A MOEAD-Based Approach to Solving the Staff Scheduling Problem

Feng Hong, Hao Chen, Bin Cao(✉), and Jing Fan

Zhejiang University of Technology, Hangzhou, China
{hongfeng,chenhao,bincao,fanjing}@zjut.edu.cn

Abstract. Due to the impact on increase of the utilization efficiency of the staff and decrease of operating cost of enterprises, the staff scheduling problem has attracted the interests of many scholars. Actually, the staff scheduling problem can be considered to be how to assign the right staff to the right shift on the right time period based on constraints, meanwhile the objectives should be optimized. Hence, designing an algorithm to satisfy all the requirements mentioned above is challenging. First, there are prohibitive combinations of assigning the staff to shifts from a sheer numbers perspective; Next, there are potential conflicts among optimization objectives, which means objectives may not reach the optimization at the same time and the optimal schedule can not be found; Finally, rare work about the fairness of optimization objectives has been studied. The existing works usually focus on optimization objectives in total, ignoring the fairness of them. A schedule with best optimization objectives can not provide the highest fairness. Hence, we propose an approach based on multi-objective evolutionary algorithm based on decomposition (MOEAD) to solve the staff scheduling problem in the fairness aspect. A series of experiments are performed and prove that the proposed method can effectively find the schedule with fairness.

Keywords: Multi-objective optimization · MOEAD · Staff scheduling · NP-hard

1 Introduction

The staff scheduling problem has drawn significant attention during the last few decades, and it can be considered to be how to assign the right staff to the right shifts on the right time period [1]. A schedule with high quality contributes to improve the utilization efficiency of the staff and reduce the operating cost of the enterprises [2,3].

The staff scheduling problem and some variants are well known as NP-hard [4] and attract the interests from both industry and academia [5,6]. However, existing works usually take into account the optimization of the objectives such as minimizing the personnel costs, while ignoring the fairness of these objectives.

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 350, pp. 112–131, 2021.
https://doi.org/10.1007/978-3-030-67540-0_7

A schedule with minimal cost does not guarantee the highest fairness, which increases the complexity of the staff scheduling problem due to the resulting trade-off between efficiency and fairness. Actually, the staff scheduling in the fairness aspect is a multi-objective optimization problem.

In this paper, what we need to take into account is: (a) the constraints that the scheduler must be considered to make a schedule, and (b) the evaluation metrics used to measure a schedule. We divided the constraints into two categories: (1) *Hard constraint*. The schedule is invalid if any one of them is broken, and (2) *Soft constraint*. It is desirable to meet the soft constraint. Hence, we make a strategy to guarantee that the hard constraint would not be broken. Each schedule will be tested for whether this schedule violates hard constraints. If the schedule violated hard constraints, the shifts of violating them will be substituted according to the strategy. Then this schedule performed by the operation of substitution will be treated as a potential schedule option. As for the soft constraint, the number of violating it is set to an evaluation metric *Shifting_Satisfaction*, which gets higher as the number of violating soft constraints is less. Besides, there are other evaluated metrics, i.e., average workload coverage *Ave_Coverage* and coverage fairness *Fairness_Coverage*. The staff deployed is asked for working as effectively as possible, *Ave_Coverage* is used to measure whether the number of the staff can cover the work demand at different times of each day. But due to constraints for scheduling, the number of the staff on the shifts of different times is varied, and the workload coverage may be different, the *Fairness_Coverage* is introduced to be an evaluation metric. Based on these metrics, the schedules will be compared with each other. The evaluation result of the schedule is worse by others, we say that this schedule is dominated by others. For example, if two schedules s_i and s_j satisfy the hard constraint, yet s_i would result in less shifting coverage, less workload coverage, and less coverage fairness than s_j, we say that s_i dominates s_j. Then, we substitute s_j and return those schedules that are *not* dominated by others as the results of our approach.

One simple method to find the optimal schedule is to compute all the combinations of shifts for the staff. However, generating the optimal schedule in this way needs prohibitive expensive as it encounters a large number of combinations of shifts assigned to the staff.

What is more, there are potential conflicts among evaluation metrics, which means that evaluation results may not reach the optimization value at the same time and the optimal schedule may not exist. Hence, finding a set of (near)optimal schedules is an acceptable way, and we propose an approach based on multi-objective evolutionary algorithm based on decomposition (MOEAD [7]) to address our issue. Our approach can be divided into three steps. In the first step, we encode the schedule as an individual, and a series of individuals constitute a population. In the second step, we perform the operations of mutation and crossover to generate a series of new schedules. During these steps, we perform our substitution strategy to ensure that each schedule is valid. Then these schedules after the strategy will be treated as potential schedule options to be compared with those schedules in the original population. In the third step, the

newly generated schedules will be measured by evaluation metrics and their evaluation results will be compared with original schedules, where the schedules with better evaluation results will be selected to be made up to a new population. These three steps will be performed consecutively until the number of repeating reaches the pre-set value. Then the schedules of the last population are the set of (near)optimal schedules.

The experimental analysis also shows the performance of our approach. In general, the contribution of this paper can be summarized as follows:

1. Previous multi-objective evolutionary algorithms (MOEAs) rarely take coverage fairness as an optimization objective used to measure the schedules. In our work, we model the fairness coverage based on the real-world scene and use it to measure the schedules, which makes the evaluation metrics more reasonable.
2. We adopt MOEAD to solve the proposed problem by converting the multiple objective into several singe-objectives. Every single objective problem is optimized, and the corresponding (near)optimal schedules are found and treated as the solutions to our problem.
3. We execute a series of extensive experimental evaluations to show the performance of MOEAD on our problem. Compared with three other MOEAs (the nominated sorting genetic algorithm II (NSGA-II [8]), the reference vector guided evolutionary algorithm (RVEA [9]) and the preference-inspired coevolutionary algorithms [10]), MOEAD has been proven that its mechanism of decomposition can effectively address our problem.

The rest of the paper is organized as follows: Sect. 2 introduces the related work of our algorithm; Sect. 3 introduces the preliminaries for helping readers to understand our algorithm; Sect. 4 introduces the main contents of the algorithm; Sect. 5 describes the experiment contents; Finally, Sect. 6 summarizes the full text.

