTSP is incorporated in our constructive heuristic by sorting all tasks $j \in J$ on start time s_j in ascending order. Ties are broken by taking into account the qualifications of employees. For this purpose, the tasks are additionally sorted based on the number of qualified personnel able to perform them, also in ascending order. This results in an ordering in which tasks with the smallest

number of feasible personnel are before others. These highly constrained tasks, which are the most difficult to assign, will then be assigned first.

An additional mechanism is introduced to ensure that the constructive heuristic finds feasible solutions in those cases where tasks can only be performed by a limited number of employees. Whenever there is a task j which cannot be assigned to a qualified employee due to other overlapping tasks, a qualified employee is randomly selected and his/her assigned tasks overlapping with j are removed. Task j is then assigned to this employee and the removed tasks are assigned to other employees.

Algorithm 1 shows pseudo code of the constructive heuristic.

Algorithm 1 Constructive heuristic

```
s_j := Start time of task j ∈ J
O_j := Jobs overlapping with task j
P_j := Employees qualified for task j
R_e := Tasks assigned to employee e

Order all j ∈ J by (s_j + |P_j|) in ascending order
while J ≠ ∅ do
   Remove task j from the first position in J
   Assign j to the first feasible employee
   if Cannot feasibly assign j then
      Select random employee e ∈ P_j
      Remove the conflicting tasks O_j from R_e
      Assign j to employee e
      Add the previously removed tasks O_J to the list of tasks to be assigned J
   end if
end while
```

Experiments performed on realistic problems from literature (Section 4.2) and problems based on real world data provided by an industrial partner [1], show that Algorithm 1 is capable of finding feasible solutions for problems with realistic dimensions. Algorithm 1 will not terminate if an instance has no feasible solution. To resolve this issue, additional mechanisms could be added to the algorithm which e.g. do not assign all visits or include dummy employees in the set of available employees.

3.2 Matheuristic based local search

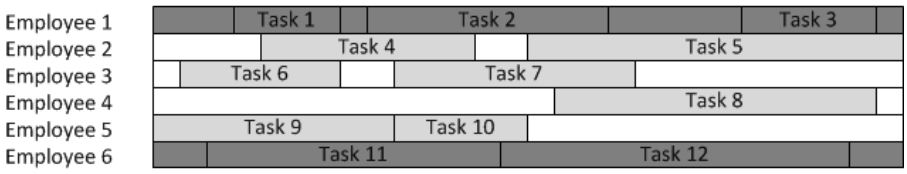
Typically, the initial solution can still be improved. For this purpose, an improvement procedure based on local search is used.

To ensure feasibilty throughout the search trajectory of the algorithm only feasible neighbouring solutions are considered. These are reached by randomly selecting k employees and optimally solving the subproblem composed of them and their assigned jobs, using a general purpose solver. The initial solution is

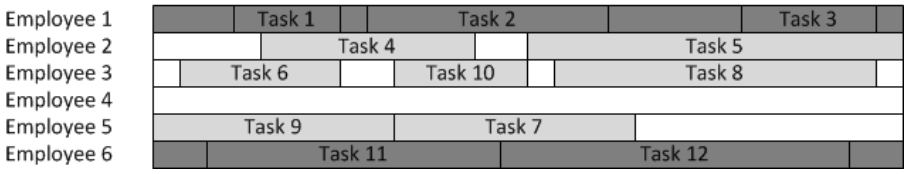
[1] SAGA Consulting

feasible and therefore solutions in this neighbourhood are also always feasible, thus forcing the local search only to explore the feasible search space.

Figure 2 illustrates the move used to reach new solutions. A gray roster indicates employees that are not considered by the move, i.e. the assignments of these employees remain unchanged during the move. In the example, the subproblem is composed of the selected employees (2, 3, 4 and 5) and their assigned jobs (4, 5, 6, 7, 8, 9 and 10). All tasks can be performed by all employees, except for task 8 for which only employees 2, 3 and 4 are qualified. This subproblem is then solved to optimality by a general purpose solver. In the resulting neighbouring solution (Figure 2b), employee 4 is no longer required to perform any tasks. The objective value of the new solution is thus one lower than the current solution. k should be limited to ensure computational feasibilty. Based on initial experimentation k was set to 40 employees. However, when $k > |W|$, k was set to the total number of employees.



