International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 14246-14250

A Deterministic Approach to Nurse Rerostering Problem

Saangyong Uhmn[1], Young-Woong Ko[2] and Jin Kim[3,*]

[1,2,3] *Department of Computer Engineering, Hallym University, Chuncheon, 24252, Republic of Korea.*

*(*Corresponding Author)*

Abstract

When a nurse at a hospital is unable to be in charge of any of his/her assigned shifts, the roster should be rescheduled to cope with it, which is called nurse rerostering problem. In this problem, we are supposed to find a new one which is close to the given as well as satisfies all the constraints that it inherits from nurse scheduling problem which built the roster. It was proved NP-complete so that many heuristic algorithms were proposed. By treating the difference between two rosters as a constraint of the highest priority, it can be thought of as a shortest path problem. In this study, we applied a deterministic algorithm, iterative deepening depth-first search, which was proved to be asymptotically optimal along all three dimensions for exponential tree searches: the amount of time it takes, the amount of space it uses, and the cost of a solution path. The results of the experiments show that it is of practical use for the problem. In terms of the number of differences, it provides us with the optimal solution for all the datasets. Based on the results, the algorithm can be applied to the problem for limited depth or time. If a solution is not found within the limit, we can apply other heuristic algorithm such as simulated annealing or genetic algorithm to find an approximate.

Keywords : Nurse rerostering problem, nurse rostering problem, iterative-deepening depth first search

INTRODUCTION

An organization providing round-the-clock services usually divides daily work into shifts to which workers are assigned. At such an organization, a schedule is constructed for a predetermined period and announced in advance so that the workers may organize their private life in accordance with their assignment. The schedule is nothing but a collection of shift assignments of all workers that must satisfy several constraints set by staffing requirements, labor contracts, and so on. However, unexpected events may occur in real life, which leads to schedule disruption and infeasibility. When it happens and there is no reserve pool of workers, rescheduling is required to make the schedule meet the constraints in consideration and, at the same time, minimize the differences between two rosters because of workers' private life organized in accordance with it [1].

Nurse rostering problem (NRP) is an instance of a scheduling problem in which each nurse is assigned to a set of shifts and days off [2-6]. Nurse re-rostering problem (NRRP) is a rescheduling problem to cope with unavailability of nurses in a given roster [7]. These problems were proved NP-complete and heuristic algorithms were applied to them. In NRRP, because the roster is usually announced in advance, a new roster must be constructed in due consideration of the previous one, which puts an additional constraint compared to NRP. However, if we treat it as one of constraints of the highest priority, the problem can be thought of as a shortest path problem in terms of distance between the given and the new roster.

This situation can be solved in two different approaches: a reactive and a proactive [8]. In this study, we deal with the reactive nurse re-rostering problem.

While NRP is dealt with in many literatures, NRRP got much less interest [1,7-12]. Since Moz proved that NRRP is NP-complete, heuristic algorithms have been applied from the theoretical point of view [1,7-15]. First, Moz and Pato formulated it as an integer multi-commodity flow problem with additional constraints in a multi-level acyclic network and applied a heuristic to real instances from a Lisbon state hospital [7]. They also presented two new integer multi-commodity flow formulations based on a directed multi-level acyclic network, besides conducting computational experiments with real data [10]. Later, they proposed genetic algorithms for the problem [1,12]. Kingston proposed a network-based mathematical programming approach to the problem and presented the results of its application to two real-life case studies from Arizona State University Computer Labs and Banner Health Baywood Hospital [11]. Maenhout and Vanhoucke proposed an evolutionary algorithm which outperformed that of Moz and Pato [15]. They also provided insights and understanding in the consequences and outcomes of various personnel re-rostering characteristics and strategies through experiments [8]. To accelerate the performance of their algorithm, Ba□umelt et al. applied a graphics processing unit to their algorithm and provided a speedup of 1.9 (2.5) times for instances with 19 (25) nurses compared to the sequential one [16].

