Proceedings of the International MultiConference of Engineers and Computer Scientists 2019
IMECS 2019, March 13-15, 2019, Hong Kong

General Task-based Shift Generation Problem

Kimmo Nurmi, Nico Kyngäs and Jari Kyngäs

***Abstract*— Workforce scheduling studies have mainly focused on staff rostering, i.e. assigning employees to shifts and determining working days and rest days. In the recent years, the generation of shifts has gained increasing interest in academic community. Shift generation is the process of determining the shift structure, along with the tasks to be carried out in particular shifts and the competences required for different shifts. Application areas of staff rostering and shift generation include hospitals, retail stores, call centers, cleaning, home care, guarding, manufacturing and delivery of goods. This paper presents the General Task-based Shift Generation Problem (GTSGP). To the best of our knowledge, the problem has not been studied in the literature. 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. Tasks may have precedence constraints and transition times between tasks are considered. The goal is to maximize the number of shifts employees are able to execute. We present the first computational results of solving GTSGP instances. We briefly describe the PEAST algorithm, which is used to solve the test instances.**

***Index Terms*— PEAST algorithm, shift generation, workforce optimization, workforce scheduling.**

I. Workforce Scheduling and Workforce Optimization

Workforce scheduling, also called staff scheduling and labor scheduling, is a difficult and time consuming problem that every company or institution that has employees working on shifts or on irregular working days must solve. Workforce scheduling studies have mainly focused on staff rostering, i.e. assigning employees to shifts and determining working days and rest days. In the recent years, the generation of shifts has gained increasing interest in academic community. Shift generation is the process of determining the shift structure, along with the tasks to be carried out in particular shifts and the competences required for different shifts. Application areas of staff rostering and shift generation include hospitals, retail stores, call centers, cleaning, home care, guarding, manufacturing and delivery of goods.

Shift generation is essential in cases where the workload is not static. On the contrary, in airlines, railways and bus companies and mostly in hospitals the demand for employees is quite straightforward because the timetables are known beforehand and the shifts are already fixed. The most important optimization target is to match the shifts to the workload as accurately as possible. The generation of shifts is based on either the number of employees working at the certain timeslots or the number of tasks that the shifts have to cover.

The generated shifts form an input for the staff rostering, where employees are assigned to the shifts. The length of the planning horizon is usually between two and six weeks. The most important constraints are employees' competences and preferences as well as the working and resting times, since these are laid down by the collective labor agreements and government regulations. Note that staff rostering also includes the scheduling of days-off and vacations.

In theory, the best results can be achieved when shift generation and staff rostering are processed and solved at the same time. However, different variations of both problems and even their sub-problems are known to be NP-hard and NP-complete [1]-[5]. Nonetheless, some interesting implementations exist. Jackson et al. [6] presented a very simple randomized greedy algorithm that uses very little computational resources. Lapegue et al. [7] introduced the Shift Design and Personnel Task Scheduling Problem with Equity objective (SDPTSP-E) and built employee timetables by fixing days-off, designing shifts and assigning fixed tasks within these shifts. They minimized the number of tasks left unassigned.

Dowling et al. [8] first created a master roster, a collection of working shifts and off shifts, and then allocated the tasks in their Task Optimiser module. Prot et al. [9] proposed a two-phase approach consisting in first computing a set of interesting shifts, then, each shift is used to build a schedule by assigning tasks to workers, and then iterating between these two phases to improve solutions. They relaxed the constraint that each task has to be assigned. Smet et al. [10] presented the Integrated Task Scheduling and Personnel Rostering Problem, in which the task demands and the shifts are fixed in time. Due to the complexity issues in large-scale practical applications, shift generation and staff rostering are mainly solved separately. Our approach is to first generate the shifts and then roster the staff as described in Section III.

Good overviews of workforce scheduling are published by Ernst et al. [11], Musliu et al. [12], Di Gaspero et al. [13] and Vanden Berghe et al [14]. Kletzander and Musliu [15] propose a very interesting framework for the general employee scheduling problem.

