An Efficient Method for Nurse Scheduling Problem using the Genetic Algorithm

Sun- Jeong Kim, Young-Woong Ko, Saangyong Uhmn and Jin Kim*
Dept. of Computer Engineering
Hallym University
Chuncheon, Gangwondo 200-702 Republic of Korea
{sunkim,yuko,suhmn,jinkim}@hallym.ac.kr

Abstract. We applied a genetic algorithm to the nurse scheduling problem. To improve performance of the genetic algorithm, we suggested an efficient operator using a cost bit matrix. The experimental results showed that the suggested method generated a nurse scheduling faster in time and better in quality compared to a traditional genetic algorithm.

Keywords: nurse scheduling problem, genetic algorithm, cost bit matrix

1 Introduction

Any organization providing a round-the-clock service divides its daily work into consecutive shifts each of which is a period of time in which a group of employees are in service, which satisfies several constraints that may be set up by staffing requirements, rules by the administration and labor contract clauses. In the nurse scheduling problem (NSP), nurses are assigned into a set of shifts and off-days in a timetable, a nurse roster. Even the simplified one was proven to be NP-hard [1].

Several techniques were applied to NSP for quality or computation time of solutions [2-5]. Jan et al. and Aickelin et al. applied genetic algorithm (GA) [6,7]. Kundu et al. applied GA and simulated annealing and compared their performances [8].

In this study, we considered the cyclic nurse scheduling problem with the constraints as in [8]. The objective of the problem is to satisfy nurses’ requests as much as possible while fulfilling the employers’ concerns. We suggested a cost bit matrix based genetic algorithm (CMGA) to find a schedule optimizing the given constraints.

2 Problem Description

2.1 Nurse Scheduling Problem

NSP is essentially a scheduling problem with a number of constraints. There are two kinds of constraints: hard constraints must be always satisfied and soft constraints are to be satisfied as much as possible. Because the main objective of this study is to show a fast and efficient GA approach, we confined the constraints as follows.

(a) Hard constraints
(i) The number of nurses for each working shift of one day and an off-day
(ii) The prohibited working patterns: Morning after night shift, evening after night, morning after evening shift and three consecutive night
2.2 Cost Function

Let \( N \) and \( D \) be the number of nurses and days and \( s \) be one of the three shifts or a day-off. Then, NSP is represented as a problem to decide an \( N \times D \) matrix \( X \) so that each \( x_{ij} \in X \) expresses that nurse \( i \) works on day \( j \) or is off where \( x_{ij} = \{ m, e, n, o \} \).

(a) We define \( m_i, e_i, n_i \) as the total number of nurses for morning, evening and night shift on day \( j \). If any of these numbers are not between minimum and maximum requirements of each shift \( (m_{\text{min}}, m_{\text{max}}, e_{\text{min}}, e_{\text{max}}, n_{\text{min}}, n_{\text{max}}) \), cost \( c_1 \) will be incremented by 1.

(b) Any violation of working patterns will increment cost \( c_2 \) by 1.

(c) We define \( M_i, E_i, N_i, O_i \) as the total number of morning, evening and night shifts and off-days for nurse \( i \) and \( M_{\text{req}}, E_{\text{req}}, N_{\text{req}}, O_{\text{req}} \) as the required number of morning, evening and night shifts and off-days for all nurses during a period of \( D \). If any of those total numbers doesn’t meet these required numbers, respectively, cost \( c_3 \) will be incremented by 1.

Different weight values can be assigned for the costs \( c_1, c_2 \) and \( c_3 \). Then, the final cost function is

\[
 f = c_1 \times w_1 + c_2 \times w_2 + c_3 \times w_3
\]

where \( w_1, w_2 \) and \( w_3 \) are weight values for \( c_1, c_2 \) and \( c_3 \), respectively.

The number of all possible nurse schedules is \( 4^{DN} \). If \( D \) and \( N \) increase, this approach is intractable, which means it is NP-hard [10]. To overcome this problem, we applied a genetic algorithm.

