Optimization of Nurse Scheduling Problem using Genetic Algorithm: A Case Study

Andriansyah¹,², N Alfadilla¹,PD Sentia¹,², D Asmadi¹
¹Industrial Engineering Department, Faculty of Engineering, University of Syiah Kuala, Darussalam, Banda Aceh 23114, Indonesia
²Laboratory of Industrial Computation and Optimization, University of Syiah Kuala, Darussalam, Banda Aceh 23114, Indonesia

*Corresponding author: nurulalfadillaa@gmail.com

Abstract. Nurse scheduling problem is an essential problem in hospital, especially in emergency department. Emergency department is very busy unit in hospital because it receives patients in emergency situation, therefore nurses in this department are obliged to be ready for 24 hours. Working hours allocation of nurses in this department is not equally, so nurse's preferences cannot be fulfilled yet. In this study, genetic algorithms are used to minimize unfulfilled preferences of nurse that called violations. Genetic algorithm represents solution candidate using random chromosomes. The selection used is elitist, the crossover used is one point cut crossover, and mutation used is reciprocal exchange mutation. Based on experiment, the optimal parameters are obtained, size of population is 80, mutation rate is 0.01, and iteration is 10000. Result is schedule of nurses for 30 days in emergency department that generated 74 violations.

1. Introduction

Scheduling the right nursing staff will improve the performance and quality of the nursing unit. Proper scheduling will help recruitment, nurse preferences, and maintain an overtime budget for a logical nurse. The most important thing in scheduling is increasing the nurse's spirit and ensuring patient safety [1]. Scheduling of nurse or commonly referred to as Nurse Scheduling Problems (NSP) are challenging because the number of nurses is relatively limited compared to the number of patients in the hospital, so the scheduling of nurses is required to get a schedule with a fair workload for each nurse and fulfill existing scheduling restrictions. Fulfillment of all scheduling restrictions is very often hampered when a limit is met, but other limitations are violated [2]. NSP is an important part of resource management in the emergency room. Good scheduling can contribute to increase the nurse welfare and distribute workload evenly [3].

NSP can be solved by using many method, one of them is using genetic algorithm (GA). Based on previous research, GA is an effective and optimal method in solving nurse scheduling problems. The results obtained from this study is an optimal schedule that is the schedule does not violate the rules that have been determined and the achievement of the objectives of the research conducted [2,4-6]. Not only used to solve nurse scheduling problems, GA is also used to solve other scheduling problems. Scheduling using GA produces a more optimal schedule compared to previous schedule [7-10].

The case study in this paper is a regional hospital located in Banda Aceh. Currently the nurse scheduling for 54 nurses in hospital is still done manually. The number of working hours for each nurse
is not evenly distributed because the working hours depend on the number of working hours the group has determined. Based on these problems, it is necessary to do research about scheduling at hospital with a genetic algorithm (GA) to provide optimal scheduling solutions.

2. Research Methodology
In the research conducted there are several steps in the steps that will be explained below.
1. Design of GA algorithm: GA algorithm is designed using MATLAB 2013a programming language by initializing the number of nurse data and algorithm GA parameters such as population size, crossover rate (Cr), mutation rate (Mr), and the number of iterations.
2. Verification of the GA Algorithm: GA algorithm is verified if the solution obtained changes when the parameter combination also changes. If the solution obtained does not change and the combination of parameters changes, it is necessary to revise the GA design that has been made.
3. Numerical example: A numerical example is done manually on the GA algorithm using hypothetical data to see GA steps.
4. Experiment and Computing Results: The experimental results are carried out by running the program 10 times and calculating the standard deviation to see the performance of the GA algorithm and see the computation time.
5. Sensitivity Analysis: The sensitivity analysis test is carried out to see how much influence the GA parameter has on the solution obtained.

3. Numerical Examples and Computational Analysis

3.1. Design of GA
Steps for GA design are following:
1. Perform data input: number of nurse nurses, shifts, days and names of nurses.
2. Determine the parameters used: population size, Cr, Mr, and the number of iterations.
3. Initialize the population.
4. Evaluate the penalty value. Penalties are violations committed.
5. Conduct a selection process to find candidates for parents. The selection process is carried out using the elitist method.
6. Perform crossover. Crossover is done using the one cut point crossover method.
7. Perform a mutation process. Mutations are done using the reciprocal exchange mutation method.
8. Conduct evaluation. Evaluate the penalty values of chromosomes resulting from crossover and mutation processes.
9. Make a selection. The selection process is carried out using the elitist method.
10. Perform population changes.
11. Termination of Iteration.
12. Get a solution.