The article is organized as follows. In the next section we introduce different shift generation scenarios and the problems triggered from these scenarios. Section III describes the general task-based shift generation problem. Section IV briefly describes the PEAST algorithm, which is used to solve wide variety of scheduling problems. In Section V we present our first computational results.

Manuscript received December 20, 2018.
Kimmo Nurmi, Nico Kyngäs and Jari Kyngäs are with the Satakunta University of Applied Sciences, Pori, Finland (phone: +358 44 710 3371, e-mail: cimmo.nurmi@samk.fi).

II. Introduction to Shift Generation Problems

The generation of shifts is based on either the number of employees working at the certain timeslots or the number of tasks that the shifts have to cover. We call these problems employee-based and task-based shift generation problems.

Numerous models and algorithms for shift generation problems have been developed. The first major contribution was published by Musliu et al. [12]. They introduced an employee-based 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. They generated solutions that contained shifts (and the number of employees per shift) that minimize the number of shifts, overstaffing, understaffing, and differences in the average number of duties per week.

Di Gaspero et al. [16] proposed an employee-based problem in which the most important issue was to minimize the number of shifts. They dealt with cyclic schedules, i.e. the last planning day of the planning horizon (e.g. one week) coincides with the first planning day of the next cycle, and the requirements are repeated in each cycle. The problem statement also includes a collection of acceptable shift types each of them characterized by the earliest and the latest start times, and a minimum and maximum length of its shifts.

Bhulai et al. [17] presented a generalized model for multi-skill shift design in call centers. Their method generated a rough match between the predicted workload and labor capacity, taking the stochastic nature of the call arrival process into account. Kyngäs et al. [18] introduced the unlimited shift generation problem in which the most important goal is to minimize understaffing and overstaffing. They define the strict version of the problem such as each timeslot should be exactly covered by the correct number of employees. In [15]-[17], the shifts are limited to a number of types, for which the length and the start time of the shifts have to be within certain ranges. In [18], the lengths and the start times of the shifts are not strictly limited.

In the person-based multitask shift generation problem with breaks presented in [19], employees can have their personal shift length constraints and competences. Even if the goal is to construct a set of shifts and not to assign them to employees, they ensure that the employees' have the ability to execute the shifts. They do this by choosing a suitable subset of the employees as a preprocessing phase for the shift generation process and creating each shift according to a single employee's personal (shift length and competence) constraints. The exact procedure can be done separately, but in a real-world case with a realistic planning horizon (usually at least a week) it is often best to schedule days-off first and then use the result as a basis for the staff rostering.

Compared to the employee-based shift generation problem, far fewer models and algorithms have been developed for the task-based shift generation problem in which a number of different tasks must to be carried out. The problem is to create shifts and assign tasks to these shifts so that employees can be assigned to the shifts.

The first major contribution of task-based problem was published by Dowling et al. [8]. They created a master roster, a collection of working shifts and off shifts, and then allocated the tasks in their Task Optimiser module, which is invoked one day before the day-of-operation. They allocated a set of tasks (with required attributes and with known start and end times) to personnel with the requisite skills who are available for work on that day (with known shift start and end times).

Krishnamoorthy and Ernst introduced a group of problems called Personnel Task Scheduling Problems (PTSP) in [20]. 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. [21] introduced a special case referred as Shift Minimisation Personnel Task Scheduling Problem (SMPTSP) in which the only cost incurred is due to the number of personnel (shifts) that are used. A similar model was earlier presented in [22] where they minimized the number of workers required to perform a machine load plan. The SMPTSP is also similar to the basic interval scheduling problem presented in [23] where the goal is to decide which jobs to process on which machines. Unlike in the interval scheduling problem, in the SMPTSP all tasks need to be assigned and not all employees can process each task.

