Large neighbourhood search for large-scale shift assignment problems with multiple tasks

Pieter Smet · Greet Vanden Berghe

Abstract Shift assignment with multiple tasks presents a challenging combinatorial optimisation problem in which two decisions must be taken simultaneously. Both tasks and shifts must be assigned to qualified employees while minimizing costs originating from employee preferences. The present paper presents a natural integer programming formulation for this problem and introduces two lower bounds on the optimal solution quality. Two exponentially-sized neighbourhoods are used in a large neighbourhood search algorithm for improving initial solutions constructed by a greedy heuristic. Extensive computational experiments are analysed to gain insights into the performance and behaviour of the proposed solution approaches. All experiments are conducted on a randomly generated benchmark dataset inspired by real cases. The results demonstrate that the presented large neighbourhood search finds optimal solutions for all instances in relatively short computation time.

Keywords Task scheduling · Personnel rostering · Large neighbourhood search · Column generation

1 Introduction and background

The multi-task shift assignment problem (MTSAP) is a daily scheduling problem which requires tasks to be assigned to qualified employees, while simultaneously assigning shifts. Tasks represent an organisation's operations whose start time and duration are decided on before the MTSAP is solved and thus may not be modified. Tasks have qualification requirements which impose a hard constraint on their assignment to employees. A shift is defined as a continuous time interval with fixed start and end times in which an employee

P. Smet and G. Vanden Berghe
KU Leuven, Department of Computer Science, CODeS & iMinds - ITEC
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
Tel.: +32 92658704
E-mail: {pieter.smet, greet.vandenberghe}@cs.kuleuven.be

can be assigned to tasks. By modelling demand in terms of tasks, a closer approximation to the practical reality of staff scheduling problems is obtained which eliminates the, often unintuitive, translation from task-based demand to shift-based demand.

Shift scheduling and assignment problems have been studied for several decades [4]. In their most basic form, shift start and end times must be determined for a single day, based on staffing demands for each time interval. Over the years, several extensions of this problem have been considered which, for example, take into account break placement [2] or consider multiple days [7].

However, few publications focus on multi-task shift assignment as it is considered in the present paper. Smet et al. [14] address the multiple day variant of the MTSAP in which violations of employees' contractual constraints are minimised. Several constructive heuristics and very large-scale neighbourhood search algorithms are proposed and evaluated on a large benchmark dataset. Krishnamoorthy et al. [9] present a heuristic algorithm based on Lagrangian relaxation for a similar task scheduling problem in which shifts are already assigned and cannot be modified. Lequy et al. [11] employ a two-stage heuristic algorithm for assigning both tasks and interruptible operations to qualified employees. The simultaneous scheduling of tasks and activities in shifts is addressed by Boyer et al. [3] through branch-and-price. Other related research can be found in literature surveys concerning personnel scheduling [5,16].

Whereas previous research efforts primarily focused on solving problem instances with a limited number of employees (20-50), the present paper emphasises solving large-scale problems with many employees (≥ 100). Problem instances of this size are often encountered in large organisations such as airline [8] or railway operations [12]. Manually constructing solutions of such a large scale represents a complex and daunting task, thereby highlighting the significant benefits associated with automating the approach.

The remainder of this paper is organised as follows. Section 2 introduces the MTSAP and discusses its complexity. A natural integer programming formulation for the MTSAP is presented in Section 3. The heuristic algorithm used for solving the large problem instances considered by this paper is discussed in Section 4. Section 5 introduces two lower bounds for the MTSAP. Experimental results are presented and discussed in Section 6. Finally, Section 7 concludes the paper and identifies future research directions.

2 Problem definition

Let E be the set of employees and T the set of tasks which need to be assigned. Each employee $e \in E$ is qualified for a (sub)set of tasks $T_e \subseteq T$. Consequently, there is a (sub)set of employees $E_t \subseteq E$ qualified for task t. Let $S = S_w \cup \{s_0\}$ be the set of shifts, consisting of the subset of working shifts S_w and the shift s_0 which, when assigned, is a dummy shift assignment indicating that an employee is not working. A task t is covered by shifts in $S_t \subset S$, indicating that when this task is assigned, a shift within S_t must be assigned to the same employee. The model assumes that a task never spans two or more shifts, that is, there always exists at least one shift which completely covers the task. Furthermore, it is assumed that a solution in which all tasks can be assigned always exists.

