Journal of Scheduling
https://doi.org/10.1007/s10951-023-00789-3

Block-based state-expanded network models for multi-activity shift scheduling

Michael Römer[1]

Accepted: 7 June 2023

Abstract
This paper presents new mixed-integer linear programming formulations for multi-activity shift scheduling problems (MASSP). In these formulations, the rules governing shift feasibility are encoded in block-based state-expanded networks in which nodes are associated with states and arcs represent assignments of blocks of work or break periods inducing state transitions. A key advantage of these formulations is that for the anonymous MASSP in which all employees are considered as equal only a single network with integer flow variables is needed as long as the network encodes all shift composition rules. A challenging aspect is that the networks can become very large, yielding huge models that are hard to solve for large problem instances. To address this challenge, this paper proposes two exact modeling techniques that substantially reduce the size of the model instances: First, it introduces a set of aggregate side constraints enforcing that an integer flow solution can be decomposed into paths representing feasible shifts. Second, it proposes to decouple the shift composition from the assignment of concrete activities to blocks of work periods, thereby removing a large amount of symmetry from the original model. In a computational study with two MASSP instance sets from the literature dealing with shift scheduling problems, we demonstrate the effectiveness of these techniques for reducing the both size of the model instances and the solution time: We are able to solve all instances, including more than 70 previously open instances, to optimality–the vast majority of them in less than 30 min on a notebook computer.

Keywords Multi-activity shift scheduling · State-expanded networks · Mixed-integer linear programming

1 Introduction

In many industries, particularly in the service sector, employees incur a major part of the direct costs and have a key impact on the quality of the products and services delivered by organizations. Making effective use of the workforce by devising high-quality personnel schedules is thus a critical success factor. For a good overview of the literature dealing personnel scheduling problems, see, e.g. (Van den Bergh et al., 2013).

This paper deals with the *Multi-Activity Shift Scheduling Problem* (MASSP) consisting of composing anonymous work shifts that cover demands specified in terms of a time period (e.g. an interval of 15 min) and an activity (a type of work). The composition of work shifts needs to account for a set of rules governing aspects such as the length and placement of breaks, the total work hours and the maximum number of consecutive periods for which a certain activity can be performed. The objective function typically involves minimizing the total work hours scheduled and/or penalties for violating rules under- and/or over-covering demand. Typically, both the shift composition rules and the objective function vary between different MASSP variants.

A classical formulation for shift scheduling problems is the set covering formulation proposed by Dantzig (1954) where each of the main decision variables corresponds to a complete feasible shift. For many shift scheduling problems, however, the number of shifts is too large to enumerate explicitly. In particular, this is the case for multi-activity shift scheduling problems. As an example, the biggest instances considered in this paper exhibit billions of feasible shifts.

To avoid an explicit enumeration of all feasible shifts, many approaches for the MASSP rely on Branch-and-Price techniques in which the shift variables are generated as needed throughout the solution process by solving column

✉ Michael Römer
michael.roemer@uni-bielefeld.de

1 Decision Analytics Group, Bielefeld University, Universitätsstr. 25, 33615 Bielefeld, Germany

Published online: 22 July 2023

generation subproblems. As an example, Demassey et al. (2005) use Constraint Programming for solving the shift generation subproblem; more specifically, they use a cost-regular constraint that allows modeling shift feasibility using a regular language. Restrepo et al. (2012) solve the subproblems using a resource-constrained shortest path algorithm on an expanded graph, and Côté et al. (2013) formulated the subproblem using a grammar and solve it by dynamic programming.

Interestingly, the idea of describing the set of feasible shifts with formal languages is also used in approaches using (monolithic) MILP formulations: As an example, Côté et al. (2011a) represent the set of feasible shifts by a network flow component where the network is a directed acyclic graph derived from an automaton. Furthermore, Côté et al. (2011b) present a formulation in which shift rules are expressed using context-free grammars. The grammars are used to derive a hypergraph which is embedded into a MIP formulation. This formulation is very efficient since it does not require using a separate hypergraph per shift/employee, but only a single hypergraph with an integer hyperflow. A set of shifts can then be determined in a postprocessing step by decomposing the hyperflow into hyperpaths.

Another stream of shift scheduling research deals with so-called implicit formulations: In these formulations, shifts are not modeled explicitly but certain aspects such as the placement of breaks are considered implicitly using certain types of constraints such as the forward- and backward constraints proposed by Bechtold and Jacobs (1990). After solving the implicit problem, shifts can be recovered from the solution in a postprocessing step. Dahmen et al. (2018) are the first to propose such an implicit model for multi-activity shift scheduling problems. Their approach relies on enumerating partial shift schedules (pre- and post-break work stretches) forming the main decision variables; the allocation of breaks between these stretches is then implicitly modeled using forward and backward constraints.

In addition to the exact approaches sketched so far, there are also some heuristic approaches for the MASSP that have been proposed in the literature, for example a large neighbourhood search (Quimper & Rousseau, 2010) and a recently published Lagrangian relaxation-based matheuristic (Hernández-Leandro et al., 2019). The latter article focuses on the *personalized* MASSP, but it also includes experiments with the MASSP instances from Demassey et al. (2005) that we also consider in this paper.

This paper presents an efficient MILP formulation for the MASSP that is based on the idea of encoding MASSP rules in state-expanded networks. Formulations based on state-expanded networks were previously applied to personnel scheduling problems such as airline crew scheduling (Mellouli, 2001) and nurse rostering (Römer & Mellouli, 2016). Recently, Porrmann and Römer (2021) introduced a formulation based on a state-expanded network for the MASSP variant introduced by Demassey et al. (2005).

In a state-expanded network, nodes are associated with rule-relevant states and arcs represent transitions between states, typically induced by assigning pieces of work (e.g. flights in airline crew scheduling or shifts in nurse rostering). The network is designed in a way that each path in the network corresponds to a schedule (or crew pairing) that is feasible with respect to a set of rules. The network is then embedded into a MILP model as a network flow component. An important advantage of this type of model is demonstrated in Mellouli (2001) for the case of the airline crew pairing chain problem: If certain employees (crew members) can be considered to be identical/anonymous and if the state-expanded network encodes the full set of schedule (pairing) legality rules, then one can use an aggregated integer-valued flow in a single network for these employees instead of introducing a separate network each employee. Similar to the implicit grammar model from Côté et al. (2011b), one can then decompose the integer flow into paths to obtain feasible schedules (crew pairing chains).

Since in the MASSP, all employees are considered identical, a state-expanded network model for this problem can also use a single network with integer flow variables. It turns out, however, that a state-expanded network that encodes all shift composition rules quickly becomes huge; this leads to the fact that for complex large-scale MASSPs, a "plain" state-expanded network model is neither practically useful nor competitive with state-of-the art approaches such as the grammar-based models from (Côté et al., 2011b) or the implicit formulation proposed in (Dahmen et al., 2018). To address this issue, Porrmann and Römer (2021) proposed to employ a machine learning approach for heuristically reducing the size of the state-expanded network. The downside of such a heuristic reduction is that the state-expanded network no longer represents all feasible shifts, rendering the whole approach non-exact. The main contribution of this paper is to propose a new formulation based on block-based state-expanded networks that can be efficiently solved without resorting to such heuristic network reduction to the extent that it outperforms other state-of-the art exact approaches.

1.1 Contributions

This paper presents a new efficient formulation for the MASSP that relies on efficiently representing shifts in block-based state-expanded networks. A key design decision in the proposed model is to use full activity blocks (consecutive assignments of the same activity) as the basis for constructing arcs of the state-expanded network. While this idea by itself does not yield a model that is competitive with the state-of-the art, it provides the basis for two exact techniques that, in particular when being combined, help to substantially reduce the size of the model instances. First, we introduce a set of constraints that implicitly ensure that the (aggregated) flow in the state-expanded network can be decomposed into shifts in which two consecutive activity blocks do not have the same activity. This means that this requirement does not need to be encoded in the network, resulting in a reduction in the number of nodes (and activity block arcs) by a factor of about $|A| - 1$ where $|A|$ is the number of activities in the instance. Second, we introduce the idea of activity block templates: An activity-block template is an activity-agnostic block associated with the arcs in the state-expanded network. By making the state-expanded network activity-agnostic, its size is reduced by a factor of $|A|-1$. The assignment of activities to the block templates is then performed by assignment variables that are linked to the flow in the state-expanded network. Finally, we show how these two ideas can be combined by embedding the activity assignment variables in a second state-expanded network that is used for composing (activity-specific) work blocks. For complex MASSP problems such as the problem considered in Dahmen et al. (2018), applying both ideas results to a reduction in the size of the main state-expanded network in the order of the square of the number of activities in a given instance.

In the experiments with two sets of problem instances from the literature that include more than 70 previously unsolved instances, we show that our approach is able to solve all instances to optimality on a notebook computer in less than one hour; more specifically, only five out of more than 1000 instances require more than 30 min to be solved to optimality.

The remainder of this paper is structured as follows: The next section provides a description of multi-activity shift scheduling problems (MASSPs), in particular including the MASSP variants introduced in Demassey et al. (2005) and (Dahmen et al., 2018) studied in our computational experiments. Section 3 describes how to construct block-based state-expanded networks encoding the shift composition rules typically arising in MASSPs, and in particular how to construct networks encoding the rules from the two problem variants presented in Demassey et al. (2005) and Dahmen et al. (2018). Section 4 presents how to embed these networks into a MILP formulation for MASSPs, and it also presents a set of implicit constraints ensuring that an aggregate flow can be decomposed into feasible paths without having to encode the activity change requirement in the network. Section 5 presents the idea of activity block templates and shows how they can be used for reducing the overall model size; furthermore it shows how they can be used to create models involving two coupled state-expanded networks operating on different levels of detail. Section 6 presents the results from our computational experiments, followed by the conclusions in Sect. 7.

2 Multi-activity shift scheduling problems

A MASSP deals with compiling (anonymous) work shifts that are used to cover a fixed work demand given per period $p \in P$ where P is a set of time periods constituting a work day. The number of shifts to be compiled can either be given or free, and the objective typically consisting of minimizing an objective involving total scheduled work time and penalties for under- or overcovering demand or for violating soft shift legality rules. What sets the MASSP apart from a classical (mono-activity) shift scheduling problem is the fact that demand in each period p is provided for different work activities $a \in A$. For each of these activities, there is usually a minimum and a maximum number consecutive periods for which it can be assigned before switching to another activity or to a break. Note that in the "plain" MASSP considered in this paper, it is assumed that each employee can be assigned to each activity. The presence of employee-specific activities turns the problem in to a personalized MASSP which is studied in Côté et al. (2013), for example. A key feature of shift scheduling problems is the fact that the shifts need to respect a (problem-specific) rule set that governs aspects such as break requirements and the number and duration of consecutive work periods. In many MASSP problem variants, the rule set involves multiple shift types (e.g. short and long shifts) affecting the parameters of certain rules. As an example, the maximum allowed shift duration is typically different for short and long shift types.

