A Heuristic Method for Nurse Rerostering Problem with a Sudden Absence for Several Consecutive Days

Manabu Kitada¹, Kazuko Morizawa²

Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai-shi, Osaka, 599-8531, Japan

Abstract—Nurse rerostering problems arise when at least one nurse is absent from his/her assigned tasks in the predetermined schedule. The scheduler should modify the task assignments in the schedule to fill up the vacancy and to make a new schedule which satisfies all important constraints such as required numbers and skill levels of nurses at every-day-shifts, the limit of workload and prohibited shift patterns for individual nurses and requests for task assignments from nurses with as few modifications as possible. This paper proposes a heuristic method for solving nurse rerostering problems with an absence of nurses for several consecutive days. The method first separates the consecutive days of absences into a set of single-day absences and then solves each of the nurse rerostering problems with one-day absence one by one by using a recursive search technique and a tree search algorithm. Numerical experiments demonstrate that the proposed method can generate new feasible schedules with a small modification of task assignments efficiently.

Keywords— Combinatorial optimization, heuristic, nurse scheduling, recursive search, rerostering

I. INTRODUCTION

Nurse scheduling problem is a combinatorial optimization problem to make rosters for nurses who provide 24-hour-a-day care service in a hospital by working in shifts. The rosters should satisfy various constraints such as required number of nurses at each shift, workable combination of nurses and requests for task assignments posed by individual nurses [2, 5]. It is very hard to make a roster satisfying all these constraints completely. Thus the schedulers spend a lot of time to make a reasonable roster which is acceptable for both nurses and the hospital. Therefore, it is crucial to develop nurse scheduling methods that generate better schedules automatically.

The nurse scheduling problem is divided into two cases, the “static” and “dynamic”. The static nurse scheduling problem is to make a roster for a given scheduling horizon under given conditions. The term “nurse scheduling problem” is usually used in referring to the static case and various kinds of static nurse scheduling methods such as integer programming approach (e.g. [6]), a constructive heuristic approaches (e.g. [10, 11]), and meta-heuristic approaches (e.g. [1, 3, 4, 13]) have been proposed.

On the other hand, the dynamic nurse scheduling problem, often called “nurse rerostering problem,” has attracted less attention than the static nurse scheduling problem. The nurse rerostering problem is to reconstruct or modify the predetermined roster for the current scheduling horizon in order to respond to changing situations triggered by some kinds of accidents. One of the most typical accidents in hospitals is an absence of nurses. Since an absence causes a constraint violation of required number of nurses on the day, the schedulers should find a nurse who can fill up the vacancy of the absentee and modify some task assignments for the related nurses on and after the day in the current roster. Since the change of task assignments should be informed to the related nurses before the shifts start, reroster should be done as soon as possible. Moreover, the change of task assignments should be as small as possible to avoid confusion in implementing the tasks according to the predetermined roster.

For nurse rerostering, meta-heuristic approaches [9, 12, 14, 15] and a heuristic approach using recursive search technique [7, 8] have been proposed. However, all of these studies focus on the nurse rerostering problem with a one-day absence of one nurse despite the fact that absences for illness and injury sometimes continue for several consecutive days.

This paper formulates a nurse rerostering problem with several consecutive days of absence and proposes a heuristic method based on a recursive search technique for generating new feasible roster by extending the nurse reroster method proposed in [8]. Numerical experiments for three-shift nurse scheduling problems with real data in a Japanese hospital are implemented to demonstrate the effectiveness of the proposed method.

II. THE NURSE REROSTERING PROBLEM

This paper considers the three-shift nurse rerostering problems with the following assumptions. These assumptions represent the actual conditions of a Japanese hospital. Please note that the term “schedule” is used as a synonym for “roster” in the rest of this paper.
A. General Assumptions

General assumptions are the same as those given by Kitada et al. [7, 8].

(i) All nurses work in three shifts: night-shift (0:00-8:00), day-shift (8:00-16:00) and evening-shift (16:00-24:00).

(ii) Each nurse $i, i=1,2,...,N$, is assigned to only one task every day from among a day-shift (shift1), an evening-shift (shift2), a night-shift (shift3), a meeting (shift4) and a day-off (shift0). Every meeting is held during 8:00-16:00 (day-shift hours).

