A Simulated Annealing Approach to the Multi-Activity Multi-Day Shift Scheduling Problem

László Kálmán Trautsch[1] [0000−0002−1589−3265] and Bence Kovari[2] [0000−0003−1555−640X]

[1] Department of Automation and Applied Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary. trautschl@edu.bme.hu

[2] Department of Automation and Applied Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary. kovari@aut.bme.hu

Abstract. This paper addresses the multi-activity multi-day shift scheduling problem with a homogeneous workforce and quadratic cost function for overstaffing. The objective of this problem is to assign shifts to employees and activities within these shifts based on short time intervals, respecting numerous hard constraints and minimizing overstaffing. We propose a multi-neighborhood Simulated Annealing algorithm as a solution method, for which we introduce eight neighborhood relations. The search space and neighborhood relations are designed so that the search algorithm can be executed efficiently even on large problem instances. The method is evaluated on a benchmark dataset consisting of problem instances with varying complexity. The results show that our approach can handle even the most complex tasks and is able to find feasible solutions for 201 out of the 225 total problem instances, of which 99 were previously unsolved. Our method outperforms the solver that produced the previous best known solutions for the benchmark dataset and finds new best solutions for 190 of the instances. The algorithm can create good schedules in a matter of a few seconds, using limited computing resources.

Keywords: Shift Scheduling, Multi-Activity, Simulated Annealing

1 Introduction

The multi-activity shift scheduling problem occurs in many industries, particularly in the service sector, commonly in retail environments. Making effective use of the workforce while satisfying various organizational and social constraints is an important task that could yield substantial cost savings. In these problems, an activity represents an interruptible operation, which can be assigned to several employees at the same time. There is a minimum required workforce for the activities at each period, to ensure an acceptable service quality. This demand may fluctuate throughout the planning horizon.

Personnel scheduling has been a widely studied problem in the literature for a long time, as shown by the numerous references given in the surveys of Ernst et al. [1, 2]. On the other hand, multi-activity shift scheduling problems were relatively under-studied until recently. One of the earliest works addressing this problem was by Loucks and

Jacobs [3]. Since then, different versions of the problem emerged with vastly different constraints, and various solving methods have been employed for solving these. Some works consider anonymous shifts, where it is not specified which employee will be assigned to a given shift. Dahmen, Rekik and Soumis [4] proposed an implicit model for this task. Other works address a variation of the problem in which interruptible activities and uninterruptible tasks should be scheduled at the same time. Lequy, Desaulniers and Solomon [5] used a two-stage heuristic for this problem. Solving shift construction and activity assignment simultaneously on a multi-day planning horizon is a challenging task, which to the best of our knowledge, has been addressed only by a few papers. The qualification to perform certain activities can also differ within the workforce in some problems, such as in the work of Dahmen and Rekik [6], where they proposed a hybrid heuristic for solving a multi-activity multi-day shift scheduling problem with a heterogenous workforce. Most recently a mathematical programming-based approach has been used for variants of multi-activity shift scheduling problems with anonymous shifts by Römer [7], who proposed block-based state-expanded network models.

This paper addresses the multi-activity shift scheduling problem with a homogeneous workforce in a multi-day environment as described in the formal description [8] of the associated benchmark problem [9]. The task is to assign shifts to employees on the given days, and to schedule the shifts and the activities within them, based on short time intervals, in a way that respects all the various hard constraints. The cost function in this problem is the quadratic penalization for overstaffing at each period for every activity. There is one existing work addressing this exact problem, by Qu and Curtois [10], in which they use Variable Neighborhood Search as a solution method.

We propose a multi-neighborhood Simulated Annealing approach for this problem. Simulated Annealing was first introduced by Kirkpatrick, Gelatt and Vecchi [11], and since then it has been successfully applied for many scheduling tasks in the literature, such as for sports timetabling [12], nurse rostering [13], course timetabling [14, 15] and most recently examination timetabling [16]. Our proposed approach for the multi-activity multi-day shift scheduling problem is based on a mathematical model which enables the efficient inspection of the various hard and soft constraints, and our introduced eight different neighborhood relations allow for an effective traversal of the state space.

The organization of this paper is as follows. Section 2 overviews the multi-activity multi-day shift scheduling problem addressed in this paper. Section 3 describes the proposed local search method and the proposed neighborhood relations in detail. In Section 4, we report and discuss the experimental results obtained on the benchmark dataset. Section 5 provides concluding remarks and future plans.