To facilitate the description of typical types of rules (and to lay the ground for the modeling approach presented later), let us introduce the following terms that view a shift as a hierarchical composition of blocks:

- An *elementary assignment* or *elementary block* is an assignment of an activity $a \in A$ to a period $p \in P$
- An *activity block* is a block of consecutive elementary assignments of one type of activity
- A *work block* is a block of consecutive activity blocks
- A *break block* is a block of consecutive break periods
- A *shift* is an alternating sequence of work and break blocks starting and ending with a work block

These components or "building blocks" of shifts can be used to express shift composition rules typically arising in an MASSP in a straightforward way. This is due to the fact that many of these rules can be expressed as constraints regarding the properties of blocks (e.g. minimum or maximum duration of activity blocks, work blocks or break blocks; and number of breaks and number of work periods in shifts) or as constraints affecting the composition of blocks from other blocks (e.g. alternation of work and break blocks in shifts, no consecutive blocks of the same work activity in one work block). To see examples for concrete rule sets and concrete MASSP variants, let us now discuss two MASSP variants from the literature.

2.1 MASSP variant from (Demassey et al., 2005)

In this problem variant, which will be referred to as the Demassey problem in the rest of this paper, each instance has a fixed number of employees and a given number of different activities. There are two shift types (short and long shifts); short shifts have a single short break and long shifts have two short breaks and one long break. Shifts must completely fall into a work day consisting of 96 periods of 15 min; demand is given for each activity and for each period. Under- and overcovering of demand are allowed, but penalized in the objective function. The other part of the (minimization) objective involves the costs associated with each scheduled work period. The shift legality rules can be stated as follows:

1. Each activity block has a minimum duration of four periods
2. Each work block is composed of a single activity block
3. The duration and composition of a shift depends on the shift type:
   (a) Short shifts exhibit at least 3 and less than 6 h of work; they consist of two work blocks separated by a short break of 15 min
   (b) Long shifts exhibit least 6 h and at most 8 h of Work; they consist of four blocks work blocks separated by two short breaks of 15 min and a long break of 1 h (breaks can placed in arbitrary order)

2.2 MASSP variant from (Dahmen et al., 2018)

In contrast to the Demassey problem, the MASSP variant presented by Dahmen et al. (2018) (subsequently called the Dahmen problem), does not deal with a fixed number of employees/shifts, but the number of shifts can be freely chosen. Depending on the instance, there are different numbers of activities and different shift types affecting the length of the break (each shift involves a single break, irrespective of the shift type), the length of the work blocks and the total shift duration.

The time granularity is either 15, 30 min or 1 h, and shifts can exceed the planning horizon of 1 day. Activity blocks exceeding the planning horizon do not contribute to demand covering, but they contribute to the objective function that consists in minimizing the total number of periods worked. The demand has to be covered (undercovering is not allowed), and overcovering is allowed and not penalized.

The shift legality rules in the Dahmen problem can be stated as follows:

1. A shift can only start at certain periods $P^{\text{start}} \subset P$.
2. There are activity-specific minimum and maximum activity block durations.
3. There are minimum and maximum durations for work blocks that depend on the type of shift and on whether the work block is before or after the break.
4. A work block can be composed of multiple consecutive activity blocks; if a work block is composed of multiple activity blocks, then two consecutive activity blocks need to have different activity types (we will subsequently call this rule the *activity change requirement*).
5. Each instance may have multiple shift types; the type of shift governs:
   (a) The minimum and maximum duration of a shift (break periods are counted).
   (b) The minimum and maximum duration of the pre- and post-break work blocks.
   (c) The (fixed) duration of the break.

In addition to these rules, Dahmen et al. (2018) discuss a "restricted" variant of their problem in which it is only allowed to assign two different activities within a single work block. In other words, in that variant, each work block contains activity blocks with at most two different activity types.

3 Modeling shifts with block-based state-expanded networks

Our approach for the MASSP relies on the central idea to encode all shift composition rules in a (directed) state-expanded network $G = (N, E)$ in a way that each path from the source node v^{source} to the sink v^{sink} corresponds to a feasible shift and the set of all source-sink paths in G corresponds to the set of all feasible shifts. Each node in $N^{\text{state}} = N \backslash \{v^{\text{source}}, v^{\text{sink}}\}$ is associated with a rule-related state s_v which typically consists of a tuple of state attributes. The arc $e^{\text{circ}} = (v^{\text{sink}}, v^{\text{source}}) \in E$ is denoted as the flow circulation arc, and in case of a given number of employees n, its flow value is fixed to n. All arcs between the nodes in N^{state} represent state transitions induced by assigning a feasible work activity block or a break block. Note that by associating arcs with full activity and break blocks, all rules related to activity blocks (e.g. minimum and maximum duration) and break blocks (e.g. the possible break lengths) are satisfied by design since only legal activity and rest blocks are considered in the network. Observe that a difference to other related approaches such as the automaton models discussed in Côté et al. (2011a) where each transition is associated

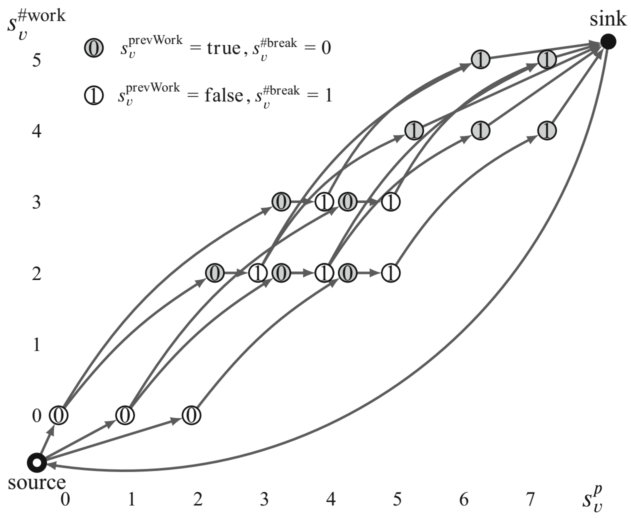
$s_v^{\#work}$
sink
$s_v^{prevWork}$ = true , $s_v^{\#break}$ = 0
$s_v^{prevWork}$ = false , $s_v^{\#break}$ = 1
5
4
3
2
1
0
source
0 1 2 3 4 5 6 7
s_v^p

Fig. 1 Block-based state-expanded network for the example

with an elementary (single-period) assignment, and also to the state-expanded network representation proposed by Pormann and Römer (2021) where transitions are associated with partial activity blocks.

Example To illustrate the construction of such a block-based state-expanded network, let us consider a small and simplified single-activity shift scheduling problem with a planning horizon of seven periods. In the example problem, a feasible shift needs to satisfy the following (hard) shift composition rules: A shift needs to contain either five or six periods of work spread across two work blocks that have to be separated by a single-period break. An activity block has a minimum duration of two periods and a maximum duration of three periods.

To model this problem, we assign each node $v \in N^{\text{state}}$ a state s_v consisting of four state attributes:

- s_v^{p} is the period index of the node.
- s_v^{prevWork} is a Boolean attribute denoting whether the last assignment was a work activity work or not.
- $s_v^{\text{\#work}}$ is a counter of the number of work periods assigned so far.
- $s_v^{\text{\#break}}$ corresponds to the number of breaks taken so far (in the example, either 0 or 1).

The resulting network with all feasible nodes and arcs is displayed in Fig. 1. The attributes s_v^{p} and $s_v^{\text{\#work}}$ of the nodes in N^{state} are visualized using the position of the node (s_v^{p} corresponds to the x-axis, $s_v^{\text{\#work}}$ to the y-axis); the attribute $s_v^{\text{\#break}}$ is used as node label and the value of s_v^{prevWork} is indicated by the color of the node.

The arcs between nodes in $v \in N^{\text{state}}$ represent state transitions induced by assigning activity blocks and break blocks. In particular, the arcs emanating from a node v for which the value of $s_v^{\text{prevWork}} = \text{false}$ represent the feasible activity blocks that can start from the state s_v and end in a feasible state (e.g. for a node v with $s_v^{\text{prevWork}} = \text{false}$ and $s_v^{\text{\#work}} = 4$, only a single activity block with a duration of two results in a state that respects the maximum number of work periods per shift).

The nodes for which $s_v^{\text{prevWork}} = \text{true}$ and $s_v^{\text{\#break}} = 0$ have an outgoing break arc representing a single-period break. The rule that shifts need to be composed of two work blocks separated by a rest block is ensured by the fact that there are only arcs to the sink from a node with $s_v^{\text{\#break}} = 1$ and $s_v^{\text{prevWork}} = \text{true}$. Similarly, the rule limiting the number of total work periods to be either five or six is ensured by the fact that there are only arcs from nodes $v \in N^{\text{state}}$ to v^{sink} if $5 \leq s_v^{\text{\#work}} \leq 6$. The remaining arcs represent the connections between v^{source} and the initial state nodes (for which $s_v^{\text{prevWork}} = \text{false}$) and the flow circulation arc e^{circ} connecting the sink to the source node.

Multi-activity work blocks. A simplifying assumption in the example above is that there is a single type of activity. In such a setting, each work block consists of a single activity block. In a multi-activity setting, a work block can consist of multiple consecutive activity blocks with different work activities. Activity blocks with the same start period and end period and different activities are represented by a separate (parallel) activity block arc for each activity.

In order to model constraints on the duration of work blocks, we introduce a state attribute $s_v^{\text{\#pWb}}$ counting the number of work periods in the work block. Given an activity block arc from node v to node w with length p, the transition function with respect to that attribute is $s_w^{\text{\#pWb}} = s_v^{\text{\#pWb}} + p$. After a break, this attribute is reset, that is, a node w that forms the target of a break arc has $s_w^{\text{\#pWb}} = 0$.

If a work block can consist of multiple activity blocks, a natural rule is that two consecutive activity blocks must be assigned to different activities. To encode this rule in the network, we can introduce a state attribute s_v^{prevAct} indicating the previously assigned activity. Then, if the previous block was assigned to activity a, that is, $s_v^{\text{prevAct}} = a$, only arcs representing blocks with activities $a' \neq a$ can emanate from v. If the number of different activities per work block is limited, this can be represented by a set-valued state attribute s_v^{actWb} that records the set of activity types assigned in a work block. That state attribute is reset to the empty set after the end of a work block.