(a) Current solution



(b) New solution

Fig. 2: Illustration of a move with $k = 4$. Employees with grayed out rosters are not considered in the move.

The aforementioned neighbourhood is explored with a very large-scale neighbourhood search algorithm. In order to further guarantee computational feasibility of the solution approach, only one neighbouring solution is sampled at each iteration. In terms of iterations per minute, a trade-off exists between the number of sampled solutions in each iteration and the number of employees k selected for composing the subproblem. Smaller values for k result in faster solution times for the subproblem, making it possible to sample more neighbouring solutions in each iteration. Initial experimentation showed that better results were achieved by limiting the number of sampled solutions and increasing the size of the subproblems.

The pseudo code of the local search algorithm is shown in Algorithm 2.

Algorithm 2 Local search algorithm

```
F(C) := Evaluation function
H := Move to reach a neighbouring solution
C_0 := Initial solution
C ← C_0
while Termination criterion not met do
   C' ← H(C)
   if F(C') ≤ F(C) then
      C ← C'
   end if
end while
```

4 Experiments

4.1 Experimental setup

We evaluate the presented solution approach using instances from a benchmark dataset [2]. According to Krishnamoorthy et al (2011) these instances are based on their experience with real world problems. Information with regard to the number of employees and the number of tasks is shown in Figure 3. The dimensions of the instances range from small (23 employees and 40 tasks) to very large (245 employees and 2105 tasks).

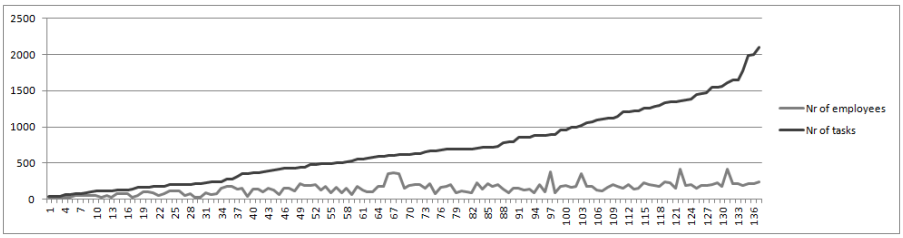


Fig. 3: Number of employees and jobs in the instances. The horizontal axis represents the different problem instances.

The experiments are carried out on an Intel Core 2 Duo at 3.16GHz with 4GB RAM operating on Windows 7. All algorithms are coded in Java and CPLEX 12.2 is used as general purpose solver. Experiments with the local search algorithm are each carried out five times. Results regarding the constructive heuristic are reported by one value since initial experimentation showed that, for the available benchmark instances, the same solution was obtained each time the algorithm was executed. The execution time for the local search procedure is limited to 1800 seconds per run.

[2] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ptaskinfo.html`

4.2 Experimental results

Tables 1, 2 and 3 show the results of the constructive heuristic (CH), the average solution quality of the matheuristic local search (LS_{avg}) and the best solution of five runs (LS_{best}) compared with 1) a lower bound obtained by CPLEX (LB) and 2) results from a Lagrangean-based heuristic as presented by Krishnamoorthy et al (2011) (LH). Furthermore, the time required by the constructive heuristic and the local search are given by t_{CH} and t_{LS}, respectively.

Figure 4 shows the relative quality gap between solutions obtained with the constructive heuristic and the lower bound (gap to LB), and the constructive heuristic and the Lagrangean heuristic (gap to LH). Positive values indicate a relatively worse solution by the constructive heuristic. The average gap between the constructive heuristic and the lower bound is 4.22% whereas the average gap between the constructive heuristic and the Lagrangean heuristic is 0.08%. Based on Figure 4, two observations can be made. First, the performance of the constructive heuristic remains relatively stable for all problem sizes. This shows that the constructive heuristic is able to generate high quality solutions, even for very large instances. Second, from a certain problem size onward, the constructive heuristic outperforms the Lagrangean heuristic. This mostly indicates that the Lagrangean heuristic has difficulties with increasing problem size. For 7 out of 137 instances, the constructive heuristic finds the optimal solution, while for 100 out of 137 instances the solution is only 5% worse than the lower bound. Note that the constructive heuristic is capable of finding feasible solutions for all benchmark instances.

