DOI: 10.15439/2021F35
ISSN 2300-5963 ACSIS, Vol. 26

The Extended Shift Minimization Personnel Task Scheduling Problem

Nico Kyngäs
Satakunta University of Applied Sciences, Satakunnankatu 23, 28130 Pori, Finland
Email: nico.kyngas@samk.fi

Kimmo Nurmi
Satakunta University of Applied Sciences, Satakunnankatu 23, 28130 Pori, Finland
Email: nico.kyngas@samk.fi

***Abstract*—In workforce scheduling, shift generation is the process of determining the shift structure, along with the tasks to be carried out in particular shifts. Application areas of shift generation include hospitals, retail stores, contact centers, cleaning, home care, guarding, manufacturing and delivery of goods. We present an extension to the Shift Minimization Personnel Task Scheduling Problem that is a problem in which a set of tasks with fixed start and finish times have to be allocated to a heterogeneous workforce. The objective in the SMPTSP is to minimize the number of employees required to carry out the given set of tasks. In the ESMPTSP, another objective is to maximize the number of feasible (shift, employee) pairs. We provide a mathematical formulation of the extended problem. We present an efficient ruin and recreate heuristic along with computational results for existing SMPTSP data sets and to a new data set. The presented heuristic is suitable for application in large real-world scenarios. The new instances, along with our best solutions, have been made available online.**

I. Introduction

Shift generation is the process of transforming the determined workload into shifts as accurately as possible. For labor-intensive industries, such as hospitals, retail stores, contact centers, cleaning, home care, guarding, manufacturing and delivery of goods, it is crucial to find a good match between the predicted and scheduled workload. The generated shifts form an input for the staff rostering, where employees are assigned to the shifts (see e.g. [1], [2] and [3]).

The generation of shifts is based on either the varying number of required employees working during the planning horizon or the tasks that the shifts must cover. We call these employee-based and task-based shift generation problems. The first major contribution for the employee-based shift generation problem was the study by Musliu et al. [4]. They introduced a problem, in which the workforce requirements for a certain period of time were given, along with constraints about the possible start times and the length of shifts, and an upper limit for the average number of duties per week per employee. Di Gaspero et al. [5] proposed a problem in which the most important issue was to minimize the number of different kinds of shifts used.

Kyngäs et al. [6] introduced the unlimited shift generation problem in which the most important goal is to minimize understaffing and overstaffing. They define a strict version of the problem, in the sense that each timeslot should be exactly covered by the correct number of employees. In the person-based multitask shift generation problem with breaks presented in [7], employees can have their personal shift length constraints and competences. The goal is to ensure that the employees can execute the shifts later in the staff rostering phase.

In the task-based shift generation problem the goal is to create shifts and assign tasks to these shifts so that the employees can be assigned to the shifts. The first major contribution of the task-based problem was the study by Dowling et al. [8]. They developed a day-to-day planning tool and to estimate a minimal staff set capable of operating as the ground staff of an international airport. Valls et al. [9] introduced a model where they minimized the number of workers required to perform a machine load plan. They presented a coloring approach to identify possible allocations along with bounds on the branch-and-bound search tree.

Krishnamoorthy and Ernst [10] introduced a similar group of problems, which they called Personnel Task Scheduling Problems (PTSP). Given the staff that are rostered on a particular day, the PTSP is to allocate each individual task, with specified start and end times, to available staff who have skills to perform the task. Later, Krishnamoorthy et al. [11] introduced a special case referred as Shift Minimization Personnel Task Scheduling Problem (SMPTSP) in which the goal is to minimize the number of employees used to perform the shifts. The SMPTSP has been studied under a few other names. Jansen [12] called SMPTSP the license and shift class design problem. Kroon et al. [13] called SMPTSP tactical fixed interval scheduling problem, and showed that solving it to optimality is NP-hard. The SMPTSP is also similar to the basic interval scheduling problem presented in [14] where the goal is to decide which jobs to process on which machines.

The General Task-based Shift Generation Problem (GTSGP) was defined in [15]. Given the tasks that should be rostered on a particular day, the GTSGP is to create anonymous shifts and assign tasks to these shifts so that employees can be assigned to the shifts. The targeted tasks must be completed within a given time window. For example, shelving in retail stores is often carried out in the forenoon. Some tasks are so-called back-office tasks. For example, in a contact center answering emails might require a given number of working hours per day dedicated to the activity but these tasks can be carried out any time of the day.

The main contributions of this paper are the following:

- a mathematical formulation of the Extended Shift Minimization Personnel Task Scheduling Problem (ESMPTSP)
- a ruin and recreate heuristic, which can successfully solve ESMPTSP instances
- a new benchmark set for the SMPTSP.

The paper is organized as follows. Section 2 first describes the Shift Minimization Personnel Task Scheduling Problem and the General Task-based Shift Generation Problem. Then we define the Extended Shift Minimization Personnel Task Scheduling Problem as an extension to the SMPTSP and as a highly simplified version of the GTSGP. In Section 3, we give the mathematical formulation of the ESMPTSP. We also present a simplified instance of the problem. Section 4 describes the most challenging SMPTSP benchmark instances, which we solve as ESMPTSP instances. Furthermore, we introduce a new benchmark instance set for the SMPTSP. This data set is generated especially for the ESMPTSP. In Section 5, we describe a ruin and recreate heuristic, which can successfully solve instances of SMPTSP, ESMPTSP and GTSGP. Finally, Section 6 presents the first computational results for solving the ESMPTSP. We also compare the results to the best-known SMPTSP results.

II. Problem Description

The Shift Minimization Personnel Task Scheduling Problem [11] can be defined formally as follows. A set of tasks $J = t_1,...,t_n$ needs to be allocated to a set of heterogeneous employees $E = e_1,...,e_m$ over a specified planning horizon. The processing time interval at which a task t has to be performed is determined by a timetable with fixed start time s_t and finish time f_t. Each employee e has a set of tasks $J_e \subseteq J$ that e can carry out. Each task t has a set of employees $E_t \subseteq E$ that can carry out t. All sets J_e and E_t are defined based on skills of employees/skill requirements of tasks and availability of employees/time windows of tasks. The objective is to minimize the number of employees required to perform the given set of tasks. The following basic assumptions hold:

A1. Preemption of tasks is not allowed.