All these algorithms tried to find the closest among the feasible solutions found in reasonable time. In this study, we took a different, practical point of view to the problem: the highest

priority to nurses' private life according to the roster. According to this point of view, we are supposed to find a feasible one from the closest candidates, not a closest one among the feasible solutions found, and applied iterative deepening depth-first search (IDDFS) with limit on depth to the problem. IDDFS was rediscovered several times and its first use documented in the literature is in Chess 4.5 [17]. Berliner and Goetsch, and Korf applied IDDFS to A* concurrently and summarized performance in their reports [18, 19]. In this study, we applied IDDFS with a limit on depth to NRRP in which if no solution is found within the limit, we may apply one of heuristic algorithms to find an approximate. In addition, to evaluate its performance, we compared it with simulated annealing.

The remainder of this paper is organized as follows. In Section 2, we describe nurse re- rostering problem, constraints, and the difference with NRP. Section 3 and 4 present IDDFS algorithm and simulated annealing briefly, and Section 6 reports on the results of the experiments. Finally, some conclusions and future works are outlined in Section 7.

NURSE REROSTERING PROBLEM

Problem Definition

A set of N nurses are scheduled over a period of D days. This schedule is called a roster and represented as a nurse-day view [3]. It is nothing but a matrix S of size $N \times D$ whose element $s_{ij} \in \{m, e, n, o\}$ for $1 \le i \le N$ and $1 \le j \le D$ where m, e, n, and o represents a morning, an evening and a night shift, and a day off, respectively. A roster S should satisfy a set of constraints. After its construction, unavailability of a nurse for the assigned task leads to schedule disruption and infeasibility. When it happens, rescheduling is necessary to make the roster meet the requirements if there is no reserve pool of nurses. In re-rostering, because the roster is announced beforehand and every nurses' private life might be organized in accordance with it, a new roster, S', must be constructed in due consideration for S in such a way that the difference between two rosters is as small as possible as well as all the constraints are satisfied.

In this study, we consider only one event, that is, an unavailability of a nurse at a shift on a day, and a period for re-rostering, so-called re-rostering time horizon, is from the day of the event to the end of the period like in Moz et al [7]. This period can be D days to the maximum.

Constraints

NRRP inherits not only characteristics from NRP but constraints. The constraints are usually dichotomized based on their possibility of violation: hard and soft constraints. Hard constraints should always be satisfied while soft ones not, but as much as possible. That is, a schedule violating a hard constraint cannot be a feasible solution, while one not satisfying only soft ones can be feasible. Possible examples of hard constraints include the limit on the number of nurses for each shift and the maximum number of shifts per a nurse in a week or a month, and some of the soft ones are nurses' requests for days off or certain shifts on certain days.

Although there are many kinds of hard and soft constraints, we only consider the followings because the main objective of this study is to estimate the applicability of IDDFS to NRRP.

(a) Hard constraints

(i) The number of nurses for a morning, an evening, and a night shift should be between the minimum and the maximum values. Let m_j, e_j, n_j and o_j for $1 \le j \le D$ be the number of nurses for each shift on a day j. Given m_{min}, e_{min}, n_{min} and m_{max}, e_{max}, n_{max} as the minimum and the maximum number of nurses for each shift, respectively, the following conditions should be met.

$$\begin{aligned} m_{min} \le m_j \le m_{max} \\ e_{min} \le e_j \le e_{max} \\ n_{min} \le n_j \le n_{max} \end{aligned} \quad (1)$$

(ii) Some working patterns must be avoided: a morning shift after a night, an evening after a night, a morning after an evening, and three consecutive nights.