Break placement is not considered in the present model. However, it is possible to model breaks if their start time and duration are fixed. To do this, shifts for which the break overlaps with task t can be simply removed from the set S_t. The objective is to minimise the costs c_{es} incurred by assigning employee e to shift s. The practical relevance of this objective is clear as it enables modelling employee preferences.

Table 1 provides an overview of the aforementioned notation.

s_0	dummy shift
c_{es}	cost for assigning shift s to employee e
T	set of tasks
S	set of shifts
E	set of employees
E_t	set of employees qualified for task t
S_w	set of working shifts, i.e. $S \backslash \{s_0\}$
S_t	set of shifts in which task t can be assigned
T_e	set of tasks for which employee e is qualified
C_e	set of maximal cliques in the interval graph defined for employee e

Table 1 Overview of notation

Theorem 1 *The MTSAP is NP-Hard.*

Proof The proof is based on restriction by showing that the Shift Minimisation Personnel Task Scheduling Problem (SMPTSP) is a special case of the MTSAP. In the SMPTSP, n tasks must be assigned to m employees, such that qualification requirements are respected and tasks overlapping in time are assigned to different employees. The objective is to minimise the number of employees that have at least one task assigned. Kroon et al. [10] showed that the SMPTSP is NP-Hard in the strong sense. A special case of the MTSAP is equivalent to an instance of the SMPTSP.

Let $S_w = \{s'\}$ be restricted to a single shift s' which can cover all tasks, or, formally, $S_t = \{s'\}$ for all $t \in T$. The set S furthermore consists of the dummy shift s_0. $c_{es'} = 1$ and $c_{es_0} = 0$ for all $e \in E$. All tasks in T and their parameters in the MTSAP are transferred directly to the SMPTSP instance.

It follows from this construction that minimizing the number of employees in the resulting SMPTSP instance is equivalent to minimizing the costs c_{es}. To construct the MTSAP solution from the SMPTSP solution, task assignments are directly transferred. Shift s' is only assigned to an employee if he has at least one task assigned. By doing so, the resulting objective value equals that of the SMPTSP solution. $\square$

3 Integer programming formulation

The natural integer programming formulation uses the maximal cliques in an interval graph $G_e = (V, A)$ to model overlapping tasks for each employee e. The graph G_e contains one node for each task in T_e. Edges are added between two nodes if their corresponding tasks overlap in time. Therefore, a maximal clique K in G_e represents a subset of tasks from which at most one may be assigned to employee e. The set containing all maximal cliques C_e in an interval graph G_e can be constructed in polynomial time [9]

Two sets of decision variables are used for assigning tasks and shifts to employees.

$$x_{te} = \begin{cases} 1 & \text{if task } t \text{ is assigned to employee } e \\ 0 & \text{otherwise} \end{cases}$$

$$y_{es} = \begin{cases} 1 & \text{if employee } e \text{ is assigned to shift } s \\ 0 & \text{otherwise} \end{cases}$$

The full integer programming formulation utilising these two sets of decision variables is denoted by NF.