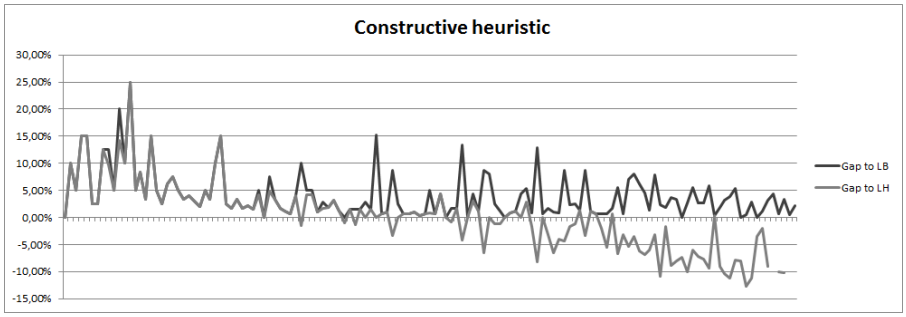


Fig. 4: Relative quality gap between the constructive heuristic and the lower bound (gap to LB) and the constructive heuristic and Lagrangean heuristic (gap to LH). The horizontal axis represents the different problem instances, from small to large.

Figure 5 shows the relative difference between the quality of solutions obtained by the local search and the lower bounds (gap to LB) and the local search and Lagrangean heuristic (gap to LH). The results clearly show that

the local search performs very well, with a maximum gap to the lower bound of 7.56% and an average gap of 0.67%. Furthermore, the presented matheuristic is capable of finding the optimal solution for 72 out of 137 instances, while for 135 out of 137 instances the local search finds solutions within 5% of the lower bound. Compared to the Lagrangean heuristic the hybrid local search also performs very well with an average improvement of 3.34%. Overall, 68 new best solutions are found by the presented matheuristic approach.

Fig. 5: Relative quality gap between the local search and the lower bound (gap to LB) and the local search and Lagrangean heuristic (gap to LH). The horizontal axis represents the different problem instances, from small to large.

When comparing the computation times in Tables 1, 2 and 3, it can be observed that, for all instances, the constructive heuristic requires less than one second of computation time to construct a solution. The computation time of the local search algorithm is plotted in Figure 6. This plot shows that if the matheuristic finds the optimal solution it finds it rather quickly. In these cases the average computation time is 74.58 seconds. However, once the dimensions of problems increase, the matheuristic does not find the optimum anymore within the time limit.

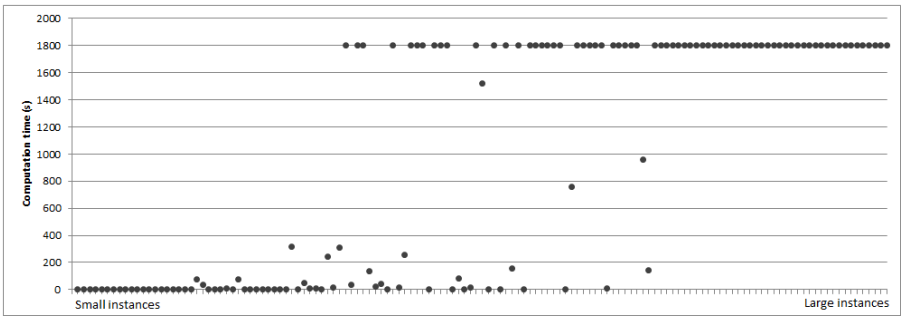


Fig. 6: Computation times of the local search algorithm (time limit set to 1800s). The horizontal axis represents the different problem instances.