2 Related Work

Current algorithms to address the staff shifting problem can be classified into three categories. The first category is the heuristic algorithm. Leksakul et al. [11] proposed a scheduling algorithm based on genetic algorithms to minimize the objective functions based on the overtime cost of nurses. Their algorithm was able to reduce 12% of the staffing cost and 13% of overtime costs. Compared with the classical search algorithms based on genetic algorithm [12], the new memetic algorithms [13] can find better results in terms of global optimization and produce more number of optimization results under the condition of balanced development and exploration in the search space. However, these algorithms do not measure the satisfaction of soft constraints, which determines the quality of the final results. Then the bee colony algorithm [14] is proposed to satisfy the above problem, but the fairness of workforce satisfaction between different days in the schedules remains an issue. Our algorithm introduces the

idea of workforce satisfaction standard deviation, and it's treated as a metric to evaluate the schedules.

The second category to address the staff shifting problem is based on mathematical programming. Various mathematical models and integer programming techniques are proposed to solve the staff shifting problem. Howell [15] proposed a cyclical model to create a schedule within a duration. Due to a large number of infeasible solutions in a cyclic shifting, a method of modeling scheduling problems with mixed integer programming is studied. Pour et al. [16] proposed a hybrid framework for staff shifting problem, they implemented the global constraint programming model to find the (near)optimal solutions. Compared with constraint programming, this model can produce better solutions by taking the requirements of staff into consideration. Hamid et al. [17] proposed a multi-objective nurse scheduling mathematical model, which takes the decision-making style of nurses into account. Then the data envelopment analysis method is used to sort the obtained Pareto solutions. Although this category of the method has high effectiveness, a large amount of computation leads to low efficiency and high responding time. However, our algorithm can generate (near)optimal solutions with less responding time, which is considered as improving the exceeding speed of the algorithm based on a bit reduction of effectiveness.

The third category is the multi-objective evolutionary algorithm. According to the evolutionary mechanism, they can be divided into MOEAs based on domination [18,19], MOEAs based on decomposition [7,20], and MOEAs based on indicators [21–23]. The main idea of MOEAs based on decomposition is to decompose multi-objective optimization problems into several scalar quantum problems for optimization. Each sub-problem is optimized by using information from its several neighboring sub-problems. And in this paper, the MOEAD is the most classical algorithm among the MOEAs based on decomposition. Though multi-objective evolutionary algorithms have natural advantages in solving the staff shifting problems, it is usually an algorithm that has the best effect in solving a specific problem due to the different constraints in the problem. Our algorithm performs the evolution based on the uniformly distributed weight vector, which makes that the solutions of the algorithm will converge to Pareto Font more quickly. Besides, MOEAD has been proven that it will perform well in dealing with similar problems [7].

3 Preliminaries

This section presents a set of preliminaries that are important to set the stage for understanding our algorithm and its vision. In particular, we will introduce our preliminaries from following aspects: basic concepts, scheduling objectives and our problem definition.

3.1 Basic Concepts

Shifts and scheduling constraints are two basic concepts for staff scheduling problems. In our work, we present them as follows:

Shifts. Each day is divided into two shifts, namely the day shift (*Day_shift*) and the night shift (*Night_shift*). Every day, each shift requires a certain number of staffs to finish the job and each staff can work at most one shift, i.e., he can take this day off if both shifts on this day have been assigned with enough workers. Note that different shifts on different days may require different numbers of staffs.

Scheduling Constraints. When assigning staffs to shifts, constraints like staffs' preferences, business demands or legal regulations should be taken into account. Usually, there are two types of constraints, i.e., hard constraints and soft constraints.

- **Hard constraints.** The entire schedule is invalid if any constraint of this type fails. In our work, we mainly consider following two hard constraints: (1) $\boldsymbol{H}_{osod}$. *Each staff can only be assigned to at most one shift per day;* and (2) $\boldsymbol{H}_{mwd}$. *Each staff can work at most k consecutive working days.* The second hard constraint actually indicates two different cases of how a staff can take holiday. Specifically, a staff must take at least one day off if he had worked k consecutive days. In the other case, it is also allowable if a staff rest a few days within a period of k consecutive days.
- **Soft constraints.** Violating these constraints will not make the schedule invalid, but it is desirable to meet them. In our work, we merely consider one soft constraint: *If one staff was assigned to the night shift (Night_shift) on one day, assigning a day shift on the next day is not allowed.*

3.2 Schedule Evaluation

Given a group of staffs N and a scheduling horizon H, it is possible to find more than one valid schedule that can satisfy the aforementioned hard and soft constraints. Hence, to determine the optimal schedule, we propose three evaluation metrics, i.e., *shifting satisfaction*, *average workload coverage*, and *coverage fairness*.

Shifting Satisfaction. For a valid schedule, though the soft constraint on shifts rotation for a staff could be violated due to reasons like insufficient workforce or the impact of hard constraints, it is better to reduce the number of violations as many as possible. Because the employee satisfaction matters a lot in real scenarios. Hence, we propose the metric of shifting satisfaction to measure the degree of soft constraint violation, and it can be computed as follows:

$$Shift\_Satisfaction = 1 - \frac{k}{|N|(|H|-1)} \times 100\% \tag{1}$$

where the $|N|$ denotes the number of total staffs, the $|H|$ is the number of days in a scheduling horizon, k represents the number of violations on the soft constraint.

Since each staff can only be assigned at most one shift per day and each shift assignment after the first day corresponds to a judgment that whether the assigned shift follows the soft constraint, i.e., whether the day shift is arranged

after the night shift for a staff, the number of soft constraints that are needed to be tested for each staff should be $|H|-1$. Then for $|N|$ staffs, the total number of judgments on the soft constraint is $|N|(|H|-1)$. Hence, the shifting satisfaction will finally be represented as $1-\frac{k}{|N|(|H|-1)}$. Obviously, the value of shifting satisfaction ranges from 0 to 1, and it will reach the maximum value of 1 if there is no soft constraint violated.