$$\text{NF} : \mathit{min} \sum_{e\in E}\sum_{s\in S} c_{es}y_{es} \tag{1}$$

$$s.t. \sum_{e\in E_t} x_{te} = 1 \qquad \forall\, t\in T \tag{2}$$

$$\sum_{t\in K} x_{te} \leq 1 \qquad \forall\, e\in E,\ K\in C_e \tag{3}$$

$$\sum_{s\in S} y_{es} = 1 \qquad \forall\, e\in E \tag{4}$$

$$\sum_{s\in S_t} y_{es} \geq x_{te} \qquad \forall\, t\in T,\ e\in E \tag{5}$$

$$\sum_{t\in T_e} x_{te} \geq 1 - y_{es_0} \qquad \forall\, e\in E \tag{6}$$

$$y_{es} \in \{0,1\} \qquad \forall\, e\in E,\ s\in S \tag{7}$$

$$x_{te} \in \{0,1\} \qquad \forall\, t\in T,\ e\in E \tag{8}$$

Objective function (1) minimises the cost incurred by shift assignments. Constraints (2) make sure each task is assigned to a qualified employee. Overlapping task assignments are forbidden by Constraints (3). Constraints (4) require each employee to be assigned one shift. Constraints (5) link the x and y variables by stating that tasks must only be assigned to employees themselves assigned to a shift in which the task may be performed. Constraints (6) ensure that shifts are only assigned if an employee is assigned to at least one task in the shift. Finally, Constraints (7) and (8) impose bounds on the decision variables.

3.1 Branching priorities

To speed up a branch-and-bound algorithm for solving NF, branching priorities are assigned to the decision variables. The priorities are set such that the y variables are branched on first, resulting in many of the x variables to be set to zero due to tasks not fitting in the selected shifts. An alternative approach would be to first branch on the x variables, but given that there are more tasks than shifts, branching on the shift assignment variables first results in faster pruning of the search tree.

4 Large neighbourhood search

As this paper aims to address large instances of the MTSAP, solving the natural integer programming formulation does not present a viable approach. Therefore, this section introduces a heuristic algorithm based on local search which can cope with increasing problem size.

Local search algorithms typically employ neighbourhoods which may be explored relatively quickly but exhibit tendencies to result in locally optimal solutions. Very large-scale neighbourhood search algorithms (VLSN) overcome this disadvantage by employing exponentially large neighbourhoods [1]. The present paper proposes a large neighbourhood search (LNS) for the MTSAP, a special type of VLSN in which neighbourhoods are implicitly defined by a destroy and repair operator [13]. Algorithm 1 outlines the general LNS framework, with σ_0 corresponding to the initial solution, $f(\sigma)$ the evaluation function and $N(\sigma)$ the neighbourhood which is explored.

Algorithm 1 Large neighbourhood search

```
Input: $\sigma_0, f(\sigma), N(\sigma)$
1: $\sigma \leftarrow \sigma_0$                          ▷ $\sigma$ maintains the current solution
2: while stop criterion not met do
3:     $\sigma' \leftarrow N(\sigma)$                    ▷ select a neighbouring solution of $\sigma$
4:     if $f(\sigma') \preceq f(\sigma)$ then
5:         $\sigma \leftarrow \sigma'$
6:     end if
7: end while
8: return $\sigma$
```

The evaluation function $f(\sigma)$ calculates the number of unassigned tasks and solution cost defined by Equation (1). A lexicographical comparison, denoted by $\preceq$, determines whether or not new solutions are accepted, based firstly on the number of unassigned tasks and, secondly, on the solution cost. A neighbouring solution σ' is thus always accepted if it contains fewer unassigned tasks. Note that the problem definition requires all tasks to be assigned; the proposed approach thus allows infeasible solutions to be considered.

4.1 Initialisation

The initial solution σ_0 is generated using a greedy heuristic which attempts to assign each task $t \in T$ to a qualified employee already assigned to a shift from S_t. If no such assignment can be made, an employee from E_t who is not yet assigned a shift is randomly selected. The task, together with the lowest cost shift from S_t are subsequently assigned to this employee. If no such employee can be found, the task remains unassigned. Algorithm 2 shows an overview of the greedy heuristic.

Algorithm 2 Greedy heuristic

```
 1: for $t \in T$ do
 2:     unassigned ← true
 3:     for $e \in E_t$ do
 4:         if $e$ is assigned to a shift from $S_t$ and there is no overlap with other tasks then
 5:             assign task $t$ to employee $e$
 6:             unassigned ← false
 7:             break
 8:         end if
 9:     end for
10:     if unassigned then
11:         randomly select a non-working employee $e' \in E_t$
12:         assign task $t$ to employee $e'$
13:         assign the shift with the smallest $c_{es}$ from the set $S_t$
14:     end if
15: end for
```

4.2 Neighbourhoods

To improve the initial solution constructed by Algorithm 2, two very large-scale neighbourhoods are proposed. While both neighbourhoods employ an optimal repair operator, they are differentiated by which part of the solution is selected to destroy.