The relative improvement obtained by the local search procedure over the constructive heuristic is shown in Figure 7. It can be seen that for the smaller instances, larger relative improvements are found than for the larger instances. These differences can possibly be explained by the value chosen for k. For the smaller instances, k is relatively large compared to the instance size, implying that each subproblem will consider a large part of the original problem. However, for the larger instances, the same value for k will consider a much smaller part of the problem as subproblem, and thus making it more difficult to optimise the solution as a whole. If the number of employees in the subproblem k would be chosen higher, larger improvements could be found in the local search phase. However, it is possible that other phenomena play an important role in the observations from Figure 7. Further investigation is required to determine other influencing factors.

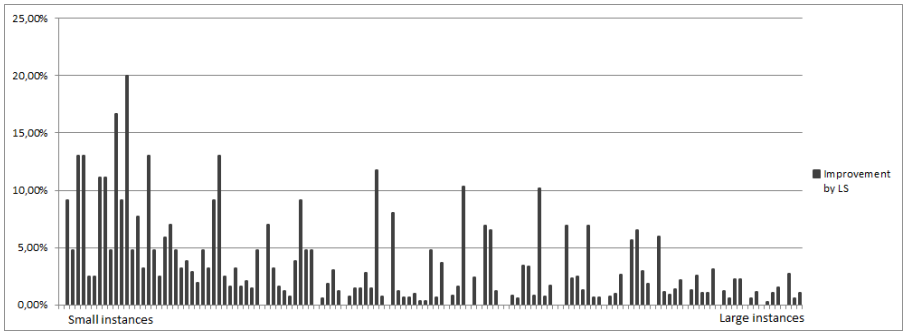


Fig. 7: Relative improvement obtained by the local search algorithm compared to the constructive heuristic. The horizontal axis represents the different problem instances.

5 Conclusions and future work

The paper is focussed on a hybrid heuristic approach to the SMPTSP. The constructive component of the heuristic is capable of generating feasible solutions in a very short computation time. Furthermore, the constructed solutions are of high quality, with an average gap to the lower bound of 4.22% on a set of benchmark instances from the literature.

A hybrid local search algorithm, based on the principles of matheuristics, is employed for further improving the initial solution. In this algorithm, neigbouring solutions are reached by randomly selecting a number of employees and solving the thus delineated subproblem to optimality. Due to computational feasibility issues, the number of sampled solutions in each neighbourhood at each iteration is limited, as well as the number of selected employees for the subproblem.

Experimental results showed that the hybrid heuristic performs very well, reaching solutions on average less than 1% worse than the lower bound. Analysis of the computation times showed that, if the algorithm is capable of reaching the optimum, it does so rather quickly with an average computation time of 74.58 seconds. Furthermore, the presented matheuristic found 68 new best solutions for the available benchmark instances.

In order to reach better solutions for larger instances, a high level strategy for the local search, i.e. a metaheuristic, can be used. Additional improvements to the existing algorithm can further increase the algorithmic performance. For example, another neighbourhood can be explored by probabilistically selecting employees in the subproblem instead of randomly selecting them.

Future work includes the incorporation of the presented solution approach for the SMPTSP in the larger problem of assigning tasks and shifts to employees in the same process. The speed of the constructive heuristic combined with the high quality solutions it generates present a particularly interesting opportunity in developing algorithms for the larger integrated problem.

Acknowledgements This research was carried out within the IWT project (IWT 110257).

References

Beer A, Gaertner J, Musliu N, Schafhauser W, Slany W (2008) Scheduling breaks in shift plans for call centers. In: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montréal

Burke E, De Causmaecker P, Vanden Berghe G, Van Landeghem H (2004) The state of the art of nurse rostering. Journal of Scheduling 7(6):441–499

Burke EK, De Causmaecker P, Petrovic S, Vanden Berghe G (2006) Metaheuristics for handling time interval coverage constraints in nurse scheduling. Applied Artificial Intelligence 20(9):743–766

Carter MW, Tovey CA (1992) When is the classroom assignment problem hard? Operations Research 40(S1):28–39

Della Croce F, Salassa F (2010) A variable neighborhood search based matheuristic for nurse rostering problems. In: McCollum B, Burke E, White G (eds) Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), Queen's University Belfast, pp 167–3175