Modelling different shift types Let us now see how we can deal with multiple shift types imposing different break patterns and shift lengths, denoting the set of shift types with Q. It turns out that we do not need to include an extra state attribute for representing shift types. Instead, for each partial shift ending at node v, we can determine the subset $Q_{s_v} \subset Q$

of shift types for which s_v is a feasible state. At the beginning of a shift, say, in the first work block, a state is likely to be feasible for various or all shift types. Later on, in particular depending on the break periods assigned, a state may only be feasible for a single shift type. In the construction of the state-expanded network, this means that it is checked for each possible transition after a node v whether the target state after the transition is feasible for at least one shift type. Only if this is the case, the corresponding target node and the arc corresponding to the transition is included in the network.

3.1 State model for the Demassey problem

The Demassey problem is a MASSP, but it has the special (and simplifying) rule that every work block consists of a single activity block. As a result, we do not have to explicitly model rules governing the composition of work blocks from activity blocks, and also the workblock duration rules are dealt with by the fact that each activity block assignment represents a full work block. The state variable thus only needs to keep track of whether the last assignment was work or not; we use the state variable s_v^{prevWork} for this purpose. There are, however, two types of shifts that not only vary with respect to the minimum and maximum duration but also with respect to their break configuration. To keep track of the break configuration, our state variable represents the number of short and long breaks assigned so far in the state attributes $s_v^{\#shortbreak}$ and $s_v^{\text{\#longbreak}}$. Finally, we use the attribute $s_v^{\text{\#work}}$ to count the total number of work periods.

To summarize, the state attributes needed are:

- s_v^{p} period p.
- s_v^{prevWork} Boolean state indicating whether the previous assignment was work or not.
- $s_v^{\text{\#shortBreak}}$ number of short breaks assigned.
- $s_v^{\text{\#longBreak}}$ number of long breaks assigned.
- $s_v^{\text{\#work}}$ number of total work periods assigned in the path so far.

3.2 State model for the Dahmen problem

While in the Demassey problem, each work block consists of a single activity block, the Dahmen problem permits work blocks with multiple consecutive activity blocks as long as two consecutive activity blocks within a work block do not exhibit the same activity (the activity change requirement) and work block length constraints are respected. As explained above, we can model these multi-activity work blocks using the state variables $s_v^{\text{\#pWb}}$ and s_v^{prevAct}. In the Dahmen problem, a shift only has a single break and the length of the break depends on the shift type. In order to determine the shift type implied by the break, we need an attribute that does not only record if there was a break but also the length of that break, we use the attribute $s_v^{\text{breakLength}}$ (which is 0 if there was no break) for this purpose.

To summarize, the state attributes needed to represent the shift rules for the Dahmen problem are:

- s_v^{p} period p.
- s_v^{prevAct} previously assigned activity if the previous assignment was work, otherwise (e.g. in case of a break), the value is *None*.
- $s_v^{\text{\#pWb}}$ number of work periods assigned in the current work block.
- $s_v^{\text{\#work}}$ number of total work periods assigned in the path so far.
- $s_v^{\text{breakLength}}$ length of the break in the path (0 if there was no break).

As explained above, the restricted variant of the Dahmen problem only permits two different activities to be assigned per work block. To model this constraint, we introduce an additional state attribute:

- s_v^{actWb} set of activity types that have occurred in the work block so far.

This state attribute is set to the empty set for each node representing the beginning of a work block. A transition induced by the assignment of a work activity adds the type of the activity to the set s^{actWb}. If the set s_v^{actWb} has a cardinality $|s^{\text{actWb}}| = 2$, only one of the two activities in the set can be chosen next, namely the activity type $s^{\text{actWb}} \setminus s_v^{\text{prevAct}}$ which does not induce a violation of the activity change requirement.

4 MILP formulations

The state-expanded network explained in the previous section constitutes a core element of our MILP formulation for the MASSP. In this section, we first describe the basic formulation and then an implicit formulation of the activity change requirement that allows moving this rule out of the state-expanded network, reducing its size by a factor of about $|A| - 1$ where $|A|$ is the number of activities in the problem.

4.1 Basic MILP formulation

The state-expanded network enters the model in form of a network flow component. The flow on an arc $e \in E$ is represented by the integer decision variable X_e. The cost of a unit flow on arc e is denoted as c_e. As an example, if arc e represents a work activity block, this cost factor may include the cost induced by the number of work periods in that block. The other two sets of decision variables are $Y^{\mathrm{u}}_{a,p}$ and $Y^{\mathrm{o}}_{a,p}$ which model the under- and overcovering of the demand $d_{a,p}$ of activity a in period p; these variables are associated with penalties c^{u} and c^{o} for under- and overcovering. Note that in case of hard demand covering limits as in the Dahmen problem, the corresponding variables can be forced to 0.

Using the described symbols, the MILP model can be written as follows:

$$\min \sum_{e \in E} c_e X_e + \sum_{a \in A} \sum_{p \in P} \left(c^{\mathrm{o}} Y^{\mathrm{o}}_{a,p} + c^{\mathrm{u}} Y^{\mathrm{u}}_{a,p}\right) \qquad (1)$$

$$\sum_{e \in v^{\mathrm{in}}} X_e = \sum_{e \in v^{\mathrm{out}}} X_e \qquad \forall v \in N \qquad (2)$$

$$X_{e^{\mathrm{circ}}} = n \qquad (3)$$

$$\sum_{e \in E^{\mathrm{cov}}_{a,p}} X_e + Y^{\mathrm{u}}_{a,p} - Y^{\mathrm{o}}_{a,p} = d_{a,p} \qquad \forall a \in A,\ p \in P \qquad (4)$$

$$X_e \in \mathbb{Z}^+_0 \qquad \forall e \in E \qquad (5)$$

$$Y^{\mathrm{o}}_{a,p} \geq 0, \quad Y^{\mathrm{u}}_{a,p} \geq 0 \qquad \forall a \in A,\ p \in P \qquad (6)$$

The objective function (1) contains the cost induced by the flow in the state-expanded network and the penalties for over- and undercovering demand. (2) are the flow balance constraints for each node v ensuring that the flow on the incoming arcs v^{in} equals the flow on the outgoing arcs v^{out}, and constraint (3) fixes the flow on the circulation arc e^{circ} to the number n of employees. In case that the number of employees is not fixed but part of the scheduling problem, this constraint can be dropped, and the cost of an employee can be modeled in the cost coefficient c^{circ}_e. Constraint set (4) models the demand covering for each activity and period; the set $E^{\mathrm{cov}}_{a,p} \subset E$ is the set of arcs representing an assignment that covers activity a in period p. The other two constraint sets determine the domains of the decision variables.

A solution to problem (1)–(6) contains the integer flow in network G; the solution flow $X^*_{e^{\mathrm{circ}}}$ on the circulation arc corresponds to the total number of units flowing through the network. Using flow decomposition, we can obtain $X^*_{e^{\mathrm{circ}}}$ paths between the source and the sink each of which corresponds to a shift. Note that in general, such a flow decomposition is not unique; that is, a given flow solution may be decomposable in different paths (representing different sets of shifts).

4.2 Implicit activity change constraints

As explained in Sect. 3, representing the rule that two consecutive work activity blocks need to have different activities requires introducing a state attribute that stores the activity assigned in the previous block. Given that $|A|$ is the number of activities, introducing this state attribute increases the number of nodes (and arcs) by a factor of about $|A| - 1$. In order to avoid this increase, we propose a set of linear constraints that implicitly enforce that the flow in the state-expanded network is decomposable into a set of shifts respecting the activity change requirement.

If we assume that the activity change rule is not embedded in the state-expanded network, it may happen that the flow through a node v representing the connection of two consecutive activity blocks is not decomposable in a way that for each path through v, the activity associated with the in-arc of v in the path is different from the activity of the out-arc of v. We refer to a node connecting two activity blocks as an *interior node* of a work block; the set of these nodes will be written as N^{inWb}. Using the state attribute $s^{\#\mathrm{pWb}}_v$ introduced in Sect. 3, we can state that N^{inWb} is the set containing all nodes with $s^{\#\mathrm{pWb}}_v > 0$ that have a least one outgoing arcs representing an activity block.

To enforce that for each of the nodes $v \in V^{\mathrm{inWb}}$, there exists a flow decomposition respecting the activity change rule, we impose a set of constraints that ensures that the total flow on the out-arcs of v representing a block with activity a (the set of these arcs is denoted as v^{out}_a) is smaller or equal than total flow on the arcs in $v^{\mathrm{in}}_{\neg a}$ representing the in-arcs associated with activity blocks for activities other than a. This set of constraints can be written as:

$$\sum_{e \in v^{\mathrm{in}}_{\neg a}} X_e \geq \sum_{e \in v^{\mathrm{out}}_a} X_e \qquad \forall v \in N^{\mathrm{inWb}}, a \in A \qquad (7)$$

Proposition 4.1 *Constraints 7 ensure that the flow through node $v \in N^{\mathrm{inWb}}$ can be decomposed in a way that the resulting (partial) work blocks respect the activity change requirement, that is, that a work block does not contain two consecutive activity blocks assigned to the same activity.*

Proof We will now give a constructive proof for the existence of a such a feasible decomposition.

We consider a node $v \in N^{\mathrm{inWb}}$ for which constraints (7) hold, and we assume that we have a non-negative flow through v. A flow unit on an incoming (outgoing) arc of such a node represents an activity block that we denote as incoming (outgoing) activity block. Recall that we want to show that each outgoing block with a certain activity a can be linked to an incoming block with an activity $a' \neq a$.

Constraints (7) ensure that in a feasible solution, the number of outgoing activity blocks with $activity(b^{\mathrm{out}}) = a$ is smaller or equal than the number of incoming blocks not assigned to a. The question we address in this proof is whether this is sufficient to guarantee that we can assign a correct incoming block b^{in} to every outgoing block b^{out} in a way that $activity(b^{\mathrm{in}}) \neq activity(b^{\mathrm{out}})$. We will show this by providing a decomposition procedure that, given that constraints (7) hold, is guaranteed to find such an assignment.

A key concept for our decomposition procedure is that of *complementary pairs*: Two activity block pairs $(b_1^{in}, b_1^{\text{out}})$ and $(b_2^{in}, b_2^{\text{out}})$ are complementary if $activity(b_1^{\text{in}}) = activity(b_2^{\text{out}})$ and $activity(b_2^{\text{in}}) = activity(b_1^{\text{out}})$. If there exists such a set of complementary pairs in the flow through a node v, we can extract the corresponding flow and obtain two partial work blocks respecting the activity change requirement.