(iii) Each nurse belongs to one of the nursing modules $m$, $m=1,2,...,M$, and any temporary change of module is not allowed.

(iv) In each shift $k$ and in each module $m$, the number of nurses required to work together on each day $t$, $N_{stm}$ is prespecified individually.

(v) Nurses are classified into five ranks according to their own skill and experience, that is, ‘module leader (leader, in short),’ ‘highly experienced,’ ‘experienced,’ ‘second year’ and ‘newcomer.’

(vi) The required mean skill level of assigned nurses is prespecified for each shift $k$.

(vii) A skillful nurse (i.e., a ‘leader’ or a ‘highly experienced nurse’) should be assigned together with each newcomer so that any newcomer can get on-the-job training.

(viii) Scheduling horizon $T$ for static scheduling is one month ($T=30$days).

(ix) Lower and upper bounds for the number of shift assigned to each nurse $i$ during each scheduling horizon $LB_{ik}$ and $UB_{ik}$ are prespecified.

(x) Prohibited shift patterns to be avoided and standard shift patterns to be maintained are given in advance.

(xi) Requests for the task assignment such as days-off, meeting or special combination of shifts are submitted to a head/chief nurse by each nurse with their priority $p$ in advance. Requests for meeting have the highest priority.

(xii) Personal relationships among nurses are known as ‘good,’ ‘neutral’ and ‘bad.’ Nurses with bad relationship should not be assigned to a night- or an evening-shift if possible.

(xiii) The following historical information on task assignment to each nurse for the last two months is available, and should be considered in assigning tasks for the next scheduling horizon.

- acceptance ratio of the requests for the task assignment
- number of evening-shift and number of night-shift - date of the latest evening-shift and date of the latest night-shift in the last scheduling horizon

(xiv) A schedule for the next scheduling horizon is presented to nurses by the end of the current scheduling horizon. The schedule satisfies the following four conditions:

(a) In each shift on each day, the number of nurses assigned to the shift is larger than or equal to the prespecified number both in total and for each module.

(b) In each shift on each day, the required mean skill level is satisfied by nurses assigned to the shift.

(c) Numbers of the day-, evening-, and night-shift assigned to each nurse in the scheduling horizon are within the prespecified range.

(d) No prohibited shift pattern is assigned to any nurse. Conditions (a) to (d) are called hard constraints.

Moreover, requests for task assignment posed by nurses should be satisfied as well as possible, and standard shift patterns for each nurse should be realized as frequently as possible in the schedule.

B. Additional Assumptions in Rerostering

The followings are assumptions for rerostering phase. Nurse rerostering will occur after the next scheduling horizon starts, and then the scheduling horizon is called the “current” scheduling horizon in discussing nurse rerostering.

(xv) Nurses usually fulfill their tasks according to a predetermined schedule for the current scheduling horizon.

(xvi) An absence of a nurse caused by an unexpected event/accident suddenly happens and is informed to a scheduler (or a head nurse) with a length of the absence before 8:00 a.m. on any day during the scheduling horizon.

(xvii) When an absence is announced, the scheduler immediately starts generating new schedules for filling up the absence by modifying some task assignments in the current schedule. The modification process is called ‘rerostering.’

(xviii) Except the night-shift and meeting on that day, any task assigned on and after that day in the predetermined schedule can be changed in rerostering.

(xix) The new schedule obtained through the rerostering process should satisfy all conditions (a) to (d) in assumption (xiv).

(xx) The number of task reassignments should be as small as possible to avoid causing additional confusion in carrying out the tasks.
C. Problem Formulation

The nurse rostering problem dealt with in this paper can be formulated as follows:

Notations:
- \( N \): number of nurses
- \( T \): scheduling horizon for static scheduling
- \( M \): number of nursing modules
- \( \delta_{itk} \): decision variable, where \( \delta_{itk} = 1 \), if nurse \( i \) is assigned to shift \( k \) on the \( t \)-th day; 0, otherwise.
- \( t_v \): date of occurrence of absence (the first day)
- \( w \): length of consecutive days of absence
- \( N_{tk} \): required number of nurses at shift \( k \) on the \( t \)-th day
- \( N_{tkm} \): required number of nurses at shift \( k \) on the \( t \)-th day for module \( m \)
- \( I_m \): set of nurses who belong to nursing module \( m \)
- \( \beta_i \): required mean skill level for shift \( k \)
- \( \beta_k \): skill level of nurse \( i \)
- \( LB_{ik} \): lower bound for number of shift \( k \) assigned to nurse \( i \) during the scheduling horizon.
- \( UB_{ik} \): upper bound for number of shift \( k \) assigned to nurse \( i \) during the scheduling horizon
- \( UB' \): upper bound for number of sum of evening- and night-shifts assigned to nurse \( i \) during the scheduling horizon
- \( req_{itk}^p \): request for task assignment from nurse \( i \) at shift \( k \) on the \( t \)-th day with priority \( p \), where \( req_{itk}^p = 1 \), if nurse \( i \) requests a task assignment at shift \( k \) on the \( t \)-th day; 0, otherwise.
- \( \delta'_{itk} \): task assignment for nurse \( i \) on the \( t \)-th day in a predetermined static schedule, where \( \delta'_{itk} = 1 \), if nurse \( i \) is assigned to shift \( k \) on the \( t \)-th day; 0, otherwise.

Minimize \( \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=0}^{M} (n_{itk} + p_{itk}) / 2 \) \hspace{1cm} (1)

Subject to

\[ \delta_{it0} = 1, \quad t = t_v, ..., t_v + w - 1 \] \hspace{1cm} (2)

\[ \sum_{t=0}^{T} \delta_{itk} = 1, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T \] \hspace{1cm} (3)

\[ \sum_{i=1}^{N} \delta_{itk} \geq N_{tk}, \quad t = 1,2, ..., T, \quad k = 1,2,3 \] \hspace{1cm} (4)

\[ \sum_{i \in I_m} \delta_{itk} \geq N_{tkm}, \quad t = 1,2, ..., T, \quad k = 1,2,3, \quad m = 1,2, ..., M \] \hspace{1cm} (5)

\[ \sum_{i=1}^{N} \beta_i \delta_{itk} \geq \beta_k \sum_{i=1}^{N} \delta_{itk}, \quad t = 1,2, ..., T, \quad k = 1,2,3 \] \hspace{1cm} (6)

\[ \sum_{t=1}^{T} (\delta_{it2} + \delta_{it3}) \leq UB'_i, \quad i = 1,2, ..., N \] \hspace{1cm} (7)

\[ LB_{ik} \leq \sum_{t=1}^{T} \delta_{itk} \leq UB_{ik}, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T, \quad k = 1,2,3 \] \hspace{1cm} (8)

\[ \sum_{k=0}^{2} \sum_{t=0}^{T} \delta_{itk} \geq 1, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T - 5 \] \hspace{1cm} (9)

\[ \delta_{it3} + \delta_{it1} + \delta_{it2} \leq 2, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T - 2 \] \hspace{1cm} (10)

\[ \delta_{it2} + \delta_{it1} \leq 1, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T - 2 \] \hspace{1cm} (11)

\[ \delta_{it4} = \delta_{it4}, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T \] \hspace{1cm} (12)

\[ \delta_{itk} = \delta'_{itk}, \quad t = 1,2, ..., t_v - 1, \quad i = 1,2, ..., N, \quad k = 0,1, ..., A \] \hspace{1cm} (13)

\[ \delta_{itv} = \delta'_{itv}, \quad i = 1,2, ..., N \] \hspace{1cm} (14)

\[ \sum_{p=0}^{2} \sum_{k=0}^{4} \sum_{t=0}^{T} req_{itk}^p \delta_{itk} - \sum_{p=0}^{2} \sum_{k=0}^{4} \sum_{t=0}^{T} req_{itk}^p \delta'_{itk} \geq 0, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T \] \hspace{1cm} (15)

\[ \delta_{itk} - \delta_{itk} - \sum_{p=0}^{2} \sum_{k=0}^{4} \sum_{t=0}^{T} req_{itk}^p \delta_{itk} + \sum_{p=0}^{2} \sum_{k=0}^{4} \sum_{t=0}^{T} req_{itk}^p \delta'_{itk} \geq -1, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T, \quad k = 0,1, ..., A \] \hspace{1cm} (16)

\[ \delta_{itk} - \delta_{itk} + n_{itk} - p_{itk} = 0, \quad i = 1,2, ..., N, \quad t = 1,2, ..., T, \quad k = 0,1, ..., A \] \hspace{1cm} (17)