A2. There are no precedence constraints among the tasks.

A3. Each task is processed only once without interruption.

A4. Each employee can execute only one task at a time.

The General Task-based Shift Generation Problem (GTSGP) [15] has the same assumptions besides (A2). However, the problem differs from the SMPTSP in several important ways:

B1. Tasks are not explicitly assigned to employees.

B2. Tasks are not fixed in time.

B3. Tasks may have shift-local precedence constraints.

B4. Transition times between tasks are considered.

B5. Employees have total working time restrictions.

B6. Employees have availability restrictions.

The GTSGP is to create anonymous shifts and assign tasks to these shifts so that employees can be assigned to the shifts. Instead of minimizing the number of employees required to carry out the given set of tasks, the objective is to maximize the number of feasible (shift, employee) pairs. The mathematical formulation of the problem was first given in [16]. The idea is to ensure that the resulting set of shifts can be carried out by the employees, i.e. each shift can be assigned to an employee s.t. all shifts are assigned to someone and no employee is assigned to multiple shifts.

In practical applications of the GTSGP, the full-time permanent and temporary employees are expected to cover 100% of the total workload in the shift generation, and later in the staff rostering phase. This is opposite to the idea behind the SMPTSP, where a large pool of casual staff is expected to be available and management would like to minimize the pool usage.

By including only requirements (B1) and (B2), we obtain the following simplified version of the GTSGP. The set of shifts S is to be generated. A set of tasks T is to be assigned to the shifts. Each task t has a duration d_t (in timeslots) and a time window $[lb_t, ub_t]$. A task t must not start before lb_t and must not end after ub_t. Each task is related to the collection of skills required by the tasks, which is a subset of the skill set C. Respectively, each employee e from the set of employees E has a collection K_e of skills. The number of shifts is usually the same as the number of available employees. In case of understaffing, additional pseudo employees can be used.

A solution to the simplified GTSGP is feasible if the following three hard constraints have no violations:

H1. The tasks in the shift do not overlap in time (overlap).

H2. Each shift can be executed by one or more employees, i.e. the skill set required by the tasks in each shift is possessed by one or more employees (shift).

H3. Each shift can be assigned to an employee s.t. all shifts are assigned to someone and no employee is assigned to multiple shifts (combination).

Modifying (B2) to allow only fixed timetables for the tasks, we have a further simplified version of the GTSGP. We call this problem the *Extended Shift Minimization Personnel Task Scheduling Problem (ESMPTSP)*. The objective is to first minimize the number of employees required to carry out the given set of tasks and then to maximize the number of feasible (shift, employee) pairs.

III. ESMPTSP Formulation

In this section, we give the mathematical formulation of the ESMPTSP. We also present a small instance of the ESMPTSP along with an example solution. We start with introducing some additional definitions and the decision variables:

P_j	The set of employees that may carry out task j.
K_t	The set of tasks active at time t.
K_t^w	The set of tasks active at time t that worker w can perform.
C	The family of sets containing all such sets K_t that $K_t \not\subseteq K_{t'}\ \forall t \neq t'$.
C^w	The family of sets containing all such sets K_t^w that $K_t^w \not\subseteq K_{t'}^w\ \forall t \neq t'$.

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

$$y_{wv} = \begin{cases} 1 & \text{if employee } w \in W \text{ is active and employee} \\ & v \in W \text{ can carry out the shift of } w \\ 0 & \text{otherwise.} \end{cases}$$

For $w \neq v$, we call x_{jwv} pseudoassignments of v to w with respect to j, as they represent whether j could be assigned to v assuming j is assigned to w. Similarly, we call y_{wv} pseudoassignments of v to w, as they represent whether all the tasks (and thus the entire shift) assigned to w could be assigned to v.

$$Z_{ESMPTSP} = min\left(\alpha * \sum_{w\in W} y_{ww} - \beta * \sum_{w,v\in W} y_{wv}\right) \quad (1)$$

$$\text{s.t.} \sum_{w\in P_j} x_{jww} = 1 \quad \forall j \in J \quad (2)$$

$$\sum_{j\in K_t^w} x_{jww} \le y_{ww} \quad \forall w \in W, K_t^w \in C_w \quad (3)$$

$$y_{wv} \le x_{jwv} - x_{jww} + 1 \quad \forall (j,w,v) \in J \times W \times W : w \neq v \quad (4)$$

$$y_{wv} \le \sum_{w\in W} x_{jww} \quad \forall (j,w,v) \in J \times W \times W \quad (5)$$

$$x_{jwv} = 0 \quad \forall (j,w,v) \in J \times W \times W : w \notin P_j \text{ or } v \notin P_j \quad (6)$$

$$x_{jwv} = x_{jww} \quad \forall (j,w,v) \in J \times W \times W : w,v \in P_j \quad (7)$$

$$0 \le y_{wv} \le 1 \quad \forall w \in W, v \in W \quad (8)$$

$$x_{jwv} \in \{0,1\} \quad \forall (j,w,v) \in J \times W \times W \quad (9)$$

The objective function (Equation 1) consists of the weighted sum of the number of used employees and the number of able (employee, shift) pairs. The rest of the equations ensure the following:

(2) Each task will be carried out by exactly one able employee.
(3) No overlapping tasks are assigned to a single employee, and the indicator for using an employee indicates employee usage, i.e. that at least one task is assigned to the employee.
(4) A shift cannot be pseudoassigned to an employee if it has shifts the employee is unable to do.
(5) Empty shifts are not counted as pseudoassignments.
(6,7) Tasks are pseudoassigned according to both actual assignments and the abilities of the employees.
(8,9) Variables must be binary**.**

Fig. 1 shows a small instance of the ESMPTSP along with an example solution. The instance and the presented example of feasible (shift, employee) pairs have the following characteristics:

- The tasks in the shift do not overlap in time (overlap).
- The planning period is divided into 18 timeslots.
- The number of tasks is 14 and the number of employees is 7 (indicated by letters from A to G).
- The duration of the tasks is given by the length of the corresponding rectangles.
- The employees able to carry out a task are indicated by the letters in the rectangles.
- The parentheses indicate a (non-unique) feasible assignment between tasks/shifts and employees.
- The colors indicate which tasks are assigned to the same shift.
- The solution is clearly optimal in the number of shifts, as at least 6 concurrent shifts are needed during slot 12.
- Furthermore, employee E can carry out the blue shift, A and C the brown shift, D and E the green shift, E the yellow shift, B, D and F the violet shift, and A, D and E the red shift, totalling 18 feasible (shift, employee) pairs.