Observe that if we have a flow solution for which the constraints (7) hold for a node v and if we extract the flow corresponding to two complementary pairs from that solution, then the two constraints associated with v and the involved activities will also hold after that operation since for both constraints, both the left-hand side and the right-hand side are reduced by one.

Our decomposition procedure for a flow through a node v starts with the full flow solution and extracts complementary pairs until no more such pairs are found. After having removed all complementary pairs, either each outgoing activity block is assigned a feasible incoming block (there is no residual flow on an activity-block arc going out of v) or there are outgoing blocks left which are not part of any complementary pair given the remaining incoming blocks. As noted above, however, the constraints are still valid for the residual flow solution, thus for each outgoing block with activity a there must exist at least one incoming block with an activity other than a. In addition, since there do not exist any complementary pairs in the residual flow, this means that the set of activities associated with residual incoming blocks is disjoint from the set of activities associated with the outgoing activity blocks. This means that we can randomly assign one of the incoming blocks represented by the residual flow to each of the outgoing blocks to obtain a feasible decomposition. $\square$

5 Reducing model size and symmetry with activity block templates and coupled networks

The state-expanded network formulation discussed above gives rise to large model instances exhibiting a considerable amount of symmetry. To illustrate the symmetry which is inherent in many MASSPs, see Fig. 2 depicting four shifts for a small example with three activities. Each shift consists of two activity blocks separated by a two-period break. In this schedule, certain activity blocks with the same start and end period can be exchanged between the shifts without affecting the quality of the solution: As an example, the first activity blocks in the first two shifts can be exchanged. In addition, the last activity block in the first shift and the first activity block in third shift can be exchanged, as well as the last block in the second shift and the first block in the fourth shift.

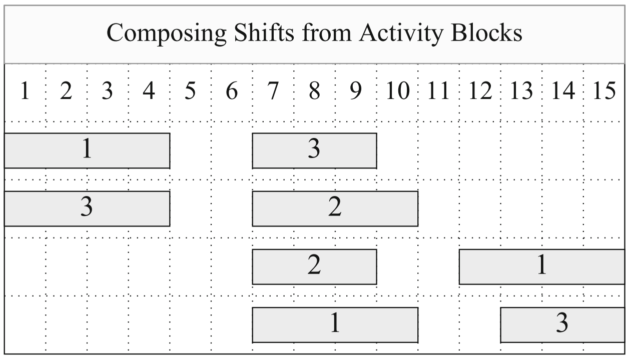

Fig. 2 Example: a multi-activity shift schedule with a lot of symmetry

In a state-expanded network in which activity blocks are associated with arcs, these symmetrical shifts correspond to symmetrical paths in the network. In particular, the exchangeable blocks in shifts one and three (and those in shifts two and four) correspond to arcs emanating from nodes associated with different states. For each of the states and for each feasible block, the network contains a parallel arc for each activity. This means that the decision which activity should be assigned to a block is "repeated" for each possible state, despite the fact that for a solution like the one displayed in Fig. 2, many solutions with arcs starting in different states are equivalent.

We propose to avoid this symmetry by moving the activity assignment decision out of the state-expanded network and replace the activity-specific activity block arcs with arcs associated with "anonymous" *activity block templates*. This means that the flow in the state-expanded network only decides that certain activity blocks with a given start and end time are placed in a shift, but not to which activity this block is assigned. The assignment of concrete activities is then delegated to a separate model part that ensures that for each activity block template (characterized by start and end period), the number of matching "concrete" activity blocks that are assigned equals the flow on all arcs representing the corresponding activity block template.

For the example from Fig. 2, this idea is illustrated in Fig. 3: In the top part, the shifts from Fig. 2 are displayed as "anonymous" blocks or activity block templates. The bottom part of the figure displays the number of times each activity type is assigned to each template block. As an example, there exist two templates representing an activity block from period 7 to period 9. The bottom part of the figure shows that these two templates will be "filled" with one block assigned to activity 2 and one block assigned to activity 3–the crucial idea here is that the template-based model does not explicitly assign the activity-specific blocks to the activity-agnostic template blocks but merely ensures that they *can* be assigned.

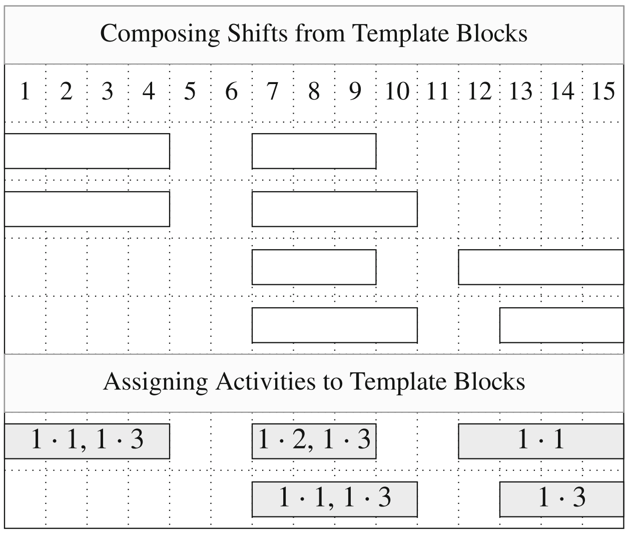
Composing Shifts from Template Blocks
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Assigning Activities to Template Blocks
1 · 1, 1 · 3
1 · 2, 1 · 3
1 · 1
1 · 1, 1 · 3
1 · 3

Fig. 3 Avoiding symmetry by decoupling shift composition and activity assignment

In other words, in contrast to the original model, the template-based model does not need to decide between solutions that are equivalent anyways: All solutions that are symmetric with respect to the assignment of activity blocks to template blocks correspond to a single solution in the template-based model, while all of them represent different solutions (paths in the state-expanded network G) in the original model.

5.1 Model reformulation based on activity block templates

To formulate the MASSP using activity block templates, we first create a state-expanded network that uses activity block templates instead of activity blocks. The network construction follows the description in Sect. 3, assuming that there is a single "template activity" (which can, depending on the rule set, appear multiple times in a single work block) that represents the activity block template. The lower / upper bound for the duration of the template activity is chosen as the minimum / maximum of the the lower / upper bounds of all activities in the problem instance under consideration, and the set of all possible activity block templates is referred to as $B^{\square}$. In the following exposition (and in the mathematical model discussed below), we use the symbol G to represent the template-based network, that is, we assume that it replaces the original activity-specific network. Note that compared to the original network, the number of activity block arcs in the template block network is reduced by a factor of $|A| - 1$, where $|A|$ is the number of activities in the problem instance under consideration.

The assignment of a concrete activity $a \in A$ to an activity block template $b \in B^{\square}$ is represented by the nonnegative integer decision variable $X^{\square}_{b,a}$. Observe that while there is a single variable $X^{\square}_{b,a}$ for each of $b \in B^{\square}$ and each activity for which b is a feasible block, every activity block template $b \in B^{\square}$ is associated with multiple arcs in G (these arcs start in nodes with different states, e.g. before and after a break); we denote the set of all arcs $e \in E$ associated with a block b as $E^{\square}_{b}$.

The new model then consists of the objective function (1), the flow balance constraints (2), the flow size constraint (3) to model a fixed number of employees, the variable domains (5) and (6), and the constraints (8)–(10) to be explained next.

$$\sum_{e \in E_b} X_e = \sum_{a \in A^b} X^{\square}_{b,a} \qquad \forall b \in B^{\square} \tag{8}$$

$$\sum_{b \in B^{\mathrm{cov}}_{a,p}} X^{\square}_{b,a} + Y^{\mathrm{u}}_{a,p} - Y^{\mathrm{o}}_{a,p} = d_{a,p} \qquad \forall a \in A,\, p \in P \tag{9}$$

$$X^{\square}_{b,a} \in \mathbb{Z}^{+}_{0} \qquad \forall a \in A,\, B \in B_a \tag{10}$$

The linking constraints (8) ensure that for each activity block template b, the total number of assigned concrete activity blocks from activities $a \in A^b$ for which block b is a valid activity block equals the total flow on the arcs $e \in E^{\square}_{b}$ in G that represents b. The constraints (9) reformulate the cover constraints (4) using the activity block assignment variables $X^{\square}_{b,a}$; the set $B^{\mathrm{cov}}_{a,p}$ is the set of blocks that are valid for activity a and demand period p. Finally, (10) establish the domains of the block assignment variables, using the set B_a representing the set of all feasible activity blocks for activity type a.

5.2 Dealing with work block composition rules by coupling state-expanded networks

The template-based state-expanded network discussed above does not consider concrete activities at all, and the mathematical model does not relate the (concrete) activity block variables $X^{\square}_{b,a}$ to each other. This means that the model above can be used for problems such as the Demassey problem, but not for problems such as the Dahmen problem in which a work block can be composed of multiple activity blocks and there are rules governing the feasibility of this composition.

In order to account for work block composition rules, we propose a more complex formulation in which the composition of activity blocks into feasible work blocks is ensured by a separate state-expanded network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that we will refer to as the *work block composition network* in the following. Like the original network, $\mathcal{G}$ contains a source and a sink node that are connected by a circulation arc from the sink to the source. All arcs $e \in \mathcal{E}$ other than the circulation arc and the arcs from the source and to the sink are associated with activity blocks assigned to concrete work activities $a \in A$ and each path in $\mathcal{G}$ corresponds to a feasible (activity-assigned) work block. Depending on the rules to be considered, the state variable s_v associated with each node $v \in \mathcal{N}^{\mathrm{state}}$ (that is, each node that is neither the source nor the sink) has (a subset of) the following attributes:

- s_v^{p} the period p.
- $s_v^{\#\mathrm{pWb}}$ The number of work periods in the current work block.
- s_v^{prevAct} The activity type of the previous activity, (*None* if it is the first activity in the block).
- s_v^{actWb} The set of activity types that have occurred in the work block so far.

