Genetic Algorithm for Rotating Workforce Scheduling Problem

Michael Mörz
Technische Universität Wien
Karlsplatz 13, 1040 Wien, Austria
Email: moerz@dbai.tuwien.ac.at

Nysret Musliu
Technische Universität Wien
Karlsplatz 13, 1040 Wien, Austria
Email: musliu@dbai.tuwien.ac.at

Abstract—In this paper a genetic algorithm based method for solving the rotating workforce scheduling problem is presented. Rotating workforce scheduling is a typical constraint satisfaction problem which appears in a broad range of workplaces (e.g. industrial plants). Solving this problem is of a high practical relevance. We propose a basic genetic algorithm for solving this problem. One mutation operator and three methods for crossover are presented. Finally we give computational results on benchmark examples from literature.

I. Introduction

Workforce scheduling, in general, includes sub-problems which appear in many spheres of life like in industrial plants, hospitals, public transport, airlines companies, universities etc. According to the area, this problem is denoted with different names in the literature. Usual used terms are: workforce scheduling, manpower scheduling, staff scheduling, employee timetabling, crew scheduling, etc. Typically a workforce schedule represents the assignments of the employees to the defined shifts for a period of time. In Table I a typical representation of workforce schedules is presented. This schedule describes explicitly the working schedule of 9 employees during one week. The first employee works from Monday until Friday in a day shift (D) and during Saturday and Sunday has days-off. The second employee has a day-off on Monday and works in a day shift during the rest of the week. Further, the last employee works from Monday until Wednesday in night shifts (N), on Thursday and Friday has days-off, and on Saturday and Sunday works in the day shift. So each row of this table represents the weekly schedule of one employee.

There are two main variants of workforce schedules: rotating (or cyclic) workforce schedules and non-cyclic workforce schedules. In a rotating workforce schedule all employees have the same basic schedule but start with different offsets. Therefore, while the individual preferences of the employees cannot be taken into account, the aim is to find a schedule that is optimal for all employees. In non-cyclic workforce schedules the individual preferences of the employees can be taken into consideration and the aim is to achieve schedules that fulfill the preferences of most employees. In both variations of workforce schedules other constraints such as the minimum number of employees required for each shift have to be met. In this paper we will consider the problem of rotating workforce scheduling. This problem is NP-complete.

TABLE I
One typical week schedule for 9 employees

Employee/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	D	D	D	-	-
2	-	D	D	D	D	D	D
3	D	-	-	N	N	N	N
4	-	-	-	-	A	A	A
5	A	A	A	A	-	-	-
6	N	N	N	N	N	-	-
7	-	-	A	A	A	A	A
8	A	A	-	-	-	N	N
9	N	N	N	-	-	D	D

Workforce schedules have an impact on the health and satisfaction of employees as well as on their performance at work. Therefore, computerized workforce scheduling has interested researchers for over 30 years. For solving the problem of workforce scheduling, different approaches were used in the literature. A survey of algorithms used for different workforce scheduling problems is given by Tien and Kamiyama [13]. Examples for the use of exhaustive enumeration for generation of rotating workforce schedules are [4] and [2]. Balakrishnan and Wong [1] solved a problem of rotating workforce scheduling by modeling it as a network flow problem. Several other algorithms for rotating workforce schedules have been proposed in the literature [6], [7], [10]. Musliu et al [12] proposed and implemented a new method for the generation of rotating workforce schedules, which is based on prunning of search space, by involving the decision maker during the generation of part solutions. The algorithm has been included in a commercial product called First Class Scheduler, which is already being used by several companies in Europe since 2000. Although this product has been shown to work well in practice for solving of most real problems, for very large instances of problems solution cannot always be

guaranteed because of the large size of the search space. Recently, Musliu [11] proposed a tabu search approach for solving the rotating workforce scheduling problem, but this method is not yet well studied in very large instances of problems. The critical features of workforce scheduling algorithms are their computational behavior and flexibility for solving a wide range of problems that appear in practice. Recently Laporte [8] assessed rotating workforce scheduling algorithms proposed in the literature saying that "[...] these are often too constraining and not sufficiently flexible for this type of problem".