2 Problem Definition

This paper addresses the Multi-Activity Multi-Day Shift Scheduling Problem, as described in the formal description [8]. The mathematical model of the problem with the formalization of the exact hard and soft constraints are available in the formal description. For clarity, we briefly summarize the key aspects of the problem. The goal is to assign shifts to employees on the given days, and activities within these shifts. An employee can work on one or more tasks during a shift, therefore activities should be scheduled within the shifts, each with an assigned task. In this context, activities and tasks are considered equivalent. Therefore, when we refer to an activity, we are indicating the duration during which an employee works on one of the specified tasks. An employee must work on exactly one task at each interval of a shift, which means that there can be no overlap between the different activities. Employees are assumed to be homogeneous in the sense that they are all qualified to perform any of the different tasks. The planning horizon is divided into 15-minute intervals and the scheduling has to be done based on these time slots. The planning horizon always starts at 6:00 a.m. on the first day and finishes at 6:00 a.m. on the last day. Therefore, if the planning horizon is 7 days long, then it runs from 6:00 a.m. on day 1 to 6:00 a.m. on day 8.

There are various hard constraints for the problem, all of which must be respected for a schedule to be considered feasible. An employee cannot start more than one shift on a day, and a shift can only start at one of the time intervals between the following times on each day: 0:00-0:00, 6:00-10:00, 14:00-18:00 and 20:00-23:45. Each shift duration should be between 6 and 10 hours. After a shift finishes, an employee cannot start another shift until at least 14 hours later. An employee cannot start shifts on more than 5 consecutive days, in other words at least one day off must be taken on each 6 consecutive days. There are no limitations on the number of activities a shift can hold or on the number of activity changes within a shift, however, every activity must be at least 1 hour long before an activity change occurs or the shift ends.

In the different problem instances, it is specified for each employee how many total minutes that employee should work at minimum and at maximum during the whole planning horizon. The minimum cover requirement is also specified for each task at each time interval, which is the minimum number of required staff to work on that task at that interval.

The objective is to minimize assigning more staff than the maximum specified for each task at each time interval. When there is overstaffing for a given task at a specific interval, the penalty is the squared difference between the maximum required number of staff and the actual number of staff. The total cost of a solution is the sum of all the penalties for every time interval and task. Thus, the cost function is quadratic to ensure that overstaffing is spread out over the planning horizon rather than occurring in a small number of tasks and intervals, as the penalty for each additional unit of overstaffing for a task at an interval increases more rapidly than linearly.

3 Solution Method

Our solution method is based on the Simulated Annealing [11] local search, for which we designed a multi-neighborhood consisting of eight different neighborhoods. The key components of the proposed method are described in this section.

3.1 Search Space

A state in the search space is the direct representation of all the shifts and activities assigned to each employee, with their respective schedules based on the time intervals

Table 1: Decision variables

Symbol	**Definition**
$s_{h,e} \in \{0, \ldots, \|H\| * \|E\| - 1\}$	Shift index of employee e on the h-th 24-hour period of the planning horizon. $\|H\|$ is the total number of 24-hour periods and $\|E\|$ is the total number of employees in the given problem instance.
$b_s \in \mathbb{Z}^+$	Start time of shift s.
$a_s \in \{0, \ldots, n\}$	Activity count of shift s. The maximum number of activities per shift (n) is a selectable parameter.
$l_{s,a} \in \mathbb{Z}^+$	Length of the a-th activity of shift s.
$t_{s,a} \in \{0, \ldots, \|T\|\}$	Task of the a-th activity of shift s. $\|T\|$ is the total number of tasks in the given problem instance.
Auxiliary variables	
$w_e \in \mathbb{Z}^+$	Workload of employee e.
$c_{t,i} \in \mathbb{Z}^+$	Cover of task t at time interval i.
$f_{d,e} \in \{0, 1, 2\}$	Count of non-empty shifts of employee e on day d.

of the planning horizon. The decision variables of our model with their descriptions are shown in Table 1.