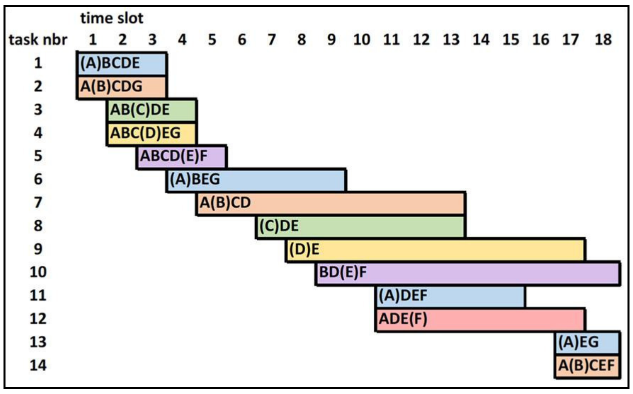


Fig. 1 A small instance of the ESMPTSP and a feasible solution. The letters indicate employees able to carry out a task. The colors indicate generated shifts.

IV. Benchmark Instances

We have basically two possibilities to create benchmark instances for the ESMPTSP: either to extend SMPTSP instances or to simplify GTSGP instances. We use both approaches. First, we use instances selected from the three current SMPTSP data sets. This approach has two benefits. The instances need no modifications whatsoever and a significant number of studies and algorithms have been designed to solve the problem. This enables the authors of the current SMPTSP algorithms to try their ideas to the ESMPTSP more easily. Second, we generate a new fourth data set for the SMPTSP, which is derived from the simplified GTSGP instances.

Krishnamoorthy et al. [11] presented the first data set of 137 instances for the SMPTSP. The data set is referred to as KEB instances. They used a Lagrangian approach to solve large instances of the SMPSTP. Smet et al. [17] generated the second data set of ten instances, because they were able to solve all KEB instances to optimality. The data set is referred to as SWMB instances. Fages and Lapegue [18] generated the third data set of 100 instances, because KEB and SWMB instances were trivial with respect to finding good quality lower bounds. This data set is referred to as FL instances.

As the first benchmark set for the ESMPTSP, we decided to select 56 very challenging instances from the three SMPTSP data sets. From the KEB data set, we selected 25 instances based on the performance of the three heuristics on the data set. An FL instance was selected if at least one of the following criteria holds:

C1. The solution (number of employees) obtained by the constructive heuristic of Lin and Ying [19] was at least 7% inferior to the optimum solution.

C2. The solution obtained by the iterative heuristic of Lin and Ying [19] was at least 2% inferior to the optimum solution.

C3. The greedy heuristic of Hojati [20] was unable to find the optimum solution.

Some of these instances are quite easy to solve as SMPTSP instances. This enables the authors of the SMPTSP algorithms to try their ideas to the ESMPTSP more easily. Note however, that we are not aware of how easy or difficult the instances are to solve as ESMPTSP instances.

From the SWMB data set, we selected all the ten instances.

From FL instances, we selected the 21 instances. We excluded instances #5 and #89, for which the minimum number of shifts is not known. Note that the instances were also unsolved by the Smet et al. method in [17]. An instance was selected if either one of the following criteria holds:

D1. The solution obtained by Fages and Lapegue [18] was at least 3% inferior to the known optimum solution.

D2. The greedy heuristic of Hojati [20] was not able find the optimum solution.

Tables I, II and III show the characteristics of the selected instances. The number of shifts denotes the known optimum value for the SMPTSP, i.e. the minimum number of shifts derived from the recent paper by Chandrasekharan et al. [21]. The @AVG measure indicates the estimated average number of tasks per non-empty shift, i.e. the number of tasks is divided by the minimum number of shifts.

The tightness level is defined as the total length of all tasks as a percentage of the total availability of all employees. The task skill level is defined as the average percentage of the total number of tasks each employee is qualified for. In addition to the task skill level, the shift skill level describes, how qualified an average employee is to carry out all the tasks of an average shift. We define shift skill level = t^a, where t = task skill level and a = @AVG. The overlap level is the probability of two random tasks to overlap. To calculate the probability, we need to iterate all task pairs once to check whether they overlap or not.

In order to make an instance challenging, the following guidelines can be drawn from the discussions and results in [11, 17, 18, 19 and 20]:

- The number of tasks and the number of employees influence the hardness of the instance, since they enlarge the search space.
- The combinatorial search space increases when the average number of tasks per shift increases.
- The tightness level should be closer to 90%.
- The task skill level should be at most 33%.
- The shift skill level should be way below 1%.
- The overlap level should be at most 40%.

However, these are general observations and no statistical evidence can be drawn from the guidelines. We generated a new fourth data set for the SMPTSP and the ESMPTSP based on the above guidelines. We had two goals for the new data set. First, the instances should be more challenging than the existing ones, and second, the characteristics of the instances should be distinctly different compared to the current data sets.

We have created an elaborate random test instance generator for the GTSGP (see [16]). The generator is guided by five parameter sets having a total number of seventeen parameters. We disabled and omitted most of the parameters to generate SMPTSP and ESMPTSP instances. The instances were generated using the following guidelines:

- The instances should be versatile.
- The minimum number of shifts should be the same as the number of employees.
- The number of employees varies from 20 to 500.
- The average number of tasks per shift is close to 5. This is about the same as the average in the other data sets.
- The tightness level should be at least 93% and in most of the cases close to 100%. This is clearly higher than in the other data sets. The level should decrease when the number of employees increases.