The objective function is to minimize the number of reassigned tasks as presented in equation (1). Equation (2) defines the consecutive days of absence in this nurse rostering problem, that is, nurse \( i \) is absent for \( w \) consecutive days starting from day \( t_v \). Equation (3) represents the constraint that at most one task must be assigned to each nurse every day. The constraints of required number of nurses for each shift and required mean skill level are expressed in equations (4) and (5) and equation (6), respectively. Workload constraints for each nurse are given in equations (7) and (8). Equations (9)-(12) represent constraints of prohibited shift patterns. Equations (13)-(17) define unmodifiable task assignments. The \( n_{itk} \) and \( p_{itk} \) in equation (18) are used for counting the number of reassigned tasks.
III. PROPOSED METHOD

A. Basic Concept

This paper proposes a heuristic method for the nurse rerostering problem triggered by a sudden absence of any nurse. The method aims to obtain a feasible schedule with the minimum number of reassigned tasks. It is assumed that the absence is not always just one day but continues for several days in this paper. The feasible schedules for the problem are defined as schedules which satisfy all hard constraints, i.e. conditions (a)-(d) in assumption (xiv) as described in the previous section.

Since it is hard to treat several consecutive days of absence of a nurse collectively in rerostering, the proposed method first separates several days of absence into a set of one-day absences, and constructs a set of nurse rerostering subproblems with a one-day absence. Then, each subproblem is solved one by one as follows.

For a nurse rerostering problem with one-day absence, if there exists more than one nurse whose task is reassigned day-off on the day when the absence occurs and who can fill up the vacancy caused by the absence without violating any hard constraints, a feasible schedule can be generated by assigning the missed task to one of the nurses. Such a problem is called ‘easily solvable’ case and the selected nurses are called ‘substitute’ nurses in this paper.

The proposed method first finds ‘easily solvable’ problems from among the set of nurse rerostering subproblems with one-day absence. If there are some easily solvable subproblems, the method selects one of them in date order of the absence and solves it by reassigning the absentee’s task to a substitute nurse. Fixing the task reassignment, the remaining easily solvable subproblems are also solved in the same manner. After that, each of remaining general case subproblems is solved by a recursive search technique and tree search algorithm in date order of the absence. The details of the recursive search technique and the tree search algorithm are described in later subsections.

The proposed method is terminated when all subproblems have been solved successfully or the method fails to generate a feasible schedule for at least one subproblem. In the latter case, the proposed method abandons to generate a feasible solution for the original nurse rerostering problem.

B. Recursive Search Procedure for Rerostering

The recursive search technique [7, 8] is applied to each of nurse rerostering subproblems with one-day absence when the problem is not easily solvable.

The procedure is summarized in the following three steps, where a nurse assigned to task $k$ on the $t$-th day in the predetermined schedule is assumed to be absent.

Step 1: Find a nurse $i$ who is assignable to task $k$ on the $t$-th day without violating the constraint of the required skill level of the shift (called an ‘assignable nurse,’ in short) and assign the nurse to the absent position.

Step 2: If the assignment in Step 1 results in any violation of the other hard constraints for the nurse $i$, then change a task assigned to the nurse $i$ on the $(t+1)$-th day in the current schedule into a ‘day-off’ so as to eliminate the violation. Let the $(t'\cdot t)$-th day be the day on which the ‘day-off’ is newly assigned, and task $k'$ be the task reassigned to nurse $i$ on the $(t'+t)$-th day.

Step 3: If the day-off assigned in Step 2 results in any violation of the other hard constraints, then deal with the ‘day-off’ as a new absence of nurse $i$, set $t := t'$ and $k := k'$, and go back to Step 1. Otherwise, terminate the procedures.

In Step 1, nurse $i$ is selected from among assignable nurses according to a dispatching rule [8], which is expected to minimize the number of task reassignments by considering the priority of hard constraints in the nurse selection process.