Although there are demand requirements for each time interval of the planning horizon, we do not directly model the assignment of employees at each interval, as this would imply an unnecessarily large model for our approach, because only the sum of the workforce is relevant at each interval. Rather, we use variables to specify which workers are assigned to which shifts at the given 24-hour periods, when these shifts start, how many activities the shifts contain, how long these activities are, and which tasks are assigned to them. Given these variables, a complete schedule can be composed, and all the different constraints can be inspected. To enable the efficient inspection of the various constraints, different auxiliary variables can be introduced, offering the necessary aggregated information directly. Our key auxiliary variables are presented in Table 1.

In our model, each employee must have one shift assigned to them at each 24-hour period, but a shift can be empty meaning that it should be ignored from the complete schedule and the relevant employee does not start a working shift at that period. A shift must start at one of the time intervals of the relevant 24-hour period, but it can extend beyond that period. We base our shifts on 24-hour periods of the planning horizon rather than on days, because this way fewer variables are needed for modeling the shifts, and all the shifts can be handled uniformly. If shifts were created for each day, then the start and the end of the planning horizon would cut into the shifts on the first and the last day respectively, making allocation to time intervals completely different on those days. The first 24-hour period starts at the beginning of the planning horizon, which is at 6:00 a.m. on the first day, and ends on the next day at 6:00 a.m. The number of 24-hour periods is one less than the number of days in a problem instance.

The problem constraints can be reformulated for our decision variables with a straightforward matching between them. The only difficulty emerges because a shift can start at 0:00 on a given day, which time interval is part of the 24-hour period of the previous day. Thus, two shifts could start on the same day, which is a constraint violation. In order to restrict this, we modified the length of the minimum rest time to 24 hours for a shift starting at 0:00. The auxiliary variables for the number of non-empty shifts of each employee on each day are created to help the inspection of the constraint on the number of consecutive working shifts. An example of two shifts of an employee starting on the same day would be that the first shift starts at 0:00 and the second one starts at 20:00. In that case the corresponding "$f_{d,e}$" value would be 2, indicating that two shifts start on that day for the employee.

To make the search space more connected, certain hard constraints are relaxed and made soft constraints, but with high weights applied to the cost for violating them. Thus, the cost function of a state in the search space is the sum of the cost induced by the soft and the hard constraints. The actual weights of the hard constraints are set by parameters associated with them. The workload constraints, the minimum cover constraint, the maximum number of consecutive shifts constraint, the minimum rest time constraint, and the constraints regarding the length of shifts and activities are relaxed. Linear cost functions are used, except for the minimum and maximum shift length constraints, for which the deviation from the minimum and maximum length is penalized quadratically.

A parameter can be set to control the maximum number of activities per shift. The variables associated with the activities are created based on this parameter, for each shift as many as the parameter specifies. Based on the maximum shift length and minimum activity length hard constraints, at maximum 10 activities per shift are needed to create any feasible solution. A higher value can be set for this parameter if we want the search algorithm to move more freely in the search space by adding more activities, but the problem constraints need to be modified in this case. A value lower than 10 can speed up the search for problem instances with few tasks, although finding the optimal solution might become theoretically impossible.

The activity count variable specifies how many activities are actually relevant from all the activities of a shift. When the activity count of a shift is lower than the maximum number of activities per shift, that means that the following activities are empty, and their tasks and lengths should be ignored. A shift is empty when its activity count is zero.

3.2 Initial Solution

For the initial solution of the search, an empty schedule is created based on the number of days and employees of the given problem instance, where each employee has an empty shift assigned to them on each 24-hour period of the planning horizon. The activity count of each shift is zero, which means that the other decision variables associated with the shift are irrelevant until an activity is assigned by a move from the neighborhood relations. The auxiliary variables also have zero values in this initial state. The solution is infeasible, and the total cost is calculated and used as the initial cost. This empty schedule is used as the initial state, which is populated with working shifts by the neighborhood relations during the search.