Horizontal neighbourhood $H(\sigma, b)$ Given a solution σ, this neighbourhood is defined by fixing all assignments of $(|E| - b)$ randomly selected employees. A neighbouring solution is obtained by solving the resulting subproblem to optimality. The parameter b has a direct impact on this neighbourhood's size, and thus also on its performance, whereby higher values of b typically result in better solutions [15]. The size of this neighbourhood is exponential, it may, however, be explored effectively by a state of the art branch-and-bound algorithm solving restricted instances of NF. Note that, if $b = |E|$, exploring $H(\sigma, b)$ is equivalent to solving the complete problem to optimality.

Hamming distance neighbourhood $HD(\sigma, h)$ Given a reference solution σ, a Hamming distance constraint limits the number of variables that may change value when (further) optimising the reference solution [6]. The neighbourhood $HD(\sigma, h)$ is defined by adding such a Hamming distance constraint (9) to NF, with $\bar{x}$ representing values of reference solution σ. By adding this constraint, at most $h\sqrt{|T|}$ tasks may be reassigned in each neighbouring solution. While this neighbourhood contains an exponential number of solutions, it may be effectively explored by using a branch-and-bound algorithm by imposing the strong Hamming distance restriction.

$$\sum_{t\in T}\sum_{e\in E}\bar{x}_{te}(1-x_{te}) \leq h\sqrt{|T|} \tag{9}$$

5 Lower bounds

To evaluate the quality of solutions obtained by the proposed LNS algorithm, two lower bounds on the optimal objective value are presented. While the first lower bound is based on solving an exponentially large formulation of the problem, the second lower bound can be easily calculated based on some of the problem's properties.

5.1 Set partitioning bound

The first lower bound is calculated by solving the linear programming relaxation of the problem's set partitioning formulation. In this formulation, each variable corresponds to a line of work which includes both task and shift assignments for a single employee. Due to the problem's combinatorial nature, this formulation contains an exponential number of variables. Therefore, column generation is used to solve the model.

Let P be the set of lines of work, with $P_e \in P$ representing a subset containing only those lines of work which are feasible for employee e. The cost c_{ep} for assigning line of work p to employee e is calculated based on the shift assignment costs c_{es}. Let a_{tp} be a binary value which equals one if task t is covered in line of work p, and zero otherwise. The integer programming model of the set partitioning formulation is denoted by SP.

$$x_{ep} = \begin{cases} 1 & \text{if line of work } p \text{ is assigned to employee } e \\ 0 & \text{otherwise} \end{cases}$$

$$u_t = \begin{cases} 1 & \text{if task } t \text{ is not assigned} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{SP}: min \sum_{e\in E}\sum_{p\in P_e} c_{ep}x_{ep} + \sum_{t\in T} Mu_t \tag{10}$$

$$s.t. \sum_{e\in E}\sum_{p\in P_e} a_{tp}x_{ep} + u_t = 1 \qquad \forall t\in T \tag{11}$$

$$\sum_{p\in P_e} x_{ep} = 1 \qquad \forall\, e\in E \tag{12}$$

$$x_{ep}\in\{0,1\} \qquad \forall\, e\in E,\ p\in P_e \tag{13}$$

$$u_t\in\{0,1\} \qquad \forall\, t\in T \tag{14}$$

Objective function (10) consists of two parts which minimise the costs incurred by assigning the selected lines of work and the cost of leaving a task unassigned. The second term has a weight M, which forces all tasks to be assigned. Constraints (11) ensure all tasks be covered by one of the selected patterns or by the task's u variable which indicates the task is unassigned. Constraints (12) require each employee to be assigned exactly one feasible line of work. Bounds on the decision variables are imposed by Constraints (13) and (14).

The column generation's pricing problem is solved to determine if there are variables with negative reduced cost defined as $\bar{c}_{ep} = c_{ep} - \sum_{t\in T} a_{tp}\pi_t - \gamma_e$ with π_t and γ_e the dual prices of Constraints (11) and (12), respectively. The resultant pricing problem is solved via the efficient decomposition approach proposed by Smet et al. [14]. The column generation's performance is further improved by stabilising the dual variables via dual smoothing and by speeding up convergence using a Lagrangian lower bound.