The above procedures are implemented recursively until the schedule has no more violated hard constraint, resulting in the “initial schedule” with an “initial search tree” to be used in the succeeding tree search algorithm. The initial search tree is constructed by generating nodes corresponding to all candidate nurses in Step 1 at each recursive level. Note that the ‘height’ of the search tree corresponds to the number of reassigned tasks in rerostering. To avoid generating an initial schedule with too much task reassignments, the height, i.e. recursive level, is limited within five in the proposed method.

C. Tree Search Algorithm

When the height of the initial search tree is larger than two, the following tree search algorithm is implemented to generate better schedules with less number of reassigned tasks. Let $n$ be number of reassigned tasks of the schedule generated by the recursive search approach and $h$ be the height of the initial search tree. Note that this algorithm is also implemented when the recursive search procedure fails to generate a feasible schedule within the limited recursive level, i.e. five.

Step 1: Find an active node $i$ with the deepest search level in the current search tree. If there are more than two active nodes with the deepest search level, select a node according to the priority of assignable nurses in the same way as used in Step 1 in the recursive search algorithm.
Step 2: Backtrack to the node \( i \), and implement the recursive search procedures given in the previous subsection provided that nurse \( i \) is set as the nurse corresponding to node \( i \) and that the related task \( k \) and date \( t \) are also set as a new task and date in node \( i \), respectively. If either the height of the current search tree reaches \( h \) or the search fails to find a feasible schedule with less than \( n \) reassigned tasks, then terminate the recursive search. Otherwise, \( h \) is set as the height of the current search tree and \( n' \) is set as the number of reassigned tasks of the current schedule.

Step 3: If \( n' < n \), then set the current schedule as the incumbent schedule with \( n := n' \).

Step 4: If there exists no active nodes in the current search tree, then terminate. Otherwise, go to Step 1.

D. Preprocessing for Skill Guaranty

To enhance the performance of the proposed method, it is important to set up a situation where there exist assignable nurses in Step 1 of the recursive search procedure. If the absentee is a high-skilled nurse, only other high-skilled nurses can fill up the vacancy without violating the hard constraint of the required skill level. In such a case, the following task reassignments are implemented to try to increase candidates of the assignable nurses. Suppose that an absence occurs in shift \( k \) on the \( t \)-th day.

Step 1: If there exist no assignable nurses, find a high-skilled nurse whose task preassigned on the \( t \)-th day is day-off. If there is such a nurse, then go to Step 2. Otherwise, terminate.

Step 2: Find a nurse with the lowest skill level among nurses whose task preassigned on the \( t \)-th day is task \( k \).

Step 3: Swap the preassigned tasks of these two nurses and check whether the task reassignments result in any violation of the hard constraints of prohibited shift patterns and number of assigned tasks for both nurses.

Step 4: If any violation of the above two hard constraints occur, then the task reassignments in Step 3 are cancelled. Otherwise, the task reassignments are accepted and fixed. Then, return to the Step 1 of the recursive search algorithm.

Experimental design is given as follows:

- Number of nurses: \( N = 24 \)
- Number of nursing module: \( M = 3 \)
- Skill levels of nurses: 5 for nurses no.1-no.3 (leader), 3 for nurses no.4-no.7 (highly- experienced), 1 for nurses no.8-no.15 (experienced), 0.8 for nurses no.16-no.20 (second-year) and 0.5 for nurses no.21-no.24 (newcomer).
- Required numbers of nurses for each shift: 10 in total and at least 3 for each module for day-shift, 3 in total and at least 1 for each module both for evening and night-shifts. Thus, 16 nurses are required to work per day in total.
- Required mean skill level of assigned nurses for each shift: \( \beta_k = 1.3, k = 1, 2, 3 \)
- Lower and upper bounds \([LB_k, UB_k]\) for the monthly number of shift assigned to each nurse: \([5, 15]\) for each day-shift \((k=1)\), \([2, 8]\) for each evening-shift \((k=2)\) and night-shift \((k=3)\) and \([4, 9]\) for the sum of evening- and night-shifts.
- Prohibited shift patterns that make schedules infeasible: (i) Six days consecutive shifts without day-off (prohibited in law); (ii) Three consecutive night-shifts; (iii) Day-shift, meeting or night-shift succeeding immediately after the evening-shift.
- Standard shift pattern: A sequence of tasks composed of up to three successive day-shifts, night-shift, evening-shift and finally day-off.
- Personal relationships between any pair of nurses and historical information on task assignment to each nurse for the last two months: given randomly but the details are omitted limit for this paper.