Della Croce F, Grosso A, Salassa F (2011) A matheuristic approach for the total completion time two-machines permutation flow shop problem. In: Merz P, Hao JK (eds) Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, vol 6622, Springer Berlin / Heidelberg, pp 38–47

Detienne B, Peridy L, Pinson E, Rivreau D (2009) Cut generation for an employee timetabling problem. European Journal of Operational Research 193(3):1178–1184

Doerner K, Schmid V (2010) Survey: matheuristics for rich vehicle routing problems. In: Blesa M, Blum C, Raidl G, Roli A, Sampels M (eds) Hybrid Metaheuristics, Lecture Notes in Computer Science, vol 6373, Springer Berlin / Heidelberg, pp 206–221

Dowling D, Krishnamoorthy M, Mackenzie H, Sier D (1997) Staff rostering at a large international airport. Annals of Operations Research 72:125–147

Ernst A, Jiang H, Krishnamoorthy M, Sier D (2004) Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153(1):3–27

Gupta U, Lee D, Leung JT (1979) An optimal solution for the channel-assignment problem. IEEE Transactions on Computers C-28(11):807–810

Guyon O, Lemaire P, Pinson E, Rivreau D (2010) Cut generation for an integrated employee timetabling and production scheduling problem. European Journal of Operational Research 201(2):557–567

Hanafi S, Lazic J, Mladenovic N, Wilbaut C, Crvits I (2010) New hybrid matheuristics for solving the multidimensional knapsack problem. In: Blesa M, Blum C, Raidl G, Roli A, Sampels M (eds) Hybrid Metaheuristics, Lecture Notes in Computer Science, vol 6373, Springer Berlin / Heidelberg, pp 118–132

Kolen A, Lenstra J, Papadimitriou C, Spieksma F (2007) Interval scheduling : a survey. Naval Research Logistics 54(5):530–543

Krishnamoorthy M, Ernst A (2001) The personnel task scheduling problem. In: Yang X, Teo K, Caccetta L (eds) Optimisation methods and application, Kluwer, pp 434–368

Krishnamoorthy M, Ernst A, Baatar D (2011) Algorithms for large scale shift minimisation personnel task scheduling problems. European Journal of Operational Research

Maniezzo V, Stutzle T, Voss S (eds) (2009) Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, Annals of Information Systems, vol 10. Springer

Meisels A, Schaerf A (2003) Modelling and solving employee timetabling problems. Annals of Mathematics and Artificial Intelligence 39:41–59

Zeitlhofer T, Wess B (2003) List-coloring of interval graphs with application to register assignment for heterogeneous register-set architectures. Signal Processing 83(7):1411 – 1425