- The task skill level should vary from 5% to 65%. In some cases, the level should be clearly lower than in the other data sets. A higher value should increase the ESMPTSP solution value.
- The shift skill level should be at most 1%, and in some cases less than 0.001%. The variation should be about the same as in the other data sets altogether. Note, that seven instances have values close to 10%, which should increase the ESMPTSP solution value.
- The overlap level should be between 30% and 40%. This is on average higher than in the other data sets.

We decided to create two distinct instance sets. The first set of instances should have a unique optimum SMPTSP solution. When the objective is treated hierarchically, i.e. $\alpha = m^2 + 1$ and $\beta = 1$, this results in a unique optimum

TABLE I.
THE CHARACTERISTICS OF THE SELECTED KEB INSTANCES

#	#Emps	#Shifts	#Tasks	@AVG	Tightness level	Task skill level	Shift skill level	Overlap level
4	23	20	59	3.0	88.3	34.4	4.3	46.7
5	25	20	60	3.0	90.2	36.2	4.7	46.9
9	49	40	104	2.6	89.9	35.0	6.5	57.3
11	24	20	119	6.0	90.0	36.2	0.2	26.0
13	25	20	120	6.0	90.2	35.8	0.2	25.9
15	72	60	126	2.1	74.5	34.2	10.5	59.6
17	23	20	139	7.0	90.0	67.7	6.7	23.6
22	47	40	180	4.5	89.9	67.4	16.9	36.1
28	75	60	208	3.5	90.1	66.8	24.7	45.0
29	22	20	219	11.0	89.8	67.2	1.3	15.0
30	25	20	219	11.0	90.1	68.8	1.7	15.0
35	171	140	280	2.0	70.3	33.2	11.0	59.3
45	67	60	420	7.0	90.0	33.9	0.05	24.1
59	70	59	525	8.9	91.4	34.4	0.01	19.4
68	359	300	613	2.0	72.7	66.1	42.9	60.2
75	72	60	665	11.1	90.0	34.2	0.001	15.4
77	180	160	688	4.3	90.0	33.4	0.9	36.8
79	94	80	689	8.6	90.1	33.6	0.01	19.8
80	112	99	691	7.0	90.9	33.8	0.05	24.4
89	88	70	788	11.3	90.1	34.0	0.001	15.3
94	93	80	881	11.0	90.0	33.8	0.001	15.6
98	91	80	896	11.2	90.0	34.2	0.001	15.3
106	121	100	1096	11.0	90.0	33.3	0.001	15.6
107	114	100	1112	11.1	90.0	33.7	0.001	15.4
108	162	128	1115	8.7	91.4	33.6	0.008	19.9

TABLE II.
THE CHARACTERISTICS OF THE SWMB INSTANCES

#	#Emps	#Shifts	#Tasks	@AVG	Tightness level	Task skill level	Shift skill level	Overlap level
1	50	40	258	6.5	89.6	19.5	0.003	25.6
2	44	40	510	12.4	87.6	19.6	0.0000002	13.3
3	102	77	525	6.8	93.5	30.0	0.03	25.4
4	113	98	647	6.6	91.7	20.0	0.002	25.7
5	77	59	777	13.2	91.5	29.7	0,00001	13.2
6	135	116	777	6.7	92.9	19.9	0.002	25.8
7	70	59	781	13.2	88.5	19.9	0.00000006	13.1
8	88	79	1022	12.8	90.0	19.9	0,0000001	13.5
9	125	98	1308	13.2	90.9	19.8	0,00000005	13.2
10	153	116	1577	13.6	93.1	19.9	0,00000003	13.1

TABLE IV.
THE CHARACTERISTICS OF THE SELECTED FL INSTANCES

#	#Emps	#Shifts	#Tasks	@AVG	Tightness level	Task skill level	Shift skill level	Overlap level
28	262	105	402	3.8	19.1	26.0	0.6	11.0
29	248	95	355	3.7	18.5	28.3	0.9	11.6
31	290	116	488	4.2	21.0	25.7	0.3	10.4
33	338	132	534	4.0	20.3	25.7	0.4	10.8
35	308	118	469	4.0	19.8	26.6	0.5	11.1
39	284	108	446	4.1	19.8	25.7	0.4	10.6
45	376	144	586	4.1	20.5	27.0	0.5	11.4
46	409	157	635	4.0	20.2	26.6	0.5	11.2
54	498	190	850	4.5	22.5	25.8	0.2	10.7
60	443	173	783	4.5	21.9	27.0	0.3	10.7
61	551	222	891	4.0	20.0	26.3	0.5	11.0
62	610	262	1096	4.2	20.8	25.5	0.3	10.2
63	524	203	905	4.5	21.9	26.5	0.3	11.1
64	366	140	570	4.1	20.2	26.2	0.3	11.0
68	561	219	958	4.4	21.4	27.3	0.4	11.0
69	550	211	891	4.2	21.1	26.1	0.3	10.8
77	648	248	1123	4.5	22.1	26.6	0.3	10.7
79	638	246	1052	4.3	21.0	26.3	0.3	10.8
80	578	222	885	4.0	19.6	27.0	0.5	11.2
84	644	247	1121	4.5	21.9	26.1	0.2	10.4
94	812	313	1394	4.5	21.9	25.7	0.2	10.6

TABLE III.
THE CHARACTERISTICS OF THE NEW KN INSTANCES. THE INSTANCES DENOTED BY * SHOULD HAVE A UNIQUE OPTIMAL SMPTSP SOLUTION. LB = LOWER BOUND OBTAINED USING SOLYALI PROCEDURE.