Eight schedules are arranged first by solving the static nurse scheduling problem randomly generated under the above conditions using a heuristic method [11]. For these schedules, \( 3840 (= 16 \text{ nurses} \times 30 \text{ days} \times 8 \text{ schedules} ) \), \( 3712 (= 16 \times 29 \times 8 ) \), \( 3584 (= 16 \times 28 \times 8 ) \) and \( 3456 (= 16 \times 27 \times 8 ) \) instances of nurse rostering problems with one-day absence and two, three and four consecutive days of absence are generated by inserting a consecutive absence starting from the \( t \)-th day to any nurse (except for the nurses who assigned ‘meeting’ or ‘day-off’ on the \( t \)-th day) in the predetermined static schedule, respectively.

The algorithm of the proposed method is coded in C-language and is run on a personal computer (Phenom II X6 3.20GHz).

IV. NUMERICAL EXPERIMENTS

A. Experimental Design

To evaluate the effectiveness of the proposed method, it is applied to some instances of a three-shift nurse rostering problem with a sudden absence of several consecutive days under real conditions.
B. Results

Table 1 shows an example of feasible schedules obtained by the proposed method. Notations ‘D’, ‘E’, ‘N’, ‘/’ and ‘M’ in this table stand for day-shift, evening-shift, night-shift, day-off and meeting, respectively. In this example, nurse no. 3 is absent for three consecutive days, from day 14 to day 16.

| Scheduling Period (20 Days) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|-----------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Night-Shift                | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| Day-Off                    | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / | / |
| Meeting                    | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M | M |

At first the method solves the easily solvable case of day 15. On day 15, nurse no. 18 who was assigned day-off can be a substitute for nurse no. 3 by assigning day-shift. Then the method deals with day 14, the earliest date among the rest of the absences. On day 14, the day-shift of nurse no. 3 is covered by nurse no. 21. In this operation ‘skill guaranty’ is implemented to avoid the constraint violation of skill level of day-shift. The day-off is assigned to nurse no. 21 instead of day-shift on nurse no. 2 on that day. By now the absence of day 14 has been fixed. Finally the proposed method deals with day 16. The “skill guaranty” is implemented again on nurses no. 4 (assigned day-shift), no. 16 (day-off), no. 9(day-shift). After that the violation of consecutive work occurs on nurse no. 9. To eliminate that violation, the task assigned to nurse no. 9 on day 18 is covered by nurse no. 6. Therefore, there is no violation of hard constraints. In this instance there are 10 reassignment tasks in the end.

Table 2 shows the results in the case of one-day absence focus on the number of reassigned tasks of generated solution for each solved instance. In this case, the proposed method can generate feasible schedules for 95.0% (3647 over 3840) of examined instances. According to the head nurse we interviewed, it is recommended that the number of reassigned tasks in rostering is at most ten in practice. From this viewpoint, the proposed method is effective in case of one-day absence.

Table 3 shows the results for the problems with consecutive days of absence. The proposed method can generate feasible solutions for 90.8% (3371 over 3712) of instances in the case of two-day absence, 86.6% (3102 over 3584) in the case of three-day absence and 82.6% (2855 over 3456) in the case of four-day absence respectively. However the longer the absence continues, the more task reassignments are needed to obtain the solutions.
The proposed method generates solutions within 10 reassigned tasks for 89.1% (3307 over 3712) of instances in the case of two-day absence, 81.1% (2905 over 3584) in the case of three-day absence and 71.4% (2466 over 3456) in the case of four-day absence, respectively.

Tables 2 and 3 indicate that there exist some instances of which the proposed method could not generate feasible solutions (5.0% of instances for one-day absence). Figure 1 shows the ratio of solvable instances by the methods in case of one-day absence. Note that 55.1% (2116 over 3840) of instances are easily solvable cases. The dotted line shows the ratio of solved instances without skill guaranty technique. The skill guaranty technique demonstrates a significant effect on generating feasible solutions.