The flow variables associated with the edges $e \in \mathcal{E}$ are denoted as $X_e^{\mathcal{G}}$, and, like for G, the network flow component associated with $\mathcal{G}$ needs respect the flow balance constraints:

$$\sum_{e \in v^{\mathrm{in}}} X_e^{\mathcal{G}} = \sum_{e \in v^{\mathrm{out}}} X_e^{\mathcal{G}} \qquad \forall v \in \mathcal{N} \tag{11}$$

To make sure that the work blocks corresponding to the flow in $\mathcal{G}$ (consisting of "concrete" activity blocks) match the work blocks corresponding to the flow in G (consisting of activity block templates), the flows in both networks need to be linked. In order to achieve this, it does not suffice to simply link blocks based on the start- and end period of the block, but one also need to account for the position of the activity blocks in the work blocks. This can be achieved by using the relative start period k of a block b within the work block as additional matching criterion. Note that this information is present in form of the state attribute $s_v^{\#\mathrm{pWb}}$ in the state nodes of both networks G and $\mathcal{G}$. Using the block b (characterized by start and end period) associated with an activity block arc and the relative position k obtained from the arc's source node, we can define the arc set $E_{b,k}^{\square}$ of all arcs in E representing a template activity block b (identified by start and end period) that starts in the relative period k of a work block. Analogously, we define the corresponding arc sets $\mathcal{E}_{b,k}^{a}$ in $\mathcal{E}$ that represent the activity blocks with start- and end period given by block b that are assigned to activity $a \in A$ and start at the relative period k in a work block.

Assuming that the set K_b represents all possible work-block relative start periods of an activity block template b, we are ready to formulate the linking constraint that connects the flows in G and $\mathcal{G}$:

$$\sum_{e \in E_{b,k}} X_e = \sum_{a \in A} \sum_{e \in \mathcal{E}_{b,k}^{a}} X_e^{\mathcal{G}} \qquad \forall b \in B_{\square}, k \in K_b \tag{12}$$

Then, using set $\mathcal{E}_{a,p}^{\mathrm{cov}}$ referring to the set of all arcs in $\mathcal{E}$ representing activity blocks that cover activity a in period p, we formulate the following covering constraints, followed by the domains of the flow variables associated with the arcs $e \in \mathcal{E}$.

$$\sum_{e \in \mathcal{E}_{a,p}^{\mathrm{cov}}} X_e^{\mathcal{G}} + Y_{a,p}^{\mathrm{u}} - Y_{a,p}^{\mathrm{o}} = d_{a,p} \qquad \forall a \in A, p \in P \tag{13}$$

$$X_e^{\mathcal{G}} \in \mathbb{Z}_0^{+} \qquad \forall e \in \mathcal{E} \tag{14}$$

In addition to the constraints (11) – (14) described in this subsection, the full model with two coupled state-expanded networks contains the objective function (1), the flow balance constraints (2) for G, the flow size constraint (3) to model a fixed number of employees, and the variable domains (5) and (6).

Finally, observe that for MASSP variants such as the flexible variant of the Dahmen problem that only need to consider the activity change rule and minimum and maximum duration rules, the implicit activity change constraints introduced in Subsection 4.2 ensuring the correct decomposability into work blocks can be applied for the work block composition network $\mathcal{G}$. This way, the state attribute s_v^{prevAct} can be dropped from the state definition of the work block composition network. The implicit activity change constraints read as follows when applied to $\mathcal{G}$:

$$\sum_{e \in v_{\neg a}^{\mathrm{in}}} X_e^{\mathcal{G}} \geq \sum_{e \in v_a^{\mathrm{out}}} X_e^{\mathcal{G}} \qquad \forall v \in \mathcal{N}^{\mathrm{inWb}}, a \in A \tag{15}$$

6 Computational results

In this section, we report the results from our experiments with two sets of instances, one for the Demassey problem, and one for the Dahmen problem. We start in Subsection 6.1 with experiments comparing the different model variants proposed in the last section both with respect to the size of the model instances and with respect to solution times. In Subsection 6.2, we compare the results from the best model variants to those obtained with state-of-the-art exact approaches from the literature.

All models were implemented in Python, and solved with Gurobi 9.1 with standard settings, except that the barrier method was used to solve the root relaxation. The computer used for the experiments was a Notebook with an Intel Core i7 10750 H processor clocked at 2.66 GHz with 6 cores and 32 GB RAM.

6.1 Experiments with different model variants

In this section, we compare the model variants proposed in the Sects. 4 and 5 for two sets of instances, one for the Demassey problem, and one for the Dahmen problem.

6.1.1 Demassey problem

The first set of experiments is conducted with the Demassey problem described in Sect. 2.1. The instances were first introduced in Demassey et al. (2005) and later used in several other papers such as Côté et al. (2011b), Côté et al. (2013) and Dahmen et al. (2018). The instance set comprises 100 instances consisting of 10 groups. Each of the groups is characterized by a given number of activities from 1 to 10, and the instances within each group vary with respect to the demand profile and with respect to the number of employees.

As described in 2.1, in the Demassey problem, each block of work needs to be assigned to a single activity, or, in other words, a change between different activities is only legal if there is a break in between. From a modeling perspective, this means that we do not need to deal with the composition of work blocks from multiple activity blocks, making it unnecessary to deal with work block composition rules such as the activity change requirement. As a consequence, we only compare two modeling approaches for the Demassey instances:

- ActivitySEN: The plain model from Sect. 4 in which all rules are encoded in a single state-expanded network that is based on concrete activity blocks.
- TemplateSEN: The model from Sect. 5.1 using a template block-based state-expanded network. Note that for the Demassey problem, we do not need a second state-expanded network for composing work blocks.

Table 1 presents the results from experiments with these two model variants. For both approaches, the table reports results for each of the 10 instance groups. Note that within one instance group, the problem structure is identical which means that for each instance group and for each model variant, the number of variables (Vars) and constraints (Cons) is identical. The other columns reported for each model variant are the number of instances solved to optimality with a time limit of 30 min and the average solution time in seconds; instances not solved to optimality are counted with 1800 s.

The results show that for the group with a single activity (yielding a mono-activity shift scheduling problem), the "plain" activity-based model yields smaller model instances and a shorter solution time. This is due to the fact that the template-based model introduces a "template flow" that is then only mapped to a single activity and thus, in that case, using the template-based model does not make much sense.

With an increasing number of activities, the template-based models can play out their strength: While the number of variables rapidly increases for the plain activity-based model, the size of the template-based model instances only grows moderately. Specifically, for the 10-activity instances, the activity-based model exhibits about eight times as many variables as the template-based model. Clearly, the model size impacts the performance: For the biggest instances, the activity-based model instances were not always solved to optimality within 30 min; in total, only 90 of the 100 instances are solved to optimality and the average solution time is 365 s. This contrasts with the template-based model: Using this model, all instances are optimally solved within half an hour, and the average solution time time is only 60 s.

6.1.2 Dahmen problem

The second set of experiments deals with the Dahmen problem described in Sect. 2.2. For this problem, Dahmen et al. (2018) presents experiments with 540 instances that vary with respect to the following features:

- Time granularity / length of a single time period (15, 30, or 60 min).
- Number of activities (3, 6 or 9).
- Degree of block flexibility (1, 2 or 3).
- Number of shift types (1 or 2).
- The set of possible shift start periods (shifts can start every ith period with $i \in \{1, 2, 3\}$).
- Demand profile index (1–5).

By grouping the 540 instances according to the first five features, one obtains 108 instance groups. Within each of these groups, all five instances have the same problem structure and only vary with respect to the demand profile.

In contrast to the Demassey problem, in the Dahmen problem, activity changes can happen within a work block. Furthermore, Dahmen et al. (2018) consider two problem variants: A basic ("flexible") variant that imposes no restriction on the number of different activities assigned in a single work block, and a "restricted" variant in which at most two different activities are allowed per work block. In this section, we first deal with deal with the flexible variant, followed by results for the restricted variant.

In our computational experiments with the flexible variant of the Dahmen problem, we use the following model variants:

- ActivitySEN: The plain model from Sect. 4 in which all rules are encoded in a single state-expanded network that is based on concrete activity blocks.
- ActivitySEN+ChangeCons: A model with a single activity block-based network that does not encode the activity change requirement and instead uses the activity change constraints introduced in Sect. 4.2.
- TemplateSEN: The model from Subsection 5.2 in which a template-block-based state-expanded network is linked to a second state-expanded network for work block composition that encodes the activity change requirement.

Table 1 Results for the Demassey instances

	ActivitySEN				TemplateSEN			
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time
1	58,979	12,413	10	24	60,909	14,344	10	42
2	133,458	15,180	10	22	62,937	14,533	10	19
3	194,407	15,276	10	127	64,965	14,722	10	74
4	255,356	15,372	10	146	66,993	14,911	10	53
5	316,305	15,468	10	194	69,021	15,100	10	64
6	377,254	15,564	9	364	71,049	15,289	10	47
7	438,203	15,660	9	548	73,077	15,478	10	57
8	499,152	15,756	9	476	75,105	15,667	10	62
9	560,101	15,852	8	631	77,133	15,856	10	65
10	621,050	15,948	5	1,120	79,161	16,045	10	116
Total	345,426	15,249	90	365	70,035	15,194	100	60

- TemplateSEN+ChangeCons: The model from Subsection 5.2 in with a template-block-based state-expanded network linked to state-expanded network for work block composition; the activity change requirement is enforced by activity change constraints 15.

The result from using these model variants on the 15-minute instances from the Dahmen instance set are displayed in Table 2. Results for the other instances (all of which are optimality, most in less than 10 s) can be found in Tables 6 and 7 in the appendix.

Regarding the model sizes, it turns out that the last model variant TemplateSEN+ChangeCons yields much smaller instances than the "plain" model: For the largest model instances, the plain model has almost 20 times as many variables and 5 times as many constraints. The difference in the model sizes is reflected in the solution time and the solution quality: With the plain model variant, only 150 of the 180 instances are solved to optimality within the time limit of 30 min. With the last model variant, in contrast, all instances are solved to optimality in slightly more than one minute on average; the biggest instances being solved in 5 min on average. Interestingly, when comparing the two variants in the middle, the variant with a single network and activity change constraints yield bigger model instances than the model variant with template blocks without the constraints but at the same time allows to solve more instances to optimality.

Let us now turn to the restricted variant of the Dahmen problem in which at most two different activities are allowed to be assigned per work block. Table 3 compares the results from using the best model (TemplateSEN) for this variant with those from the best model for the flexible variant (TemplateSEN+ChangeCons). Note that we do not use the model with the activity change constraints here since the state information needed for modeling the "at-most-two activities" rule already contains all the information needed to model the activity change requirement. See Sect. 3.2 for a description of this state model and 5.2 for a description of the mathematical model involving two coupled state-expanded networks. Since the model instances for the restricted model are bigger (on average, about twice as big in terms of variables and constraints) and thus harder to solve, we increased the solution time limit to one hour for these models. The results show that while the model instances are bigger and the solution times are higher, still all instances could be solved to optimality in less than 30 min on average for each instance group—only for five instances, more than 30 min were needed.