5.2 Shift cost bound

The second lower bound is derived from the shift assignment costs c_{es}. Since the problem definition requires each employee to be assigned one shift, the sum of each employee's best possible shift assignment is a valid lower bound on the optimal solution quality. Formally, the shift cost bound LB^{sc} is calculated via Equation (15).

$$\text{LB}^{\text{sc}} = \sum_{e\in E} \min_{s\in S} c_{es} \tag{15}$$

6 Computational experiments

A series of computational experiments are analysed to evaluate the performance of the new contributions.

6.1 Experimental setup

Given the unavailability of a publicly available benchmark dataset, the instance generator from Krishnamoorthy et al. [9] was adapted for the MTSAP. Table 2 shows which instance characteristics were varied and identifies the resulting 12 instance classes. The skilling level refers to the average percentage of tasks each employee is qualified for. Task length is sampled from a triangular distribution $tri(\alpha, \beta, \gamma)$, for which two settings were used. In each instance, three shifts are defined: an early shift (7:00 am - 3:00 pm), day shift (10:00 am - 6:00 pm) and late shift (1:00 pm - 9:00 pm). For each class, ten random instances were generated in which shift start time is advanced or postponed at most one hour and shift duration is shortened or extended at most 30 minutes with the aim of diversifying the set of instances.

Instance class id	Number of employees	Number of tasks	Number of shifts	Skilling level	Task length distribution
1	120	391	3	0.6	tri(25,75,150)
2	123	296	3	0.6	tri(50,100,175)
3	119	394	3	1.0	tri(25,75,150)
4	119	303	3	1.0	tri(50,100,175)
5	243	783	3	0.6	tri(25,75,150)
6	236	598	3	0.6	tri(50,100,175)
7	243	795	3	1.0	tri(25,75,150)
8	240	609	3	1.0	tri(50,100,175)
9	365	1158	3	0.6	tri(25,75,150)
10	360	897	3	0.6	tri(50,100,175)
11	366	1169	3	1.0	tri(25,75,150)
12	368	904	3	1.0	tri(50,100,175)

Table 2 Instance classes

All experiments were performed on a Dell Poweredge T620, 2x Intel Xeon E5-2670 with 128GB ram. CPLEX 12.6.1 was used as an integer programming solver, configured to use one thread and default settings. Linear programming problems were solved with the primal simplex algorithm in CPLEX 12.6.1. Whenever a time limit was imposed, it was set to 1800 seconds. In the LNS, the number of non-improving iterations was considered an additional stop criterion.

6.2 Lower bound quality

The first series of computational experiments compares the different lower bounds on the optimal solution quality. Specifically, emphasis is placed upon the linear programming (LP) relaxations of the NF and SP formulations. Table 3 shows, for the different instance classes, the objective value and calculation time in seconds for the LP relaxations of NF, denoted as $\tilde{\text{NF}}$, and of SP, denoted as $\tilde{\text{SP}}$. The reported values are the average value of all instances per class, with the best results highlighted in bold. Finally, the last column shows the value of the shift cost bound LB^{sc}.

Class id	$\tilde{\text{NF}}$		$\tilde{\text{SP}}$		LB^{sc}
	Obj	Time	Obj	Time	
1	**165.8**	200.6	**165.8**	16.8	165.8
2	**166.5**	4.9	**166.5**	7.6	166.5
3	**166.7**	568.3	**166.7**	305.2	166.7
4	**165.9**	183.4	**165.9**	105.5	165.9
5	67.9	1448.3	**339.6**	165.2	339.6
6	36.5	1623.7	**329.1**	70.0	329.1
7	**103.0**	1371.4	0.0	1801.5	333.6
8	102.1	1280.9	**262.6**	1448.3	328.5
9	0.0	1803.9	**507.0**	594.4	507.0
10	51.3	1629.9	**496.3**	277.9	496.3
11	**0.0**	1800.8	**0.0**	1811.2	503.3
12	**53.3**	1683.7	0.0	1803.7	506.8

Table 3 Comparison of lower bounds

Table 3's results demonstrate the tighter formulation of SP, compared to NF. For almost all instance classes, the best LP relaxation value is obtained by solving SP. Moreover, column generation achieves this in less calculation time on average. Such results confirm the potential of embedding the proposed column generation in branch-and-price with the aim of obtaining optimal integer solutions.