Lin and Ying [24] developed a three-phase heuristic for the SMPTSP. They obtain an initial solution using a simple but very effective construction heuristic, which is then improved using an iterated greedy heuristic, which in turn is used as an initial upper bound while solving the MIP model of the problem. Lapegue et al. introduced an equity objective to the SMPTSP [25]. The idea is to find a solution where employees have approximately the same amount of work, thus generating more fair schedules. However, they relax the constraint that all tasks need to be assigned. The SMPTSP problem with Equity objective minimizes the number of unassigned tasks and the inequity among workers.

The next section describes more general shift generation problem in which the tasks are not fixed in time. Furthermore, the tasks are not explicitly assigned to employees.

III. The General Task-based Shift Generation Problem

In this section, we present the General Task-based Shift Generation Problem (GTSGP). To the best of our knowledge, the problem has not been studied in the literature. However, numerous models, algorithms and implementations for specific shift generation problems have been developed as presented in Section II. Due to the complexity issues in large-scale practical applications, shift generation and staff rostering are mainly solved separately. In most cases, the first problem consists of choosing a small subset of shifts from a pre-defined set of shift types that is common to all days of the planning horizon and then for each day of the planning horizon, deciding which employees execute which tasks in which shifts. For

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

example, Dowling et al. [8] and Prot et al. [9] first create a set of shifts and then assign the tasks to the shifts. Our approach is the opposite. We first generate the shifts and then roster the staff. This approach is due to customer needs in retail stores, cleaning, home care and guarding.

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. It is obvious that we will need far more employees if we fix the starting times of the tasks compared to the dynamic starting times. Some tasks are so-called back-office tasks. In a contact center, for example, answering emails might require a given number of working hours per day dedicated to the activity even though the distribution of those hours within the day does not matter. The back-office work could be preempted as stated in [9]. However, in real-world applications, we should have a freedom in determining the starting times of back-office tasks.

The GTSGP differs from the SMPTSP in several ways:

1) tasks are not explicitly assigned to employees
2) tasks are not fixed in time
3) tasks may have precedence constraints
4) transition times between tasks are considered
5) the number of shifts employees are able to execute is maximized.

We consider the shift structure separately for each day, so there is no connection between the shifts of different days. Recall, that we roster the staff after the shifts have been generated for each day. This is why we must create as versatile shifts as possible to insure that the rostering of the staff can be completed. For each day, the goal is to maximize the number of shifts employees are able to execute. We have to ensure that

- no such combination of tasks in a shift exist that no employee has ability to execute
- combination of tasks in shifts are such that each employee can execute at least one shift
- at least one shift is assigned to each employee and at least one employee to each shift.

The GTSGP covers a one-day planning period of t timeslots. The set of shifts S is to be generated. 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 set of tasks T is to be assigned to the shifts. Each task t_i has a duration d_i (in timeslots), a time window [ts_i, te_i], a task type y_i and a location l_i. A task t_i must not start before ts_i and must not end after te_i. A task type y_i is related to the collection C_i of skills, which is a subset of the skill set C. It is easier to manage the required skills for the tasks by first classifying them to task types. Respectively, each employee e_i from the set of employees E is related to the collection D_i of skills. In addition, each employee e_i has a time constraint [ws_i, we_i] for the total working time and an availability set A_i of timeslots. An employee e_i must not work less than ws_i or more than we_i timeslots in the targeted day. An employee e_i cannot execute tasks that are assigned to the timeslots not in A_i. Furthermore, each employee e_i has a unique transition matrix M_i, where M_{ijk} indicates the number of timeslots needed for the employee i to transit from location j to location k. For example, one employee can use a car or a bus while the other can use a bicycle. In summary, an employee e_i can be assigned to the shift s only if the following criteria hold:

(C1) He/she possesses all the skills indicated by the task types of the tasks assigned to the shift (*skill*).
(C2) The total working time of the tasks in the shift is within [ws_i, we_i] (*working time*).
(C3) All the timeslots of the tasks in the shift are included in A_i (*availability*).
(C4) He/she has enough transition time to move between the tasks in the shifts (*transition time*).