3.3 Neighborhood Relations

We propose a multi-neighborhood on the previously described search space, composed of the union of eight neighborhoods:

- **AddShiftActivity:** A random empty shift is selected, and an activity is added to it with a random task. A random start is also assigned to the shift and a random length to the activity. The start of the shift is selected from the valid shift start times of the day. The length is selected from the possible shift lengths that do not extend beyond the planning horizon, given the already selected shift start time.
- **RemoveShiftActivities:** A random non-empty shift is selected, and all its activities are removed.
- **ChangeShiftStart:** A random non-empty shift is selected, and its start is changed to a different, random start. The start is selected from those valid shift start times of the day that would not make the shift extend beyond the end of the planning horizon, given its current length.
- **SwapShifts:** A random non-empty shift is selected, and its assignment is swapped between its original employee and the employee of an other random shift from the same day. The other shift can be either empty or not.
- **ChangeActivityLength:** A random non-empty activity is selected, and its length is changed to a different, random length. The length is selected so that the shift would not extend beyond the planning horizon, and the length of the shift up to the end of the selected activity would not be longer than the maximum shift length and shorter than the minimum shift length. The minimum activity length is also respected when the previous criteria enable it.
- **ChangeActivityTask:** A random non-empty activity is selected, and its task is changed to a different, random task.
- **AddLastActivity:** A random non-empty and non-full shift is selected, and a new activity is added to its end with a random task, which is different than the task of the previous activity. A random length is also assigned to the activity, and it is selected so that the shift would not extend beyond the planning horizon, and the shift would not be longer than the maximum shift length. When the previous criteria enable it, the minimum activity length is also respected.
- **RemoveLastActivity:** A random shift is selected from the shifts that have at least two activities, and the last activity of that shift is removed.

At each iteration step of the search, one of the eight neighborhood types is selected with probabilities specified by associated parameters, then a move is randomly drawn from the selected neighborhood. When the random selection of a variable cannot be made during a move, the move is instantly rejected. For example, if there are no shifts with at least one activity assigned to them, then moves from the *RemoveShiftActivities* neighborhood are rejected.

3.4 Simulated Annealing

As the metaheuristic to guide the search, we implemented the Simulated Annealing algorithm [11]. The method starts from an initial random state and at each iteration selects a random move from its neighborhood as explained above. Calling Δf the change in cost induced by the selected move, the move is always accepted if $\Delta f \leq 0$, and it is accepted with probability (1) when $\Delta f > 0$, where T_a is the temperature parameter controlled by the algorithm.

$$e^{-\Delta f / T_a} \tag{1}$$

We implemented the Fast Simulated Annealing [17] cooling scheme to determine the temperature at each iteration, based on the number of the current iteration (t) and the initial temperature (T_0), as described by equation (2).

$$T_a(t) = \frac{T_0}{(1+t)} \tag{2}$$

The search is repeated for a set number of iterations, which number is a parameter of the metaheuristic.

3.5 Efficient Implementation

The neighborhood relations and decision variables were designed to enable efficient implementation of the search algorithm, so that a high number of iterations could be executed even on large problem instances. The change in cost induced by new candidate moves should be calculated only based on the constraints and variables directly relevant to the actual neighborhood type and the exact move, and the state should be modified only if the move is accepted. The proposed auxiliary variables are used for inspecting the relevant constraints of the actual neighborhood relation. Shift indexes were introduced for achieving low computational complexity when executing a *SwapShifts* move between two employees. We also used other auxiliary variables and structures to help select random variables and calculate the changes in cost during the search, but these are not reported in this paper for the sake of brevity.

4 Experimental Results

4.1 Problem Instances

The algorithm was tested on the instances of the publicly available multi-activity shift scheduling benchmark dataset [9]. The benchmark contains 225 different problem instances, with varying difficulty. There are instances with lengths of 7, 14, and 28 days. The number of staff varies from 10 to 150, and the number of tasks varies from 1 to 19. The problem size tends to increase with the instances. The features of the instances are shown in Table 4. in the Appendix. It is known that every instance has a feasible solution, due to the way the instances were created [10].

4.2 Parameter Settings

A single parameter configuration was tested during the experiments on the different problem instances, which is shown in Table 2. The table includes the parameters for the Simulated Annealing metaheuristic and the weights assigned to the various hard constraints.

Table 2. Parameter configuration

Parameter	Assigned value
Initial temperature	800,000
Number of iterations	10,000,000
Maximum number of activities per shift	10
Each neighborhood relation probability	0.125
Weight for minimum rest time constraint	15,000,000
Weight for minimum workload constraint	1,500,000
Weight for maximum number of consecutive working days constraint	1,000,000
Weight for shift length constraint	225,000
Weight for maximum workload constraint	150,000
Weight for minimum activity length constraint	150,000
Weight for minimum activity cover demand constraint	10,000

The hard constraint weights are based on the problem instance files in XML format found in the benchmark dataset, except for the weight for violating the minimum rest time, for which we assigned a weight higher than the others. The neighborhood relation probabilities were selected uniformly. The number of iterations was set so that the runtime of the search on even the hardest problem instance would take no longer than 5 seconds. The initial temperature was chosen intuitively, based on trial runs on the hardest problem instance. It is important to note that the presented method could significantly benefit from parameter tuning, and employing a different cooling scheme or stopping criterion might further improve the results.