3.2. GA Verification
GA verification is done to see the suitability of the program model with the desired model. If the program is run by changing the parameter combination and producing different outputs, it shows that the program has been verified. The verification phase is done by changing the parameter combination. Table 1 shows the parameter scenario. From the table, it can be seen that there is a change from the smallest total violation if the different parameter combinations. So that it can be said that the program has been verified.
Table 1. Scenario parameter

| No | Population Size | Mr   | Iteration | Computation Time(s) | Number of Violation |
|----|-----------------|------|-----------|---------------------|---------------------|
| 1  | 5               | 0.01 | 50        | 4,405               | 412                 |
|    | 100             |      |           | 8,686               | 302                 |
|    | 500             |      |           | 33,590              | 231                 |
|    | 1,000           |      |           | 65,638              | 131                 |
| 2  | 10              | 0.05 | 50        | 5,093               | 247                 |
|    | 100             |      |           | 8,660               | 202                 |
|    | 500             |      |           | 35,416              | 171                 |
|    | 1,000           |      |           | 68,447              | 176                 |
| 3  | 5               | 1    | 50        | 11,867              | 231                 |
|    | 100             |      |           | 22,824              | 224                 |
|    | 500             |      |           | 110,245             | 213                 |
|    | 1,000           |      |           | 223,100             | 215                 |
| 4  | 10              | 0.01 | 50        | 5,273               | 395                 |
|    | 100             |      |           | 8,321               | 290                 |
|    | 500             |      |           | 33,754              | 159                 |
|    | 1,000           |      |           | 65,193              | 158                 |
| 5  | 80              | 0.01 | 50        | 9,376               | 398                 |
|    | 100             |      |           | 12,020              | 286                 |
|    | 500             |      |           | 38,074              | 152                 |
|    | 1,000           |      |           | 70,164              | 134                 |
| 6  | 200             | 0.01 | 50        | 15,260              | 405                 |
|    | 100             |      |           | 18,968              | 300                 |
|    | 500             |      |           | 45,747              | 166                 |
|    | 1,000           |      |           | 79,360              | 142                 |

3.3. Numerical Examples

Numerical examples are done by manual calculation using hypothetical data. The population size used is 5 individuals with crossover rate (Cr) 1 and mutation rate (Mr) 0.01. Iteration is done 1 time. The following are the steps taken.

3.3.1 Population Initialization

Chromosome representation is depicted in table 2.

Table 2. Chromosome representation

| 1st day | ... | 30th day |
|---------|-----|---------|
| Shift 1 | Shift 2 | Shift 3 |
| 1       | 2    | 7       |
| 3       | 4    | 1       |
| 2       | 7    | 8       |

3.3.2 Calculation of penalty and parent selection

Penalty is the value of a violation that is not in accordance with the rules. 10 rules or nurse scheduling criteria can be seen in table 3.

The violation value is calculated based on the violations that occur on each chromosome. Following are the results of the recapitulation of the penalty results from the 5 initial population chromosomes shown in table 4.
Table 3. Nurse scheduling criteria

| Violations | Remarks |
|------------|---------|
| P1         | One nurse in one shift and one working day. |
| P2         | Each nurse was not assigned more than two consecutive morning shifts. |
| P3         | Each nurse is not assigned more than two consecutive afternoon shifts. |
| P4         | Each nurse is not assigned more than two consecutive night shifts. |
| P5         | Every nurse is not closed for more than two consecutive days. |
| P6         | Every nurse on the night shift is in accordance with a predetermined portion. |
| P7         | After having two consecutive night shifts, each nurse must be closed. Nurses are not assigned on the night shift on day i then continue the morning shift or afternoon shift the following day. |
| P8         | Each nurse gets the same number of shifts. |
| P9         | Every nurse gets a minimum of one time off on weekends in one month. |
| P10        | |

Table 4. Violation each chromosome

| Individu | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | Number of Violation |
|----------|----|----|----|----|----|----|----|----|----|-----|---------------------|
| P1       | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 2  | 1  |     | 5                   |
| P2       | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 2  | 1  |     | 5                   |
| P3       | 1  | 0  | 0  | 0  | 5  | 0  | 2  | 4  | 3  |     | 15                  |
| P4       | 34 | 0  | 4  | 0  | 9  | 0  | 22 | 8  | 0  |     | 77                  |
| P5       | 34 | 0  | 0  | 7  | 0  | 8  | 24 | 6  | 0  |     | 79                  |

From five individuals, it can be seen that individu 1 and individu 2 are the best chromosomes because they have the smallest violations. Whereas individu 4 and individu 5 are chromosomes that have the most violations. Then chromosome 1 and chromosome 2 were chosen to be parent.

3.3.3 Crossover(C) and Mutation(M)

The crossing process is done by selecting two individuals to cross. The method used is a cut point crossover. The cut point of the crossover cut point is point 18. The mutation process is done by the reciprocal exchange mutation method, which is by selecting two positions in a particular gene randomly and then exchanging it. In this mutation, chromosome 1 is chosen as the parent, with the exchange point that is point 3 and point 15. The number of chromosomes that can make a mutation is 1 because the number of population X Mr = 5 X 0.01 = 0.05~1.