The GTSGP has four basic assumptions:

(B1) Each task will be assigned to a shift.
(B2) Preemption of tasks is not allowed.
(B3) Each task is processed only once without interruption.
(B4) Each employee can execute only one task at a time.

A solution to the GTSGP is feasible, if the following five hard constraints have no violations:

(H1) The tasks in the shift do not overlap in time.
(H2) Some tasks may have precedence constraints, that is, a task may not be executed after some other tasks in the same shift.
(H3) Each employee can execute at least one shift, i.e. C1-C4 hold.
(H4) Each shift can be executed (C1-C4 hold) by one or more employees.
(H5) One or more shifts is assigned to each employee AND one or more employees to each shift.

Lunch and other breaks can also be created using the idea given in [19]. To evaluate the hard constraint H5, we have to solve the corresponding assignment problem. Note, that the criterion H5 actually includes the criteria H3 and H4. Figure 1 shows a solution to an assignment problem with six shifts and six employees. The corresponding assignment problem would have no solution, if employee A could not execute shift 5, even though criteria H3 and H4 would still hold.

	Shift 1	Shift 2	Shift 3	Shift 4	Shift 5	Shift 6	
Emp A	1100	1001	0110	1111	1111 x	0101	2
Emp B	1110	1101	1011	0110	1010	1111 x	1
Emp C	0000	1111 x	0000	0000	1110	1111	2
Emp D	1111 x	1111	1111	0100	0100	1111	4
Emp E	0011	0111	0000	1111 x	1101	1111	2
Emp F	1100	0011	1111 x	0101	0111	1111	2
	1	2	2	2	1	5	Total 13

Fig. 1. An assignment problem with six shifts and six employees. The cells indicate the values for criteria C1-C4 (1 = criterion holds). An employee

can execute the shift if all the cell values are one. A solution to the assignment problem is denoted with x.

The GTSGP can now be stated as follows:

1) Maximize the sum of number of shifts that could be assigned to each employee, over all employees
2) Satisfy the hard constraints H1-H5.

The number of shifts employees are able to execute in the example given in Figure 1 is 13. Note, that in the GTSGP, the shift structure is implicitly generated from the skills, working times, availabilities and transition times of the employees (criteria C1-C4).

The GTSGP can also include the same soft constraints as for the employee-based shift generation problem (see [19]), for example the following:

(S1) Shifts of less than k_1 and over k_2 timeslots in length must be minimized.
(S2) The average shift length should be as close to k_3 timeslots as possible.
(S3) Shifts that start between timeslots k_4 and k_5 must be minimized.
(S4) Shifts that end between timeslots k_6 and k_7 must be minimized.
(S5) Each shift should contain at most k_8 switches from one task to another.

IV. Solution Method

The search space of the GTSGP is enormously larger than that of the SMPTSP. For example, consider an instance with ten shifts and with one hundred tasks each having a duration of ten timeslots and a time window of nineteen timeslots. In the SMPTSP, we have ten possible assignments for each task totaling 10^{100} solution candidates. In the GTSGP, however, we have 10 x 20 possible assignments for each task totaling 200^{100} solution candidates, i.e. 20^{100} times more candidates.

We solve the GTSGP using the PEAST algorithm described in [26]. The algorithm is a population-based metaheuristic. The acronym PEAST stems from the methods used: Population, Ejection, Annealing, Shuffling and Tabu. It has been used in staff rostering [27], employee-based shift generation [19], professional sports league scheduling [28] and school timetabling problems [29]. Furthermore, the algorithm has been used to solve somewhat more academic problems, such as balanced incomplete block design [30], single round robin tournaments with balanced home-away assignments and pre-assignments [30] and constraint minimum break problems [31].

