International Journal of Software Engineering and Its Applications
Vol.8, No.1 (2014), pp.53-62
http://dx.doi.org/10.14257/ijseia.2014.8.1.05

# A Strategy to Improve Performance of Genetic Algorithm for Nurse Scheduling Problem

Sun-Jeong Kim[1], Young-Woong Ko[2], Saangyong Uhmn[2] and Jin Kim[2]

[1]*Dept. of Ubiquitous Computing,* [2]*Dept. of Computer Engineering*
*Hallym University*
*Chuncheon, Gangwondo 200-702 Republic of Korea*
*{sunkim,yuko,suhmn,jinkim}@hallym.ac.kr*

***Abstract***

*We applied genetic algorithm to nurse scheduling problem. For time complexity problem of genetic algorithm, we suggested efficient operators using a cost bit matrix of which each cell indicates any violation of constraints. A cell with 1 indicates that the corresponding assignment violates constraints and needs no further consideration. The experimental results showed that the suggested method generated a nurse scheduling faster in time and better in quality compared to the traditional genetic algorithm.*

***Keywords:*** *nurse scheduling problem (NSP), genetic algorithm, cost bit matrix*

## 1. Introduction

Any organization providing a round-the-clock service divides its daily work into consecutive shifts. A shift is a period of time in which a group of employees is in-service. In such an organization, an employee is assigned to the set of shifts, which satisfies several constraints that may be set up by staffing requirements, rules by the administration and labor contract clauses.

Nurse scheduling problem (NSP) is an instance of a scheduling problem in which each nurse is assigned to the set of shifts and rest days in a timetable called a nurse roster [1-5]. It was proven to be NP-hard even with only subset of real world constraints [6].

Miller *et al.*, and Warner *et al.*, formulated NSP as the selection of a timetable that minimized an objective function that balanced the trade-off between staffing coverage and nurses' preferences [7, 8]. They simplified the problem in which they included too small or ignored too many constraints to be practical. Abdannadher *et al.*, applied Constraint Logic Program (CLP) framework and Li *et al.*, employed Bayesian optimization algorithm [9, 10]. Jan *et al.*, and Aickelin *et al.*, applied the genetic algorithms (GA) to NSP [11, 12]. Kundu *et al.*, applied genetic algorithm and simulated annealing (SA) to the same problem instances and compared their performances with others [13, 14].

Because NSP may include many constraints and there can be several different instances with different set of constraints, the problem instance must be defined clearly. In this study, we consider the cyclic nurse scheduling problem with the following constraints as in [8]. An instance includes three components (1) the preference of each nurse as the aversion to work on a particular days and shifts, (2) minimal coverage constraints of minimal required nurses per shift and per day, (3) case-specific constraints by personal time requirements, specific workplace conditions, *etc*. The objective of this problem is to satisfy nurses' requests as much as possible while fulfilling the employers' concerns.

ISSN: 1738-9984 IJSEIA

GA is adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetics [15]. The basic concepts of the GA were designed to simulate those processes in natural evolution system, specifically those that follow the principles by Charles Darwin of survival of the fittest. GA is a powerful tool to solve optimization problem with multiple variables. GA does not need to calculate the differential value of the fitness function, because the natural selection process is determined by the fitness of chromosomes. GA was applied several scheduling problems [16, 17].

In this paper, we suggest an improvement to genetic algorithm with a cost bit matrix (CMGA) to find a schedule that optimizes a set of constraints. In the next section, we will briefly introduce NSP and its cost function and in Section 3, GA, the cost bit matrix and operators in CMGA. Section 4 provides computational experiments and results. Finally, conclusions and further work are discussed in Section 5.