In this paper, we investigate the use of genetic algorithms (GA) for solving the rotating workforce scheduling problem. Although the GA were used to solve similar problems with rotating workforce scheduling (e.g. nurse scheduling), to our best knowledge the genetic algorithms was not used to solve directly the problem we consider in this paper. The rotating workforce scheduling involves typically different constraints compared to nurse scheduling, and what also makes the rotating workforce scheduling different from nurse scheduling is that it is cyclic (see the problem definition in next section).

We proceed in this paper as follows: In the next section we give a precise definition of the problem for which the genetic algorithm is applied. In Section 2 the genetic algorithm used for solving this problem is described. In Section 3 the computational results are given and finally in the last section concluding remarks are presented.

A. Rotating workforce scheduling

In this section, we describe the problem of assigning shifts and days-off to employees in case of rotating workforce schedules. This is a specific problem of a general workforce-scheduling problem. The definition is given below ([12]):
Instance:

- Number of employees: n.
- Set A of m shifts (activities) : $a_1, a_2, \ldots, a_m$, where a_m represents the special day-off "shift".
- w: length of schedule. The total length of a planning period is $n \times w$ because of the cyclic characteristics of the schedules.
- A cyclic schedule is represented by an $n \times w$ matrix $S \in A^{nw}$. Each element $s_{i,j}$ of matrix S corresponds to one shift. Element $s_{i,j}$ shows which shift employee i works during day j or whether the employee has time off. In a cyclic schedule, the schedule for one employee consists of a sequence of all rows of the matrix S. The last element of a row is adjacent to the first element of the next row, and the last element of the matrix is adjacent to its first element.
- Temporal requirements: $(m-1) \times w$ matrix R, where each element $r_{i,j}$ of matrix R shows the required number of employees for shift i during day j.
- Constraints:
  - Sequences of shifts permitted to be assigned to employees (the complement of inadmissible sequences): Shift change $m \times m \times m$ matrix $C \in A^{(m^3)}$. If element $c_{i,j,k}$ of matrix C is 1, the sequence of shifts (a_i, a_j, a_k) is permitted, otherwise it is not.
  - Maximum and minimum length of periods of consecutive shifts: Vectors $MAXS_m$, $MINS_m$, where each element shows the maximum respectively minimum permitted length of periods of consecutive shifts.
  - Maximum and minimum length of blocks of workdays: $MAXW$, $MINW$.

Problem: Find a cyclic schedule (assignment of shifts to employees) that satisfies the requirement matrix, and all other constraints.

Note that in [12], finding as many non-isomorphic cyclic schedules as possible that satisfy all constraints, and are optimal in terms of weekends without scheduled work shifts (weekends off), is required. But in this paper we consider the generation of only one schedule, which satisfies all constraints. Furthermore, we do not consider the optimization of weekends off.

II. A Genetic Algorithm for the rotating workforce scheduling problem

Genetic algorithm ([5], [3]) is a powerful modern heuristic technique, which has been used successfully for many practical problems. The basic idea of this technique is to use the natural selection of the fittest population members for advancing to a better set of solutions.

Further, we present the main characteristics of a genetic algorithm, which is applied for solving the rotating workforce scheduling problem. We describe how the individuals are represented, the generation of initial population, mutation and crossover operators and the selection process during each iteration.

A. Representation of Individuals

Each population member stores one cyclic schedule that is represented by an $n \times w$ matrix S where each element $s_{i,j}$ of matrix S corresponds to one shift or day off (for exact definition see section I-A).

B. Generation of Initial Population

The cyclic schedule of each individual is filled on a day (column) per day basis. For every day a vector v is generated with $v \in A^{Numberofemployees}$, that is filled randomly with shifts until the requirements for each shift for the specific day are fulfilled. This kind of initialization always fulfills the workforce requirements. Actually as we will see later, we also define the crossover and mutation operators not to derange the workforce requirements, when they are applied. This design assures, that the constraint about the workforce requirements is fulfilled during each iterations, what makes the problem easier to solve.

Considering the number of individuals in the population, we relate this parameter with the size of the problem. In our current implementation of algorithm we choose the number of individuals in a population to be twice the number of employees.