2.3 Cost Bit Matrix and Mutation for CMGA

A selection and a crossover operation generate two new schedules. With mutation probability \( P_m \), the two schedules can be mutated. We use a cost bit matrix \( V \) which is also an \( N \times D \) matrix to apply a mutation operator to the current schedule to make a new one more efficiently. The value of each cell in \( V \) is assigned to 0 or 1 when the cost of a new schedule is calculated. Initially, the values of all cells in \( V \) are set to 0. The value 1 in a cell of \( V \) shows any violation of the constraints. Now we apply a mutation rule to the cell \( x_{ij} \) of a schedule with cell change probability \( p_{cc} \) only if the value of the corresponding cell \( v_{ij} \) is 1.

3 Experiments and results

Each method was applied to 100 instances of NSP. The number of nurses was 15 and the number of weeks were from 1 to 4. The weights were \( w_1=5, w_2=5, w_3=1 \). Hard constraints are same for all the problems \((d_{\text{min}}=4, d_{\text{max}}=6, e_{\text{min}}=3, e_{\text{max}}=5, n_{\text{min}}=3, n_{\text{max}}=5)\) and soft constraints are proportional to the period \((D_{\text{req}}=2, E_{\text{req}}=2, N_{\text{req}}=2, O_{\text{req}}=1 \) for 1 week). The crossover probability \( P_c = 0.03 \), the mutation probability \( P_m = 0.01 \) and the cell change probability \( p_{cc} = 0.01 \) were applied. We applied a simple crossover, a multi-point crossover and a uniform crossover operation in our GA.
The methods were compared on four criteria: number of problem solved with cost=0 ($f_{opt}=0$), average cost of solutions, average number of iterations to reach a final state $s_{opt}$ and time $T_{opt}$. Judged on the basis of these four criteria, the CMGA is the better method. Table 1 shows the performance of CMGA and traditional GA (TGA).

<table>
<thead>
<tr>
<th>Period</th>
<th>Method</th>
<th>$f_{opt}$</th>
<th>$I_{opt}$ / $k$</th>
<th>$T_{opt}$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 week</td>
<td>CMGA</td>
<td>79/100</td>
<td>305612 / $10^6$</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>TGA</td>
<td>4/100</td>
<td>546054 / $10^6$</td>
<td>1.9</td>
</tr>
<tr>
<td>2 weeks</td>
<td>CMGA</td>
<td>54/100</td>
<td>2900782 / $5 \times 10^6$</td>
<td>23.9</td>
</tr>
<tr>
<td></td>
<td>TGA</td>
<td>0/100</td>
<td>2312737 / $5 \times 10^6$</td>
<td>30.6</td>
</tr>
<tr>
<td>3 weeks</td>
<td>CMGA</td>
<td>72/100</td>
<td>12192435 / $20 \times 10^6$</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>TGA</td>
<td>0/100</td>
<td>9588709 / $20 \times 10^6$</td>
<td>121</td>
</tr>
<tr>
<td>4 weeks</td>
<td>CMGA</td>
<td>97/100</td>
<td>51631978 / $100 \times 10^6$</td>
<td>926</td>
</tr>
<tr>
<td></td>
<td>TGA</td>
<td>0/100</td>
<td>56574940 / $100 \times 10^6$</td>
<td>1607</td>
</tr>
</tbody>
</table>

CMGA generated schedules with the optimal cost ($f_{opt}=0$) in all periods whereas TGA did not except 1 week. Also, the average costs of $f_{opt}$ from CMGA were smaller than TGA and CPU time $T_{opt}$ of CMGA was faster than that of TGA. Among four crossover operations, the best one was the simple crossover. Judged from the results, CMGA was very efficient compared to TGA.

4 Conclusion and Future Work

In this paper, we proposed an efficient method for NSP using GA. In this method, a cost bit matrix is used for operators in GA. This approach generated a nurse schedule faster in speed and better in quality than traditional GA. Although we have presented this work in terms of nurse scheduling, it should be noticed that the main idea of the approach could be applied to many other scheduling problems.

References