## 2. Problem Description

### 2.1. Nurse Scheduling Problem

NSP is to create weekly or monthly schedules for *n* nurses by assigning one out of a number of possible shift patterns to each nurse. These schedules have to satisfy working contacts and meet the requirements for the number of nurses of different grades for each shift, while being seen to be fair by the staff concerned. Therefore, NSP is essentially a scheduling problem that suits a number of constraints. Constraints are usually categorized into two categories: soft and hard constraints. Hard constraints should be always satisfied in any working schedule so that there will be no breaches. Any schedule that does not satisfy any of hard constraints cannot be a feasible one. Possible examples include restrictions on the number of nurses for each shift; the maximum number of shifts in a week, a month, *etc*. On the contrary, soft constraints can be violated but as minimal as possible. In other words, the soft constraints are expected to be satisfied, but violation does not make it an infeasible solution. Some examples are the demand for a desired day off, the demand for doing certain shift on a certain day with a certain nurses, *etc*. Generally, every nurse works on three shifts, morning, evening, and night, and some holidays.

There are various kinds of hard and soft constraints we may consider. Because the main objective of this study is to show fast and efficient GA approach, we confined the constraints as follows.

(a) Hard constraints

(i) There are constraints on the number of nurses for each working shift per day. The number of nurses for morning, evening, and night shift should be between the minimum and maximum values.

(ii) There are constraints for the working patterns. Morning after night shift, evening after night, morning after evening shift and three consecutive night shifts are restricted combination of working patterns.

(b) Soft constraints

There are constraints for the total number of off-days (*o*), night (*n*), morning (*m*) and evening (*e*) shifts during the certain period of days for each nurse.

 

### 2.2. Cost Function

We have to define a cost function to optimize to obtain optimal schedules for each nurse. Let $N$, $D$ be the number of nurses and days, and $s$ be one of the three shifts or a day-off required to be scheduled. Then, NSP may be represented as a problem to decide an $N \times D$ matrix, so that each element of the matrix, $x_{ij}$ expresses that nurse $i$ works on day $j$ where $x_{ij} = \{m, e, n, o\}$.

(a) To evaluate the violation of hard constraint (i), we define $m_j$, $e_j$, $n_j$ as the total number of nurses for morning, evening, and night shift on day $j$. If any of these numbers are not between the minimum and maximum number of nurses for each shift ($m_{\min}$, $m_{\max}$, $e_{\min}$, $e_{\min}$, $n_{\min}$, $n_{\max}$), cost $c_1$ will be incremented by 1.

(b) To evaluate the violation of hard constraint (ii), working patterns are examined. Any violation of working patterns will increment cost $c_2$ by 1. Figure 1 shows the description of calculation of $c_1$ *and* $c_2$.

(c) To evaluate the violation of soft constraint, we define $M_i$, $E_i$, $N_i$, $O_i$ as the total number of the corresponding shifts, morning, evening and night and off-days for nurse $i$ during the period of $D$ and $M_{req}$, $E_{req}$, $N_{req}$, $O_{req}$ as the required number of morning, night and night shifts and off-days for all nurses during the period of $D$. If any of these numbers $M_i$, $E_i$, $N_i$, $O_i$ does not meet $M_{req}$, $E_{req}$, $N_{req}$, $O_{req}$ respectively, cost $c_3$ will be incremented by 1. Figure 2 shows the description of calculation of $c_3$.

```
C1(){
   for(j=1; j<=N; j++){
      if ((mj<dmin)||(dmax<mj)||
          (ej<emin)||(emax<ej)||
          (nj<nmin)||(nmax<nj)) {
        c1=c1+1;
      }
      if (c1 increased) {
         for (i=1; i<=D; i++) {
            vij=1;
         }
      }
}
C2(){
   For (i=1; i<=D; i++) {
      For (j=1; j<=D; j++) {
         if ((xij-1==n)&&(xij==m)) { c2=c2+1; }
         if ((xij-1==e)&&(xij==n)) { c2=c2+1; }
         if ((xij-1==m)&&(xij==e)) { c2=c2+1; }
         if ((xij-2==n)&&(xij-1==n)&&(xij==n)) { c2=c2+1; }
         if (c2 increased) { vij=1; }
      }
   }
}
```

**Figure 1. Description of Cost Functions c1() and c2()**

```
C3(){
   for (i=1; i<=D; j++){
      if ((Mi!=Mreq)||(Ei!=Ereq)||(Ni!=Nreq)||(Oi!=Oreq)) { c3=c3+1; }
      if (c3 increased) {
         for (j=1; j<=N; j++) {
            vij=1;
         }
      }
   }
}
```

**Figure 2. Description of the Cost Function c3()**

Different weight values can be assigned for the costs *c1*, *c2* and *c3*. Then, the final cost function is

$$f = c_1 * w_1 + c_2 * w_2 + c_3 * w_3$$

where $w_1$, $w_2$ and $w_3$ are weight values for $c_1$, $c_2$ and $c_3$, respectively.

