Abstract— Workforce management has become increasingly important for companies. It is vital to optimize the performance of staff on both financial efficiency and customer satisfaction. Furthermore, it is important to optimize employer and employee satisfaction simultaneously. Good staff schedules are very important for the welfare of the staff, resulting in increased job satisfaction and reduction of sick-leaves. Good optimization methods are needed to roster the staff efficiently. This paper defines the core staff rostering problem, introduces the necessary terminology and details the constraints of the problem. The first set of random benchmark instances for the problem is introduced. The instances are simplified to the point where the used constraints comprise the core of almost any staff rostering problem. This enables a large number of researchers to find out how well their optimization methods work. We publish the best solutions we have found. The instances are available online.

Index Terms—Staff rostering, scheduling, benchmark instances, PEAST algorithm

I. INTRODUCTION

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 optimization is the key to efficient use of workforce, customer satisfaction, employee satisfaction and control over rapidly changing situations, duties and competence demands. The workforce scheduling problem has a fairly broad definition. Most of the studies focus on assigning employees to shifts, determining working days and rest days or constructing flexible shifts and their starting times. Different variations of the problem and subproblems are NP-hard and NP-complete [1]-[6], and thus extremely hard to solve. The first mathematical formulation of the problem based on a generalized set covering model was proposed by Dantzig [7]. Good overviews of workforce scheduling are published by Alfares [8], Ernst et al. [9], Meisels and Schaerf [10] and De Causmaecker and G. Vanden Berghe [11].

The real-world workforce scheduling process starts from three entry points (see Figure 1). First, workload prediction (or demand forecasting) is the phase of determining the staffing levels - that is, how many employees are needed for each timeslot in the planning horizon. Shift generation is the phase of determining the shift structure, along with the activities to be carried out in particular shifts and the competences required for different shifts. Second, the HR master system provides necessary employee data, such as labor contract, work unit and working hours. Third, the competences and preferences of the employees are maintained by the workforce management system itself. These three entry points gather the required information for shifts and employees (see Figure 1).

Fig. 1. The preprocessing phases of the real-world workforce scheduling process.

The nature of determining the amount and type of work to be done at any given time during the next planning horizon depends greatly on the nature of the job. Some form of workload prediction is called for if the workload is overly uncertain. Some examples of this are the calls incoming to a call center or the customer influx to a hospital. Modern workforce management systems (WFM) use simulation to calculate forecast values.

Shift generation transforms the determined workload into shifts. This includes deciding break times when applicable. Shift generation is essential especially in cases where the workload is not static. A basic shift generation problem
includes a variable number of activities for each task in each timeslot. Activities may require competences. The most important optimization target is to match the shifts to the workload as accurately as possible. The shifts are generally created for each day separately, each shift corresponding to a single employee’s competences and preferences.

Staff rostering is the phase where the shifts are assigned to the employees. 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.

Unfortunately, the optimized staff rosters need to be changed. Daily rescheduling deals with ad hoc changes that are necessary due to sick leaves and other no-shows. The changes are usually carried out manually. Still, the system should suggest suitable substitutes considering the qualifications, employment contract, legal limitations and salaries. The goal is to find the most economical candidates.

Finally, the completed working times will be booked and made available for payroll accounting system. When necessary, the workload prediction or/and the shift generation phases may be restarted. In any case, the staff rostering will be restarted for the next planning horizon.

As stated earlier, most of the academic studies focus on staff rostering. Some academic researchers have even announced that their optimization methods are in commercial use [12-15]. Some real-world benchmark instances [16-19] exist to help researchers to test their optimization methods. To the best of our knowledge, no simple random test instances exist. The goal of this paper is to introduce a new set of benchmark instances for the core staff rostering problem. The test instances are simple enough to ensure that they will not remain unsolved by other researchers because of the complexity of the instance.

Section II describes the core staff rostering problem and introduces the necessary terminology. Section III introduces the benchmark instances. In Section IV we briefly describe our PEAST algorithm which is used to solve the benchmark instances. Section V sets out the computational results.