Instance	LB	LH	CH	LS_{avg}	LS_{best}	t_{CH} (s)	t_{LS} (s)
data_1_23_40_66	20.00	20.00	20.00	20.00	20.00	0.00	0.03
data_2_24_40_33	20.00	20.00	22.00	20.00	20.00	0.00	0.02
data_3_25_40_66	20.00	20.00	21.00	20.00	20.00	0.00	0.03
data_4_23_59_33	20.00	20.00	23.00	20.00	20.00	0.00	0.08
data_5_25_60_33	20.00	20.00	23.00	20.00	20.00	0.00	0.03
data_6_48_80_66	40.00	40.00	41.00	40.00	40.00	0.00	0.10
data_7_51_80_66	40.00	40.00	41.00	40.00	40.00	0.00	0.13
data_8_48_85_33	40.00	40.00	45.00	40.00	40.00	0.00	0.38
data_9_49_104_33	40.00	41.00	45.00	40.00	40.00	0.00	0.84
data_10_51_111_66	40.00	40.00	42.00	40.00	40.00	0.00	2.19
data_11_24_119_33	20.00	21.00	24.00	20.00	20.00	0.00	0.30
data_12_49_119_33	40.00	40.00	44.00	40.00	40.00	0.00	0.95
data_13_25_120_33	20.00	20.00	25.00	20.00	20.00	0.00	0.24
data_14_75_124_33	60.00	60.00	63.00	60.00	60.00	0.00	0.38
data_15_72_126_33	60.00	60.00	65.00	60.00	60.00	0.00	0.69
data_16_75_131_66	60.00	60.00	62.00	60.00	60.00	0.00	2.19
data_17_23_139_66	20.00	20.00	23.00	20.00	20.00	0.00	1.08
data_18_48_160_66	40.00	40.00	42.00	40.00	40.00	0.00	1.19
data_19_97_160_33	80.00	80.00	82.00	80.00	80.00	0.00	0.44
data_20_99_163_33	80.00	80.00	85.00	80.00	80.00	0.00	0.92
data_21_93_175_33	80.00	80.00	86.00	80.00	80.00	0.00	78.58
data_22_47_180_66	40.00	40.00	42.00	40.00	40.00	0.00	33.88
data_23_74_180_66	60.00	60.00	62.00	60.00	60.00	0.00	1.48
data_24_110_200_33	100.00	100.00	104.00	100.00	100.00	0.00	0.75
data_25_120_200_33	100.00	100.00	103.00	100.00	100.00	0.00	1.05
data_26_116_203_66	100.00	100.00	102.00	100.00	100.00	0.00	11.81
data_27_49_204_66	40.00	40.00	42.00	40.00	40.00	0.00	3.73
data_28_75_208_66	60.00	60.00	62.00	60.00	60.00	0.00	73.82
data_29_22_219_66	20.00	20.00	22.00	20.00	20.00	0.00	2.73
data_30_25_219_66	20.00	20.00	23.00	20.00	20.00	0.00	3.31
data_31_90_230_66	80.00	80.00	82.00	80.00	80.00	0.00	0.88
data_32_70_236_66	60.00	60.00	61.00	60.00	60.00	0.00	0.81
data_33_76_240_66	60.00	60.00	62.00	60.00	60.00	0.00	2.08
data_34_152_240_33	120.00	120.00	122.00	120.00	120.00	0.00	1.06
data_35_171_280_33	140.00	140.00	143.00	140.00	140.00	0.02	0.30
data_36_175_280_33	140.00	140.00	142.00	140.00	140.00	0.00	2.57
data_37_145_321_33	120.00	121.00	126.00	120.67	120.00	0.00	316.42
data_38_147_347_66	120.00	120.00	120.00	120.00	120.00	0.02	0.37
data_39_45_351_66	40.00	41.00	43.00	40.00	40.00	0.00	49.97
data_40_138_360_33	120.00	120.00	124.00	120.00	120.00	0.00	9.48
data_41_144_360_66	120.00	120.00	122.00	120.00	120.00	0.00	9.13
data_42_101_380_66	80.00	80.00	81.00	80.00	80.00	0.00	0.70
data_43_156_387_66	140.00	140.00	141.00	140.00	140.00	0.00	240.18
data_44_121_400_33	100.00	100.00	104.00	100.00	100.00	0.00	16.32
data_45_67_420_33	60.00	67.00	66.00	60.00	60.00	0.02	308.35
data_46_147_423_33	120.00	121.00	126.00	120.67	120.00	0.02	1800.00
data_47_150_430_33	120.00	121.00	126.00	120.00	120.00	0.02	34.63
data_48_120_434_66	100.00	100.00	101.00	101.00	101.00	0.02	1800.00
data_49_211_446_66	180.00	182.00	185.00	184.33	184.00	0.02	1800.00
data_50_187_447_66	160.00	160.00	163.00	160.00	160.00	0.02	136.13