Our goal is to minimize the cost function *f* so as to find an optimal nurse schedule. The simplest method to find the solution is a brute force approach evaluating all possible nurse schedules and finding the feasible one with the minimum cost among them. It guarantees that it finds a feasible schedule with the minimum cost. However, the number of all possible nurse schedules is $4^{DxN}$. If *D* and *N* increase, this approach is intractable. This is a class of problems for which it is believed that no efficient algorithm exists, called *NP-hard* [18]. In other words, the algorithms that guarantee to find an optimal solution with the size of *D* and *N* in reasonable time may not exist. To overcome this problem, we use genetic algorithm which is an approximation algorithm. GA provides an approximate solution rather than an optimal one in acceptable time.

## 3. Algorithmic Flow of CMGA

### 3.1. Genetic Algorithms

GA is a search algorithm to simulate the process of natural selection. GA starts with the set of potential solutions called population and evolves toward more optimal solutions. The solutions are evaluated by a fitness function. The fitness value represents the quality measure of a solution so that the algorithm can use it to select ones with better genetic material for producing new solutions and further generations. This simulation of evolution allows survival of better solutions and extinction of inferior ones. The goal is to find better solutions in each generation. The process of evolution is carried out by selection, crossover and mutation. In terms of GA, those processes are called genetic operators. The selection chooses superior solutions in every generation and assures that inferior solutions extinct. The crossover operator chooses two solutions from current population and generates a new solution based on their genetic material. Selection and crossover operators will expand good features of superior individuals through the whole population. They will also direct the search process towards a local optimum. The mutation operator changes the value of some genes in a solution and help to search other parts of problem space.

The main disadvantage of GA is its requirement for a large amount of computation time [10]. This is because GA is based on Monte Carlo method, which allows a new state with a

 

higher cost than that of a current state. To reduce this computation time, we proposed a speedup strategy using a cost bit matrix.

### 3.2. Selection and Crossover for CMGA

Initial population, the first *n* schedules for nurses that are $N \times D$ matrices, is generated by randomly assigning each nurse to one of the three shifts or a day-off on each day. The costs of these schedules are calculated by cost functions *c1*, *c2* and *c3*. There are many different types of selection. In this study, roulette wheel selection that is the most common type of selection method is applied. Two schedules, $p_1$ and $p_2$, are chosen randomly based on their costs and used to produce offspring. One schedule can be selected for a parent more than once.

Crossover is an operator by which new individuals are created from the information contained in the parents. There are many different types of crossover including single-point crossover, two-point crossover and uniform crossover. In this study, we use single-point crossover. With crossover probability of $P_c$, crossover occurred at single point of $p_1$ and $p_2$ to form two children. In other words, the first $n/2$ of a nurse schedule of $p_1$, and the second $n/2$ of a nurse schedule of $p_2$ make one offspring, as well as the second $n/2$ of a nurse schedule of $p_2$ and the second n/2 of a nurse schedule of $p_2$ make another offspring. If no crossover takes place, the exact copies of $p_1$ and $p_2$ can be two children.

### 3.3. A Cost Bit Matrix and Mutation for CMGA

After selection and crossover, two new schedules can be obtained. With mutation probability $P_m$, the two schedules can be mutated. In this study, we use a cost bit matrix, *v*, which is an $N \times D$ matrix to apply mutation operator to the current schedule to make a new state more efficiently. The value of each cell in the matrix *v* is assigned to 0 or 1 when the cost of new schedule is calculated. Initially, the value in each cell in *v* is set to 0. When the cost functions `c1`(), `c2`(), and `c3`() are applied to the a new schedule, if cost is increased by 1, the value of the corresponding cell in *v* is set to 1. Table 1 shows a sample schedule and corresponding cost bit matrix.