#	#Emps	#Shifts LB	#Tasks	@AVG	Tightness level	Task skill level	Shift skill level	Overlap level
1*	20	20	105	5.3	100	24.5	0.06	33.0
2	25	25	125	5.0	93.9	19.8	0.03	32.8
3*	30	30	146	4.9	100	60.3	8.5	35.9
4	35	35	174	5.0	94.5	62.0	9.3	33.8
5*	40	40	200	5.0	100	18.8	0.02	35.2
6	45	45	216	4.8	95.5	17.7	0.02	35.2
7*	50	50	266	5.3	100	32.1	0.2	33.5
8	55	55	265	4.8	95.0	63.3	11.1	35.0
9*	60	60	297	5.0	99.5	20.3	0.04	35.6
10	65	65	335	5.2	95.5	54.9	4.6	33.2
11*	70	70	352	5.0	99.5	19.8	0.03	35.2
12	75	75	363	4.8	94.0	39.9	1.2	34.6
13*	80	80	417	5.2	99.0	15.0	0.005	34.0
14	85	85	434	5.1	95.0	36.5	0.6	33.3
15*	100	100	506	5.1	98.5	10.8	0.001	34.7
16	110	110	583	5.3	95.5	42.5	1.1	32.4
17*	120	120	613	5.1	98.0	7.7	0.0002	34.4
18	130	130	641	4.9	94.5	31.1	0.3	34.5
19*	140	140	670	4.8	97.6	5.2	0.00007	36.4
20	150	149	782	5.2	95.0	29.4	0.2	32.9
21	160	160	802	5.0	97.3	62.7	9.6	34.8
22	180	180	843	4.7	97.0	37.8	1.1	36.9
23	200	199	967	4.8	95.0	19.9	0.04	35.3
24	240	239	1157	4.8	96.6	57.9	7.1	35.9
25	280	279	1419	5.1	96.3	32.5	0.3	34.3
26	320	309	1611	5.0	95.0	5.4	0.00004	32.7
27	360	356	1763	4.9	96.0	63.7	11.0	35.2
28	400	399	2012	5.0	96.0	40.0	1.0	34.4
29	450	443	2231	5.0	94.5	12.0	0.003	34.4
30	500	488	2473	4.9	93.5	8.6	0.0005	34.1

ESMPTSP solution as well. The second set of instances can have many different optimum SMPTSP solutions per instance, from which we should find the best ESMPTSP solution. These instances should be easier to solve as tightness level decreases and shift skill level increases.

The generator creates instances where the minimum number of shifts should be the same as the number of employees. The basic idea is to construct an instance along with a feasible solution. Applying the lower bounding procedure of Solyali [22] confirmed this for 21 instances. However, when the number of employees increases and the tightness level decreases, the generator may create instances where the minimum number of shifts is less than the number of employees.

Table IV shows the characteristics of the thirty instances generated. We refer to this data set as KN instances. The characteristics clearly show that the instances are distinctly different from the existing data sets. The instances denoted by * belong to the first set of instances, which should have a unique optimum SMPTSP solution.

The existing three data sets can be found in [23] and the new KN data set in [24]. We provide the new KN instances in the traditional SMPTSP format as well as in the GTGSP format.

V. Ruin and Recreate Heuristic

We use a ruin and recreate heuristic similar to that described in [25] to solve the ESMPTSP. Pseudo code for the algorithm is given in Fig. 2. Our version of ruin and recreate heuristic (2RH) has been created to solve practical GTSGP instances. In practical applications of the GTSGP, we should

- generate as versatile shifts as possible to
- ensure that the rostering of the staff can be completed, so that
- the computation time is still acceptable considering the release time of the rosters.

Therefore, we do not seek the fastest possible solution method. Instead, it is advantageous to use more computation time in order to achieve shifts that are more versatile.

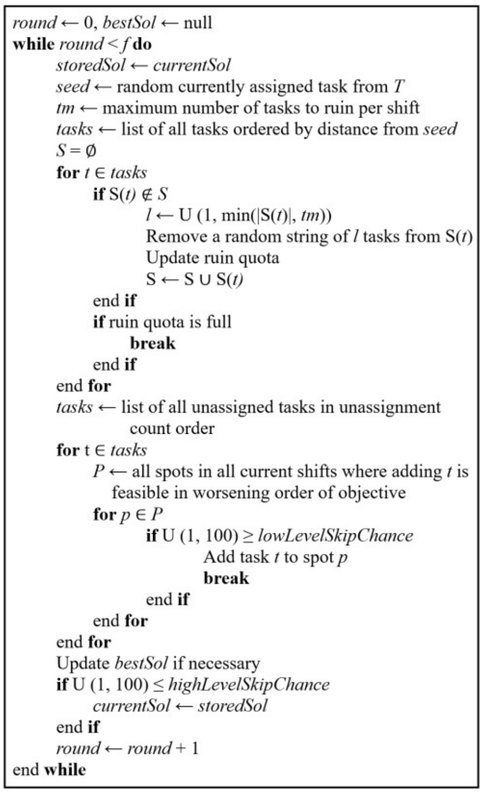
The ruin operator first chooses a random task t_0 assigned to some shift s_0 and removes a random sequence of adjacent tasks from s_0 containing t_0. For each task sorted by closeness to t_0 w.r.t. time windows and skills, similar removal is done if the corresponding shift has not been removed from yet. The underlying idea is that by removing whole strings of tasks from shifts at a time, room is created for new tasks to be inserted, and due to the way removed tasks are selected, at least some of them are more or less interchangeable between shifts. When the total number of removed tasks exceeds the given parameter, the ruin operator quits.

The recreate operator adds free tasks one by one to their respective best positions in the incumbent solution. First, all free tasks are sorted in order of how many times they have been unassigned after recreation during the solution process. This is done in order to emphasize tasks that seem more difficult to place in the solution. For each task *t*, all feasible addition positions in the incumbent solution are evaluated.

The concept of a position depends on the exact problem. For the GTSGP, a position is determined by an immediate predecessor, e.g. a task or an employee. Note that the tasks can have wide time windows in the GTSGP, hence the order of tasks within a shift is not predetermined. In the SMPTSP there are no time windows, which fixes the order of tasks within a shift. Task *t* is then added to the position that leads to the best objective function value, with a small chance to skip over to the next best position. Consecutive skipping is not constrained in any way, so *t* might not get assigned even if it has feasible addition positions. When all free tasks have been processed, the recreate operator quits.