The shift cost bound consistently improves upon the two considered LP relaxations. This improvement is clear compared to $\tilde{\text{NF}}$, however, improvement compared to $\tilde{\text{SP}}$ only occurs when the column generation algorithm fails to converge within the time limit. Nevertheless, these results highlight the usefulness of the shift cost bound, as its required calculation times are orders of magnitude smaller.

6.3 Neighbourhood parametrisation

A series of experiments were conducted to gain insight into how parameter settings influence the neighbourhoods and affect their overall performance.

6.3.1 Tuning the horizontal neighbourhood

Neighbourhood $H(\sigma, b)$ is parametrised by b. Figure 1 shows the influence of increasing values for b on both the objective value and calculation time. For these experiments, there was no limit on the maximum number of non-improving iterations. Consequently, the calculation time is always 1800 seconds. Once b is greater than or equal to 20, the same objective value is always found.

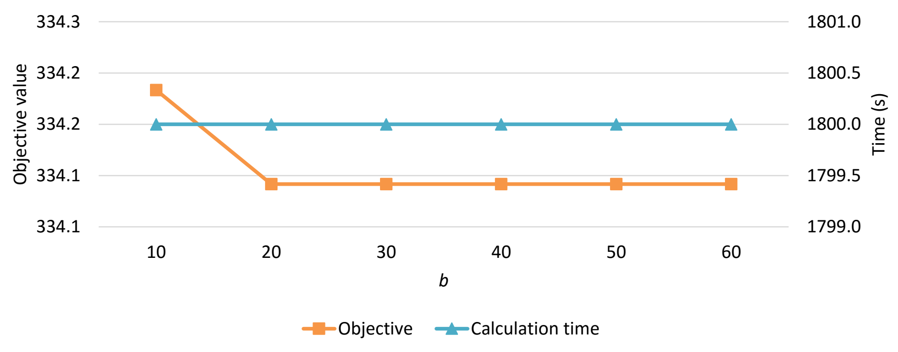

Fig. 1 Influence of b in $H(\sigma, b)$

Exploring this neighbourhood results in relatively quick convergence of the LNS. This insight may be exploited to reduce the calculation time by terminating the algorithm earlier than the maximum time limit by restricting the number of non-improving iterations. Figure 2 empirically demonstrates the trade-off between the maximum number of non-improving iterations and the average objective value and calculation time when $b = 20$.

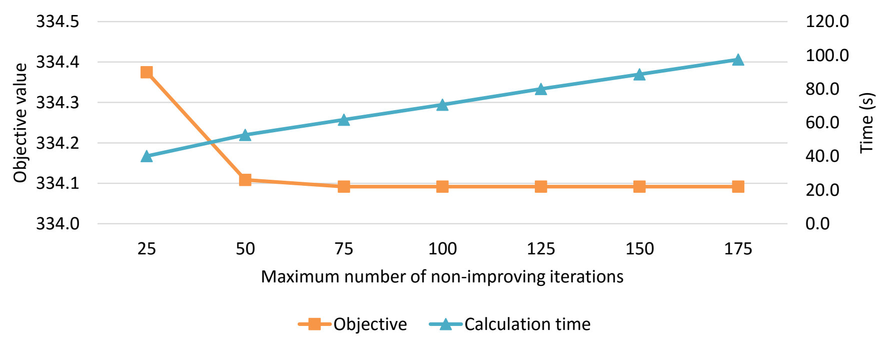

Fig. 2 Influence of the maximum number of non-improving iterations in $H(\sigma, 20)$

The obtained results demonstrate the expected linear relationship between the maximum number of non-improving iterations and the required calculation time. Regarding objective value, there is a clear convergence when 75 or more non-improving iterations are permitted. Based on these empirical observations, a maximum number of 75 non-improving iterations will be employed for the remaining computational experiments.