**Table 1. A Sample Schedule Table and its Corresponding Cost Bit Matrix**
**$m_{min}=m_{max}=2$, $e_{min}=e_{max}=1$, $n_{min}=n_{max}=1$, $M_{req}=2$, $E_{req}=2$, $N_{req}=2$, $O_{req}=1$**

| nurse | M | T | W | T | F | S | S | m | e | n | o |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n1 | *n* | *n* | *o* | *m* | *m* | *e* | *e* | 2 | 2 | 2 | 1 |
| n2 | *e* | *e* | *m* | *o* | *m* | *n* | *n* | 2 | 2 | 2 | 1 |
| n3 | *m* | *m* | *e* | *e* | *n* | *n* | *o* | 2 | 2 | 2 | 1 |
| n4 | *o* | *m* | *m* | *n* | *e* | *e* | *n* | 2 | 2 | 2 | 1 |
| n5 | *m* | *o* | *n* | *m* | *n* | *e* | *e* | 2 | 2 | 2 | 1 |
| m | 2 | 2 | 2 | 2 | 2 | 0 | 0 | | | | |
| e | 1 | 1 | 1 | 1 | 1 | 3 | 2 | | | | |
| n | 1 | 1 | 1 | 1 | 2 | 2 | 2 | | | | |
| o | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | | | |

| | M | T | W | T | F | S | S |
|---|---|---|---|---|---|---|---|
| n1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| n2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| n3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| n4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| n5 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

```
Mutation(){
   For(i=1; i<=M; i++) {
      For(j=1; j<=N; j++) {
         If ((rand()%100<p)&&(v_ij==1) {
            x_ij=random(d,e,n, o);
}
```

**Figure 3. Mutation for GA**

Figure 1 shows how to set the value of *v*. The value 1 in a cell of the matrix *v* shows the violation of constraints. Now, we apply mutation rule to the cell $x_{ij}$ in a schedule with cell change probability $P_{cc}$ only if the corresponding cell $v_{ij}$=1. Figure 3 shows the description of our mutation rule.

Figure 4 shows the pseudo code of GA for the experiments in this paper.

**Procedure GA**

1. Start with *n* nurse schedules generated randomly.
2. Calculate the fitness *f*(*y*) of each nurse schedule *y* in the population.
3. Repeat the following steps until *n* children are created.
   (a) Select a pair of nurse schedule $p_1$ and $p_2$ from the current population with the probability of selection that is an increasing function of fitness.
   (b) With probability $P_c$ (crossover probability), cross over $p_1$ and $p_2$ at the middle to form two children. If no crossover takes place, form two children that are exact copies of their respective parents.
   (c) Mutate the two children with probability $P_m$ (mutation probability) and place the resulting chromosomes in the new population.
4. Replace the current population with new population.
5. Go to step 3 until a desirable solution is found or the maximum number of generations is completed.

**Figure 4. Pseudocode for GA**

## 4. Experiments and Results

The proposed CMGA and traditional GA (TGA) were implemented in C language and run on a PC with an Intel® Core(TM) i5-2520M 2.5 Ghz CPU and 4GB of memory. All the constraints described in this paper were included in both methods with exact same conditions. The goal was to check whether the proposed algorithm could actually generate an acceptable NSP and compare the proposed GA with traditional one. The random number generator `rand()` was used. Identical problem instances were given to both methods. Each set of instances is consisted of 100 problems generated randomly. The number of nurses, *N*, is 15 and the number of weeks, *D*, is from one to four. The weights for cost functions are $w_1$=5, $w_2$=5 and $w_3$=1, respectively. Hard constraints are same for all the problems ($d_{\min}$=4, $d_{\max}$=6, $e_{\min}$=3, $e_{max}$=5, $n_{\min}$=3, $n_{\max}$=5) and soft constraints for one week are $D_{req}$=2, $E_{req}$=2, $N_{req}$=2, and $O_{req}$=1, and are proportional to the period for two to four weeks. The crossover probability $P_c$=0.03, mutation probability $P_m$=0.01 and cell change probability $p_{cc}$=0.01 were applied. We applied simple crossover, multi-point crossover and uniform crossover in our GA and compared their effects.