C. Fitness Function

During the generation of a new population (next generation), the evaluation of solutions (population members) is most time consuming, because each solution has to be checked for many constraints. For each violation of a constraint a determined number of points (penalty) is given, based on the constraint and the degree of the constraint violation. The fitness of a population member is calculated as the sum of those points for the population member. So the fitness represents the sum of all penalties caused by the violation of constraints. As the problem, we want to solve, has only hard constraints, the solution will be found when the fitness of the solution reaches the value 0. The fitness is calculated as follows [11]:

$$Fitness = \sum_{i=1}^{NW} P1 \times Distance(WB_i, WorkBBRange) + \sum_{i=1}^{ND} P2 \times Distance(DOB_i, DayOffBRange) + \sum_{j=1}^{NumOfShifts} (\sum_{i=1}^{NS_j} P3 \times Distance(SB_{ij}, ShiftSeqRange_j)) + P4 \times NumOfNotAllowedShiftSeq$$

NW, ND, represent respectively, the number of work blocks, and days off blocks, whereas NS_j, represents number of shift sequences blocks of shift j. WB_i, DOB_i, represent respectively, a work block i, and days-off block i. SB_{ij}, represents the i-th shifts block of shift j. The penalties of the violation of constraints are set as follow: $P3 = P1 = P2 = P4 = 1$. The function $Distance(XBlock, range)$ returns 0 if the length of the block $XBlock$ is inside the range of two numbers ($range$), otherwise returns the distance of length of $XBlock$ from the range. For example if the legal range of work blocks is $4-7$ and length of work block $XBlock$ is 3 or 8 then this function will return value 1.

D. Mutation

The mutation operator is a simple move, which swaps the contents of two elements in the $n \times w$ matrix S of an individual. The swapping of elements is done only inside of a particular column to be sure that the workforce requirements are always fulfilled. The column in which the swap is done, is selected randomly and the elements to swap are also chosen randomly.

Because our experiments have shown that the mutation operator is very powerful, the probability of mutation is set to be very high (0.8).

The pseudo code for the the mutation operator is presented below.

For FirstChild to LastChild
 if $randompercentage < mutationprobability$ then
  $column = randomday \in [1, ..., NumberOfColumns]$;
  $employee1 = randomemployee \in [1, ..., Employees]$;
  $employee2 = randomemployee \in [1, ..., Employees]$;
  $swapshift\,(column, employee1, employee2)$;
 endif
Next

E. Crossover

We have experimented with three different types of crossover operators. All of them take two individuals as an input. The crossover probability is set to be 0.5. The individuals which will take part in crossover are chosen randomly from the set of parents (population).

Types of crossover:

Random crossover

1) Choose a random number x ($x \in [1, ..., w]$)
2) Select x times a random column
3) Swap the chosen columns between the individuals

Block crossover

1) Choose a column x ($x \in [1, ..., w]$) randomly
2) Choose a length $y \in [1, ..., w - x]$ for a block of columns
3) Swap the selected block of columns between the individuals

Separate columns crossover

1) Choose a random number x ($x \in [1, ..., 3]$)
2) Select x times a random column that is a not a neighbor column of an already chosen one
3) Swap the selected columns between the population members

F. Selection

To acknowlegde the fittness and their survival, all children are evaluated regarding the errors that are present in their schedule. Then they are subdivided into classes by their fittness. That results in p classes where p can be any number between 1 and the number of children.

Class x, where $x \in 1, ..., p$, has the probability of $\frac{1}{2}^x$ to get choosen. When a class has been choosen the first child of that class is copied into the new population and the child gets removed from the class. If the class is empty then, it will get removed from the class subdivision.

III. Computational results

In this section we report on computational results obtained with the current version of genetic algorithm we have implemented. We give the results for the three benchmark problems from the literature. Our first aim in this paper was to see if the genetic algorithm can find

at all solutions for real-world rotating workforce scheduling problems. Another aim was to compare the genetic algorithm with the existing algorithms. For comparison we take in considering the number of evaluations needed to find the solution. We compare our results with those reported with tabu search in [11], and give some remarks considering the performance of genetic algorithm compare to First Class Scheduler [12]. Direct comparison with First Class Scheduler can not be done, because the FCS is based on the interaction with the decision maker.

For each problem 10 independent runs were executed. Number of iterations is limited to 500000 for each run.