3.3.4 Penalty

Penalty calculation evaluation is done to assess the chromosomes resulting from the crossover process and the mutation process. Number violations of each chromosome show in table 5.
Table 5. Violation each chromosome after crossover and mutation

| Individu | Violations | Number of Violation |
|----------|------------|---------------------|
|          | P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 |                     |
| P1       | 1 0 0 0 0 0 0 1 2 1 5            |                     |
| P2       | 1 0 0 0 0 0 0 0 1 2 1 5            |                     |
| P3       | 1 0 0 0 0 0 0 5 0 2 4 3 15        |                     |
| P4       | 34 0 4 0 9 0 0 22 8 0 77          |                     |
| C1       | 1 1 0 1 0 6 2 2 6 1 20            |                     |
| C2       | 1 1 0 1 0 6 2 2 6 1 20            |                     |
| M1       | 2 1 0 1 0 6 2 2 6 1 21            |                     |

3.3.5 Selection
The method used is elitist. The elitist selection process is carried out by filtering all individu from the GA process to form a new generation. Individu chosen are individu with the least number of violations, shown in Table 6.

Table 6. Selection

| Individu | Number of violation |
|----------|---------------------|
| P1       | 5                   |
| P2       | 5                   |
| P3       | 15                  |
| C1       | 20                  |
| C2       | 20                  |

Table 7. GA experiment

| Number of test | Computation time(seconds) | Violations |
|----------------|---------------------------|------------|
| 1              | 640,160                   | 74         |
| 2              | 629,515                   | 81         |
| 3              | 613,176                   | 88         |
| 4              | 619,344                   | 90         |
| 5              | 617,163                   | 74         |
| 6              | 641,348                   | 102        |
| 7              | 642,964                   | 99         |
| 8              | 635,247                   | 101        |
| 9              | 621,227                   | 99         |
| 10             | 634,929                   | 102        |
| Rata-rata      | 629,507                   | 91         |
| Standar Deviasi| 10,995                    | 11,343     |

3.3.6 Computational Results
From the experimental results, it is found that the population size is 80, \( Mr = 0.01 \) and the iteration of 10,000 is the best solution because it has the smallest total violation of 74 violations with a computing time of 617.163 seconds. The average violation from 5 tests was 81.4 violations with an average computation time of 623.872 seconds.
GA was tested 10 times to see the consistency of the performance of GA with a population size of 80, $Mr = 0.01$ and iteration of 10,000. These results will indicate whether there is a significant difference or not. Table 7 will show the test results in 10 times.

4. Conclusions
From the trials that have been conducted using several different parameter values, nurse scheduling using optimal GA with a population size of 80, $Mr = 0.01$, and a maximum of 10,000 iterations with a total violation of 74 violations. Of the 74 violations obtained, the most violations occurred in the 8th rule, namely 43 violations. Rule P8 is that the Nurse is not assigned on the night shift on that day then continued the morning shift or afternoon shift the following day.

References
[1] Hallah, R. M. and Alkhbazz, A. (2013) ‘Scheduling of Nurses: A Case Study of a Kuwaiti Health Care Unit’, Operations Research for Health Care. Elsevier Ltd, 2(1–2), pp. 1–19.
[2] Ilmi, R. R., Mahmudy, W. F. and Ratnawati, D. E. (2015) ‘Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika’, (13). [In Indonesian]
[3] Wong, T. C., Xu, M. and Chin, K. S. (2014) ‘A Two-Stage Heuristic Approach for Nurse Scheduling Problem: A Case Study in an Emergency Department’, 51, pp. 99–110.
[4] Aickelin, U. and Dowsland, K. A. (2004) ‘An Indirect Genetic Algorithm for a Nurse-Scheduling Problem’, Computers and Operations Research, 31(5), pp. 761–778.
[5] Leksakul, K. and Phetsawat, S. (2014) ‘Nurse Scheduling Using Genetic Algorithm’, Mathematical Problems in Engineering.
[6] Moz, M. and Vaz Pato, M. (2007) ‘A Genetic Algorithm Approach to A Nurse Rerostering Problem’, Computers and Operations Research, 34(3), pp. 667–691.
[7] Afshar-Nadjafi, B., Rahimi, A. and Karimi, H. (2013) ‘A Genetic Algorithm for Mode Identity and The Resource Constrained Project Scheduling Problem’, Scientia Iranica. Elsevier B.V., 20(3), pp. 824–831.
[8] Damayanti, C. P. et al. (2017) ‘Implementasi Algoritma Genetika Untuk Penjadwalan Customer Service (Studi Kasus: Biro Perjalanan Kangaroo)’, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1(6), pp. 456–465.[In Indonesian]
[9] Kadri, R. L. and Docter, F. F. (2018) ‘An Efficient Genetic Algorithm to Solve The Resource-Constrained Project Scheduling Problem with Transfer Times: The Single Mode Case’. Elsevier B.V., 265, pp. 454–462.
[10] Kaiafa, S. and Chassiakos, A. P. (2015) ‘A Genetic Algorithm for Optimal Resource-driven Project Scheduling’, Procedia Engineering. Elsevier B.V., 123, pp. 260–267.