6.3.2 Tuning the Hamming distance neighbourhood

Neighbourhood $HD(\sigma, h)$ also has one parameter: the scaling factor h. Figure 3 shows the objective value and calculation time for increasing values of h. Given the nature of the Hamming distance neighbourhood, it suffices to set the maximum number of non-improving iterations to two, without risking prematurely stopping the algorithm before it has converged.

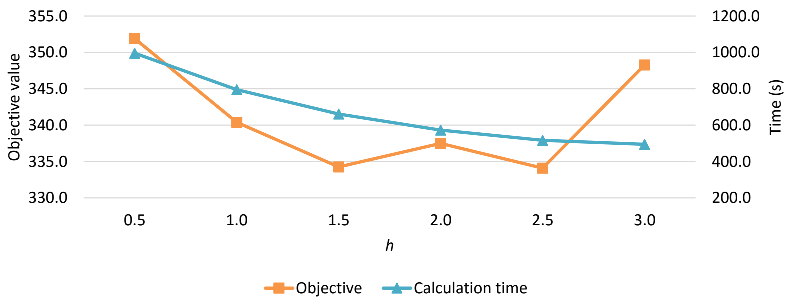

Fig. 3 Influence of scaling factor h in $HD(\sigma, h)$

Figure 2 visualises how increasing h enables more tasks to be reassigned by one move in this neighbourhood, resulting in lower calculation times. The trend demonstrated by the objective value is less clear. The lowest average objective values are obtained when f is greater than or equal to 1.5. While it is expected that the larger the neighbourhood the better the solutions, the increase at $h = 3.0$ is remarkable. The best solutions were obtained with $h = 2.5$.

6.4 Overall performance evaluation

A final series of experiments compares the performance of the different algorithms proposed in this paper: directly solving the natural formulation (NF), the greedy constructive heuristic (*Greedy*), LNS with the horizontal neighbourhood ($H(\sigma, 20)$), and LNS with the Hamming distance neighbourhood ($HD(\sigma, 2.5)$). Table 4 supplies an overview of the results. For all approaches, the average objective value and average calculation time in seconds are shown. Whenever an approach failed to find a feasible solution for all instances in a class, the percentage of feasible instances is given in brackets instead of the objective value. The final column shows a lower bound, calculated as $\mathrm{LB}^* = \max(\tilde{\mathrm{NF}}, \tilde{\mathrm{SP}}, \mathrm{LB}^{\mathrm{sc}})$.

By directly solving NF, large problem instances with 200 or more employees are not solved consistently. Moreover, for most instances, the maximum permitted calculation time is required. The LNS algorithms, by contrast,

Class id	NF		Greedy		$H(\sigma, 20)$		$HD(\sigma, 2.5)$		LB*
	Obj	Time	Obj	Time	Obj	Time	Obj	Time	
1	**165.8**	86.7	304.8	0.0	**165.8**	40.7	**165.8**	34.4	165.8
2	**166.5**	10.6	307.5	0.0	**166.5**	27.2	**166.5**	19.6	166.5
3	**166.7**	212.1	300.5	0.0	**166.7**	50.3	**166.7**	67.6	166.7
4	**165.9**	69.2	301.4	0.0	**165.9**	37.5	**165.9**	50.5	165.9
5	(30%)	1290.1	609.6	0.1	**339.6**	63.9	**339.6**	322.5	339.6
6	(90%)	996.7	584.2	0.0	**329.1**	44.2	**329.1**	182.7	329.1
7	(50%)	1329.3	604.7	0.1	**333.6**	83.0	**333.6**	638.8	333.6
8	(50%)	1133.5	600.7	0.1	**328.5**	53.4	**328.5**	417.7	328.5
9	(10%)	1801.0	916.5	0.1	**507.0**	84.5	**507.0**	990.6	507.0
10	(40%)	1395.8	900.0	0.1	**496.3**	78.6	**496.3**	742.6	496.3
11	(0%)	1801.5	906.4	0.1	**503.3**	111.8	**503.3**	1550.5	503.3
12	(10%)	1801.3	925.0	0.1	**506.8**	66.9	**506.8**	1179.8	506.8
Average	(57%)	994.0	605.1	0.1	334.1	61.8	334.1	516.5	334.1

Table 4 Comparison of different algorithmic approaches for the MTSAP

find feasible solutions for all instances. While the greedy heuristic responsible for generating the initial solutions proves successful, these solutions are of poor quality. Both neighbourhoods find optimal solutions for all considered instances. The time required by the LNS increases with larger problem sizes, however, for $H(\sigma, 20)$, the average calculation time is only little more than one minute.