4.3 Experimental Setup

The solution method was implemented in C++ and compiled using g++. The experiments were run on a machine with 16 GB of RAM and a 3.3 GHz Intel Core i5-4590 processor, using a single core during the tests. The number of iterations was selected so that the search on each instance would take no longer than 5 seconds. A single run of our solution method was performed on each problem instance of the dataset.

4.4 Results

The results of our solution method on each problem instance are shown in Table 4. in the Appendix. We compare our results achieved by Simulated Annealing (SA) to the solver that produced the existing best known solutions for the benchmark dataset, the method by Qu and Curtois [10], which uses Variable Neighbourhood Search (VNS). The best result found for a problem instance is highlighted in bold and underlined. Only feasible solutions are reported, in which none of the hard constraints are violated. A cell contains "-" if no feasible solution was found by a method under its time limit.

The authors of the VNS method used a time limit of 10 minutes for their experiments on each instance, and they conducted their tests on a comparable machine (Intel Core i5-4690K CPU 3.50GHz) to the one used in our tests.

For our solution method, we selected the number of iterations so that the runtime of the search on each instance would not take longer than 5 seconds. The actual runtimes ranged from 2.079 seconds on the simplest instance to 4.073 seconds on the most complex one. It should be noted that the runtime scales well with the problem complexity when using a fixed number of iterations for the search. Less than twice as much time was needed for a problem instance with a four times longer planning period, fifteen times as many staff, and nineteen times as many tasks, and it should be also considered that the simplest instances had one task to be scheduled, meaning that moves from three neighborhood types were always rejected in those cases, reducing the runtime. The memory usage of the algorithm was also efficient, less than 4 MB of memory was needed during the searches.

Table 3. shows an overview of the results on the benchmark dataset, comparing our approach to the VNS method. Simulated Annealing was able to find feasible solutions for most of the problem instances (201 out of the 225 total), of which 190 are the best solutions found so far. It was able to find feasible solutions for many previously unsolved problem instances, even for the most complex ones.

Table 3. Overview of the comparative results on the benchmark dataset

Time limit	**VNS [10]** 10 minutes	**SA** 5 seconds
Number of instances for which feasible solutions are found	107	<u>**201**</u>
Number of instances for which best solutions are found	16	<u>**190**</u>

On the other hand, our approach could not find feasible solutions for some simpler instances, 5 of which have been solved by VNS. The size of a problem instance does not seem to affect finding a feasible solution for the Simulated Annealing. The search randomly gets stuck in an infeasible local optimum in some cases, which could be due to the selected cooling scheme or the parameters of the metaheuristic. Only a single parameter configuration with equal probabilities for all neighborhood types was tested, which implies that applying parameter tuning could greatly improve the results. Improving the neighborhood relations and introducing new ones could also help escaping the local optima, and different selection of weights for the hard constraints should also be inspected. A single search was conducted on each problem instance, but the method could benefit from selecting the best solution from more searches, run both in parallel and by applying restarts.

The reported solutions were all validated using a verification software, which is available for the benchmark dataset [9]. The software can be used to view and verify the solutions created for the problem instances. It is able to identify any hard constraint violations and calculate the cost function of a complete schedule. The accuracy of our new computational results was ensured by using this validation.

5 Conclusions and Future Work

In this paper, we presented a multi-neighborhood Simulated Annealing method for addressing a multi-activity multi-day shift scheduling problem. We introduced eight different neighborhood relations for the search algorithm. We tested our approach on a benchmark dataset and compared the results with the best existing solution available for the problem. Our method was able to outperform the previously developed algorithm on most of the problem instances and was able to find feasible solutions for many of the unsolved ones. The results show that our approach can produce good schedules in a matter of a few seconds, using limited computing resources. The method would be able to find solutions for even larger problems than the most complex instances of the benchmark dataset, as the results suggest. However, in some cases, it was not able to produce feasible solutions even for some simpler instances, therefore there is still room for improvement.