Table 1: Detailed computational results for SMPTSP benchmark instances

Instance	LB	LH	CH	LS_{avg}	LS_{best}	t_{CH} (s)	t_{LS} (s)
data_51_196_480_33	160.00	160.00	165.00	160.00	160.00	0.02	23.31
data_52_205_480_66	160.00	160.00	162.00	160.00	160.00	0.02	45.65
data_53_127_487_66	100.00	101.00	100.00	100.00	100.00	0.02	1.97
data_54_175_492_66	140.00	140.00	142.00	141.00	141.00	0.02	1800.00
data_55_85_493_66	70.00	72.00	71.00	70.33	70.00	0.00	17.10
data_56_163_500_66	140.00	140.00	142.00	140.33	140.00	0.02	255.40
data_57_88_508_66	70.00	72.00	72.00	70.67	70.00	0.02	1800.00
data_58_158_517_66	140.00	140.00	142.00	140.67	140.00	0.02	1800.68
data_59_70_525_33	59.00	68.00	68.00	60.00	60.00	0.02	1800.00
data_60_181_549_66	139.00	139.00	140.00	139.33	139.00	0.02	2.87
data_61_121_557_66	100.00	100.00	101.00	101.00	101.00	0.02	1800.00
data_62_101_571_33	80.00	90.00	87.00	80.67	80.00	0.02	1800.00
data_63_97_577_66	80.00	82.00	82.00	81.00	81.00	0.02	1800.00
data_64_176_595_66	160.00	160.00	161.00	160.00	160.00	0.03	1.34
data_65_179_596_66	159.00	159.00	160.00	159.00	159.00	0.03	85.10
data_66_348_600_33	300.00	300.00	303.00	300.00	300.00	0.03	5.62
data_67_371_600_66	300.00	300.00	301.00	300.00	300.00	0.05	16.72
data_68_359_613_66	300.00	300.00	302.00	301.00	301.00	0.06	1800.00
data_69_148_614_33	120.00	125.00	126.00	120.33	120.00	0.02	1518.01
data_70_192_623_66	160.00	160.00	161.00	160.00	160.00	0.03	3.74
data_71_197_624_33	158.00	158.00	165.00	159.00	159.00	0.02	1800.00
data_72_205_624_66	160.00	160.00	160.00	160.00	160.00	0.03	0.70
data_73_155_661_66	120.00	123.00	122.00	121.67	121.00	0.03	1800.00
data_74_209_664_33	180.00	180.00	183.00	180.00	180.00	0.02	159.15
data_75_72_665_33	60.00	71.00	68.00	61.00	61.00	0.02	1800.00
data_76_162_683_66	140.00	140.00	140.00	140.00	140.00	0.03	1.83
data_77_180_688_33	160.00	162.00	167.00	163.00	163.00	0.02	1800.00
data_78_199_688_66	160.00	160.00	162.00	162.00	162.00	0.03	1800.00
data_79_94_689_33	80.00	93.00	87.00	81.00	81.00	0.00	1800.00
data_80_112_691_33	99.00	107.00	107.00	100.00	100.00	0.02	1800.00
data_81_97_692_66	80.00	83.00	82.00	81.00	81.00	0.02	1800.00
data_82_89_697_66	80.00	82.00	81.00	81.00	81.00	0.03	1800.00
data_83_222_700_66	180.00	180.00	180.00	180.00	180.00	0.05	0.87
data_84_136_718_66	120.00	120.00	121.00	120.67	120.00	0.05	761.00
data_85_217_720_66	180.00	180.00	182.00	181.00	181.00	0.05	1800.00
data_86_178_721_33	140.00	146.00	146.00	141.33	141.00	0.03	1800.00
data_87_203_735_33	170.00	174.00	179.00	173.00	173.00	0.03	1800.00
data_88_137_777_66	120.00	123.00	121.00	120.67	120.00	0.03	1800.00
data_89_88_788_33	70.00	86.00	79.00	71.33	71.00	0.02	1800.00
data_90_157_791_66	139.00	140.00	140.00	139.67	139.00	0.05	8.35
data_91_147_851_66	118.00	124.00	120.00	119.33	118.00	0.05	1800.00
data_92_126_856_66	98.00	106.00	99.00	99.00	99.00	0.03	1800.00
data_93_141_856_66	119.00	125.00	120.00	120.00	120.00	0.05	1800.00
data_94_93_881_33	80.00	91.00	87.00	81.00	81.00	0.02	1800.00
data_95_204_882_33	170.00	177.00	174.00	171.33	170.00	0.03	1800.00
data_96_98_886_66	80.00	83.00	82.00	80.33	80.00	0.05	962.12
data_97_383_895_33	300.00	300.00	304.00	300.00	300.00	0.06	146.39
data_98_91_896_33	80.00	90.00	87.00	81.33	81.00	0.03	1800.00
data_99_176_956_66	160.00	160.00	162.00	161.33	161.00	0.08	1800.00
data_100_194_956_66	160.00	160.00	161.00	160.00	160.00	0.08	1800.00

