Nurse Rostering Problem Using Cuckoo Search

Yun-hee WOO, Saang-yong Uhmn, Zin-Woo MIN and Jin KIM

Dept. of Computer Engineering, Hallym University
Chuncheon, Gangwondo 24252 Republic of Korea
*Corresponding author

Keywords: Nurse rostering, Cuckoo search algorithm, Nature-inspired algorithm, Optimization Algorithm

Abstract. Nurse rostering problem is to construct a nurse roster for a given period, which is NP-hard. In this paper, we apply cuckoo search algorithm to the problem and adjust the parameters of the algorithm through experiments. We show that the proposed algorithm performed well under many circumstances by comparison between our algorithm and other approximation algorithms.

Introduction

In organizations providing round-the-clock services such as hospitals, workers are assigned to a set of shifts in a roster which is announced in advance and should satisfy government regulations and contracts clauses. The problem of creating the roster in a hospital is called nurse rostering problem. The goal of the problem is simple: given the number of nurses and the period that they should work, create schedules for nurses. It is one of the most popular NP-hard problems which may not have a polynomial algorithm so that the time to find a solution grows exponentially with problem size. Because of its complexity, many approximation algorithms were applied [1,2,3,4]. A bibliographical survey of the problem can be found in [5]. Because of the page limit, we briefly present the problem and its conditions in the following section. To find a feasible solution, we applied cuckoo search algorithm which was introduced by Yang and Deb [7]. It is a nature-inspired optimization algorithm which is briefly described in Section 3. To justify its applicability to the problem, we compared its performance with simulated annealing algorithm in Section 4.

Nurse Rostering Problem

The goal of the nurse rostering problem is that for the given number of nurses and the period that they should work, schedules for the nurses need to be made. However, it is not that simple to solve this problem because there are several conditions to consider. For example, there should be several days-off in a week or limitation of working hours. There are various conditions that can be applied by the hospital and must be satisfied by laws or the terms of contracts. These are called hard constraints. On the other hand, there are others that do not need to be observed. These are called soft constraints. The environment that is usually addressed in nurse rostering problem is three shifts 24 hours a day such as in a large hospital. Sometimes nurses may want to work at certain shifts on certain days because of personal reasons. These needs should be met as well as possible. Although there can be various constraints, we will consider three constraints that are commonly applied in real life. The constraints are as follows.

a. The number of nurses for each shift of a day is specified by the upper and lower limits.
b. There are restrictions on the pattern of shifts of a nurse. Morning shift or evening shift after night shift, morning shift after evening shift and three consecutive night shifts should be avoided.
c. The number of morning, evening, and night shifts and days-off of each nurse should be fixed.

The constraint (a) means the required number of nurses to provide adequate medical care. we set 4-6 nurses for the morning shift, 3-5 nurses for the evening, 1-3 nurses for the night. Constraint (b) must also be met because nurses must have enough rest after work. Therefore, constraints (a) and (b)
are hard constraints. In the case of constraint (c), nurses are basically on two morning shifts, two evening, two days-off and one night shift in a week. These numbers increase as the period gets longer. For example, the numbers are doubled for two weeks. The final one, (c), is considered as a soft constraint in this study. When the numbers of violations of constraint (a), (b) and (c) are \( f_1, f_2, \) and \( f_3 \), respectively, the cost of a roster, \( C \), is:

\[
C = f_1 \cdot w_1 + f_2 \cdot w_2 + f_3 \cdot w_3
\]

The weight implies the importance of each constraint. Since the constraints (a) and (b) are hard constraints, \( w_1 \) and \( w_2 \) are set to 5, and the constraint (c) is soft constraint so that \( w_3 \) to 1. Our goal is to find a nurse schedule that minimizes the cost function \( C \).

The schedule that \( N \) nurses can work for \( D \) days is a matrix of size \( N \times D \). There are four possible values that the element of the matrix can have: morning\((m)\), evening\((e)\), night\((n)\) and a day-off\((o)\). The number of possible schedules is \( 4^{(N \times D)} \) and the size of search space increases exponentially as \( N \) and \( D \) grow. Therefore, it is very inefficient to create all possible schedules, calculate the cost of them, and find the schedule with the minimum cost. To solve this problem, we use cuckoo search algorithm, which is an approximation algorithm.

**Cuckoo Search Algorithm**

Cuckoo search (CS) is one of nature-inspired heuristic algorithms which was developed by Yang and Deb [7]. It is derived from cuckoo’s habit of laying eggs in other bird’s nest and letting host birds grow eggs. It can be described by three rules:

1. A cuckoo lays one egg at a time, and a nest to lay an egg is randomly selected.
2. Nests with high quality eggs remain in the next generation.
3. The number of nests is fixed and the host can find cuckoo eggs with a probability \( P_a \in [0, 1] \). In this case, the host throws away the eggs or leaves the nest and builds a new nest.