As part of our future work, first, we will investigate the possibility of improving our proposed multi-neighborhood by introducing new types of relations and modifying the existing ones. Secondly, we plan to configure our algorithm by using automated parameter tuning to find better values for the probabilities of the neighborhood types, the hard constraint weights, the initial temperature, and finally for the maximum number of activities within the shifts. Applying a different cooling scheme and stopping criterium might also improve the results. We will conduct further evaluations of our approach, including experiments with longer runtimes, enabling the restart of the search method multiple times on each problem instance, from which the best solution can be selected. We also plan to extend our method to allow running searches on multiple threads in parallel. Finally, we will create an iterative procedure for populating the initial state of the search. This procedure would take every constraint into account for creating a good initial solution in a short timeframe, thus speeding up the search method and possibly enabling the production of better solutions.

Acknowledgements This research was supported by the Ministry of Culture and Innovation and the National Research, Development and Innovation Office within the Cooperative Technologies National Laboratory of Hungary (Grant No. 2022- 2.1.1-NL-2022-00012).

Appendix

Table 4. Features of the problem instances and comparative results

Inst.	Days	Staff	Tasks	VNS [10]	SA	Inst.	Days	Staff	Tasks	VNS [10]	SA
1	7	10	1	387	**383**	114	14	80	8	-	**5799**
2	7	10	1	176	**140**	115	14	80	10	-	**4006**
3	7	10	1	317	**290**	116	14	90	3	5598	**5214**
4	7	10	1	328	**304**	117	14	90	5	8818	**7985**
5	7	10	2	115	**37**	118	14	90	6	-	**8663**
6	7	20	1	900	**779**	119	14	90	9	-	**5399**
7	7	20	1	818	**783**	120	14	90	12	-	**3175**
8	7	20	2	884	**775**	121	14	100	4	-	**6946**
9	7	20	2	500	**353**	122	14	100	5	-	**9009**
10	7	20	3	268	**59**	123	14	100	7	-	**9779**
11	7	30	1	844	**788**	124	14	100	10	-	**7867**
12	7	30	2	1541	**1501**	125	14	100	13	-	**4061**
13	7	30	2	**1440**	-	126	14	110	4	7573	**7151**
14	7	30	3	**1469**	-	127	14	110	6	-	**9879**
15	7	30	4	553	**270**	128	14	110	8	-	**10015**
16	7	40	2	1883	**1580**	129	14	110	11	-	**7898**
17	7	40	2	1831	**1713**	130	14	110	14	-	**3758**
18	7	40	3	1737	**1457**	131	14	120	4	7475	**6877**
19	7	40	4	1437	**1034**	132	14	120	6	-	**11057**
20	7	40	5	955	**457**	133	14	120	8	-	**11847**
21	7	50	2	1740	**1647**	134	14	120	12	-	**7340**
22	7	50	3	2646	**2596**	135	14	120	15	-	-
23	7	50	4	2446	**2115**	136	14	130	5	-	**8764**
24	7	50	5	1795	**1395**	137	14	130	7	-	**12460**
25	7	50	7	1344	**758**	138	14	130	9	-	**11958**
26	7	60	2	1734	**1594**	139	14	130	13	-	**7345**
27	7	60	3	2904	**2622**	140	14	130	17	-	**5093**
28	7	60	4	3248	**2836**	141	14	140	5	**8013**	8859
29	7	60	6	2463	**1918**	142	14	140	7	-	**12725**
30	7	60	8	-	**1121**	143	14	140	10	-	**13013**
31	7	70	3	2574	**2466**	144	14	140	14	-	**9022**
32	7	70	4	3288	**3182**	145	14	140	18	-	**5706**
33	7	70	5	3170	**3025**	146	14	150	5	-	**8763**
34	7	70	7	-	-	147	14	150	8	-	**14321**
35	7	70	9	-	**1386**	148	14	150	10	-	**14824**
36	7	80	3	2709	**2536**	149	14	150	15	-	-
37	7	80	4	**3335**	3422	150	14	150	19	-	**6012**
38	7	80	6	3894	**3610**	151	28	10	1	1677	**1486**
39	7	80	8	-	**2709**	152	28	10	1	1509	**1341**
40	7	80	10	-	**1787**	153	28	10	1	1729	**1597**
41	7	90	3	**2575**	2643	154	28	10	1	1535	**1299**
42	7	90	5	4317	**4302**	155	28	10	2	-	**255**
43	7	90	6	4877	**4463**	156	28	20	1	3766	**3565**