Table 2: Detailed computational results for SMPTSP benchmark instances

Instance	LB	LH	CH	LS_{avg}	LS_{best}	t_{CH} (s)	t_{LS} (s)
data_101_166_997_66	140.00	144.00	141.00	141.00	141.00	0.03	1800.00
data_102_179_997_66	138.00	147.00	139.00	138.33	138.00	0.03	1800.00
data_103_348_1024_33	300.00	303.00	305.00	302.67	302.00	0.05	1800.00
data_104_181_1057_33	146.00	165.00	154.00	150.67	150.00	0.02	1800.00
data_105_173_1075_66	150.00	156.00	151.00	151.00	151.00	0.05	1800.00
data_106_121_1096_33	100.00	113.00	107.00	101.67	101.00	0.02	1800.00
data_107_114_1112_33	100.00	112.00	108.00	101.33	101.00	0.02	1800.00
data_108_162_1115_33	128.00	145.00	136.00	132.00	132.00	0.02	1800.00
data_109_205_1115_33	157.00	176.00	164.00	161.33	161.00	0.03	1800.00
data_110_183_1143_66	155.00	167.00	157.00	157.00	157.00	0.05	1800.00
data_111_155_1211_33	139.00	155.00	150.00	141.67	141.00	0.03	1800.00
data_112_200_1213_33	169.00	194.00	173.00	171.00	171.00	0.03	1800.00
data_113_141_1221_66	110.00	114.00	112.00	111.33	111.00	0.05	1800.00
data_114_157_1227_33	138.00	157.00	143.00	141.33	141.00	0.03	1800.00
data_115_228_1257_33	177.00	199.00	183.00	180.33	179.00	0.03	1800.00
data_116_205_1262_66	176.00	190.00	176.00	176.00	176.00	0.08	1800.00
data_117_192_1285_33	149.00	170.00	153.00	152.00	151.00	0.03	1800.00
data_118_180_1302_33	147.00	165.00	155.00	151.67	151.00	0.03	1800.00
data_119_236_1335_33	188.00	208.00	193.00	191.67	191.00	0.05	1800.00
data_120_228_1341_33	187.00	208.00	192.00	190.67	190.00	0.05	1800.00
data_121_147_1345_33	120.00	140.00	127.00	123.33	123.00	0.03	1800.00
data_122_422_1358_66	324.48	348.00	349.00	349.00	349.00	0.33	1800.00
data_123_187_1376_33	159.00	178.00	162.00	161.00	160.00	0.08	1800.00
data_124_198_1383_33	158.00	182.00	163.00	162.33	162.00	0.08	1800.00
data_125_157_1448_33	130.00	152.00	135.00	132.33	132.00	0.08	1800.00
data_126_193_1462_33	167.00	191.00	176.00	172.33	172.00	0.09	1800.00
data_127_192_1472_66	167.83	185.00	170.00	170.00	170.00	0.19	1800.00
data_128_207_1542_66	175.29	205.00	179.00	178.67	178.00	0.22	1800.00
data_129_233_1546_33	178.00	206.00	183.00	182.00	181.00	0.11	1800.00
data_130_176_1562_66	138.35	145.00	140.00	140.00	140.00	0.17	1800.00
data_131_415_1610_33	344.07	359.00	352.00	351.00	351.00	0.34	1800.00
data_132_216_1645_33	186.00	211.00	192.00	190.67	190.00	0.13	1800.00
data_133_211_1647_33	185.00		193.00	190.33	190.00	0.14	1800.00
data_134_184_1776_66	157.56	179.00	161.00	161.00	161.00	0.27	1800.00
data_135_213_1988_33	179.00	206.00	185.00	181.67	180.00	0.19	1800.00
data_136_216_2000_66			180.00	179.67	179.00	0.36	1800.00
data_137_245_2105_33	190.00	223.00	194.00	193.33	192.00	0.22	1800.00

Table 3: Detailed computational results for SMPTSP benchmark instances