(b) Soft constraints

(i) The number of morning, evening, and night shifts and days off for each nurse in a roster must meet the requirement. Let M_i, E_i, N_i and O_i for $1 \le i \le N$ be the number of shifts for nurse i, and M_{req}, E_{req} and N_{req} be the requested number of each shift. Then,

$$\text{Minimize}(|M_i - M_{req}| + |E_i - E_{req}| + |N_i - N_{req}|) \quad (2)$$

Objective Function

We have to define an objective function to measure the quality of a reconstructed roster. Before we define the function, we have to define costs for each constraint. Let C_1, C_2 and C_3 be the costs for the above constraints. The costs indicate the number violations of the constraints as in Ko et al [22]. The additional C_4 is the number of different shifts in two rosters. Based on these costs, we defined the function as follows.

$$C_{total} = w_1 \cdot C_1 + w_2 \cdot C_2 + w_3 \cdot C_3 + w_4 \cdot C_4 \quad (3)$$

where weight values w_1, w_2, and w_3 for the cost C_1, C_2, and C_3, respectively as in Ko et al [21, 22]. In addition, w_4 is needed for the cost C_4. These weights reflect the priority of costs. By the definition of the problem, C_1 and C_2 should be zero and C_3 and C_4 must be as small as possible.

ITERATIVE DEEPENING DEPTH FIRST SEARCH (IDDFS)

Search Space

IDDFS was proposed to cope with the drawbacks of the space complexity of breadth-first search and the time complexity of depth-first search for a graph. To apply IDDFS to NRRP, we first define its search space T as a tree whose node represents a roster as follows.

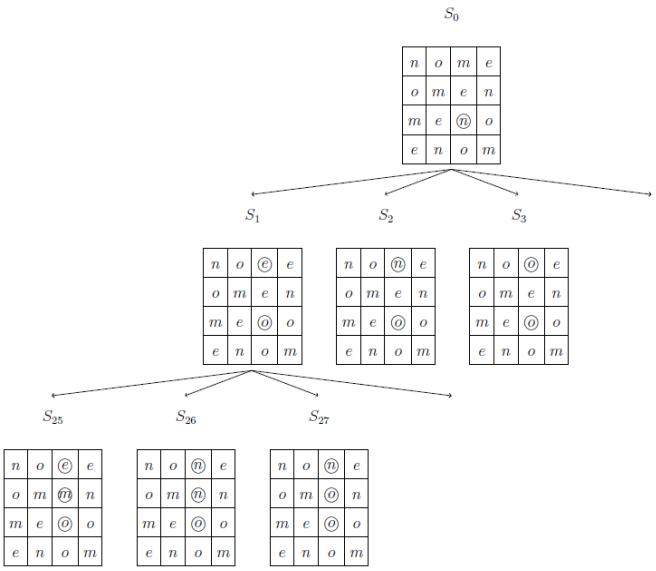

Figure 1: The search space of NRRP.

A node S_k for $0 \le k \le |T|$-1 represents a roster for N nurses over a D-day period. Each cell s_{ij} of S_k denotes an assigned shift for a nurse i on a day j where $s_{ij} \in \{m, e, n, o\}$. The root node, S_0, is the given roster and the depth of each node is same as the number of different shifts between the root and a roster that the node represents. Figure 1 represents its structure. The assignment of $s_{33} = n$ should be reassigned in S_0, and $s_{13} = e$ in S_1, $s_{13} = n$ in S_2 and $s_{13} = o$ on depth one indicate reassigned shifts. In this example, all the nodes cannot be a solution because s_{43} and s_{44} are both o, which violates hard constraint (1).

Algorithm

In IDDFS, each node is visited in depth-first fashion within a predefined limit of depth. If no solution is found, the limit is increased and all the nodes including previously visited and newly added are visited in the order. By revisiting nodes in lower depth, IDDFS requires less space than breadth-first search. By depth-first search fashion within the limit, it may find a solution on the lower depth faster than the depth-first search to the depth of a given tree. The algorithm is briefly described in Algorithm 1. The procedure IDDFS is called with the root and the limit of depth which calls the procedure DFS iteratively. In line 11 of DFS, the current node is checked if it satisfies all hard constraints. In line 20, a newly found roster, found, is compared to the previous one, solution, in terms of C_4: The smaller C_4, the better. Because we put C_4 the higher priority than C_3, the first one found in line 4 is the answer.