**TABLE II**
The Number Of Reassigned Tasks For One-Day Absence

<table>
<thead>
<tr>
<th>number of reassignments</th>
<th>no. 1</th>
<th>no. 2</th>
<th>no. 3</th>
<th>no. 4</th>
<th>no. 5</th>
<th>no. 6</th>
<th>no. 7</th>
<th>no. 8</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>262</td>
<td>285</td>
<td>265</td>
<td>314</td>
<td>267</td>
<td>233</td>
<td>248</td>
<td>242</td>
<td>2116</td>
</tr>
<tr>
<td>2</td>
<td>55</td>
<td>43</td>
<td>46</td>
<td>50</td>
<td>36</td>
<td>64</td>
<td>80</td>
<td>40</td>
<td>414</td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>86</td>
<td>80</td>
<td>53</td>
<td>80</td>
<td>75</td>
<td>81</td>
<td>81</td>
<td>607</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>17</td>
<td>22</td>
<td>16</td>
<td>40</td>
<td>31</td>
<td>27</td>
<td>23</td>
<td>202</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>14</td>
<td>21</td>
<td>14</td>
<td>15</td>
<td>19</td>
<td>16</td>
<td>17</td>
<td>135</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>13</td>
<td>89</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>15</td>
<td>5</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>total</td>
<td>458</td>
<td>458</td>
<td>453</td>
<td>466</td>
<td>463</td>
<td>443</td>
<td>471</td>
<td>435</td>
<td>3647</td>
</tr>
</tbody>
</table>

**TABLE III**
The Number Of Reassigned Tasks For Consecutive Absences

<table>
<thead>
<tr>
<th>length of absence</th>
<th>1 day</th>
<th>2 days</th>
<th>3 days</th>
<th>4 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of reassigned tasks</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2116</td>
<td>606</td>
<td>187</td>
<td>34</td>
</tr>
<tr>
<td>2</td>
<td>414</td>
<td>909</td>
<td>535</td>
<td>227</td>
</tr>
<tr>
<td>3</td>
<td>607</td>
<td>521</td>
<td>541</td>
<td>417</td>
</tr>
<tr>
<td>4</td>
<td>202</td>
<td>446</td>
<td>448</td>
<td>410</td>
</tr>
<tr>
<td>5</td>
<td>135</td>
<td>282</td>
<td>340</td>
<td>365</td>
</tr>
<tr>
<td>6</td>
<td>89</td>
<td>175</td>
<td>265</td>
<td>282</td>
</tr>
<tr>
<td>7</td>
<td>58</td>
<td>169</td>
<td>204</td>
<td>221</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>85</td>
<td>169</td>
<td>216</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>70</td>
<td>118</td>
<td>168</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>44</td>
<td>98</td>
<td>126</td>
</tr>
<tr>
<td>10+</td>
<td>0</td>
<td>64</td>
<td>197</td>
<td>389</td>
</tr>
<tr>
<td>total</td>
<td>3647</td>
<td>3371</td>
<td>3102</td>
<td>2855</td>
</tr>
</tbody>
</table>

Table 4 shows the distribution of unsolvable instances among the schedule no.1 for one-day absence. The black cells present unsolvable instances for the proposed method, and grey cells present those for the method without skill guaranty technique. The unsolvable instances distribute evenly during the scheduling horizon. That means the timing of occurrence of the absence is not crucial.
The computation time of the proposed method is at most ten seconds per instance. It is efficient enough from a practical viewpoint.

V. CONCLUSIONS

A heuristic method for nurse rorostering problem with an absence for several consecutive days is proposed in this paper. The proposed method first separates several days of absence into a set of one-day absences, and then applies the revised recursive search technique with tree search algorithm to each of nurse rorostering problems one by one. Numerical experiments in the three-shift nurse rorostering problems with real data in a Japanese hospital showed that the proposed method is effective to find feasible schedules for the nurse rorostering problems even if the sudden absence has continued for several consecutive days.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 22510157.

**TABLE IV**

UNSOLVABLE INSTANCES ON SCHEDULE NO.1 FOR THE CASE OF THREE CONSECUTIVE DAYS OF ABSENCE

| scheduling period (30 days) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|-----------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

REFERENCES