Essential parameters of 2RH and the values used in our computational experiments include

- average number of tasks removed per ruin operator (10),
- maximum length of task string to remove from a single shift (8),
- recreate skipping chance (1%), and
- move skipping chance (1%).



```
round ← 0, bestSol ← null
while round < f do
    storedSol ← currentSol
    seed ← random currently assigned task from T
    tm ← maximum number of tasks to ruin per shift
    tasks ← list of all tasks ordered by distance from seed
    S = ∅
    for t ∈ tasks
        if S(t) ∉ S
            l ← U (1, min(|S(t)|, tm))
            Remove a random string of l tasks from S(t)
            Update ruin quota
            S ← S ∪ S(t)
        end if
        if ruin quota is full
            break
        end if
    end for
    tasks ← list of all unassigned tasks in unassignment
        count order
    for t ∈ tasks
        P ← all spots in all current shifts where adding t is
          feasible in worsening order of objective
        for p ∈ P
            if U (1, 100) ≥ lowLevelSkipChance
                Add task t to spot p
                break
            end if
        end for
    end for
    Update bestSol if necessary
    if U (1, 100) ≤ highLevelSkipChance
        currentSol ← storedSol
    end if
    round ← round + 1
end while
```

Fig. 2 The pseudo-code of the ruin and recreate heuristic.

VI. Computational Results

This section presents our computational results for the ESMPTSP benchmark instances introduced in Section 4. As was stated in previous section, our implementation of 2RH requires sufficient running time to tackle the largest practical instances. For each ESMPTSP benchmark instance, we execute eight parallel 2RH runs exactly four hours. We also register the running time elapsed to reach the first solution to the SMPTSP as well as the value for the ESMPTSP at that time.

Note that our implementation of 2RH is such that the best solution is generally reached at the later stages of the optimization run. We could find the first solution to the SMPTSP faster if we used less maximum running time for 2RH. It should also be noted here, that running parallel 2RHs one hour instead of four hours seems to weaken ESMPTSP solutions only at most 5%. This could be acceptable for practical applications, since after the shift generation and staff rostering have completed, new tasks will most certainly arise and some of the tasks need to be changed or removed.

Tables V, VI, VII and VIII show our results for the benchmark instances. We ran the instances using hierarchical objective, i.e. $\alpha = m^2 + 1$ and $\beta = 1$, i.e. only solutions optimal w.r.t. to the underlying SMPTSP need to be considered. The optimum values for the underlying SMPTSPs were derived from the recent paper by Chandrasekharan et al. [21]. The test runs were carried out on a workstation with AMD Ryzen 9 3950X 16-Core Processor at 3.49GHz and with 64GB RAM running Windows 10 using default settings. 2RH is implemented in C++.

We used no domain specific knowledge in order to generate better solutions, nor did we fine-tune any parameter. However, we also experimented with different numbers of parallel runs, different parameter values and different running times. We publish the best solutions we have found during these experiments, but we emphasize that these solutions are shown only for comparison purposes. For comparison purposes, we also conducted 30-minute test runs with Gurobi 8.1. Gurobi was able to verify, that we have found the optimum ESMPTSP solution for 16 instances. For the other 70 instances, we do not know the optimum values. Note that Gurobi was not able to find optimum solutions to any of the SWMB and FL instances.

First, the results show that 2RH can solve the underlying SMPTSP instances extremely well and sufficiently fast. With respect to the SMPTSP, the heuristic can successfully solve the most challenging existing benchmark instances as well as the new KN instances. The SWMB instances are very challenging. These instances have a high average number of tasks per shift and low task skill levels, which implies very low shift skill levels.

With respect to the running time required to reach the SMPTSP optimum, SWMB4 and SWMB8 are easier, and SWMB3 and SWMB6 harder. We could not solve SWMB7 and SWMB10 instances within the given time limit using the given parameter set. However, we did find the SMPTSP optimum by increasing the time limit.

The FL instances are quite easy to solve. The instances have very low tightness levels and shift skill levels are not too low. The KEB instances are easy to solve. This is true even for those instances that have low shift skill levels. We have no other reason for this than the high task skill levels. With respect to the running time required to reach the SMPTSP optimum, KEB080 and FL77 are the hardest ones.

Our goal in creating the KN data set was to generate instances that are more challenging. The results indicate this to be true at least for our ruin and recreate heuristic. There are mainly two reasons for this: higher tightness levels and lower task skill levels.

With respect to the running time required to reach the SMPTSP optimum, the easiest instances are KN5 and KN19. Among the instances with equal number of shifts and employees, the hardest instances are KN7 and KN22. We could not solve KN26, KN29 and KN30 instances to the lower bound value. We suspect that our solutions to these instances are not optimal.

We define the ratio $r = e/s$, where s = the best-known SMPTSP value and e = the number of feasible pairs in the best-known ESMPTSP solution. In general, we could argue that instances with high r values should be easier to solve as SMPTSP and as ESMPTSP instances, because several employees may carry out several shifts. This seems to be true, because the FL instances have a very high r value, and the SWMB instances have r values very close to one. However, for the KEB and KN instances, there is no correlation between r values and hardness of instances.

We only know the optimum ESMPTSP values for 16 of the 86 benchmark instances. It should be clear, that the combinatorial search space for an ESMPTSP instance increases when its r value increases. Therefore, we speculate that our solution to the ESMPTSP should be closer to the optimum value for those instances that have low r values.

The KN instances intended to have a unique optimum SMPTSP solution turned out to be almost trivial for Gurobi. The corresponding SMPTSP instances were solved in a few seconds each. It seems likely that the instances are so heavily constrained that methods focused on reducing the search space are far superior to any pure heuristics.

VII. Conclusions

We presented a mathematical formulation of the Extended Shift Minimization Personnel Task Scheduling Problem (ESMPTSP), which in turn is a highly simplified version of the GTSGP. We showed that the presented 2RH heuristic can successfully solve ESMPTSP benchmark instances. Furthermore, we showed that the heuristic was able to find optimal solutions to the SMPTSP instances.