Algorithm 1 Iterative deepening depth-first search.

```
 1: procedure IDDFS(root, depthLimit)
 2:   for depth = 1 to depthLimit do
 3:     found ← DLS(root, 1, depth)
 4:     if found ≠ null then
 5:       return found
 6:     end if
 7:   end for
 8: end procedure

 9: procedure DFS(node, currentDepth, maxDepth)
10:   if currentDepth = maxDepth then
11:     if node is a goal then
12:       return node
13:     else
14:       return null
15:     end if
16:   else
17:     solution ← null
18:     for each child of node do
19:       found ← DFS(child, currentDepth + 1, maxDepth)
20:       if (found ≠ null) and (found is better than solution) then
21:         solution ← found
22:       end if
23:     end for
24:   end if
25:   return solution
26: end procedure
```

Simulated Annealing

To evaluate the performance of IDDFS to NRRP, we compare it with simulated annealing (SA). SA is a probabilistic metaheuristic to find a good approximation to the global minimum of a function for combinatorial optimization problem. The problem defines a set of states $S = \{S_1, \cdots, S_L\}$ and a cost function $C : S \rightarrow R$, where R is the set of real numbers. For each state $S_k \in S$, a real value $C(S_k)$ is the cost. The goal of the problem is to find an optimal state S_{opt} whose score is

$$\min(\max)\{C(S_k) \mid 1 \le k \le L\}.$$

The algorithm generates a new candidate S_{new} from a current state $S_{current}$ by applying transition rules and acceptance rules. The candidate is accepted if ΔE is less than or equal to zero, or in the probability of

$$P(\Delta E) = \exp(-\frac{\Delta E}{k_B T})$$

where T is a temperature, k_B is Boltzmann's constant, and $\Delta E = C(S_{new}) - C(S_{current})$ is a cost difference.

The algorithm avoids getting stuck in a local minimum by probability. A state is called a local minimum if there is no new state in S that is generated from it by applying the rules and has a lower cost. Temperature T controls the probability to accept a new state in such a way that it starts from a high temperature and decreases based on annealing schedule at each iteration and becomes zero eventually. The probability to accept the new state with the higher cost also decreases as it does. If annealing schedule and a number of iterations are chosen deliberately, it

converges to a state with a global optimum. Because of efficiency in performance, it has been applied to many combinatorial problems. However, SA has a main disadvantage of the large amount of computation time20. This is because SA is based on Monte Carlo method.

EXPERIMENTS

We implemented IDDFS and SA with the constraints described above in C and carried out experiments on Linux.

Test datasets

We generated 100 rosters of five nurses (N = 5) for one to four weeks (D = 7, 14, 21, and 28) by the program explained in Ko et al [22]. We applied the values for constraints as follows: $m_{min} = e_{min} = n_{min} = m_{max} = e_{max} = n_{max} = 1$ and M_{req}, E_{req}, N_{req} varies according to the number of weeks. In addition, we used the following values for weights of an objective function: $w_1 = 5$, $w_2 = 5$, $w_3 = 1$ and $w_3 = 3$.

Roster disruption

As we noticed in Section 2, we considered only one absence for all datasets. The shift was chosen randomly among active nurses. However, we chose a day randomly in the first half of the period because the day close to the end of the period may result in a roster without a feasible solution which is meaningless to the study.

EXPERIMENT RESULTS

To compare the performance of IDDFS with SA, we conducted experiments on 100 datasets and compared their results in three criteria: the number of instances each algorithm solved, the average cost of solutions found, and average elapsed time. Because of its dependence on the random number, we ran SA on the dataset 10 times with different random number sequences and chose one with the smallest cost. Table 1 shows the results of the algorithms.