In the algorithm, a cuckoo randomly chooses one nest for every generation and lays an egg. The next host is selected by Lévy flight. Lévy flight is a type of random walk that normally searches around an existing area but has a low probability of jumping largely to distant areas. Several studies have shown that it is similar to the hunting and flight paths of many animals and insects [6]. The nests with cuckoos’ eggs of high quality remain for the next generation. The eggs are solutions. This means that a better solution than an existing one will replace the worse and be carried over to the next generation. In addition, the host discovers cuckoo’s egg with a certain probability, which is the process of replacing bad solutions with new ones. The interested readers may refer to others for the detailed description of the algorithm [7,8]. The algorithm using Lévy flight is shown in Algorithm. 1.
Algorithm 1 Cuckoo search algorithm for nurse rostering problem

Input: The number of nests, \( n \)
The maximum number of generations, \( MaxGeneration \)
An object function \( f(x) \)
Output: An optimized roster

Generate an initial set of \( n \) rosters
Evaluate the cost and the rank of the rosters
while (\( t < MaxGeneration \)) or (stop criterion is not satisfied) do
Select a roster randomly by Lévy flight
Make a new roster (say, \( i \)) from the selected roster by Lévy flight
Evaluate cost \( F_i \)
Choose a roster (say, \( j \)) randomly among \( n \) rosters
if \( F_i < F_j \) then
Replace \( j \) by the new roster
endif
Replace a fraction(\( P_a \)) of worse rosters with newly generated ones
Rank the rosters and find the roster with the lowest cost
end while
return Roster with the lowest cost

Experiment

The algorithm was implemented in C on Ubuntu 16.04 LTS with Intel Core i7-920 2.66 GHz CPU and 16 GB of memory. The number of nurses is set at 15 and the numbers of weeks are one, two, three, and four. Since CS uses Lévy flight to explore the search space and the exploration in this problem is creating rosters evenly over the space, we must decide how to use it to generate new schedules. First, we used it as the probability of mutation. Each element of a roster mutates to other possible shift based on the value from Lévy flight. Although this method provided solutions, the more the number of weeks, the more it was called. As an alternative, we used it as the degree of variation, the number of elements to mutate for a generation. Because the function provides integer values, we take an absolute value of it.

In order to ensure the stability of modified schedule, the variation is set to occur in the areas that do not match the best schedule so far, but also even in matching part with a probability of 10%. In CS, there is a probability \( P_a \in [0, 1] \) that the host bird will notice the cuckoo’s egg with low quality. However, we found through separate experiment that it is better to set \( P_a \) to 0 for the current conditions. Fig. 1 shows the effect of \( P_a \) in time: for the solutions of the same quality, the higher \( P_a \) is, the longer in time it took. We conjectured that this was due to the size of the search space. Because we didn’t find a reason to apply other values except zero for \( P_a \), we conducted experiments with \( P_a = 0\% \), that is, no random drop of eggs. Initial schedules are randomly generated. All experiments are set 106
for the maximum number of generations and carried out 100 times. The followings show the averages of the results. First, we had to find a suitable value for the number of nests. We conducted experiments with several values and the results are depicted in Fig. 2 (a) and Fig. 2 (b).

![Figure 2](image)

(a) Average execution time by period per nest

(b) Average cost by period per nest

Figure 2. Average execution time and cost by period per nest.

Under the given conditions, the smaller the number of nests, the better the performance in general: faster in time and smaller in cost. It is also caused by the size of the solution space: The larger the number of nests, the bigger the size of the space. So, we set the number of nests to 10. Table 1 shows the differences in the results between the proposed method and simulated annealing (SA). The criteria for comparing the two algorithms are the average execution time and the average cost. The execution time is the time from the initial schedule creation to the final schedule. Both CS and SA found solutions with the average cost of zero, which means the algorithms found optimal solutions. However, we can see that the average execution time of CS is much smaller than that of SA. Fig. 3 shows the difference in the average execution time of CS and SA more clearly.

![Figure 3](image)

Figure 3. Average execution time for CS and SA.

<table>
<thead>
<tr>
<th>Period</th>
<th>Algorithm</th>
<th>Average Cost</th>
<th>Average Time[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 week</td>
<td>CS</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td>0</td>
<td>1.23</td>
</tr>
<tr>
<td>2 weeks</td>
<td>CS</td>
<td>0</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td>0</td>
<td>3.16</td>
</tr>
<tr>
<td>3 weeks</td>
<td>CS</td>
<td>0</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td>0</td>
<td>8.71</td>
</tr>
<tr>
<td>4 weeks</td>
<td>CS</td>
<td>0</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td>0</td>
<td>12.17</td>
</tr>
</tbody>
</table>
Conclusion

In this paper, we applied cuckoo search to solve nurse rostering problem and compared the results with those of simulated annealing. To achieve better performance, we adjusted the parameters of the algorithm through experiments. Although both algorithms provided the solutions with similar quality, cuckoo search algorithm found the solutions faster in time. With this success, we are going to apply this approach to other scheduling and application problems for further research.

Acknowledgement

This research was supported by Hallym University Research Fund, 2014 (HRF-201407-015).

References