Average Workload Coverage. One of the critical issues that a schedule must follow is to have the staff deployed as effectively as possible, and this usually means whether the number of the staff at work can cover the work demand of different times of the day. Here the work demand in our paper refers to the workload of different shifts on different days, e.g., the minimal number of required staffs to finish the job for the corresponding shift. Considering that the workload in real scenarios may vary by different shifts, we use the average workload coverage for the past $|H|$ days to reflect the overall coverage level, and it can be computed as follows:

$$Ave\_Coverage = \frac{1}{2|H|} \times \sum_{j=1}^{|H|} (Coverage_{H_j}^{Day\_shift} + Coverage_{H_j}^{Night\_shift}) \quad (2)$$

where $Coverage_{H_j}^{Day\_shift}$ and $Coverage_{H_j}^{Night\_shift}$ respectively denotes the coverage of the day shift and night shift on day H_j. Specifically, the day/night shift coverage equals to the ratio of the number of assigned staffs to the number of required staffs. Note that, since two shifts are involved on each day, the average workload coverage for the scheduling horizon of H days should be divided by 2.

Coverage Fairness. Due to the existance of hard and soft constraints for scheduling, the workload coverage for different shifts on different days may vary a lot. For example, suppose the workload for the day shift and night shift is 30 and 20 respectively for one day, and 40 staffs in total are available for scheduling, how many would really be put in place? If we choose to have schedules that fully cover the day shift, then the maximal number of staffs that can be arranged for the night shift would be only 10, and the average coverage of this day is 0.75 $((30/30+10/20)/2)$. Compared with the 100% coverage of the day shift, the night shift coverage is merely 50%. The coverage deviation of these two shifts are too huge to be applied in real scenarios, in fact, this deviation can cause the problem of overstaffing and understaffing. To avoid above problems and achieve the fairness in staff scheduling, we calculate the standard deviation for the coverage of all the shifts as follows:

$$Coverage\_Fairness = \left\{ \frac{1}{2|H|} \times \sum_{j=1}^{|H|} \left[(Coverage_{H_j}^{Day\_shift} - Ave\_Coverage)^2 + (Coverage_{H_j}^{Night\_shift} - Ave\_Coverage)^2 \right] \right\}^{1/2} \quad (3)$$

Take the above example again, the coverage fairness is $[(1-0.75)^2+(0.5-0.75)^2]^{1/2}/2 = 0.18$. Now suppose the day shift and night shift are now assigned

with 21 and 16 staffs, though the average coverage is still 0.75, the coverage deviation becomes smaller, i.e., the day shift coverage is 0.7 while it is 0.8 for the coverage of night shift, hence the coverage fariness is 0.04. Moreover, three staffs in this schedule can take off this day, which then can provide some room for future staffing.

3.3 Problem Definition

Definition 1. Given a group of staffs N and a scheduling horizon H, the shifts *Day_shift* and *Night_shift* are assigned to staffs, along with shifting satisfaction *Shift_Satisfaction*, average workload coverage *Ave_Coverage* and coverage fairness *Coverage_Fairness*, the goal is to find a set of schedules S, where each schedule ∀s∈S, the following hold:

1. Following hard constraints must be satisfied,
   - Each staff can only be assigned to at most one shift per day;
   - Each staff can work at most k consecutive working days;
2. argmax *Shift_Satisfaction*;
3. argmax *Ave_Coverage*;
4. argmin *Coverage_Fairness*;
5. s is in the set of Pareto solutions.

There are many staffs remaining to be assigned to shifts, it's difficult from a sheer number perspective with so many different possible combinations of schedules to optimize all the evaluated metrics. Besides, there may be potential conflicts among the evaluated metrics, which means that all the evaluated metrics may not reach the optimization at the same time. Under this situation, finding the (near)optimal schedules is acceptable way. The evaluated results of different metrics are hard to be compared with each other for selecting (near)optimal schedules, which makes finding (near)optimal schedules challenging. Actually, some variants of staff scheduling problem have been proven NP-hard [4].

3.4 Problem Definition

Definition 2. Given a group of employees E and a scheduling horizon H, where each employee $e \in E$ has the performance P_e, the shifts *day_Shift* and *Night_shift* are assigned to employees, along with shifting satisfaction *Shift_Satisfaction*, average workload coverage *Ave_Coverage* and coverage fairness *Coverage_Fairness*, our algorithm finds the optimal schedule, the following hold:

1. Following hard constraints must be satisfied,
   - Each staff can only be assigned to at most one shift per day;
   - Each staff can work at most k consecutive working days;
2. argmax *Shift_Satisfaction*;
3. argmax *Ave_Coverage*;
4. argmin *Coverage_Fairness*;

Scheduler often face the situation about assigning lots of employees to the shifts, it's a difficult job from a sheer numbers perspective. Since so many possibilities and combinations of potential options need to be taken into account while maximizing the Coverage_Fairness, minimizing the Ave_Coverage and the Shifting_Satisfaction. Besides, these optimization goals may potentially conflict with each other, which means that they may not reach the optimal value at the same time. What's more, not only are there countless possibilities for scheduling options, but juggling these and deciding on the right combination is very challenging because of all the employees involved in the process. For example, the workload of day shift on Monday is 2, the performances of employees A, B and C are 1, 1 and 2. Employee A and B can work together to satisfy the demand of day shift, employee C can finish the job alone. Although different employees can be assembled to perform the job, how to choose right combination remains to be a challenging problem.

4 MOEAD-Based Shifting

A naive way to find the optimal schedule is to compute all the combinations about assigning staffs to shifts. Though the solution looks simple, if all the combinations are taken into consideration, the computation is so prohibitive that we can't afford to support it. Actually, the staff scheduling problem is a multi-objective optimization problem. We adopt MOEAD to convert our problem into several single-objective optimization problems based on the idea of decomposition. And the original evaluation metrics are transformed into a certain metric called mapping distance by the Tchebyshev function [24], each single objective optimization problem will be optimized by this metric, we will get a set of (near)optimal schedules as the solutions to our problem.