Problem 1: The first problem from literature for which we discuss computational results is a problem solved by Butler [2] for the Edmonton police department in Alberta, Canada. Properties of this problem are:
Number of employees: 9
Shifts: 1 (Day), 2 (Evening), 3 (Night)

Temporal requirements:

$$R_{3,7} = \begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 3 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Constraints: Length of work periods should be between 4 and 7 days; Only shift 1 can precede the rest period preceding a shift 3 work period; Before and after weekends off, only shift 3 or shift 2 work periods are allowed; At least two consecutive days must be assigned to the same shift; No more than two 7-day work periods are allowed and these work periods should not be consecutive

Before we give our computational results some observations should be made. First constraint two and three cannot be represented in our problem definition. Let us note here that in all three examples given, we cannot model the problem exactly (the same was true for Balakrishnan and Wong's [1] approach to the original problems), which is to a high degree due to the different legal requirements found in the U.S./Canadian versus those found in the European context, but we tried to mimic the constraints as closely as possible or to replace them by similar constraints that appeared more meaningful in the European context. Having said this, let us proceed as follows: The other constraints can be applied in our model and are left like in the original problem. As mentioned, we include additional constraints about maximum length of successive shifts and minimum and maximum length of days-off blocks. In summary, additional constraint used for our algorithm are:

- Not allowed shift changes: (N D), (N A), (A D)
- Length of days-off periods should be between 2 and 4
- Vector $MAXS_3 = (7, 6, 4)$

The genetic algorithm could find a solution for this problem in all 10 runs. GA needed in average 4850 iterations (485070 evaluations) to find the solution. The solution for this problem found by genetic algorithm in the first run is presented in Table II.

The solver based on tabu search [11] finds the solution in only 25 iterations (5250 evaluations).

TABLE II
Genetic algorithm solution for the problem 1

Employee/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	A	A	N	N	-
2	-	D	D	A	A	A	-
3	-	-	D	D	A	A	A
4	-	-	-	A	A	N	N
5	N	-	-	D	D	D	D
6	A	A	-	-	D	D	D
7	D	N	N	N	-	-	N
8	N	N	N	-	-	A	A
9	A	A	A	N	N	-	-

Problem 2 (Laporte et al. [9]): There exist three non overlapping shifts D, A, and N, 9 employees, and requirements are 2 employees in each shift and every day. A week schedule has to be constructed that fulfills these constraints: (1) Rest periods should be at least two days-off, (2) Work periods must be between 2 and 7 days long if work is done in shift D or A and between 4 and 7 if work is done in shift N, (3)Shift changes can occur only after a day-off, (4)Schedules should contain as many weekends as possible, (5)Weekends off should be distributed throughout the schedule as evenly as possible, (6) Long (short periods) should be followed by long (short) rest periods, (7)Work periods of 7 days are preferred in shift N.

Let as note that we cannot model the problem given in this section exactly in our solver. Here we also mimic the constraints as closely as possible or replace them by similar constraints which can be taken in consideration in our genetic algorithm. In our case constraint 1 is straightforward. Constraint 2 can be approximated if we take the minimum of work blocks to be 4. Constraint 3 can also be modeled if we take the minimum length of successive shifts to be 4. For maximum length of successive shifts we take 7 for each shift. Other constraints can not be modeled in our solver.

The genetic algorithm could find a solution for this problem in all 10 runs. GA needed in average 5318 iterations (531840 evaluations) to find the solution. One solution for this problem found by genetic algorithm is presented in Table III.

The solver based on tabu search [11] finds the solution in 585 iterations (122850 evaluations).

Problem 3: This problem is a larger problem first reported in [4]. Characteristics of this problem are:
Number of employees is 17 (length of planning period is 17 weeks).

TABLE III
Genetic algorithm solution for the problem 2

Employee/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	-	-	-	N	N	N	N
2	-	-	D	D	D	D	D
3	D	D	-	-	A	A	A
4	A	A	A	A	-	-	N
5	N	N	N	-	-	-	-
6	D	D	D	D	D	D	D
7	-	-	-	A	A	A	A
8	A	A	A	-	-	-	-
9	N	N	N	N	N	N	-

Three nonoverlapping shifts.
Temporal requirements are:

$$R_{3,7} = \begin{pmatrix} 5 & 4 & 4 & 4 & 4 & 4 & 3 \\ 5 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 3 & 3 & 3 & 4 & 4 & 4 \end{pmatrix}$$

Constraints: Rest-period lengths must be between 2 and 7 days; Work-periods lengths must be between 3 and 8 days; A shift cannot be assigned to more than 4 consecutive weeks in a row; Shift changes are allowed only after a rest period that includes a Sunday or a Monday or both; The only allowable shift changes are 1 to 3, 2 to 1, and 3 to 2.