**II. THE CORE STAFF ROSTERING PROBLEM**

The staff rostering problem consists of assigning employees to shifts and tasks over a period of time. The planning horizon is the time interval over which the employees have to be scheduled. Each employee has a total working time that he/she has to work during the planning horizon. Furthermore, each employee has competences (qualifications and skills) that enable him/her to carry out certain tasks. Days are divided into working days (days-on) and rest days (days-off). Each day is divided into periods or timeslots. A timeslot is the smallest unit of time and the length of a timeslot determines the granularity of the schedule. A shift is a contiguous set of working hours and is defined by a day and a starting period on that day along with a shift length (the number of occupied timeslots) or shift time. Shifts are sometimes grouped into shift types, such as morning, day and night shifts. Each shift is composed of tasks and breaks. The sum of the length of a shift’s tasks is called working time. A shift or a task may require the employee assigned to it to possess one or more competences. A sequence of working days with one shift each day is called a work stretch. A work schedule over the planning horizon for an employee is called a roster. A roster is a combination of shifts and days-off assignments that
covers a fixed period of time. Figure 3 clarifies the terminology.

![Staff rostering terminology](image)

There are hundreds of staff rostering solutions commercially available and in widespread use. Only few of them include optimization. It is difficult to incorporate the experience and expertise of the personnel managers into a staff rostering system. Personnel managers have extremely valuable knowledge, experience and detailed understanding of their specific staffing problem, which will vary from company to company. To formalize this knowledge into constraints is not an easy task.

Dozens of optimization models for staff rostering have been introduced by academics (see [8-11]). These models present a wide variety of real-world constraints. The goal is to build up a solid foundation for staff rostering scenarios. The most important goal of the models is to minimize understaffing and overstaffing without violating work contracts and government regulations. Some of the models also pay attention to employee requests.

We model the core staff rostering problem as a constraint satisfaction problem. The model includes eight constraints that comprise the core of almost any staff rostering problem:

**Hard constraints**

1. **H1** An employee cannot be assigned to more than one shift on a day and cannot be assigned to overlapping shifts
2. **H2** Maximum of one employee can be assigned to a shift
3. **H3** An employee must not work consecutively for more than the given number of days
4. **H4** The given number of rest time between two shifts should be respected

**Soft constraints**

1. **S1** Each employee’s working time should be within the given limits (one violation for each starting 10 minutes below or above)
2. **S2** Single days-off should be avoided (one violation for each such case)
3. **S3** Single working days should be avoided (two violations for each such case)
4. **S4** The given (minimum) number of free weekends, both Saturday and Sunday free, for each employee should be respected (one violation for each case below).

The constraints state that the task is to meet the daily staffing requirements without violating work contracts. The goal is to find a solution that has no hard constraint violations and that minimizes the sum of the soft constraint violations. Note, that single days-off and single working days at the start or at the end of the planning period are not calculated.

### III. The Benchmark Instances

We believe that introducing simple benchmark instances of the core staff rostering problem will be very valuable to researchers. This enables a large number of researchers to find out how well their optimization methods work. The number of constraints in the current staff rostering benchmark instances is quite large and some of the models are also quite complicated.

Researchers quite often only solve one real-world case. The strength of random test instances is the ability to produce many problems with many different properties. Still, they should be sufficiently simple for each researcher to be able to use them in their test environment.

The No Free Lunch Theorem [20] states that all search algorithms that look for an optimum of some cost function perform exactly identically when averaged over all possible cost functions. As we know, this result implies three important conclusions:

1. If a search algorithm works well for a given problem type, it is not guaranteed to work as well for a different problem type
2. There exists no superior optimization method
3. If we are faced with a new problem area, we should first analyze the problem and then choose the most applicable optimization method, not vice versa.

However, there is still room for comparisons between different optimization methods, albeit not in the sense of finding the overall best one, but to find out on which problem areas a particular method is likely to be better than others. We invite the workforce scheduling community to try their methods on the core staff rostering problem.