In this section, we first overview MOEAD, then modeling our problem based on MOEAD. The detail of modeling can be divided into three steps: (a) encoding the individual and population, (b) the operations of mutation and crossover, and (c) the selection of a set of (near)optimal schedules.

4.1 Overview of MOEAD

MOEAD adopts the mechanism of decomposition to convert the multi-objective optimization problem into multiple single objective optimization sub problems and works concurrently to solve these sub problems. Each sub problem is optimized with the help of the information gained from its neighborhood [25].

MOEAD uses the mechanisms inspired by biological evolution, such as individual, population, generation, mutation, crossover, fitness function and selection. An individual represents a solution to the problem, the population consists of a series of individuals and denotes a set of solutions. During the process of MOEAD, the initialized population is called the first generation. Then by the operation of mutation and crossover, the individuals in initialized population will be generated new ones. All the individuals will get scored by the fitness

function, where some individuals with higher score will be selected to make up of a new population. This new population is considered as the second generation. MOEAD will repeat the above operations on the previous generation to generate the next until the number of generations reaches the pre-set value. The individuals in the final generation are considered as the (near)optimal solutions.

However, the fitness function is composed of different evaluation metrics, and the evaluated results of different solutions are hard to be compared with each other. For example, the *Shifting_Satisfaction* and *AVE_Coverage* of solution A are 0.7 and 0.5, and those of solution B are 0.5 and 0.7. weighing up with these two solutions, B has a higher AVE_Coverage, but A owns better Shifting_Satisfaction. It's a difficult job to compare the solutions from different evaluation metrics. MOEAD adopts idea of mechanism to use the Tchebyshev function [24] to solve this situation. The function is as follows:

The Tchebyshev function is shown as follows:

$$C(x \mid w) = \max_{i \in \{1,\ldots,m\}} w_i \mid f_i(x) - P_i \mid \tag{4}$$

where x represents the set of solutions, the $f_i(x)$ is the computing functions of metrics and m is the number of metrics, w_i denotes the weight on the metric $f_i(x)$, the $C(x|w)$ is the mapping distance between schedules x and weight vector w, P_i represents the best evaluated result of the metric $f_i(x)$ among all the solutions.

The weight vector represents a sub problems, and is uniformly distributed in the space of solution, each weight vector corresponds to each individual The original three evaluation metrics are transformed into only one metric called mapping distance, as Fig. 1 shows. Suppose that there are two evaluation metrics F1 and F2, a schedule is positioned as Fig. 1 shows. The schedule is mapped to the vector w_1, and the mapping point on the w_1 is c, the distance between the origin and c is called mapping distance. If this schedule is comared with others, these schedules will be mapped to the weight vector w_1 corresponds to the schedule and be computed to the mapping distances, which are used to be compared with the schedule'. Thus, the mechanism of decomposition contributes to selecting the feasible schedules.

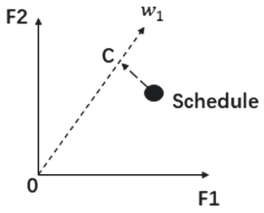


Fig. 1. The diagram of schedule mapping to the weight vector

Original evaluated results of different metrics are converted into a mapping distance on the weight vector. The solution gets better as the mapping distance

is shorter. Then the solutions can be compared with each other by their mapping distances, and the solutions with shorter mapping distance will be selected to make up of the next generation.

4.2 Encoding the Individual and Population

In our problem, the individual denotes a schedule, and the population represents a set of schedules. Then we will introduce the process of encoding the individual and population as the following steps: (1) Encoding the shifts of schedules; (2) Splicing the shift sequence of each day; (3) Setting the constrained control of schedules.

Encoding the Shifts of Schedules. Schedule is fulfilled with shifts, dates and staffs as shown in Fig. 2(a). Note that the staffs can not be assigned to two different shifts on one day in the schedule of our problem as the hard constraint H_{osod} asks. In MOEAD, the individuals are asked for being transformed into a vector. Hence, both the individual and the population need to be encoded. We use the example of schedule as Fig. 2(a) to explain the operations of encoding.

Timetable T	Staff 1	Staff 2
Day 1	Night shift	Night shift
Day 2	Day shift	Rest-day

(a) the example of schedule

Timetable T	Staff 1	Staff 2
Day 1	2	2
Day 2	1	0

(b) the example of schedule where the shifts are encodeed

Fig. 2. The diagram of encoding the shifts

The shifts consist of *Rest-day*, *Day shift* and *Night shift*. We use the number 0 to represent the *Rest-day*, number 1 and 2 denote the *Day shift* and *Night shift*. Then the example of schedule is shown in Fig. 2(b) after the shifts are encoded.

Splicing the Shift Sequence of Each Day. The schedule needs to be transformed into a vector. A schedule can be treated as the combination of shift sequence on each staff, e.g., the shift sequence of staff 1 is 21, and staff 2 has the shift sequence 20. Then the shift sequence of each staff is spliced according to the staffs' number, the quantities of staffs and days in the schedule are unique. Hence, we can know that each number on the sequence represents this shift is performed by which staff on which day, and the information of the schedule will not be lost. As Fig. 3 shows, the schedule is transformed into a vector, where the shift sequence of each day is connected according to the staffs' number. By the mentioned above operations, the new schedule T' is generated. The number of days in the original schedule T is 2, every two numbers on the new schedule T' is a shift sequence and represents the shifts performed by a staff. For example, the first two numbers 2 and 1 denote the shift sequence of staff 1, and the shift sequence of staff 2 is the next two numbers 20. Note that the shift sequence of

each staff is arranged according to the day, and the shift sequence 12 makes no sense. The shift sequence (21) of staff 1 represents that the staff 1 performs the night shift on Day 1 and the day shift on Day 2. All the individuals are encoded into a vector based on the above transformation.

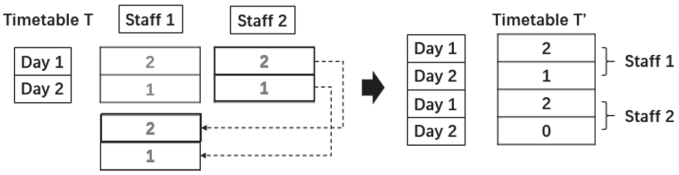


Fig. 3. The diagram of splicing the shift sequence of each day

The population is composed of individuals, each individual is a vector, and the population is a set of vectors.