(Dahmen et al., 2018) raise the interesting question if the two-activity restriction has a negative impact on solution quality. Dahmen et al. (2018) suspected that there was no negative impact, but they were not able to solve all instances to optimality and thus could not give a definite answer for all instances in the problem set. In the experiments presented here, however, all instances were solved to optimality for both variants. The rightmost column (ObjDiff) in Tables 3, 6 and 7 reports the average difference in the objective between the restrictive and the flexible variant; and it turns out that indeed, for all instances, the optimal objective function value from the restricted variant is the same as the optimal objective from the flexible variant.

6.2 Comparison to other exact approaches from the literature

In this section, we compare the results from our experiments to the results reported in the literature for existing state-of-the-art exact approaches for the MASSP. To the best of our knowledge, while exact approaches for the Demassey problem were considered in various publications, for example in Demassey et al. (2005), Demassey et al. (2006),Côté et al. (2011b), Côté et al. (2013) and Dahmen et al. (2018), the only

Table 2 Results for the Dahmen instances with time granularity of 15 min (flexible variant of the Dahmen problem)

	ActivitySEN				ActivitySEN+ChangeCons				TemplateSEN				TemplateSEN+ChangeCons			
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time
3.1.1.3	18,081	7,586	5	3	21,217	9,378	5	4	27,306	12,454	5	7	28,280	14,832	5	4
3.1.1.2	27,121	11,234	5	3	31,825	13,922	5	4	36,802	14,665	5	5	37,784	17,063	5	4
3.1.1.1	54,241	22,178	5	7	63,649	27,554	5	8	65,290	21,298	5	8	66,296	23,756	5	6
3.1.2.3	31,905	13,730	5	9	39,009	17,730	5	9	43,306	16,934	5	8	44,280	19,312	5	10
3.1.2.2	47,857	20,450	5	7	58,513	26,450	5	8	60,802	21,385	5	11	61,784	23,783	5	9
3.1.2.1	95,713	40,610	5	18	117,025	52,610	5	27	113,290	34,738	5	43	114,296	37,196	5	38
3.2.1.3	52,897	11,234	5	9	43,105	14,626	5	8	42,208	15,698	5	10	34,782	16,110	5	10
3.2.1.2	79,345	16,706	5	11	64,657	21,794	5	8	51,831	17,936	5	13	44,338	18,349	5	8
3.2.1.1	158,689	33,122	5	21	129,313	43,298	5	24	80,700	24,650	5	20	73,006	25,066	5	20
3.2.2.3	94,913	22,178	5	18	73,633	25,410	5	22	58,208	20,178	5	12	50,782	20,590	5	19
3.2.2.2	142,369	33,122	5	34	110,449	37,970	5	33	75,831	24,656	5	60	68,338	25,069	5	63
3.2.2.1	284,737	65,954	5	86	220,897	75,650	5	132	128,700	38,090	5	19	121,006	38,506	5	25
6.1.1.3	76,705	17,026	5	22	47,713	17,282	5	13	52,160	18,208	5	19	36,446	17,216	5	10
6.1.1.2	115,057	25,250	4	535	71,569	25,634	5	34	61,869	20,465	5	92	46,016	19,462	5	98
6.1.1.1	230,113	49,922	5	200	143,137	50,690	5	62	90,996	27,236	5	93	74,726	26,200	5	45
6.1.2.3	134,113	32,226	5	48	81,121	29,506	5	33	68,160	22,688	5	50	52,446	21,696	5	40
6.1.2.2	201,169	48,050	5	224	121,681	43,970	5	51	85,869	27,185	5	27	70,016	26,182	5	44
6.1.2.1	402,337	95,522	5	326	243,361	87,362	5	118	138,996	40,676	5	87	122,726	39,640	5	74
6.2.1.3	234,145	22,146	5	75	84,321	23,682	5	30	116,908	20,168	5	91	49,026	19,192	5	28
6.2.1.2	351,217	32,930	5	142	126,481	35,234	5	43	127,171	22,441	5	142	58,700	21,454	5	34
6.2.1.1	702,433	65,282	4	706	252,961	69,890	5	111	157,960	29,260	5	418	87,722	28,240	5	57
6.2.2.3	410,401	43,618	2	1,195	143,201	40,770	5	152	132,908	24,648	5	448	65,026	23,672	5	131
6.2.2.2	615,601	65,138	3	1,286	214,801	60,866	5	466	151,171	29,161	5	494	82,700	28,174	5	115

Table 2 continued

	ActivitySEN				ActivitySEN+ChangeCons				TemplateSEN				TemplateSEN+ChangeCons			
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time
6.2.2.1	1,231,201	129,698	1	1,567	429,601	121,154	5	451	205,960	42,700	5	215	135,722	41,680	5	82
9.1.1.3	174,369	25,378	5	191	70,625	23,138	5	48	92,458	21,698	5	78	44,400	19,310	5	31
9.1.1.2	261,553	37,634	4	102	105,937	34,274	5	32	102,513	23,982	5	58	54,036	21,571	5	17
9.1.1.1	523,105	74,402	4	597	211,873	67,682	5	71	132,678	30,834	5	225	82,944	28,354	5	48
9.1.2.3	301,569	47,970	5	526	119,649	39,234	5	59	108,458	26,178	5	380	60,400	23,790	5	133
9.1.2.2	452,353	71,522	5	974	179,473	58,418	5	285	126,513	30,702	5	189	78,036	28,291	5	65
9.1.2.1	904,705	142,178	3	1,084	358,945	115,970	5	338	180,678	44,274	5	494	130,944	41,794	5	210
9.2.1.3	543,777	33,058	5	448	125,537	32,738	5	45	243,076	24,638	5	321	63,270	22,274	5	21
9.2.1.2	815,665	49,154	5	993	188,305	48,674	5	110	254,421	26,946	5	373	73,062	24,559	5	54
9.2.1.1	1,631,329	97,442	2	1,361	376,609	96,482	5	212	288,456	33,870	5	678	102,438	31,414	5	52
9.2.2.3	946,497	65,058	1	1,696	212,769	56,130	5	447	259,076	29,118	0	1800	79,270	26,754	5	304
9.2.2.2	1,419,745	97,154	0	1,817	319,153	83,762	3	1,543	278,421	33,666	4	1,357	97,062	31,279	5	230
9.2.2.1	2,839,489	193,442	2	1,645	638,305	166,658	5	507	336,456	47,310	2	1,272	150,438	44,854	5	307
Total	461,292	51,647	150	500	162,234	49,878	178	154	127,156	26,687	171	267	73,404	26,019	180	68

Table 3 Results for the flexible and the restricted variant of the Dahmen problem: instances with time granularity of 15 min

	Flexible				Restricted				
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	ObjDiff
3.1.1.3	28,280	14,832	5	4	27,766	12,800	5	6	0
3.1.1.2	37,784	17,063	5	4	37,266	15,014	5	5	0
3.1.1.1	66,296	23,756	5	6	65,766	21,656	5	7	0
3.1.2.3	44,280	19,312	5	10	43,766	17,280	5	27	0
3.1.2.2	61,784	23,783	5	9	61,266	21,734	5	14	0
3.1.2.1	114,296	37,196	5	38	113,766	35,096	5	43	0
3.2.1.3	34,782	16,110	5	10	47,788	18,740	5	10	0
3.2.1.2	44,338	18,349	5	8	57,459	21,004	5	9	0
3.2.1.1	73,006	25,066	5	20	86,472	27,796	5	22	0
3.2.2.3	50,782	20,590	5	19	63,788	23,220	5	19	0
3.2.2.2	68,338	25,069	5	63	81,459	27,724	5	61	0
3.2.2.1	121,006	38,506	5	25	134,472	41,236	5	40	0
6.1.1.3	36,446	17,216	5	10	58,876	23,308	5	20	0
6.1.1.2	46,016	19,462	5	98	68,643	25,609	5	79	0
6.1.1.1	74,726	26,200	5	45	97,944	32,512	5	50	0
6.1.2.3	52,446	21,696	5	40	74,876	27,788	5	100	0
6.1.2.2	70,016	26,182	5	44	92,643	32,329	5	59	0
6.1.2.1	122,726	39,640	5	74	145,944	45,952	5	134	0
6.2.1.3	49,026	19,192	5	28	151,128	39,616	5	286	0
6.2.1.2	58,700	21,454	5	34	161,685	42,055	5	95	0
6.2.1.1	87,722	28,240	5	57	193,356	49,372	5	408	0
6.2.2.3	65,026	23,672	5	131	167,128	44,096	5	511	0
6.2.2.2	82,700	28,174	5	115	185,685	48,775	5	671	0
6.2.2.1	135,722	41,680	5	82	241,356	62,812	5	598	0
9.1.1.3	44,400	19,310	5	31	112,306	37,040	5	61	0
9.1.1.2	54,036	21,571	5	17	122,532	39,456	5	59	0
9.1.1.1	82,944	28,354	5	48	153,210	46,704	5	277	0
9.1.2.3	60,400	23,790	5	133	128,306	41,520	5	252	0
9.1.2.2	78,036	28,291	5	65	146,532	46,176	5	278	0
9.1.2.1	130,944	41,794	5	210	201,210	60,144	5	567	0
9.2.1.3	63,270	22,274	5	21	328,996	73,856	5	518	0
9.2.1.2	73,062	24,559	5	54	341,079	76,584	5	606	0
9.2.1.1	102,438	31,414	5	52	377,328	84,768	5	856	0
9.2.2.3	79,270	26,754	5	304	344,996	78,336	5	1581	0
9.2.2.2	97,062	31,279	5	230	365,079	83,304	5	1288	0
9.2.2.1	150,438	44,854	5	307	425,328	98,208	5	1051	0
Total	73,404	26,019	180	68	152,978	42,323	180	296	0

publication dealing with the Dahmen problem is the original publication.