Table 1: Results of TSA and IDDFS (The numbers in parenthesis are the results for the instances solved by TSA)

# weeks	Algorithm	# solved	Cost	Time
1	TSA	78	12.12	1.81
	IDDFS	100 (78)	13.59 (12.08)	42.66 (2.02)
2	TSA	100	12.60	16.65
	IDDFS	100 (100)	12.00 (12.00)	1.18(1.18)
3	TSA	66	17.55	90.51
	IDDFS	100 (66)	12.09 (12.14)	101.29 (148.02)
4	TSA	30	17.10	598.33
	IDDFS	100 (30)	12.03 (12.10)	319.37 (1,004.61)

First, IDDFS outperformed SA in terms of the number of instances solved. IDDFS provided us with a solution for all datasets while SA didn't for three and four weeks. Because the size of search space is increased at an exponential rate as the number of weeks increases, the probability to find a solution for SA decreases drastically. Second, IDDFS always provided us with the better solutions than SA. As the number of week increases, the difference in the average cost between the solutions is getting bigger, which is caused by the same factor, the increased size of search space. The bigger the size of search space is, the higher in probability it found a solution far from the given roster. Third, it showed the remarkable difference in the average elapsed time; the bigger the number of weeks, the more time both algorithms took. This difference is caused by the number of iterations each algorithm takes to find a solution. Because for IDDFS, the first solution found in the process is the answer, it is executed until one is found. However, SA should perform the search until the temperature reaches zero based on the annealing schedule, which results in the larger number of iterations compared to IDDFS, that is, the huge amount of time. However, as expected for the problems of NP-completeness, the time for IDDFS increased steeper than that for SA, while IDDFS ran rather faster than SA. We can easily infer that it is because the number of nodes at each depth increases exponentially which means the number of nodes IDDFS visits increases accordingly. However, because of the priority for C_4, the depth visited by IDDFS was low enough to finish its execution in very small amount of time.

CONCLUSIONS

We applied one of brute force approach, iterative deepening depth-first search, to nurse rerostering problem to pursue optimality. In NRRP, we are supposed to find a closest roster to the given one which also satisfies all the constraints inherited from nurse rostering problem. From the practical point of view, the difference in the number of assigned shifts should be minimized to preserve nurses' private life organized in accordance to the roster. In this study, by considering the difference between two rosters with priority over soft constraint, we may apply an iterative deepening depth-first search to the problem, even though the problem is proved NP-complete. However, as the number of weeks or nurses or the re-rostering period increases, the size of search space will grow exponentially, which makes the algorithm hard to be applied to the problem. In addition, there may be a case that no feasible solution exists, which makes the algorithm visit the search space thoroughly. For these cases, this algorithm may be inapplicable and a heuristic algorithm should be applied to. For these cases, we can apply this algorithm in such a way that, for limited depth, the iterative deepening depth first search is applied first, then other heuristic algorithm if no solution is found.

As a future work, we will apply this algorithm to real scheduling data with more constraints to assess its applicability to real problem.

ACKNOWLEDGEMENT

This research was supported by The Leading Human Resource Training Program of Regional Neo industry through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and future Planning(2016H1D5A1910630).

REFERENCES

[1] Moz, M., and Vaz Pato, M., 2007, "A genetic algorithm approach to a nurse reros- tering problem," Comput Oper Res, 34 (3), pp.667–691.

[2] Warner, D. M., and Prawda, J., 1972, "A mathematical programming model for scheduling nursing personnel in a hospital," Manag Sci, 19 (4-Part-1), pp.411–422.

[3] Cheang, B., Li, H., Lim, A., and Rodrigues, B., 2003, "Nurse rostering problems—a bibliographic survey," Eur J Oper Res, 151 (3), pp.447–460.

[4] Ernst, A. T., Jiang, H., Krishnamoorthy, M., and Sier, D., 2004, "Staff scheduling and rostering: A review of applications, methods and models," Eur J Oper Res, 153 (1), pp.3–27.