Setting the Constrained Control of Schedules. For our problem, there are several constraints, which can be divided into two categories: hard constraints and soft constraints. The hard constraints must be followed. However, the original operations of mutation, crossover and selection in MOEAD can not provide the guarantee that each individual will follow the hard constraints. Hence, we need to set a optimization strategy to ensure that the hard constraints will be followed.

There are two hard constraints in our problem: (1)H_{osod}, each staff can only be assigned to one shift per day; (2)H_{mwd}, the maximal number of consecutive working days do not exceed k days. First, during the process of encoding the individuals, the hard constraint H_{osod} has bee set to be followed. All the schedules used in MOEAD will not appear that one staff is assigned to two different shifts on one day. As for the hard constraint H_{mwd}, we traverse each individual, where the examination is performed in a sliding window of k+1 consecutive numbers and its step is one number. More in detail, the rest-day is encoded to the number 0, we multiply these numbers and get a value. When the value equals to 0, some these numbers is 0, which means the the assignment of these k consecutive shifts obeys the hard constraint H_{mwd}. Otherwise, the last one of these $\mathrm{k}+1$ consecutive numbers will be replaced with the number 0. We repeat above operations until the lower bound of the window reaches the last number of this staff. The sliding window will take the first number as the upper bound to continue the examination.

For example, as shown in Fig. 4. Suppose that the maximal consecutive working days is 1, and the population consists of three staffs and each staff is assigned to shift on three consecutive days. The length of sliding window is 2, and the upper and lower bound is the first two numbers as Fig. 4(a) shows. The value of multiplying first two numbers is 4, which do not equal to 0. The night shift represented by the lower bound of sliding window break the hard constraint H_{mwd}.

Then this night shift will be reset to the rest-day and the window will slides down one number as Fig. 4(b). When the lower bound of sliding window reaches the last shift performed by staff 1, it will continue the examination from the first shift of next staff (staff 2) as Fig. 4(c) shows. All the shifts of schedule T1 are examined as Fig. 4(d) shows, the sliding window will start the examination from the first shift of next schedule until all the schedules are performed the examination.

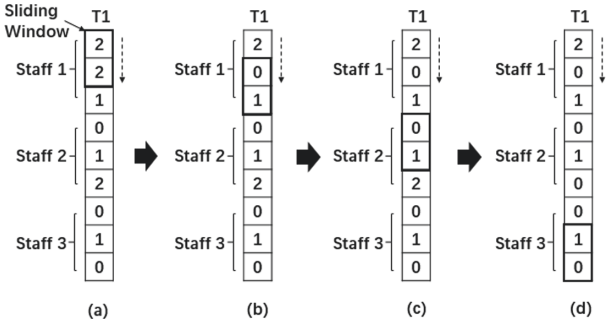


Fig. 4. The diagram of examining the individuals

Compared with abandoning the schedules which break the hard constraint H_{mwd}, substituting the shifts breaking H_{mwd} is more reasonable. Because the number of individuals in the population is fixed. If the schedules which do not obey the hard constraints H_{mwd} are abandoned, we need to supplement the same number of schedules to keep the number of individuals in the population constant. This way works inefficiently and costs a lot of extra computations, and our substitution strategy is more feasible.

4.3 The Operations of Mutation and Crossover

The operations of mutation and crossover are performed on individuals of the previous population to generate new schedules. The mutation can be considered that some of a schedule is replaced with the part of another one according to the mutation point, note that the mutation point is randomly generated. For example, as shown in Fig. 5, there are three encoded schedules T1, T2 and T3. We generate randomly the mutation point of T1, the corresponding shift in the position of mutation point is 0, and the shifts of T2 and T3 with the same position are 1 and 2, which means the shift 0 of the position of mutation point on T1 can be substituted by the shifts 1 and 2. Which shift will be used to substitute the shift 0 depends on the polynomial of MOEAD. The shift 0 is an input to the polynomial, the computed result is the shift 2, then the shift 0 with the position of mutation point on T1 can be substituted by the shift 2.

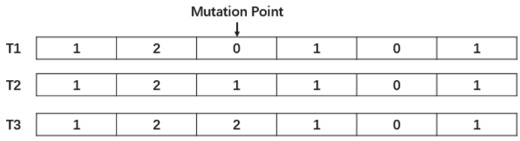


Fig. 5. The example operation for mutation

As for the crossover, MOEAD adopts the simulated binary crossover strategy to achieve it. It can be treated as exchanging some shift sequences of two schedules to generate new schedules according to the crossover point, note that the crossover point is randomly generated.

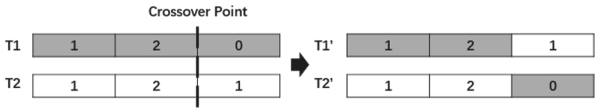


Fig. 6. The example operation of crossover

For example, there are two schedules T1 and T2, we randomly generate the crossover point on these two schedules. The schedules will be divided int shift sequences according to the crossover point. Thus, we get four shift sequences 12, 12, 0, and 1. Then we splice these shift sequences which are not in the same schedule as Fig. 6 shows. We get two new schedules T1′ and T2′.

4.4 The Selection of a Set of (Near)optimal Schedules

The input to this section is the new schedules generated by the operations of mutation and crossover, and they will be compared with the schedules in the previous population. As the first subsection in this section shows, the Chebyshev function converts the evaluated metrics into a mapping distance between weighted vectors and three evaluated metrics, and this mapping distance is used to be a metric for schedules to compared with each others.

This selection can be divided into two steps: (1) Converting the evaluated metrics into the mapping distance; (2) Comparison between original schedules and new generated ones.

Converting the Evaluation Metrics into the Mapping Distance. Each schedule can be considered as a point in the solution space, and the mapping distance can be understood as an Euclidean distance between the point and origin point of the solution space. Note that this Euclidean distance needs to be converted into a distance on the corresponding weight vector by the Chebyshev function.