We published a new data set for the SMPTSP and the ESMPTSP. We provide the new instances in the traditional SMPTSP format as well as in the GTGSP format. The instances, along with our best solutions, have been made available online [24].

This was the first encounter of solving the SMPTSP instances as ESMPTSP instances. Even though the computational results were encouraging, we suspect that better solutions for most of the instances exist. Furthermore, there should be room for both more efficient solution methods and efficient lower bounding methods. These would also bring more insight to the hardness of the problem as well as to the hardness of the current benchmark instances.

No matter the point during the planning process at which the problem is solved, there will always be changes, be it to the tasks themselves due to e.g. changed customer expectations or the employees at our disposal due to e.g. sick leaves. In practice, solutions with equal number of shifts and larger number of feasible pairs are better because the flexibility of the assignment of the shifts is increased, making it easier to assign existing shifts to different employees. This justifies the ESMPTSP in the big picture of the workforce optimization and the real-world workforce scheduling process. We believe that the presented method is suitable for application in large real-world scenarios.

TABLE V.
THE RESULTS FOR KEB INSTANCES. TIMES GIVEN IN MINUTES. THE FINAL ESMPTSP SOLUTION IS THE BEST OF EIGHT FOUR-HOUR PARALLEL 2RH RUNS. THE BEST SOLUTIONS WE HAVE FOUND IN OUR OTHER TEST RUNS ARE ALSO REPORTED. THE SOLUTIONS DENOTED BY ^{O} ARE OPTIMUM.

#	SMPTSP optimum	Time to reach the SMPTSP optimum	# of feasible pairs at that time	Final # of feasible pairs	Best # we have found
4	20	0.001	30	53^{o}	
5	20	0.001	41	62^{o}	
9	40	0.06	161	233	
11	20	0.03	20	29	30^{o}
13	20	0.03	20	31	32^{o}
15	60	0.02	506	684	694
17	20	0.1	58	97	
22	40	0.1	351	518	524
28	60	0.3	1152	1489	1497
29	20	0.2	33	68	69
30	20	0.3	31	78	
35	140	0.06	2701	3534	3558
45	60	2	63	110	114
59	59	1.7	60	79	85
68	300	0.2	46422	49448	49539
75	60	3	60	76	84
77	160	1.8	485	1021	1053
79	80	3	84	127	131
80	99	9	121	214	219
89	70	0.9	70	92	95
94	80	4	80	110	115
98	80	6	81	109	113
106	100	3	100	140	146
107	100	4	100	140	146
108	128	7	136	220	232

TABLE VI.
THE RESULTS FOR SWMB INSTANCES. TIMES GIVEN IN MINUTES. THE FINAL ESMPTSP SOLUTION IS THE BEST OF EIGHT FOUR-HOUR PARALLEL 2RH RUNS. THE BEST SOLUTIONS WE HAVE FOUND IN OUR OTHER TEST RUNS ARE ALSO REPORTED. SWMB7 AND SWMB10 COULD NOT BE SOLVED WITHIN THE GIVEN TIME LIMIT.

#	SMPTSP optimum	Time to reach the SMPTSP optimum	# of feasible pairs at that time	Final # of feasible pairs	Best # we have found
1	40	67	40	41	45
2	40	30	40	41	42
3	77	139	83	101	117
4	98	15	98	111	114
5	59	86	59	60	
6	116	207	116	129	
7	59	*	*	*	59
8	79	30	79	80	
9	98	102	98	99	
10	116	*	*	116	

TABLE VII.
THE RESULTS FOR FL INSTANCES. TIMES GIVEN IN MINUTES. THE FINAL ESMPTSP SOLUTION IS THE BEST OF EIGHT FOUR-HOUR PARALLEL 2RH RUNS. THE BEST SOLUTIONS WE HAVE FOUND IN OUR OTHER TEST RUNS ARE ALSO REPORTED.

#	SMPTSP optimum	Time to reach the SMPTSP optimum	# of feasible pairs at that time	Final # of feasible pairs	Best # we have found
28	105	0.1	3181	4034	4051
29	95	0.1	3389	4285	4296
31	116	0.1	3614	4505	4535
33	132	0.2	4728	5915	5933
35	118	0.1	4418	5374	5393
39	108	0.2	3318	4215	4224
45	144	0.3	6526	8456	8461
46	157	0.7	7306	9699	9702
54	190	1	9560	11321	
60	173	0.6	8102	9648	9698
61	222	2	14084	17590	17646
62	262	3	16776	19419	19497
63	203	1	11140	13088	13191
64	140	0.3	5981	7499	7524
68	219	2	14331	18576	18682
69	211	1	12506	15945	16045
77	248	14	17619	20824	20986
79	246	2	16974	21917	22056
80	222	1	16096	20034	20154
84	247	4	17515	21953	22110
94	313	6	25574	30006	30342

TABLE VIII.
THE RESULTS FOR KN INSTANCES. THE INSTANCES DENOTED BY * SHOULD HAVE A UNIQUE OPTIMAL SMPTSP SOLUTION. TIMES GIVEN IN MINUTES (X = NOT FOUND WITHIN THE GIVEN TIME LIMIT). THE FINAL ESMPTSP SOLUTION IS THE BEST OF EIGHT FOUR-HOUR PARALLEL 2RH RUNS. THE BEST SOLUTIONS WE HAVE FOUND IN OUR OTHER TEST RUNS ARE ALSO REPORTED. KN26, KN29 AND KN30 COULD NOT BE SOLVED WITHIN THE GIVEN TIME LIMIT. THE SOLUTIONS DENOTED BY ^{o} ARE OPTIMUM.