Inst.	Days	Staff	Tasks	VNS [10]	SA	Inst.	Days	Staff	Tasks	VNS [10]	SA
44	7	90	9	-	**3109**	157	28	20	1	3523	**3312**
45	7	90	12	-	**1539**	158	28	20	2	3327	**2663**
46	7	100	4	**3471**	3635	159	28	20	2	2989	**2071**
47	7	100	5	**4837**	-	160	28	20	3	1803	**696**
48	7	100	7	5302	**4331**	161	28	30	1	3505	**3264**
49	7	100	10	-	**3662**	162	28	30	2	6551	**5499**
50	7	100	13	-	**2171**	163	28	30	2	6209	**5370**
51	7	110	4	**3338**	3553	164	28	30	3	-	**4163**
52	7	110	6	**5084**	5460	165	28	30	4	-	-
53	7	110	8	6237	**4980**	166	28	40	2	7613	**7173**
54	7	110	11	-	**3388**	167	28	40	2	7317	**7177**
55	7	110	14	-	**2694**	168	28	40	3	8270	**6710**
56	7	120	4	3486	**3410**	169	28	40	4	-	**5940**
57	7	120	6	5991	**5267**	170	28	40	5	-	**2960**
58	7	120	8	6749	**5931**	171	28	50	2	**6843**	-
59	7	120	12	-	**4643**	172	28	50	3	-	**8896**
60	7	120	15	-	**2714**	173	28	50	4	-	**7765**
61	7	130	5	4932	**4485**	174	28	50	5	-	**6374**
62	7	130	7	6720	**6366**	175	28	50	7	-	**3293**
63	7	130	9	7086	**6264**	176	28	60	2	7179	**6861**
64	7	130	13	-	**4449**	177	28	60	3	-	-
65	7	130	17	-	-	178	28	60	4	-	-
66	7	140	5	**4057**	4432	179	28	60	6	-	**8477**
67	7	140	7	**6009**	6370	180	28	60	8	-	**4513**
68	7	140	10	-	**6719**	181	28	70	3	-	**10393**
69	7	140	14	-	**4462**	182	28	70	4	-	-
70	7	140	18	-	**2685**	183	28	70	5	-	**13913**
71	7	150	5	**4063**	4419	184	28	70	7	-	**10329**
72	7	150	8	7590	**7367**	185	28	70	9	-	**5941**
73	7	150	10	-	**7330**	186	28	80	3	11181	**10544**
74	7	150	15	-	**5493**	187	28	80	4	-	**14658**
75	7	150	19	-	**2955**	188	28	80	6	-	**15811**
76	14	10	1	598	**550**	189	28	80	8	-	**10153**
77	14	10	1	814	**775**	190	28	80	10	-	**6690**
78	14	10	1	634	**581**	191	28	90	3	-	**10314**
79	14	10	1	607	**509**	192	28	90	5	-	-
80	14	10	2	292	**88**	193	28	90	6	-	**16767**
81	14	20	1	1659	**1580**	194	28	90	9	-	**13170**
82	14	20	1	1643	**1561**	195	28	90	12	-	**5338**
83	14	20	2	1387	**1053**	196	28	100	4	-	**14039**
84	14	20	2	1168	**906**	197	28	100	5	-	-
85	14	20	3	520	**123**	198	28	100	7	-	**20037**
86	14	30	1	1738	**1725**	199	28	100	10	-	**13458**