As Fig. 7 shows, suppose that there is a schedule which is represented by the point X in the coordinate system. The weight vector w responds to point

X. First, we connect the point X and the origin o. Then take one point A on w, vectors $\overrightarrow{OX}$ is perpendicular to $\overrightarrow{XA}$. The distance of OX can be computed based on Euclidean distance function, and the angle between the vector $\overrightarrow{OX}$ and weight vector w can be computed. Then we can get the length of OA, which is the mapping distance. The evaluated results of three metrics are converted into a distance OA.

Similarly, the schedule will be compared with its neighboring schedules, they will be mapped to the weight vector w and are generated to the mapping distances. These mapping distances will be compared with that of x. If none of them has shorter mapping distance than x, x will be keep. Otherwise, any of the schedules that have shorter mapping distance can be selected to substitute x.

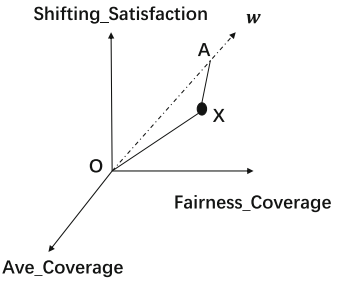


Fig. 7. The diagram of one schedule mapping to weight vector w

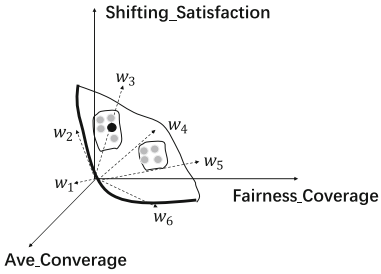


Fig. 8. The diagram of selecting the schedules

Comparison Between Original Schedules and New Generated Ones. Combined with Fig. 8, we explain the detail of replacing. Suppose that a schedule (black point) corresponds to w_3, and the number of weight vector in the domain of w_3 is 4, the schedule will be selected randomly from four schedules (four gray point around black point) as a parent, and will be performed the operation of mutation and crossover with black point to generate a new schedule. Then we perform the operation of replacing. Three schedules (a selected gray point, black point and a new one) will be computed the mapping distance by Chebyshev function. Then the mapping distance of new schedule will be compared with parents'. If one of parents has longer mapping distance than the new one's, this parent will be replaced the new schedule. If both of them are longer than the new one's, choose any one of parents to be replaced. If the mapping distance of new schedule is longer than each parent's, the new schedule will be abandoned.

The operation of replacement will be performed in the order of weight vectors on each generation. When all the operations of replacement were finished, the next generation would have been generated.

5 Experiments

This section describes the experiments used to test the ideas outlined in the previous section. All experiments are performed based on a mixture of real and synthetic data sets. The real part comes from Hangzhou center of China Telecom, containing the call arrivals of each day from June 2019 to November 2019. The synthetic data set is the staff of each month during the above time period.

There are a series of measures to evaluate the quality of Pareto Fronts (PFs), and the hypervolume (HV) [26] is used into the following experiments. As for a three-dimension problem, HV is treated as the volume surrounded by each point on the PF and a point. HV is adopted to compare the performance of PFs. The larger the value of HV and the better the PF is.

In our work, in order to simulate the real scene, the schedule horizon of the experiment is set to one month. Besides, we assume that the performance of each staff is the same, and the number of staff is fixed. Each experiment will start from the same set of the initial population, and be executed 20 times. Our model is compared with four very effective algorithms for solving staff scheduling problems, including the nondominated sorting genetic algorithm II (NSGA-II [8]), the reference vector guided evolutionary algorithm (RVEA [9]) and the preference-inspired coevolutionary algorithms [10]. NSGA-II relies on its elite selection strategy, fast non-dominant sorting, and crowded distance calculation to find the (near)optimal schedules, these characteristics guarantee the generated schedules effectiveness and efficiency. RVEA introduces the idea of the reference vector, and achieve the search of (near)optimal schedules based on angle constrained local search. As for PICEAg, it finds the (near)optimal schedules based on co-evolutionary of candidate solutions and optimization functions. These three MOEAs achieve better performance than others in practice. All experiments are evaluated on a laptop with Intel(R) Core(TM) CPU i3-2310M 2.10 GHz processor and 6 GB RAM with Windows 10.

5.1 Parameter Setting

It's significant to choose proper parameters for evolutionary algorithms. In our work, we determine parameters, the rate of mutation $rate_m$, the rate of crossover $rate_c$ and the size of population though experiments on a instance. In order to find the optimal value for $rate_m$ and $rate_c$ for all the algorithms, a large number of experiments are constructed. The maximal number of generations is set to 1000, first, and $rate_m$, $rate_c$ are increased from 0.1 to 1 in the step of 0.1. For each combination of $rate_m$ and $rate_c$, the average HV (AVE.HV) which is the average value of 10 HVs over 20 independent runs. For each algorithm, we choose the most suitable $rate_m$ and $rate_c$ based on the value of HVs, and the detail value is as Table 1 shows.

With the above parameters, experiments are further conducted to test the effect of population size. The population size is set to 100, 200 and 400, and 20 independent runs are conducted. The AVE.HV are reported in Table 2. Compared with other algorithms, the values of AVE.HV for MOEAD is better than

Table 1. The rates of mutation and crossover used in each algorithm

Parameters	MOEAD	NSGA-II	RVEA	PICEAg
Rate_m	0.1	0.2	0.9	0.7
Rate_c	0.7	0.8	0.8	0.8

others in general, and its HV gets better as the population size increases. The number of solutions in a larger population is more, which strengthens the ability to search for feasible schedules.