#	Best found SMPTSP solution	Time to reach the best found SMPTSP solution	# of feasible pairs at that time	Final # of feasible pairs	Best # we have found
1*	20	0.002	43	43^{o}	
2	25	0.03	55	57^{o}	
3*	30	26	235	235^{o}	
4	35	1	365	456	
5*	40	0.08	89	89^{o}	
6	45	0.3	96	101^{o}	
7*	50	192	147	147^{o}	
8	55	0.6	1063	1369	1375
9*	60	21	76	76^{o}	
10	65	3.8	753	1259	1271
11*	70	36	118	118^{o}	
12	75	0.9	945	1361	1378
13*	80	43	220	220^{o}	
14	85	29	791	1106	1160
15*	100	37	128	128^{o}	
16	110	34	1446	2562	2621
17*	120	11	131	131^{o}	
18	130	10	1471	3012	3150
19*	140	1	152	152^{o}	
20	149	128	1620	2513	2933
21	160	42	8473	11858	12101
22	180	179	4411	5952	6888
23	199	148	1852	2163	2654
24	239	92	18041	24858	25835
25	279	110	6452	9674	11778
26	316	*	*	*	795
27	356	156	47213	63392	66171
28	399	92	24160	44071	51277
29	445	*	*	*	5190
30	495	*	*	*	2731

REFERENCES

[1] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models", European Journal of Operational Research vol. 153, no. 1, pp. 3-27, 2004.

[2] J. Van den Bergh, J. Belin, P. De Bruecker, E. Demeulemeester and L. De Boeck, "Personnel scheduling: A literature review", European Journal of Operational Research vol. 226, no. 3, pp 367-385, 2013.

[3] L. Kletzander and N. Musliu, "Solving the General Employee Scheduling Problem", Computers and Operations Research, vol. 13: 104794, 2020.

[4] N. Musliu, A. Schaerf and W. Slany, "Local search for shift design", European Journal of Operational Research, vol. 153, no. 1, pp. 51-64, 2004.

[5] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf and W. Slany, "The minimum shift design problem", Annals of Operations Research, vol. 155, no. pp. 79-105, 2007.

[6] N. Kyngäs, D. Goossens, K. Nurmi and J. Kyngäs, "Optimizing the unlimited shift generation problem", In: Di Chio C. et al. (eds) Applications of Evolutionary Computation. EvoApplications, Lecture Notes in Computer Science, vol. 7248, pp. 508-518, 2012.

[7] N. Kyngäs, K. Nurmi and J. Kyngäs, "Solving the person-based multitask shift generation problem with breaks", Proceedings of the 5th International Conference On Modeling, Simulation And Applied Optimization, pp. 1-8, 2013.

[8] D. Dowling, M. Krishnamoorthy, H. Mackenzie and H. Sier, "Staff rostering at a large international airport", Annals of Operations Research vol. 72, pp. 125-147, 1997.

[9] V. Valls, A. Perez and S. Quintanilla, "A graph colouring model for assigning a heterogenous workforce to a given schedule", European Journal of Operations Research vol. 90, pp. 285-302, 1996.

[10] M. Krishnamoorthy and A.T. Ernst, "The personnel task scheduling problem", Optimization Methods and Applications, pp. 343–367, 2001.

[11] M. Krishnamoorthy, A.T. Ernst and D. Baatar, "Algorithms for large scale Shift Minimisation Personnel Task Scheduling Problems", European Journal of Operational Research, vol. 219, no. 1, pp. 34-48, 2012.

[12] K. Jansen, "An approximation algorithm for the license and shift class design problem", European Journal of Operational Research vol. 73, pp. 127-131, 1994.

[13] L.G. Kroon, M. Salomon and L.N. Van Wassenhove, "Exact and approximation algorithms for the tactical fixed interval scheduling problem", Operations Research vol. 45, no. 4, pp. 624-638, 1997.

[14] A.W.J. Kolen, J.K. Lenstra, C.H. Papadimitriou and F.C.R. Spieksma, "Interval Scheduling: A Survey", Naval Research Logistics vol. 54, no. 5, pp. 530-543, 2007.

[15] K. Nurmi, N. Kyngäs and J. Kyngäs, "Workforce Optimization: the General Task-based Shift Generation Problem", IAENG International Journal of Applied Mathematics, vol. 49, no. 4, pp. 393-400, 2019.

[16] N. Kyngäs, K. Nurmi and D. Goossens, "The General Task-based Shift Generation Problem: Formulation and Benchmarks", Proceedings of the 9th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA), pp. 301-319, 2019.

[17] P. Smet, T. Wauters, M. Mihaylov and G. Vanden Berghe, "The shift minimization personnel task scheduling problem: A new hybrid approach and computational insights", Omega vol. 46, pp. 64-73, 2014.

[18] J.G. Fages and T. Lapegue, "Filtering Atmostnvalue with Difference Constraints: Application to the Shift Minimisation Personnel Task Scheduling Problem", Lecture Notes in Computer Science, vol. 8124, pp. 63-79, 2013.

[19] S.-W. Lin and K.-C. Ying, "Minimizing Shifts for Personnel task Scheduling Problems: A three-Phase Algorithm", European Journal of Operational Research vol. 237, pp. 323-334, 2014.

[20] M. Hojati, "A greedy heuristic for shift minimization personnel task scheduling problem", Computers and Operations Research vol. 100, pp. 66-76, 2018.

[21] R. Chirayil Chandrasekharan, P. Smet, and T. Wauters, "An automatic constructive matheuristic for the shift minimization personnel task scheduling problem", J Heuristics, Feb. 2020, doi: 10.1007/s10732-020-09439-9.

[22] O. Solyali, "The Shift Minimization Personnel Task Scheduling Problem: An Effective Lower Bounding Procedure", Hacettepe Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, vol. 34, no. 2, Jun. 2016, doi: 10.17065/huniibf.259136.

[23] T. Lapègue: "Personnel Task Scheduling Problem Library", [Online]. Available: https://sites.google.com/site/ptsplib/smptsp/instances, (Last access 15-January-2021).

[24] K. Nurmi: "The General Task-based Shift Generation Problem - Benchmark Instances", [Online]. Available: http://web.samk.fi/public/tkiy/GTSGP/, (Last update 26-July-2021).

[25] K. Sörensen, M. Sevaux and F. Glover, "A History of Metaheuristics", In: R. Martí, P. Pardalos and M. Resende (eds) Handbook of Heuristics, pp. 791-808, 2018.