 

The methods were compared on the bases of four criteria, number of problem solved with cost=0 ($f_{opt}$=0), the average cost of the solution obtained, the average number of iterations to reach the final state $s_{opt}$ and the execution time $T_{opt}$. The CMGA is the better method, judged on the bases of the four criteria. Table 1 shows the performance results of two methods.

Both methods solved all the given problem instances and the results did not violate any of hard constraints in all periods. CMGA generated schedules with optimal cost ($f_{opt}$=0) in all periods whereas TGA could not generate schedules with optimal cost except 1 week. Also, the average costs, $f_{opt}$, from CMGA were smaller than TGA and the execution times, $T_{opt}$, of CMGA were much faster than those of TGA. We applied simple crossover, multi-point crossover and uniform crossover in our GA and the best one was simple crossover.

CMGA was very effective compared to TGA based on this comparison. In all the quality of the solution, CMGA was very impressive due to its powerful operators with a cost bit matrix.

**Table 2. Comparison of CMGA and TGA**

| Period | Method | $f_{opt}$=0 | $f_{opt}$ | $I_{opt}/k$ | $T_{opt}$ (sec) |
|---|---|---|---|---|---|
| 1week | CMGA | 79/100 | 8.14(2) | 305612 / 1x10$^6$ | 1.1 |
| | TGA | 4/100 | 8.12(2) | 546054 / 1x10$^6$ | 1.9 |
| 2 weeks | CMGA | 54/100 | 7.96(0) | 2900782 / 5 x10$^6$ | 23.9 |
| | TGA | 0/100 | 11.53(4) | 2312737 / 5 x10$^6$ | 30.6 |
| 3 weeks | CMGA | 72/100 | 11.02(4) | 12192435 / 20 x10$^6$ | 155 |
| | TGA | 0/100 | 18.03(8) | 9588709 / 20 x10$^6$ | 121 |
| 4 weeks | CMGA | 97/100 | 23.91(6) | 51631978 / 100 x10$^6$ | 926 |
| | TGA | 0/100 | 27.83(18) | 56574940 / 100 x10$^6$ | 1607 |

## 5. Conclusion and Future Work

In this paper, we proposed a strategy to improve performance of GA for NSP. In the improved GA, a cost bit matrix was used for genetic operators. The selection and crossover operators were applied based on the probability only. On the contrary, the mutation operator was applied based on the probability and the value in the cost bit matrix. This usage of a matrix resulted in pruning of search space that was the main cause of reduction in execution time. In addition, the result of pruning increased the possibility to find feasible solutions, which made our algorithm find solutions satisfying all the constraints. 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. Future research aims at experiments on the real hospital data with more constraints and diversity of requirements.

## Acknowledgements

This research was supported by Hallym University Research Fund, 2012 (HRF-201208-004).