Table 2. The AVE.HV generated by varying the size of population in each algorithm

Population size	MOEAD	NSGA-II	RVEA	PICEAg
100	**3.362**	3.32	3.29	3.346
200	**3.384**	3.341	3.326	3.351
400	**3.413**	3.351	3.348	3.361

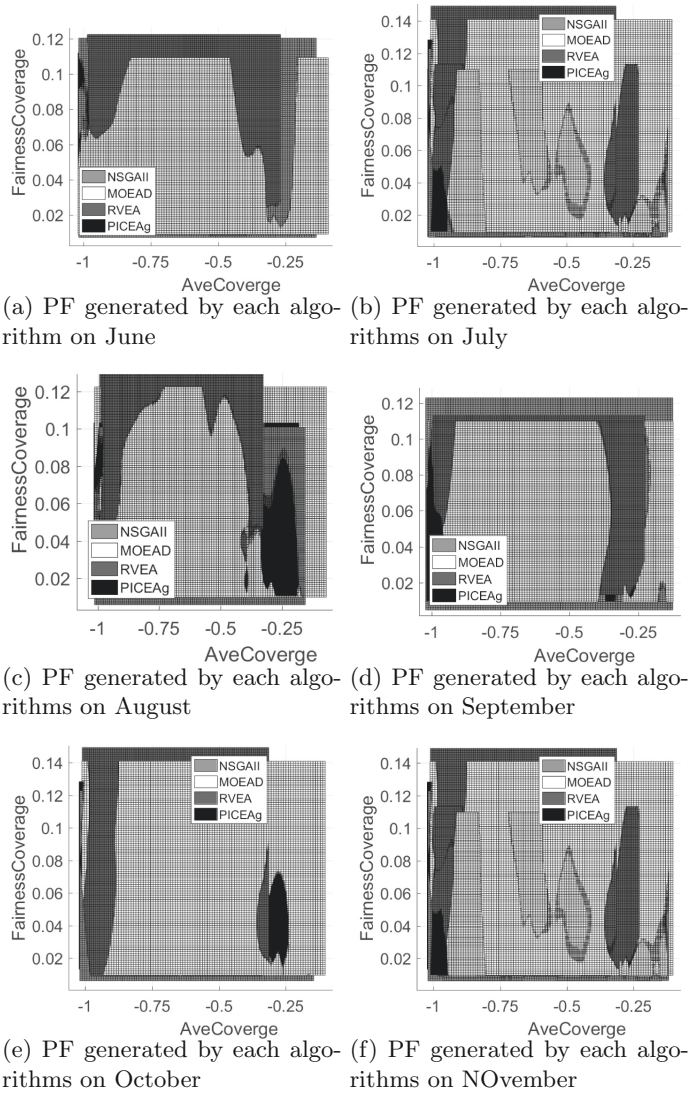
5.2 Overall Evaluation on Each Data Set

The scheduler needs to choose a feasible schedule from a set of metric schedules. Here we suppose that according to the actual requirement of *Shifting_Satisfaction*, *Ave_Coverage* and *Fairness_Coverage*, the scheduling scheme which makes *Shifting_Satisfaction*, *Ave_Coverage* be maximum, and *Fairness_Coverage* be minimum will be chosen. 20 independent runs of each algorithm are conducted with the data set of each month, respectively. Table 3 shows the AVE_HV.

Table 3. The AVE.HV produced by adopting different algorithms on each data set

Dataset	MOEAD	NSGA-II	RVEA	PICEAg
June	**7.1831**	6.6484	6.7101	6.8251
July	**6.2350**	6.1819	5.5967	5.8449
August	6.6285	**7.3530**	7.2048	6.5801
September	**6.1173**	2.705	2.6505	2.4207
October	**6.3404**	5.953	5.7377	5.829
November	**5.6613**	3.1141	3.3386	3.389

With the AVE.HVs of data set of each month, MOEAD performs better than other algorithms in general. As for the performance of August, it's caused by the weight vectors. Whatever the shape of PF is, MOEAD will use a fixed



(a) PF generated by each algorithm on June

(b) PF generated by each algorithms on July

(c) PF generated by each algorithms on August

(d) PF generated by each algorithms on September

(e) PF generated by each algorithms on October

(f) PF generated by each algorithms on NOvember

Fig. 9. The PFs of each algorithm on the data sets of 6 months

set of weight vectors and all the sub-problems are assigned the same amount of computation in each generation, which causes that some of PF is harder to be converged and the reduction of AVE.HV.

Besides, we select the best result of these algorithms for comparison, and Fig. 9 lists the comparison of PFs of all the algorithms. For facilitating observing the differences among PFs, we transform the Shifting_Coverage into the coverage of colors and present the PFs in a two-dimension chart. In this

two-dimension (Fairness_ Coverage and Ave_Coverage) aspect, a solution with lower Shifting_Coverage will be covered by others with a higher value. The colors shown in the figure represent that the corresponding solutions have higherShifting_Coverage. Then we can see the distribution of each PF directly and the proportion of its solutions on the approximate PF for our problem. Then as Fig. 9 shows, the color representing MOEAD has the largest area among all the algorithms, and the distribution of solutions towards argmax Ave_Converage and argmin Fairness_Coverage is superior to other algorithms in general. The decomposition mechanism of MOEAD transforms our problem into several single objective optimization problems, which explores the space of solutions. As for others, the non-dominant schedules of NSGA-II increases exponentially, which makes trouble in distinguishing the quality of schedules. RVEA is performed in the local search to find the (near)optimal solutions based on the reference vector, which reduces the space of solutions. And PICEAg is more suitable for dealing with the high-dimension problem, solving our problem does not make the co-evolutionary works well.

5.3 Evaluation of Varying Maximal Number of Consecutive Working Days

We add more choices for the scheduler about the maximal number of consecutive days based on the real-world scenes. Hence, we add the experiments about the maximal number of consecutive working days, which are set from 3 to 7. The parameters for each algorithm are set to a suitable situation. Each experiment is performed as the number of evolution is set to 5000, and runs 10 times. All the data in the table is the average of experimental results. According to the data in Table 4.