As pointed out in Dahmen et al. (2018), probably the best existing exact approach for the Demassey problem is the implicit grammar model proposed in Côté et al. (2011b). In Table 4, we compare the results from our best model variant to those reported in Dahmen et al. (2018) for the implicit grammar model which are, to the best the of our knowledge, the most recent results reported for the grammar model. Like in the original paper (Côté et al., 2011b; Dahmen et al., 2018) solved the grammar model only until a MIP gap of 1% was reached. To allow a better comparison, we also ran a series of experiments with the same MIP gap the results of which are reported in Table 4. When comparing the number of variables and the number of constraints, our model is smaller than the grammar model; this difference increases with the number of activities per instance. For the 10-activity instances, the grammar model instances exhibit about 25% more variables

Table 4 Comparing our approach to the state-of-the-art for the Demassey instances

	SEN+Template				Grammar			
Group	Vars	Cons	1% Opt	Time	Vars	Cons	1%-Opt	Time
1	60,909	14,344	10	38	68,200	16,277	10	7
2	62,937	14,533	10	19	71,465	18,131	10	22
3	64,965	14,722	10	74	74,734	19,987	10	36
4	66,993	14,911	10	53	78,013	21,845	10	26
5	69,021	15,100	10	65	81,300	23,707	10	21
6	71,049	15,289	10	44	84,591	25,569	10	74
7	73,077	15,478	10	54	87,898	27,437	10	77
8	75,105	15,667	10	56	91,148	29,286	10	27
9	77,133	15,856	10	63	94,387	31,132	10	65
10	79,161	16,045	10	98	97,712	33,006	10	152
Total	70,035	15,194	100	56	82,945	24,638	100	51

and 50% more constraints than our model instances. Both approaches solved all instances to 1%-optimality within less than a minute on average.

Observe, however, that the results were obtained on different hardware, and with different MILP solvers. Specifically, regarding hardware, our computer is a notebook with a newer processor clocked at 2.66 GHz with 6 cores, 12 threads and 32 GB RAM, while the experiments from Dahmen et al. (2018) with the grammar models were carried out on a server with a Dual Intel Xeon X5650 processor clocked at 2.66 GHz with in total 12 cores and 24 threads and 72 GB of RAM. With respect to the solvers, it can be expected that the solver used in our experiments (Gurobi 9.1) is faster than CPLEX 12.6 used in Dahmen et al. (2018). Taking into account all these aspects, at this point, we cannot tell which of the models performs better with respect to solution time. However, it appears unlikely that if evaluated with the same hardware and the same solver, one approach would completely outperform the other.

Table 5 deals with the Dahmen instances. For these instances, Dahmen et al. (2018) show that their implicit formulation yields much better results than the grammar formulation. Table 5 compares the results obtained with this implicit formulation to those obtained with our best model (TemplateSEN+ChangeCons) for the instances with a time granularity of 15 min intervals for the flexible variant of the Dahmen problem.

It turns out that the implicit model has much less constraints than our model for all instances. This is due to the fact that in the implicit model, all pre- and post-break work blocks are explicitly enumerated while in our model, the work blocks are composed by a flow in a state-expanded network in which each node corresponds to a constraint. When it comes to the number of variables, for the smallest instances with three activities, the instances from the implicit formulation also exhibit a much smaller number of variables. This is again related to the enumeration of the pre- and post-break work blocks in the implicit model: For the instances with only three activities and little flexibility, the number of enumerated work blocks is relatively small. The opposite is the case for the instances with nine activities and a lot of flexibility where the number of work blocks to enumerate is very large. For these instances, our model exhibits much less variables; for the biggest instance, the implicit model exhibits more than 20 times as many variables as our model.

With regard to the solution performance of the models, the statements from above regarding hardware and solver software also hold here. However, the differences in performance are much bigger than those reported for the Demassey problem, in particular when it comes to the large instances: With our model, all instances are solved to optimality, and even the largest instances are optimally solved in 5 min or less on average. The implicit model from (Dahmen et al., 2018), however, seems to struggle with the large instances: Even within 3 h of computation time, 45 of the instances are not solved to optimality. These results, in combination with the huge difference with respect to the number of variables, indicate that our model performs substantially better than the implicit model for large instances, despite the fact that one needs to be careful with such statements given differences in hardware and software mentioned above.

Regarding the “restricted” variant, we do not include an explicit comparison here since Dahmen et al. (2018) do not report detailed figures with respect to model size and instances solved to optimality per instance group. Nonetheless, it is interesting to note that while our model instances for the restricted variant are larger than those for the basic flexible variant (see above), the opposite is the case for the implicit model from (Dahmen et al., 2018): Since it relies on enumerating feasible work blocks, the instances get smaller for restricted variant (29 % on average as reported in the paper). This makes the restricted models easier to solve: instead of

Table 5 Comparing our approach to the state-of-the-art for the Dahmen instances with time granularity of 15 min

	TemplateSEN+ChangeCons				Implicit formulation			
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time
3.1.1.3	28,280	14,832	5	4	6041	657	5	2
3.1.1.2	37,784	17,063	5	4	6995	802	5	1
3.1.1.1	66,296	23,756	5	6	9678	1235	5	5
3.1.2.3	44,280	19,312	5	10	12,946	1426	5	25
3.1.2.2	61,784	23,783	5	9	15,099	1908	5	15
3.1.2.1	114,296	37,196	5	38	21,190	3350	5	55
3.2.1.3	34,782	16,110	5	10	29,598	657	5	27
3.2.1.2	44,338	18,349	5	8	33,829	802	5	13
3.2.1.1	73,006	25,066	5	20	45,522	1235	5	249
3.2.2.3	50,782	20,590	5	19	60,450	1426	5	199
3.2.2.2	68,338	25,069	5	63	69,071	1908	5	377
3.2.2.1	121,006	38,506	5	25	93,284	3350	5	523
6.1.1.3	36,446	17,216	5	10	38,164	894	5	33
6.1.1.2	46,016	19,462	5	98	43,515	1039	5	113
6.1.1.1	74,726	26,200	5	45	58,440	1472	5	188
6.1.2.3	52,446	21,696	5	40	77,466	1663	5	901
6.1.2.2	70,016	26,182	5	44	88,312	2145	5	2094
6.1.2.1	122,726	39,640	5	74	118,942	3587	5	1661
6.2.1.3	49,026	19,192	5	28	282,116	899	3	4538
6.2.1.2	58,700	21,454	5	34	319,779	1046	3	4539
6.2.1.1	87,722	28,240	5	57	425,918	1479	4	2615
6.2.2.3	65,026	23,672	5	131	562,752	1670	1	10,779
6.2.2.2	82,700	28,174	5	115	637,588	2152	1	9361
6.2.2.1	135,722	41,680	5	82	850,568	3594	3	5790
9.1.1.3	44,400	19,310	5	31	129,511	1090	4	2597
9.1.1.2	54,036	21,571	5	17	147,029	1235	5	523
9.1.1.1	82,944	28,354	5	48	196,242	1668	5	870
9.1.2.3	60,400	23,790	5	133	259,450	1859	3	6213
9.1.2.2	78,036	28,291	5	65	294,375	2341	4	4029
9.1.2.1	130,944	41,794	5	210	393,542	3783	5	4363
9.2.1.3	63,270	22,274	5	21	1,048,918	1113	3	6705
9.2.1.2	73,062	24,559	5	54	1,186,487	1258	1	9941
9.2.1.1	102,438	31,414	5	52	1,577,274	1691	0	10,800
9.2.2.3	79,270	26,754	5	304	2,082,958	1882	0	10,800
9.2.2.2	97,062	31,279	5	230	2,356,341	2364	0	10,800
9.2.2.1	150,438	44,854	5	307	3,138,116	3806	0	10,800
Total	73,404	26,019	180	68	464,375	1791	135	3404

45 instances not being solved to optimality within 3 h for the flexible problem, only 28 instances cannot be solved to optimality within that time for the restricted variant. Recall, however, that our model instances for the restricted problem can be solved to optimality within 296 s on average for the 15-minute instances; for the largest instance groups, the average solution time to optimality is around 20 min.

7 Conclusions

This paper presents a new MILP modeling approach for multi-activity shift scheduling problems based on state-expanded networks. In particular, it presents two techniques for dealing with the explosion of the size of the networks for large-scale instances: A set of implicit constraints ensur-

Table 6 Results for both variants of the Dahmen problem: instances with time granularity of 30 min

	Flexible				Restricted				
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	ObjDiff
3.1.1.3	4427	3126	5	0.1	4754	2948	5	0.2	0
3.1.1.2	5647	3541	5	0.2	5980	3360	5	0.3	0
3.1.1.1	9271	4747	5	0.3	9616	4560	5	0.6	0
3.1.2.3	6411	3846	5	0.7	6738	3668	5	0.9	0
3.1.2.2	8623	4621	5	0.4	8956	4440	5	0.6	0
3.1.2.1	15,223	6907	5	1.3	15,568	6720	5	3.2	0
3.2.1.3	5571	3466	5	0.6	8665	4214	5	0.6	0
3.2.1.2	6809	3885	5	0.4	9960	4648	5	0.7	0
3.2.1.1	10,469	5099	5	0.7	13,734	5892	5	1.1	0
3.2.2.3	7555	4186	5	1.0	10,649	4934	5	1.2	0
3.2.2.2	9785	4965	5	0.8	12,936	5728	5	1.1	0
3.2.2.1	16,421	7259	5	2.8	19,686	8052	5	3.5	0
6.1.1.3	6155	3894	5	0.7	12,119	5864	5	1.2	0
6.1.1.2	7403	4318	5	0.6	13,476	6325	5	0.6	0
6.1.1.1	11,083	5542	5	1.1	17,374	7623	5	1.9	0
6.1.2.3	8139	4614	5	2.3	14,103	6584	5	3.1	0
6.1.2.2	10,379	5398	5	2.0	16,452	7405	5	2.3	0
6.1.2.1	17,035	7702	5	6.0	23,326	9783	5	9.6	0
6.2.1.3	8303	4384	5	1.5	29,739	10,094	5	5.6	0
6.2.1.2	9587	4816	5	0.9	31,408	10,629	5	3.1	0
6.2.1.1	13,339	6056	5	1.9	35,930	12,075	5	7.2	0
6.2.2.3	10,287	5104	5	2.1	31,723	10,814	5	7.0	0
6.2.2.2	12,563	5896	5	2.2	34,384	11,709	5	5.6	0
6.2.2.1	19,291	8216	5	7.4	41,882	14,235	5	29.0	0
9.1.1.3	7813	4567	5	0.5	24,526	10,120	5	1.6	0
9.1.1.2	9089	5000	5	0.8	26,104	10,654	5	2.4	0
9.1.1.1	12,825	6242	5	2.1	30,444	12,098	5	5.1	0
9.1.2.3	9797	5287	5	1.5	26,510	10,840	5	3.7	0
9.1.2.2	12,065	6080	5	2.4	29,080	11,734	5	4.5	0
9.1.2.1	18,777	8402	5	7.8	36,396	14,258	5	20.7	0
9.2.1.3	11,035	5302	5	1.6	65,653	19,708	5	11.8	0
9.2.1.2	12,365	5747	5	1.3	67,960	20,410	5	9.4	0
9.2.1.1	16,209	7013	5	2.4	73,758	22,190	5	18.7	0
9.2.2.3	13,019	6022	5	3.4	67,637	20,428	5	14.2	0
9.2.2.2	15,341	6827	5	3.6	70,936	21,490	5	15.6	0
9.2.2.1	22,161	9173	5	10.3	79,710	24,350	5	87.9	0
Total	11,118.7	5,479.2	180	2.1	28,552	10,294	180	7.9	0

ing that an aggregated flow can be decomposed into shifts respecting the activity change requirement, and the concept of template blocks that allows modeling different aspects of composing shifts in different and coupled state-expanded network. When combined, the two techniques allow reducing the size of the main state-expanded network by a factor in the order of the square of the number of activities in the instance under consideration.