[5] Özcan, E., 2005, "Memetic algorithms for nurse rostering," in: Yolum, P., Güngör, T., Gürgen, F., and Özturan, C. (Eds.), ISCIS'05: Proceedings of the 20th international conference on Computer and Information Sciences, Springer-Verlag, pp.482–492.

[6] Causmaecker, P., and Berghe, G., 2011, "A categorisation of nurse rostering problems," J Sched, 14, pp.3–16.

[7] Moz, M., and Pato, M. V., 2003, "An integer multicommodity flow model applied to the rerostering of nurse schedules," Ann Oper Res, 119 (1/4), pp.285–301.

[8] Maenhout, B., and Vanhoucke, M., 2013, "Reconstructing nurse schedules: Computational insights in the problem size parameters," Omega, 41 (5), pp. 903–918.

[9] Moz, M., 2003, "Técnicas de Investigação Operacional Aplicadas a um Problema de Escalonamento de Pessoal em Contexto Hospitalar," Ph.D. thesis, Instituto Superior de Economia e Gestão, Universidade Técnica de Lisboa, Portugal.

[10] Moz, M., and Pato, M. V., 2004, "Solving the problem of rerostering nurse schedules with hard constraints: new multicommodity flow models," Ann Oper Res, 128 (1-4), pp.179–197.

[11] Knighton, S. A., 2005, "An optimal network-based approach to scheduling and re-rostering continuous heterogeneous workforces," Ph.D. thesis, Arizona State University, Tempe AZ.

[12] Pato, M. V., and Moz, M., 2008, "Solving a bi-objective nurse rerostering problem by using a utopic Pareto genetic heuristic," J Heuristics, 14 (4), pp.359–374.

[13] Moz, M., and Pato, M. V., 2008, "On modelling and solving nurse rostering and rerostering problems," in: Proceedings of the 1st International Conference on Applied Operational Research, 9–17.

[14] Chiaramonte, M. V., 2008, "Competitive nurse rostering and rerostering," Ph.D. thesis, Arizona State University, Tempe AZ.

[15] Maenhout, B., and Vanhoucke, M., 2011, "An evolutionary approach for the nurse rerostering problem," Comput Oper Res, 38 (10), pp.1400–1411.

[16] Bäumelt, Z., Dvořák, J., Šůcha, P., and Hanzálek, Z., 2013, "An acceleration of the algorithm for the nurse rerostering problem on a graphics processing unit," in: Proceedings of the 5th International Conference on Applied Operational Research, pp.101–110.

[17] Slate, D. J., and Atkin, L. R., 1983, "CHESS 4.5—The Northwestern University chess program," in: P. W. Frey (Ed.), Chess Skill in Man and Machine, Springer New York, New York, NY, pp.82–118.

[18] Berliner, H., and Goetsch, G., 1984, "A quantitative study of search methods and the effect of constraint satisfaction," Tech. Rep. CMU-CS-84-187, Carnegie-Mellon University, Pittsburgh, PA.

[19] Korf, R. E., 1985, "Depth-first iterative-deepening: An optimal admissible tree search*," Artif Intell, 27 (1), pp.97–109.

[20] Osogami, T., and Imai, H., 2000, "Classification of various neighborhood operations for the nurse scheduling problem," in: ISAAC '00: Proceedings of the 11th International Conference on Algorithms and Computation, Springer-Verlag, pp.72–83.

[21] Kim, S.-J., Ko, Y.-W., Uhmn, S., and Kim, J., 2013, "An efficient method for nurse scheduling problem using the genetic algorithm," ASTL, 23, pp.252– 255.

[22] Ko, Y.-W., Kim, D., Jeong, M., Jeon, W., Uhmn, S., Kim, J., 2013, "An improvement technique for simulated annealing and its application to nurse scheduling problem," IJSEIA, 7 (4), pp.269–278.