Table 4. The AVE.HV generated by varying the maximal number of consecutive working days of each algorithm

Working days	MOEAD	NSGA-II	RVEA	PICEAg
3	3.211	3.215	3.17	2.991
4	**2.865**	2.736	2.754	3.341
5	**2.8398**	2.1629	2.7732	2.8327
6	**3.3419**	3.3391	3.3389	3.0839
7	**3.3843**	3.1225	3.3120	3.3277

The days need to be assigned shift for the staff increases as the maximal number of consecutive working days decreases. For example, the maximal number of consecutive working days is set to 3, then for a month, the hard constraint *H_mwd* must be met, there are 10 days to rest and 20 days needs to be assigned to shifts at least. And the value is set to 7, there are 4 days to rest and 26 days needs to be assigned to shifts at least. When the number of working days increasing, the space of solutions will grow up. Similarly, the decomposition mechanism

of MOEAD works well in this situation. As for other algorithms, their working mechanism results in the worse schedules than MOEAD's.

6 Conclusion

In this paper, we adopt the MOEAD to address the staff scheduling problem, where MOEAD can optimize multiple objectives simultaneously and provides a set of feasible schedules instead of only one optimal schedule. The numerical experiment and a case study on 6 data sets from the real-world scene shows that MOEAD can find PFs of high quality than other classical MOEAs for this problem. Besides, we perform the experiments of varying the maximal number of consecutive working days, which verifies the applicability of our approach in the real-world scene. What's more, a series of experiments prove the effectiveness of the decomposition mechanism of MOEAD.

Acknowledgment. This research was partially supported by: National Key R&D Program of China (2018YFB1402800), and the Fundamental Research Funds for the Provincial Universities of Zhejiang (RF-A2020007).

References

1. Strandmark, P., Yi, Q., Curtois, T.: First-order linear programming in a column generation based heuristic approach to the nurse rostering problem. Comput. Oper. Res. **120**, 104945 (2020)
2. Akbarzadeh, B., Moslehi, G., Reisi-Nafchi, M., Maenhout, B.: A diving heuristic for planning and scheduling surgical cases in the operating room department with nurse re-rostering. J. Schedul. **23**(2), 265–288 (2020). https://doi.org/10.1007/s10951-020-00639-6
3. Maenhout, B., Vanhoucke, M.: An exact algorithm for an integrated project staffing problem with a homogeneous workforce. Kluwer Academic Publishers (2016)
4. Augustine, L., Faer, M., Kavountzis, A., Patel, R.: A brief study of the nurse scheduling problem (NSP). University of Pittsburgh Medical Center (2009)
5. Solos, I.P., Tassopoulos, I.X., Beligiannis, G.N.: A generic two-phase stochastic variable neighborhood approach for effectively solving the nurse rostering problem (2013)
6. Zhou, Y., Liu, J., Zhang, Y., Gan, X.: A multi-objective evolutionary algorithm for multi-period dynamic emergency resource scheduling problems. Transp. Res. Part E Logs Transp. Rev. **99**, 77–95 (2017)
7. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
9. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. **20**(5), 773–791 (2016)
10. Wang, R., Purshouse, R.C., Fleming, P.J.: Preference-inspired coevolutionary algorithms for many-objective optimization. IEEE Trans. Evol. Comput. **17**(4), 474–494 (2013)

11. Leksakul, K., Phetsawat, S.: Nurse scheduling using genetic algorithm,". Mathematical Problems in Engineering **2014**(pt.21), 1–16 (2014)
12. Aickelin, U., Dowsland, K.A.: An indirect genetic algorithm for a nurse-scheduling problem. Comput. Oper. Res. **31**(5), 761–778 (2004)
13. Lü, Z., Hao, J.-K., Glover, F.: Neighborhood analysis: a case study on curriculum-based course timetabling. J. Heurist. **17**(2), 97–118 (2011)
14. Todorovic, N., Petrovic, S.: Bee colony optimization algorithm for nurse rostering. IEEE Trans. Syst. Man Cybern.: Syst. **43**(2), 467–473 (2012)
15. Howell, J.P.: Cyclical scheduling of nursing personnel. Hospitals **40**(2), 77 (1966)
16. Pour, S.M., Drake, J.H., Ejlertsen, L.S., Rasmussen, K.M., Burke, E.K.: A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem. Eur. J. Oper. Res. **269**(1), 341–352 (2018)
17. Hamid, M., Tavakkoli-Moghaddam, R., Golpaygani, F., Vahedi-Nouri, B.: A multi-objective model for a nurse scheduling problem by emphasizing human factors. Proc. Inst. Mech. Eng. Part H: J. Eng. Med. **234**(2), 179–199 (2020)
18. Zhang, Z., Qin, H., Yao, L., Liu, Y., Jiang, Z., Feng, Z., Ouyang, S.: Improved multi-objective moth-flame optimization algorithm based on r-domination for cascade reservoirs operation. J. Hydrol. **581**, 124431 (2020)
19. Yu, X., Yao, X., Wang, Y., Zhu, L., Filev, D.: Domination-based ordinal regression for expensive multi-objective optimization. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 2058–2065. IEEE (2019)
20. Mashwani, W.K., Salhi, A., Yeniay, O., Jan, M.A., Khanum, R.A.: Hybrid adaptive evolutionary algorithm based on decomposition. Appl. Soft Comput. **57**, 363–378 (2017)
21. Tian, Y., Cheng, R., Zhang, X., Cheng, F., Jin, Y.: An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. IEEE Trans. Evol. Comput. **22**(4), 609–622 (2017)
22. Jiang, S., Zhang, J., Ong, Y.-S., Zhang, A.N., Tan, P.S.: A simple and fast hypervolume indicator-based multiobjective evolutionary algorithm. IEEE Trans. Cybern. **45**(10), 2202–2213 (2014)
23. Luo, J., Yang, Y., Li, X., Liu, Q., Chen, M., Gao, K.: A decomposition-based multi-objective evolutionary algorithm with quality indicator. Swarm Evol. Comput. **39**, 339–355 (2018)
24. Apostol, T.M.: Introduction to Analytic Number Theory. Springer, Heidelberg (2013). https://doi.org/10.1007/978-1-4757-5579-4
25. Omran, S.M., El-Behaidy, W.H., Youssif, A.A.A.: Decomposition based multi-objectives evolutionary algorithms challenges and circumvention. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) SAI 2020. AISC, vol. 1229, pp. 82–93. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52246-9_6
26. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Fonseca, V.G.D.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)