We show how this approach can be used to model different MASSP problems. In particular, we provide experimental results for two big sets of MASSP instances with 100 and 540 instances, respectively.

Our experiments show that our approach is at least competitive with the best approach on the first set of instances, and is clearly better than the previously best approach for large instances from the second set. This is shown by the fact that we are able to solve all instances to optimality, including

Table 7 Results for both variants of the Dahmen problem: instances with time granularity of 60 min

	Flexible				Restricted				
Group	Vars	Cons	Opt	Time	Vars	Cons	Opt	Time	ObjDiff
3.1.1.3	945	805	5	0.1	1155	830	5	0.1	0
3.1.1.2	,165	906	5	0.1	1383	932	5	0.1	0
3.1.1.1	1785	1,167	5	0.1	2011	1194	5	0.1	0
3.1.2.3	1265	941	5	0.1	1475	966	5	0.1	0
3.1.2.2	1645	1110	5	0.1	1863	1136	5	0.1	0
3.1.2.1	2745	1575	5	0.1	2971	1602	5	0.1	0
3.2.1.3	1185	909	5	0.1	2043	1131	5	0.1	0
3.2.1.2	1412	1012	5	0.1	2304	1244	5	0.1	0
3.2.1.1	2039	1275	5	0.1	2965	1517	5	0.1	0
3.2.2.3	1505	1045	5	0.1	2363	1267	5	0.1	0
3.2.2.2	1892	1216	5	0.1	2784	1448	5	0.1	0
3.2.2.1	2999	1683	5	0.1	3925	1925	5	0.1	0
6.1.1.3	1413	1090	5	0.1	3307	1783	5	0.1	0
6.1.1.2	1648	1197	5	0.1	3616	1918	5	0.1	0
6.1.1.1	2283	1464	5	0.1	4325	2213	5	0.1	0
6.1.2.3	1733	1226	5	0.2	3627	1919	5	0.3	0
6.1.2.2	2128	1401	5	0.2	4096	2122	5	0.2	0
6.1.2.1	3243	1872	5	0.1	5285	2621	5	0.3	0
6.2.1.3	1827	1214	5	0.1	7271	2923	5	0.2	0
6.2.1.2	2076	1325	5	0.1	7728	3100	5	0.3	0
6.2.1.1	2725	1596	5	0.1	8585	3437	5	0.3	0
6.2.2.3	2147	1350	5	0.2	7591	3059	5	0.5	0
6.2.2.2	2556	1529	5	0.2	8208	3304	5	0.5	0
6.2.2.1	3685	2004	5	0.1	9545	3845	5	0.4	0
9.1.1.3	1848	1333	5	0.1	6887	3238	5	0.2	0
9.1.1.2	2098	1446	5	0.1	7331	3425	5	0.2	0
9.1.1.1	2748	1719	5	0.1	8175	3772	5	0.3	0
9.1.2.3	2168	1469	5	0.2	7207	3374	5	0.4	0
9.1.2.2	2578	1650	5	0.2	7811	3629	5	0.3	0
9.1.2.1	3708	2127	5	0.2	9135	4180	5	0.3	0
9.2.1.3	2469	1519	5	0.1	16,115	5839	5	0.6	0
9.2.1.2	2740	1638	5	0.1	16,904	6122	5	0.5	0
9.2.1.1	3411	1917	5	0.1	18,093	6565	5	1.0	0
9.2.2.3	2789	1655	5	0.1	16,435	5975	5	0.9	0
9.2.2.2	3220	1842	5	0.2	17,384	6326	5	1.4	0
9.2.2.1	4371	2325	5	0.2	19,053	6973	5	1.0	0
Total	2283	1432	180	0.1	6971	2968	180	0.3	0

70 previously unsolved instances from the second set within less than 45 min of computation time, most of the instances being solved much faster.

Preprint and relation to prior work

A preprint version of this paper can be found at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3798667. The previous title of the preprint was "State-Expanded Network Formulations for Multi-Activity Shift Scheduling". In its latest revision, the title was changed to "Block-based state-expanded network formulations for multi-activity shift scheduling" in order to emphasise the important modeling decision of basing the state-expanded network on blocks of consecutive assignments and to highlight the difference to the state-expanded network model used in (Porrmann & Römer, 2021).

The article (Porrmann & Römer, 2021) was published before submitting this work to the Journal of Scheduling. The key differences between the contributions of (Porrmann & Römer, 2021) and those presented in this manuscript are discussed in the main text and can be summarised as follows:

- Porrmann and Römer (2021) use a different state model which implies a different structure of the state-expanded network underlying the MILP formulation. In (Porrmann & Römer, 2021), a single activity block, that is, a sequence of consecutive assignments of the same activity, can be composed of multiple arcs. This requires that the state model contains a state attribute that allows computing the duration of an activity block. In the model in the present paper, every arc that represents an assignment corresponds to a full activity or break block, making it unnecessary to track any activity block-related aspects in the state model.
- This block-based network structure allows us to develop the exact techniques (the implicit modeling of the activity change rule and the two-level model combining a model layer based on template blocks with an activity assignment layer) that form key contributions of this paper. While, as can be seen in the computational results, the block-based model itself still struggles with the largest instances, applying both techniques eventually permits solving all Demassey and Dahmen instances to optimality, many of them for the first time. Since using the model employed in (Porrmann & Römer, 2021) did not permit to efficiently solve all Demassey instances, the key contribution of that paper was to introduce a Machine Learning-based approach to heuristically remove nodes and arcs from the state-expanded network.
- (Porrmann & Römer, 2021) only considers the Demassey MASSP variant; the present paper considers both the Demassey and the Dahmen variants and the respective instances.

Acknowledgements This research was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft), Grant Number 398911053 and by the Return Programme of the Federal State of North Rhine Westphalia (NRW Rückkehrprogramm).
In addition, the author would like to thank Sana Dahmen for providing the instances and helpful explanations regarding the Dahmen variant of the multi-activity shift scheduling problem.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The author obtained the problem instances from the authors of other papers dealing with those instances. Upon request, the author will provide access to both the instances, the solution files as well as to code for checking and verifying solutions.

Declarations

Conflict of interest The author has no competing interests to declare that are relevant to the content of this article.

A Additional result data

Table 3 discussed in Sect. 6 presented results for the hardest instances from the Dahmen instance set, namely the instances with a time granularity of 15 min. The following tables present the results for the remaining instances with a time granularity of 30 and 60 min.

References

Bechtold, S. E., & Jacobs, L. W. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science, 36*(11), 1339–1351. https://doi.org/10.1287/mnsc.36.11.1339

Côté, M. C., Gendron, B., Quimper, C. G., et al. (2011). Formal languages for integer programming modeling of shift scheduling problems. *Constraints, 16*(1), 54–76. https://doi.org/10.1007/s10601-009-9083-2

Côté, M. C., Gendron, B., & Rousseau, L. M. (2011). Grammar-based integer programming models for multiactivity shift scheduling. *Management Science, 57*(1), 151–163. https://doi.org/10.1287/mnsc.1100.1264

Côté, M. C., Gendron, B., & Rousseau, L. M. (2013). Grammar-based column generation for personalized multi-activity shift scheduling. *INFORMS Journal on Computing, 25*(3), 461–474. https://doi.org/10.1287/ijoc.1120.0514

Dahmen, S., Rekik, M., & Soumis, F. (2018). An implicit model for multi-activity shift scheduling problems. *Journal of Scheduling, 21*(3), 285–304. https://doi.org/10.1007/s10951-017-0544-y

Dantzig, G. B. (1954). Letter to the editor–a comment on Edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America, 2*(3), 339–341. https://doi.org/10.1287/opre.2.3.339

Demassey, S., Pesant, G., & Rousseau, L.M. (2005). Constraint programming based column generation for employee timetabling. In: Barták, R., & Milano, M. (eds) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, pp 140–154, https://doi.org/10.1007/11493853_12

Demassey, S., Pesant, G., & Rousseau, L. M. (2006). A cost-regular based hybrid column generation approach. *Constraints, 11*(4), 315–333. https://doi.org/10.1007/s10601-006-9003-7

Hernández-Leandro, N. A., Boyer, V., Salazar-Aguilar, M. A., et al. (2019). A matheuristic based on Lagrangian relaxation for the multi-activity shift scheduling problem. *European Journal of Operational Research, 272*(3), 859–867. https://doi.org/10.1016/j.ejor.2018.07.010

Mellouli, T. (2001). A network flow approach to crew scheduling based on an analogy to a train/aircraft maintenance routing problem. In: Voss, S., & Daduna, J. (eds) Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, vol 505. Springer, Berlin, pp 91–120, https://doi.org/10.1007/978-3-642-56423-9_6

Porrmann, T., & Römer, M. (2021). Learning to reduce state-expanded networks for multi-activity shift scheduling. In: Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Lecture Notes in Computer Science, vol 12735. Springer-Verlag, Berlin, Heidelberg, pp 383–391, https://doi.org/10.1007/978-3-030-78230-6_24

Quimper, C. G., & Rousseau, L. M. (2010). A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics, 16*(3), 373–392. https://doi.org/10.1007/s10732-009-9106-6

Restrepo, M. I., Lozano, L., & Medaglia, A. L. (2012). Constrained network-based column generation for the multi-activity shift scheduling problem. *International Journal of Production Economics, 140*(1), 466–472. https://doi.org/10.1016/j.ijpe.2012.06.030

Römer, M., & Mellouli, T. (2016). A direct MILP approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. In: Burke, E.K., Di Gaspero, L., Özcan, E., et al (eds) PATAT 2016: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, Udine, Italy, pp 549–552

Van den Bergh, J., Beliën, J., De Bruecker, P., et al. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research, 226*(3), 367–385. https://doi.org/10.1016/j.ejor.2012.11.029