Table I shows ten random test instances for the core staff rostering problem. The number of employees \( E \) varies between 50 and 100. The planning period \( W \) is 4, 6 or 8 weeks. The length of a shift is a uniform random number on the interval given by \( L \). The required working time of an employee is given by an interval, for example between 10000 and 11000 minutes. The intervals are randomly created based on the total number of working time required by the shifts. The average required working time of the employees is given by \( T_{avg} \). The minimum and maximum intervals of the working times are given by \( I_{min} \) and \( I_{max} \). The data for the test instances is available online [26].

Table II shows the parameters and weights of the constraints. The hard constraints \( H1 \) and \( H2 \) have no additional parameters. The maximum number of consecutive working days is given in \( H3 \) and the minimum rest time in minutes between two shifts is given in \( H4 \). The
weights of the soft constraints are given by $S1$, $S2$, $S3$ and $S4$. The minimum number of free weekends for $S4$ is given in parentheses.

<table>
<thead>
<tr>
<th>ID</th>
<th>E</th>
<th>W</th>
<th>S</th>
<th>L-avg</th>
<th>I-min</th>
<th>I-max</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>100</td>
<td>4</td>
<td>2000</td>
<td>450 - 600</td>
<td>10499</td>
<td>25</td>
</tr>
<tr>
<td>02</td>
<td>50</td>
<td>8</td>
<td>2000</td>
<td>240 - 720</td>
<td>19211</td>
<td>307</td>
</tr>
<tr>
<td>03</td>
<td>50</td>
<td>4</td>
<td>1000</td>
<td>460 - 500</td>
<td>9598</td>
<td>15</td>
</tr>
<tr>
<td>04</td>
<td>100</td>
<td>4</td>
<td>2000</td>
<td>440 - 520</td>
<td>9569</td>
<td>133</td>
</tr>
<tr>
<td>05</td>
<td>100</td>
<td>4</td>
<td>2000</td>
<td>360 - 540</td>
<td>9093</td>
<td>25</td>
</tr>
<tr>
<td>06</td>
<td>50</td>
<td>8</td>
<td>2000</td>
<td>360 - 540</td>
<td>17806</td>
<td>110</td>
</tr>
<tr>
<td>07</td>
<td>50</td>
<td>4</td>
<td>1000</td>
<td>450 - 550</td>
<td>9997</td>
<td>15</td>
</tr>
<tr>
<td>08</td>
<td>75</td>
<td>6</td>
<td>2250</td>
<td>100 - 700</td>
<td>11988</td>
<td>7</td>
</tr>
<tr>
<td>09</td>
<td>100</td>
<td>4</td>
<td>2000</td>
<td>100 - 700</td>
<td>8058</td>
<td>92</td>
</tr>
<tr>
<td>10</td>
<td>75</td>
<td>6</td>
<td>2250</td>
<td>700 - 900</td>
<td>24556</td>
<td>23</td>
</tr>
</tbody>
</table>

$E =$ number of employees, $W =$ number of weeks, $S =$ number of shifts, $L =$ shift length interval, $T$-avg $=$ average required working time, $I$-min and $I$-max $=$ minimum and maximum interval of the working time

<table>
<thead>
<tr>
<th>ID</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>on</td>
<td>on</td>
<td>6</td>
<td>540</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>02</td>
<td>on</td>
<td>on</td>
<td>6</td>
<td>720</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>03</td>
<td>on</td>
<td>on</td>
<td>6</td>
<td>720</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3 (2)</td>
</tr>
<tr>
<td>04</td>
<td>on</td>
<td>on</td>
<td>6</td>
<td>720</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>05</td>
<td>on</td>
<td>on</td>
<td>6</td>
<td>510</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>06</td>
<td>on</td>
<td>on</td>
<td>8</td>
<td>540</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>07</td>
<td>on</td>
<td>on</td>
<td>5</td>
<td>720</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>08</td>
<td>on</td>
<td>on</td>
<td>5</td>
<td>700</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>09</td>
<td>on</td>
<td>on</td>
<td>5</td>
<td>1000</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>on</td>
<td>on</td>
<td>6</td>
<td>360</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

$H3 =$ maximum number of consecutive working days, $H4 =$ minimum rest interval in minutes between two shifts, $Sx =$ weight of the soft constraint $x$

IV. THE PEAST ALGORITHM

This section briefly describes the PEAST algorithm which is used to solve the benchmark instances presented in Section III. The PEAST algorithm has been used to solve several real-world scheduling problems (see eg. [20-24]) and it is in industrial use.