The heart of the PEAST is the local search called GHCM, which is used to explore promising areas in the search space. Another important feature of the algorithm is the use of shuffling operators, which assist in escaping from local optima. Furthermore, simulated annealing and tabu search are used to avoid staying stuck in promising search areas too long. The algorithm uses ADAGEN, the adaptive genetic penalty method, which assigns dynamic weights to the hard constraints based on the constant weights assigned to the soft constraints. For the detailed discussion of the algorithm, we refer to [26] and [32]. The pseudo-code of the algorithm is given in Figure 2.

```
Set the iteration limit t, cloning interval c, shuffling interval s, ADAGEN
update interval a and the population size n
Generate a random initial population of schedules S_i for 1 <= i <= n
Set best_sol = null, round = 1
WHILE round ≤ t
   index = 1
   WHILE index++ <= n
      Apply GHCM to schedule S_index to get a new schedule
      IF Cost(S_index) < Cost(best_sol) THEN Set best_sol = S_index
   END REPEAT
   Update simulated annealing framework
   IF round ≡ 0 (mod a) THEN Update the ADAGEN framework
   IF round ≡ 0 (mod s) THEN Apply shuffling operators
   IF round ≡ 0 (mod c) THEN Replace the worst schedule with the
                              best one
   Set round = round + 1
END WHILE
Output best_sol
```

Fig. 2. The pseudo-code of the PEAST algorithm.

In the GHCM search the basic hill-climbing step is extended to generate a sequence of moves in one step, leading from one solution candidate to another. In the GTSGP, the GHCM search moves a task t_1, from its currents shift s_1, to a new shift s_2, and then moves another task t_2, from shift s_2 to a new shift s_3, and so on, ending up with a sequence of moves. The first task is selected by tournament selection. The shift and the starting timeslot that receives the task is selected by considering all the possible shifts and in those shifts all the starting timeslots that are not booked, and selecting the one that causes the least increase in the cost function. Then, a task from that shift is selected by considering all the tasks in that shift and picking the one for which the removal causes the most decrease in the cost function. Next, a new shift for that task is selected, and so on. The sequence of moves stops if the last move causes an increase in the cost function value and if the value is larger than that of the previous non-improving move, or if the maximum number of moves is reached. Then, a new sequence of moves is started.

Due to the complexity issues and the very large search space of the GTSGP, we have to consider five calculation key points when solving real-world instances. First, whenever possible, we should reduce the number of tasks by grouping or sequencing smaller tasks into bigger single tasks. Second, we should similarly group skills to larger skill groups. This is not as vital as with the number of tasks. Third, without losing too much important information, the slot size should as long as possible, i.e. the number of timeslots should be as small as possible.

Fourth, when generating a sequence of moves, we have to calculate the cost function many times during one GHCM operation. Furthermore, we have to calculate the cost function while rollbacking the moves, often down to the starting point. The computational resources are too high to calculate the solution value. Therefore, we should recalculate only those parts of the solution, which are changed due to the single moves of the move sequence. This

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

is very tough to implement, but it is vital for real-world use of the PEAST algorithm.

Finally, to check out the hard constraint H5, we have to solve the assignment problem as described in Section III. The problem was originally solved in $O(n^4)$ time, but it can be solved in $O(n^3)$ time using appropriate data structures [33]. Unfortunately, this is still far too slow since we have to solve the problem in each single move in the move sequence. One possibility would be to use a greedy heuristic, but it does not guarantee that we can generate such shifts that the staff rostering can be completed. Fortunately, implementing the move sequence in such a way that we can apply the ideas presented in [34], we are able to calculate only the changes incurred to the initial assignment problem. In our implementation the calculation of changes requires $O(n^2)$.

V. First Computational Results

In this section we present our first computational results. We first try to solve some of the SMPTSP benchmark instances and then we present one GTSGP benchmark instance. Real-world benchmark instances for SMPTSP do not exist at the moment, but three artificial benchmark instances have been published.

Krishnamoorthy et al. [21] presented a data set of 137 instances for the SMPTSP. The data set is referred to as KEB instances. Smet et al. [35] generated ten more difficult instances than KEB instances, referred to as SWMB instances. Furthermore, Fages and Lapegue [36] generated a new data set of 100 instances, because the KEB and SWMB instances are trivial with regard to finding good quality lower bounds. This data set is referred to as FL instances. A good summary of the instances and an excellent greedy algorithm for the SMPTSP can be found in [37].