## References

[1] O. Ender, "Memetic algorithms for nurse rostering", Lecture notes in computer science, 20th International Symposium on Computer and Information Sciences, Springer-Verlag, **(2005)**, pp. 482-492.
[2] A. T. Ernst, H. Jiang, M. Krishamoorty, B. Owens and D. Sier, "An annotated bibliography of personnel scheduling and rostering", Annals of Operations Research, vol. 127, no. 1, **(2004)**, pp. 21-144.
[3] E. K. Burke, P. D. Causmaecker, S. Petrovic and G. V. Berghe, "Eta-heuristics for handling time interval coverage constraints in nurse scheduling", Applied Artificial Intelligence, vol. 20, no. 9, **(2006)**, pp. 743-766.
[4] B. Cheang, H. Li, A. Lim and B. Rodrigues, "Nurse rostering problems-A bibliographic survey", European Journal of Operational Research, vol. 151, no. 3, **(2003)**, pp. 447-460.
[5] J. F. Bard and H. W. Purnomo, "Cyclic preference scheduling of nurses using a lagrangianbased heuristic", Journal of Scheduling, vol. 10, no. 1, **(2007)**, pp. 5-23.
[6] T. Osogami and H. Imai, "Classification of Various Neighborhood Operations for the Nurse Scheduling Problem", ISAAC '00: Proceedings of the 11th International Conference on Algorithms and Computation, Springer-Verlag, **(2000)** December, pp. 72-83.
[7] H. E. Miller, W. P. Pierskalla and G. J. Rath, "Nurse Scheduling using Mathematical Programming", Operations Research, vol. 24, no. 5, **(1976)**, pp. 857-870.
[8] D. M. Warner and J. Prawda, "A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital", Management Science, vol. 19(4-Part-1), **(1972)** December, pp. 411-422.
[9] S. Abdennadher and H. Schlenker, "Nurse Scheduling using Constraint Logic Programming", Proc. AAAI '99/IAAI '99, **(1999)**, pp. 838-843.
[10] J. Li and U. Aickelin, "A Bayesian Optimization Algorithm for the Nurse Scheduling Problem", Evolutionary Computation, 2003. In: CEC '03. The 2003 Congress on 3, **(2003)**.
[11] A. Jan, M. Yamamoto and A. Ohuchi, "Evolutionary Algorithms for Nurse Scheduling Problem", Evolutionary Computation, 2000. Proc. The 2000 Congress, **(2000)**, pp. 196-203.
[12] U. Aickelin and K. A. Dowsland, "An indirect Genetic Algorithm for a Nurse-Scheduling Problem", Computers & Operations Research, vol. 31, no. 5, **(2004)** April, pp. 761-778.
[13] S. Kundu, M. Mahato, B. Mahanty and S. Acharyya, "Comparative Performance of Simulated Annealing and Genetic Algorithm in Solving Nurse Scheduling Problem", Proc. Int'l Multi Conference of Engineers and Computer Scientists 2008 1, **(2008)** January, pp. 1-5.
[14] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, **(1989)**.
[15] U. Aickelin and K. A. Dowsland, "Exploiting Problem structure in a genetic algorithm approach to a nurse rostering problem", Journal of Scheduling, vol. 3, **(2000)**, pp. 139-153.
[16] A. Brezulianu, F. Monica and F. Lucian, "A Genetic Algorithm Approach for a Constrained Employee Scheduling Problemm as Applied to Employees at Mall Type Shops", IJAST, vol. 14, **(2010)**, pp. 1-14.
[17] H. Heidari and A. Chalechale, "Scheduling in Multiprocessor System Using Genetic Algorithm", IJAST, vol. 43, **(2012)**, pp. 81-94.
[18] C. H. Papadimitriou, "Computational Complexity", Addison Wesley, **(1993)**.

## Authors

**Sun-Jeong Kim** received an MS degree in Computer Science at the Korea University in 1998, and a PhD degree from the Korea University in 2002. Since 2005 she has been working as an associate professor in Ubiquitous Computing at the Hallym University. Her research includes Scientific Visualization, Virtual Reality, and Bioinformatics.

 

**Young-Woong Ko** received the B.S., M.S. and Ph.D. degrees in Department of Computer Engineering from Korea University, Korea, in 1997, 1999 and 2003 respectively. Since 2003, he has been with the Department of Computer Engineering, Faculty of Engineering, Hallym University, where he is currently a professor. His research interests are in the areas of storage system, operating systems, and embedded systems

**Saangyong Uhmn**, he received an MS degree in Computer Engineering at Hallym University in 1997 and worked as a lecturer at Hallym University. His research interests include distributed computing, algorithms and bioinformatics.

**Jin Kim** received an MS degree in computer science from the college of Engineering at the Michigan State University in 1990, and in 1996 a PhD degree from the Michigan State University. Since then he has been working as a professor on computer engineering at the Hallym University. His research includes Bioinformatics and data mining.