The pseudo-code of the algorithm is given in Figure 4. The algorithm is a population-based local search method. In every $c^n$ iteration, the least fit schedule is replaced with a clone of the fittest individual. This operation is completely irrespective of the globally fittest schedule ($best_S$) found by that in the search process.

The heart of the algorithm is the local search operator called GHCM (greedy hill-climbing mutation). It extends the basic hill-climbing step to generate a sequence of moves in one step, leading from one solution to another.

The PEAST algorithm applies a number of shuffling operators to perturb a solution into a potentially worse solution in order to escape from local optima. The operators are called after a given number of iterations have passed. For example, in staff rostering two random employees are selected and all their jobs are swapped.

The algorithm avoids staying stuck in the same areas of the search space using tabu search and the refined simulated annealing method. A tabu list is used to prevent reverse order moves in a single application of the GHCM operator. The simulated annealing refinement is used to decide whether or not to commit to a sequence of moves in the GHCM operator.

The PEAST algorithm uses the adaptive genetic penalty method (ADAGEN) to solve multi-objective problems such as the constraint satisfaction problem. The method assigns dynamic weights to the hard constraints based on the search trajectory and the constant weights assigned to the soft constraints. The soft constraints are assigned fixed weights according to their significance. The significance is given by the problem owner (end-user).

The acronym PEAST stems from the methods used: Population, Ejection, Annealing, Shuffling and Tabu. To the best of our knowledge, the heart of the algorithm, the GHCM operator, is one of a kind. The same applies to our implementation of the shuffling operators, the simulated annealing refinement and the penalty method. For the detailed discussion of the PEAST algorithm and its components we refer to [25].

V. COMPUTATIONAL RESULTS

This section summarizes the best solutions we have found for the benchmark instances. We solved the instances using the PEAST algorithm described in Section IV. We used exactly the same version and parameters of the algorithm that are in use in the real-world workforce management system.

We found a feasible solution to all instances. This was actually not difficult. Table III shows our best solutions from ten runs. The instances 05 and 10 were the most difficult to solve. The most difficult task was to minimize the number of single days-off. We believe that the number of violations for the soft constraint S1 for the instances 05, 06 and 08 is optimal. We are quite certain that better overall solutions for all the test instances exist.
TABLE III

<table>
<thead>
<tr>
<th>ID</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>18</td>
<td>234</td>
</tr>
<tr>
<td>02</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>13</td>
<td>101</td>
</tr>
<tr>
<td>03</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>63</td>
<td>261</td>
</tr>
<tr>
<td>04</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>0</td>
<td>0</td>
<td>112</td>
</tr>
<tr>
<td>05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>264</td>
<td>13</td>
<td>106</td>
<td>907</td>
</tr>
<tr>
<td>06</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>683</td>
<td>37</td>
<td>1</td>
<td>10</td>
<td>1441</td>
</tr>
<tr>
<td>07</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>683</td>
<td>37</td>
<td>1</td>
<td>10</td>
<td>1441</td>
</tr>
<tr>
<td>08</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2543</td>
<td>167</td>
<td>0</td>
<td>134</td>
<td>5989</td>
</tr>
<tr>
<td>09</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>161</td>
<td>0</td>
<td>221</td>
<td>382</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>63</td>
<td>5</td>
<td>0</td>
<td>78</td>
</tr>
</tbody>
</table>

The cells give the number of violations for each constraint.

VI. CONCLUSIONS

We defined the core staff rostering problem as a constraint satisfaction problem that includes eight constraints that comprise the core of almost any staff rostering problem. We introduced ten random test instances for the problem.

We solved the instances using the PEAST algorithm. The workforce scheduling community is invited to challenge our solutions. We hope that the instances will help researchers to test the value of their optimization methods. The instances are available online.

REFERENCES