It is obvious, that the PEAST algorithm designed for the GTSGP cannot compete with the specifically tailored SMPTSP algorithms described in [9], [21], [24], [25], [35] and [37]. However, as a first test, we decided to select seven instances from the KEB data set. The instances were selected to cover different task sizes and employee sizes and their ratio. In addition, four of the instances are such that the heuristic presented in [21] were not able to solve to optimality. Table 1 shows the test instances and our first test runs. We were able to solve six of the seven instances.

TABLE I
Test results for the seven KEB instances

KEB	#Tasks	#Emps	Optimum	Heuristic	PEAST
1	40	23	20	20	20
9	104	49	40	41	40
26	203	116	100	100	100
27	204	49	40	40	40
39	351	45	40	41	40
63	577	97	80	82	80
75	665	72	60	71	65

KEB = KEB instance id, *#Tasks* = number of tasks, *#Emps* = number of employees, *Optimum* = optimum value, *Heuristic* = the value obtained using the heuristic in [21], *PEAST* =the value obtained using the PEAST

Next, we present one GTSGP benchmark instance, which introduces all the features and hard constraints of the problem described in Section III. We have implemented a GTSGP test generator, which we used to generate the test instance. Test instances are generated in a way that at least one solution with no hard constraint violations exists. The instance was created using the following settings:

- Number of task type precedence constraints: 13
- Minimum (max) number of skills in task types: 5 (10)
- Average number of employees fit for a shift: 3
- Average gap of employees' minimum and maximum working time: 20%
- Average difference of shift durations: 20%
- Percentage of backup-office tasks: 0%
- Percentage of fixed tasks: 50%
- Average task window deviation: 50%
- Probability of a location change between the tasks in the same shift: 50%
- Number of transition timeslots required between the tasks in the same shift: 0, 1 or 2.

Table 2 shows the characteristics of the test instance. The data for the instance is available online [38]. We were able to find a solution with no hard constraint violations (see Figure 3).

TABLE II
Characteristics of the test instance

Timeslots	20	Min (Max) number skills of employees (see C1)	20 (29)
Tasks	100	Min (Max) working time limits of employees (see C2)	11-13 (18-20)
Shifts	20	Min (Max) employee available timeslots (see C3)	13 (20)
Employees	20	Min (Max) number of employee transition times of length 2 (see C4)	18 (44)
Task types	10	Min (Max) number of skills related to task types	5 (10)
Precedences	13	Min (Max) time window length of tasks	1 (17)
Skills	30	Min (Max) duration of tasks	1 (16)
Locations	11		

```
 1: 53(3) 44(6) 76(8) 75(11) 26(12) 11(15) 77(16)
 2: 50(1) 85(12)
 3: 72(1) 28(5)
 4: 32(3) 33(15) 94(19)
 5: 60(4) 61(6) 98(16)
 6: 1(1) 80(4) 81(6) 3(8) 37(10) 17(13) 62(14) 93(16) 92(19)
 7: 19(2) 95(3) 96(6) 97(10)
 8: 6(3) 7(5) 8(8) 10(9) 15(11) 18(14) 16(16)
 9: 99(1) 23(6) 24(8) 49(10) 88(12) 46(13) 48(16) 78(17)
10: 5(1) 2(4) 52(13) 29(19)
11: 13(1) 73(5) 84(7) 34(18)
12: 27(3) 14(5) 43(6) 36(9) 66(10) 67(12) 51(13) 57(14)
13: 58(1) 64(4) 65(9) 45(10) 90(11) 68(13) 69(14) 70(18)
14: 42(3) 21(4) 22(5) 35(6) 38(14) 55(16) 40(17) 63(18)
15: 30(1) 31(2) 20(3) 89(4) 54(5)
16: 79(2) 12(4) 91(5) 87(7) 9(9) 71(11) 74(12) 4(14)
17: 25(3) 100(5)
18: 82(8) 83(16) 39(18)
19: 56(2) 47(14) 86(16)
20: 59(4) 41(20)
```