Inst.	Days	Staff	Tasks	VNS [10]	SA	Inst.	Days	Staff	Tasks	VNS [10]	SA
87	14	30	2	2672	**2541**	200	28	100	13	-	-
88	14	30	2	2780	**2539**	201	28	110	4	-	**14678**
89	14	30	3	**2551**	-	202	28	110	6	-	-
90	14	30	4	-	**1145**	203	28	110	8	-	**22346**
91	14	40	2	3514	**3324**	204	28	110	11	-	**15528**
92	14	40	2	3767	**3588**	205	28	110	14	-	**8984**
93	14	40	3	3820	**3232**	206	28	120	4	-	**14038**
94	14	40	4	3980	**3417**	207	28	120	6	-	**22210**
95	14	40	5	-	**1264**	208	28	120	8	-	-
96	14	50	2	3666	**3390**	209	28	120	12	-	**15592**
97	14	50	3	4921	**4278**	210	28	120	15	-	**12832**
98	14	50	4	4802	**4095**	211	28	130	5	-	**17786**
99	14	50	5	-	**3602**	212	28	130	7	-	-
100	14	50	7	-	**947**	213	28	130	9	-	**25974**
101	14	60	2	3419	**3327**	214	28	130	13	-	**15203**
102	14	60	3	5473	**5309**	215	28	130	17	-	-
103	14	60	4	5942	**5914**	216	28	140	5	-	**18010**
104	14	60	6	5620	**4278**	217	28	140	7	-	-
105	14	60	8	-	-	218	28	140	10	-	**25463**
106	14	70	3	**5137**	5170	219	28	140	14	-	**19802**
107	14	70	4	6892	**6546**	220	28	140	18	-	-
108	14	70	5	-	**5705**	221	28	150	5	-	-
109	14	70	7	-	**4684**	222	28	150	8	-	**29983**
110	14	70	9	-	**3434**	223	28	150	10	-	**28523**
111	14	80	3	5510	**5310**	224	28	150	15	-	**19259**
112	14	80	4	**6748**	7113	225	28	150	19	-	**13429**
113	14	80	6	8124	**7034**						

References

1. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D.: An annotated bibliography of personnel scheduling and rostering. Annals of Operations Research 127(1), 21–144 (2004).
2. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153(1), 3–27 (2004).
3. Loucks, J.S., Jacobs, F.R.: Tour scheduling and task assignment of a heterogeneous work force: A heuristic approach. Decision Sciences 22(4), 719–738 (1991).
4. Dahmen, S., Rekik, M., Soumis, F.: An implicit model for multi-activity shift scheduling problems. Journal of Scheduling 21(3), 285–304 (2018).
5. Lequy, Q., Desaulniers, G., Solomon, M.M.: A two-stage heuristic for multi-activity and task assignment to work shifts. Computers & Industrial Engineering 63(4), 831–841 (2012).

6. Dahmen, S., Rekik, M.: Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic. Journal of Scheduling 18(2), 207-223 (2015).
7. Römer, M: Block-based state-expanded network models for multi-activity shift scheduling. Journal of Scheduling, 1-21 (2023).
8. Qu, Y., Curtois, T.: Multi-Activity, Multi-Day Shift Scheduling Problem Definition, http://www.schedulingbenchmarks.org/matsp/ProblemDefinition.pdf, last accessed 2024/03/13.
9. Multi-activity, Multi-day Shift Scheduling Benchmark Instances, http://www.schedulingbenchmarks.org/matsp/, last accessed 2024/03/13.
10. Qu, Y., Curtois, T.: Solving the Multi-Activity Shift Scheduling Problem Using Variable Neighbourhood Search. In: Proceedings of the 9th International Conference on Operations Research and Enterprise Systems, pp. 227-232. SCITEPRESS, Valetta (2020).
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671-680 (1983).
12. Rosati, R.M., Petris, M., Di Gaspero, L., Schaerf, A.: Multi-neighborhood simulated annealing for the sports timetabling competition ITC2021. Journal of Scheduling 25(2), 1-19 (2022).
13. Ceschia, S., Guido, R., Schaerf, A.: Solving the static inrc-ii nurse rostering problem by simulated annealing based on large neighborhoods. Annals of Operations Research 288(1), 95–113 (2020).
14. Cambazard, H., Hebrard, E., O'Sullivan, B., Papadopoulos, A.: Local search and constraint programming for the post enrolment-based timetabling problem. Annals of Operational Research 194(1), 111–135 (2012).
15. Lewis, R., Thompson, J.: Analysing the Effects of Solution Space Connectivity with an Effective Metaheuristic for the Course Timetabling Problem. European Journal of Operational Research 240(3), 637-648 (2015).
16. Van Bulck, D., Goossens D., Schaerf, A.: Multi-neighbourhood simulated annealing for the ITC-2007 capacitated examination timetabling problem. Journal of Scheduling, 1-16 (2023).
17. Szu, H., Hartley, R.: Fast simulated annealing. Physics Letters A 122(3-4), 157-162 (1987).