For the considered problem instances, the results demonstrate that the lower bound based on shift costs always corresponds to the optimal objective value. This observation warrants further research into the empirical hardness of the used MTSAP instances and the employed instance generation procedure.

7 Conclusions and future work

The present paper addressed a large-scale multi-task shift assignment problem. A natural integer programming formulation was introduced, in addition to lower bounds based on linear programming relaxations and derived from the problem's structure. Computational results demonstrated that the bound based on the problem's structure was equal to the optimal solution cost for all considered problem instances. To find high quality solutions, two very large-scale neighbourhoods were proposed which were explored in a large neighbourhood search algorithm. This algorithm succeeded in solving problem instances with up to 368 employees to optimality within reasonable calculation time.

Computational experiments revealed the potential for embedding the presented column generation in a branch-and-price algorithm. Moreover, the problem's structured nature lends itself to a variety of other decomposition approaches, such as, for example, Benders decomposition. From a practical perspective, considering the problem's stochastic variant in which, for example, task durations are not deterministic, presents a relevant and challenging research objective.

Acknowledgements This work was supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX (http://comex.ulb.ac.be) and funded by research project 110257 of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Editorial consultation provided by Luke Connolly (KU Leuven).

References

1. Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics **123**(1-3), 75 – 102 (2002)
2. Bechtold, S.E., Jacobs, L.W.: Implicit modeling of flexible break assignments in optimal shift scheduling. Management Science **36**(11), 1339–1351 (1990)
3. Boyer, V., Gendron, B., Rousseau, L.M.: A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. Journal of Scheduling **17**(2), 185–197 (2014)
4. Dantzig, G.B.: A comment on Edie's 'Traffic delays at toll booths'. Journal of the Operations Research Society of America **2**(3), 339–341 (1954)
5. Ernst, A., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research **153**(1), 3–27 (2004)
6. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming Series B **98**, 23–47 (2003)
7. Jacobs, L.W., Brusco, M.J.: Overlapping start-time bands in implicit tour scheduling. Management Science **42**(9), 1247–1259 (1996)
8. Klinkert, A.: Large-scale rostering in the airport industry. In: Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling, pp. 493–494 (2014)
9. Krishnamoorthy, M., Ernst, A., Baatar, D.: Algorithms for large scale shift minimisation personnel task scheduling problems. European Journal of Operational Research **219**(1), 34–48 (2012)
10. Kroon, L.G., Salomon, M., Van Wassenhove, L.N.: Exact and approximation algorithms for the tactical fixed interval scheduling problem. Operations Research **45**(4), 624–638 (1997)
11. Lequy, Q., Desaulniers, G., Solomon, M.M.: A two-stage heuristic for multi-activity and task assignment to work shifts. Computers & Industrial Engineering **63**(4), 831 – 841 (2012)
12. Macedo, R., Benmansour, R., Urosevic, D., Artiba, A., Mladenovic, N.: Scheduling preventive railway maintenance activities with resource constraints. In: Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications, pp. 782–784 (2015)
13. Pisinger, D., Ropke, S.: Large neighborhood search. In: M. Gendreau, J.Y. Potvin (eds.) Handbook of metaheuristics, pp. 399–419. Springer (2010)
14. Smet, P., Ernst, A., Vanden Berghe, G.: Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. Computers & Operations Research (2016). DOI http://dx.doi.org/10.1016/j.cor.2016.05.016
15. Smet, P., Wauters, T., Mihaylov, M., Vanden Berghe, G.: The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. Omega **46**, 64–73 (2014)
16. Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L.: Personnel scheduling: A literature review. European Journal of Operational Research **226**(3), 367–385 (2013)