We cannot model constraints 3, 4, and 5 in their original form. We allow changes in the same block and for these reason we have other shift change constraints. In our case the following shift changes are not allowed: 2 to 1, 3 to 1, and 3 to 2. Additionally, we limit the rest period length from 2 to 4 and work periods length from 4 to 7. Maximum and minimum length of blocks of successive shifts are given with vectors $MAXS_3 = (7, 6, 5)$ and $MINS_3 = (2, 2, 2)$.

The genetic algorithm could find a solution for this problem in all 10 runs. GA needed in average 17383 iterations (3302941 evaluations) to find the solution. The solution for this problem found by genetic algorithm in the first run is presented in Table IV.

The solver based on tabu search [11] finds the solution 1179 iterations (880713 evaluations).

In the Table V a summary of results for our implementation of genetic algorithm in three benchmark problems is given. For each problem the average number of iterations and evaluations out of 10 runs is presented.

IV. Conclusion

Our current implementation of genetic algorithm gives promising results for the problem of rotating workforce scheduling. Computational results show that the genetic algorithm can successfully generate solutions for typical real life problems from practice. Compared to tabu search approach [11] the current version of algorithms needs more evaluations of candidate solutions to reach the solutions. First Class Scheduler [12] still outperforms in most real life problems tabu search approach and also the approach we presented in this paper. However the current version of genetic algorithm can still be improved. In particularly a comprehensive study of the impacts of different GA parameters is needed and the adaptive change of these parameters during the iterations would be a possible extension of our algorithm.

TABLE IV
Genetic algorithm solution for the problem 3

Employee/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	A	A	A	A	-
2	-	-	D	D	D	A	A
3	N	N	-	-	-	D	D
4	D	D	D	-	-	-	A
5	A	A	A	N	N	-	-
6	D	D	A	A	N	N	N
7	-	-	-	A	A	N	N
8	N	-	-	-	A	A	N
9	N	-	-	-	A	A	A
10	A	N	N	-	-	D	D
11	D	D	D	D	-	-	-
12	A	A	N	N	N	-	-
13	A	A	A	A	N	N	-
14	-	-	D	D	D	N	N
15	N	-	-	D	D	D	A
16	A	A	-	-	D	D	D
17	D	N	N	N	-	-	-

TABLE V
Genetic algorithm results for 10 runs

Example	Iterations	Evaluations
Butler Problem	4850	485070
Laporte Problem	5318	531840
Heller Problem	17383	3302941

Acknowledgment

This work was supported by FWF (Austrian Science Fund) Project No. Z29-N04.

References

[1] Nagraj Balakrishnan and Richard T. Wong. A network model for the rotating workforce scheduling problem. Networks, 20:25–42, 1990.

[2] B. Butler. Computerized manpower scheduling. Master's thesis, University of Alberta, Canada, 1978.

[3] D. E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[4] N. Heller, J. McEwen, and W. Stenzel. Computerized scheduling of police manpower. St. Louis Police Department, St. Louis, MO, 1973.

[5] J. H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, 1975.
[6] R. Hung. A multiple-shift workforce scheduling model under 4-day workweek with weekday and weekend labour demands. J Opl Res Soc, (45):1088–1092, 1994.
[7] P. Knauth. Designing better shift systems. Appl Ergonom, (27):39–44, 1996.
[8] G. Laporte. The art and science of designing rotating schedules. Journal of the Operational Research Society, 50:1011–1017, 1999.
[9] G. Laporte, Y. Nobert, and J. Biron. Rotating schedules. Eur. J. Ops. Res., 4:24–30, 1980.
[10] Hoong Chuin Lau. Combinatorial aproaches for hard problems in manpower scheduling. Journal of Operations Research Society of Japan, 39(1):88–98, 1996.
[11] Nysret Musliu. Applying tabu search to the rotating workforce scheduling problem. In The 5th Metaheuristics International Conference (MIC'03), Kyoto, Japan, 2003.
[12] Nysret Musliu, Johannes Gärtner, and Wolfgang Slany. Efficient generation of rotating workforce schedules. Discrete Applied Mathematics, 118(1-2):85–98, 2002.
[13] James M. Tien and Angelica Kamiyama. On manpower scheduling algorithms. SIAM Review, 24(3):275–287, 1982.