Fig. 3. A solution with no hard constraint violations.
Numbers in parentheses denote the starting timeslot of task.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

VI. Conclusion

We presented the General Task-based Shift Generation Problem (GTSGP). To the best of our knowledge, the problem has not been studied in the literature. We briefly described the PEAST algorithm, which was used to solve the presented test instances. We first solved seven SMPTSP instances, which are very special cases of GTSGP. Then we solved one GTSGP instance, which introduced all the features and hard constraints of the GTSGP. The computational results were encouraging.

The PEAST algorithm for staff rostering has been integrated into Visma Numeron WFM market-leading workforce management software in Finland. This research has contributed to better systems for our industry partners. We are currently working on integrating the shift generation and the GTSGP to the WFM software.

Our near future direction is to apply the PEAST algorithm to all the KEB, SWMP and FL instances, and to use the GTSGP test generator to create a large set of benchmark instances for the GTSGP.

References

[1] M.R. Garey and D.S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness,” Freeman, 1979.

[2] J. Tien and A. Kamiyama, “On Manpower Scheduling Algorithms,” in SIAM Rev. 24 (3), pp. 275–287, 1982.

[3] H.C. Lau, “On the Complexity of Manpower Shift Scheduling,” Computers and Operations Research 23(1), pp. 93-102, 1996.

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

[5] P. Bruecker, R. Qu and E. Burke, “Personnel scheduling: Models and complexity”, European Journal of Operational Research 210 (3), pp. 467-473, 2011.

[6] W.K. Jackson, W.S. Havens and H. Dollard, “Staff scheduling: A simple approach that worked”, Technical Report CMPT97-23, School of Computing Science, Simon Fraser University, Canada, 1997.

[7] T. Lapegue, O. Bellenguez-Morineau and D. Prot, “A constraint-based approach for the Shift Design Personnel Task Scheduling Problem with Equity”, Computers and Operations Research 40 (10), pp. 2450-2465, 2013.

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

[9] D. Prot, T. Lapgue and O. Bellenguez-Morineau, “A two-base method for the shift design and personnel task scheduling problem with equity objective”, International Journal of Production Research 53 (24), pp. 7286-7298, 2015.

[10] P. Smet, A.T. Ernst and G. Vanden Berghe, “Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem”, Computers & Operations Research 76, pp. 60-72, 2016.

[11] 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 153 (1), pp. 3-27, 2004.

[12] N. Musliu, A. Schaerf and W. Slany, “Local search for shift design,” European Journal of Operational Research, 153(1), pp. 51–64, 2004.

[13] L. Di Gaspero, J. Gärtner, N. Musliu, A. Schaerf, W. Schafhauser and W. Slany, “Automated Shift Design and Break Scheduling”, In: Uyar A., Ozcan E., Urquhart N. (eds) Automated Scheduling and Planning. Studies in Computational Intelligence, vol. 505, Springer, Berlin, Heidelberg, 2013.

[14] 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 226 (3), pp 367-385, 2013.

[15] L. Kletzander and N. Musliu, “Solving the General Employee Scheduling Problem”, Preprint submitted to Computers and Operations Research, Vienna, Austria, 2018.

[16] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf and W. Slany, “The minimum shift design problem,” Annals of Operations Research, 155(1), pp. 79–105, 2007.

[17] S. Bhulai, G. Koole and A. Pot, "Simple Methods for Shift Scheduling in Multiskill Call Centers", Manu-facturing and Service Operations Management 10 (3), pp. 411-420, 2008.

[18] N. Kyngäs, D. Goossens, K. Nurmi and J. Kyngäs, “Optimizing the unlimited shift generation problem”, Applications of Evolutionary Computation: EvoApplications, Springer, pp. 508-518, 2012.

[19] N. Kyngäs, K. Nurmi and J. Kyngäs, “Solving the person-based multitask shift generation problem with breaks”, In Proc. of the 5th International Conference On Modeling, Simulation And Applied Optimization, Hammamet, Tunis, pp. 1-8, 2013.

[20] M. Krishnamoorthy, A.T. Ernst and D. Baatar, “The personnel task scheduling problem”, Optimization Methods and Applications, pp. 343–367, 2001.

[21] M. Krishnamoorthy and A.T. Ernst, “Algorithms for large scale Shift Minimisation Personnel Task Scheduling Problems”, European Journal of Operational Research, 219 (1), pp. 34-48, 2012.

[22] 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 90, pp. 285-302, 1996.

[23] A.W.J. Kolen, J.K. Lenstra, C.H. Papadimitriou and F.C.R. Spieksma, “Interval Scheduling: A Survey”, Naval Research Logistics 54 (5), pp. 530-543, 2007.

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

[25] Lapègue, T., Prot, D., Bellenguez-Morineau, O.: A constraint-based approach for the Shift Design Personnel Task Scheduling Problem with Equity. Computers and Operations Research 10(40), 2450–2465 (2013)

[26] N. Kyngäs, K. Nurmi and J. Kyngäs, “Crucial Components of the PEAST Algorithm in Solving Real-World Scheduling Problems”, Journal of Lecture Notes on Software Engineering 1(3), pp. 230-236, 2013.

[27] N. Kyngäs, K. Nurmi and J. Kyngäs, “Workforce Scheduling Using the PEAST algorithm”, in Ao, Sio-Iong (ed.): IAENG Transactions on Engineering Technologies, Lecture Notes in Electrical Engineering Volume 275, Springer, USA, 2014, pp 359-372,

[28] K. Nurmi, J. Kyngäs and A.I. Järvelä, “Ten-year Evolution and the Experiments in Scheduling a Major Ice Hockey League”, in Daniel Hak (ed.): An in Depth Guide to Sports, pp 169-207, Nova Science Publishers, USA, 2018.

[29] K. Nurmi and J. Kyngäs, ”A Conversion Scheme for Turning a Curriculum-based Timetabling Problem into a School Timetabling Problem” in Proc of the 7th Conference on the Practice and Theory of Automated Timetabling (PATAT), Montreal, Canada, 2008.

[30] K. Nurmi, D. Goossens and J. Kyngäs, “Scheduling a Triple Round Robin Tournament with Minitournaments for the Finnish National Youth Ice Hockey League”, Journal of the Operational Research Society, Vol. 65(11), 2014, pp. 1770-1779,

[31] K. Nurmi, D. Goossens, T. Bartsch, F. Bonomo, D. Briskorn, G. Duran, J. Kyngäs, J. Marenco, C.C. Ribeiro, F. Spieksma, S. Urrutia and R. Wolf-Yadlin, “A Framework for Scheduling Professional Sports Leagues”, in Ao, Sio-Iong (Eds.). IAENG Transactions on Engineering Technologies 5, Springer, USA, 2010, pp. 14-28.

[32] K. Nurmi, “Genetic Algorithms for Timetabling and Traveling Salesman Problems”, Ph.D. dissertation, Dept. of Applied Math., University of Turku, Finland, 1998.

[33] C.H. Papadimitriou and K. Steiglitz, “Combinatorial Optimization: Algorithms and Complexity”, Dover Publications, 1998.

[34] I.H. Toroslu and G. Ucoluk, “Incremental assignment problem”, Information Sciences 177(6), pp. 1523-1529, 2007.

[35] 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 46, pp. 64-73, 2014.

[36] 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, 8124, pp. 63-79, 2013.

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

[38] K. Nurmi: “The General Task-based Shift Generation Problem – Benchmark Instances” [Online]. Available: http://web.samk.fi/public/tkiy/GTSGP/, (Last access December 2018).