Comparative Performance of Simulated Annealing and Genetic Algorithm in Solving Nurse Scheduling Problem

S. Kundu, M. Mahato, B. Mahanty and S. Acharyya

***Abstract*— Nurse Scheduling Problems (NSP) represent a subclass of scheduling problems that are hard to solve. The goal is to find high quality shift and resource assignments, in accordance with the labor contract rules, satisfying the requirements of employees as well as the employers in health-care institutions. The Nurse Scheduling Problems (NSP) can be viewed as Constraint Satisfaction Problem (CSP) where the constraints are classified as hard and soft constraints. In this paper, a real case of a cyclic nurse Scheduling problem is introduced. This means that the generated roster can be repeated indefinitely if no further constraint is introduced. We use two different methods, namely, Simulated Annealing and Genetic Algorithm to solve this problem and compared their performances at different difficulty levels.**

***Index Terms*— constraints, genetic algorithm, nurse scheduling, simulated annealing.**

I. Introduction

In organizations that operate continuously, daily work is divided into shifts. In such a context, the Scheduling problem consists in assigning a schedule to each worker, which involves building a timetable for a specified period. The timetable should comply with staffing requirements, the rules laid down by the administration and the labour contract clauses.

A *nurse roster* is a timetable consisting of shift assignments and rest days of nurses working at a hospital. In nurse scheduling, the ultimate aim is to create high quality timetables, taking well-being of nurses *[20]* as a basis without discarding the concerns of employers.

Work schedules directly affect the employees' pay, quality of life, and structure of work, family, and leisure activities. Effective scheduling of employees can reduce both, the size and the cost of the workforce. The objective is to satisfy the daily labor demands with the minimum size or minimum cost of the workforce. Typically, personnel scheduling problems are *highly constrained* and NP-hard. In this paper, two different procedures are presented for solving the cyclic *nurse-scheduling problem (NSP)*, which involves the construction of duty rosters for nursing staff over a pre-defined period.

Scheduling nurses to staff shifts involves considerable time and resources, and it is often difficult to create schedules that satisfy all the requirements.

The nurse scheduling is achieved based on requests from all nurses. Schedulers, who typically are head or chief nurses in the units, must assign nurses to each shift according to numbers and skill levels required while at the same time balancing the workload among the nurses involved and considering staff nurses' preferences such as providing requested days-off *[5]*.

The NSP under study is incorporated with three main components, i.e.

- Each nurse needs to express her *preferences* as the aversion to work on a particular day and shift.
- The *minimal coverage constraints* embody the minimal required nurses per shift and per day, and are inherent to any shift scheduling problem.
- The *case-specific constraints* are not inherent to any NSP instance but are rather *case-specific*, i.e. determined by personal time requirements, specific workplace conditions, national legislation, etc.

Hence, the objective of the NSP is to satisfy nurses' requests as much as possible without discarding the concerns of employers.

II. Problem Description

We are interested in those classes of NSP, which are abstracted from real-life problems. There are two types of nurse rosters, a two-shift system and a three-shift system. In this paper, we focus on the 3-shift system *[9]*. Each day consists of three shifts: a *morning-shift or M* (8 a.m.–3 p.m.), an *evening-shift* or *E* (1:30 p.m.–8:30 p.m.), and a *night-shift* or *N* (8:00 p.m.–8 a.m.). Nurses have to be assigned to these shifts or give off-days. On any day only one shift can be assigned to any nurse. The scheduling period is usually one or two week or one month. These schedules have to satisfy working contracts and meet the demand for a given number of nurses on each shift, and to be accepted by the staff concerned. The latter objective is achieved by meeting as many of the nurses' requests as possible. Few things are taken into account for generating a roster. These include:

– The number of nurses assigned to each working shift must be within the range of a certain maximum value and a certain

S. Kundu, West Bengal University of Technology, Calcutta, India, e-mail: sudip.wbut@ gmail.com.
M. Mahato, West Bengal University of Technology, Calcutta, India e-mail: mihir.mahato@gmail.com.
B. Mahanty, West Bengal University of Technology, Calcutta, India e-mail: mahantybiswajit@gmail.com
S. Acharyya, West Bengal University of Technology, Calcutta, India e-mail: srikalpa8@ yahoo.co.in

minimum value.

– The number of shifts assigned to each nurse must be within the limits of legal regulation.

– *Prohibited working patterns* must be prevented. A "working pattern" is a sequence of working shifts over several days.

– Requests from nurses should be satisfied as much as possible.

We will evaluate our method under different conditions. For example we consider a scheduling term of 2 weeks, and there are 15 nurses to be scheduled. Accordingly, the problem is to assign shift numbers (morning/evening/night/day off), i.e., values, to the *number_of_nurses*number_of_days* (15*14 =210) variables. Different hard and soft constraints must be taken into account for generating a roster. We've considered the following types of constraints:

A. Hard constraints:

All the hard constraints must be satisfied to obtain a feasible solution.

1) Type1-constraints:

– Constraints on the number of nurses for each working shift per day: For each shift, the number of nurses has to be within the range of maximum and minimum values (morning: 4-6, evening: 3-5, night: 3-5).

2) Type2-constraints:

The roster must be cyclic and avoid the following prohibited working patterns–

"morning-shift after night-shift",
"evening-shift after night-shift",
"morning -shift after evening-shift", and
"3 consecutive night-shifts".

B. Soft constraints:

The feasibility of a solution is determined by the satisfaction of hard constraints. But the quality of a feasible solution depends on the degree to which the soft constraints are satisfied.

Type3-constraints:

For each nurse in this example of two weeks rostering problem, the typical type-3 constraints are

Total number of off-days = 4;
Total number of night-shifts = 3;
Total number of morning-shifts and evening-shifts for each nurse is between 3 and 4.

This will vary depending on the scheduling period and number of nurses.

As we mentioned earlier, the roster is generated based on the satisfaction of a number of constraints. All the hard constraints must be satisfied and soft constraints should be satisfied.

C. Cost function

We have designed a cost function depending on the different types of constraints mentioned earlier. Total cost will be calculated by combining following three types of cost:

cost1 is the penalty cost for violating most important type of constraint i.e. *type1 constraint*. If the shift assignments on any day violate any of the *type1 constraints*, then for each violation the value of *cost1* will be incremented by 1.

cost2 is the penalty cost for violating *type2 constraints*. If shift sequence for any nurse violates any of the *type2 constraints*, then for each violation *cost2* will be incremented by 1.

cost3 is the penalty cost for violating *type3 constraints*. For example, if for any nurse the total number of "day off" or any other shift does not fall within the mentioned limit, then for each violation the value of cost3 is incremented by 1.

We have assigned three different weights to three different types of constraints. Let w_1, w_2, w_3 are the weights assigned to type1, type2, type3 constraints respectively. Then our objective is to minimize the

total cost , $C = w_1*cost1 + w_2*cost2 + w_3*cost3$

We'll consider a solution feasible only if all the hard constraints are satisfied (i.e. cost1 and cost2 are minimized to 0). Solution strategy will also try to satisfy as many soft-constraints as possible.

III. Solution Methods

A satisfactory analytical solution procedure for the problem, even in its idealized form, has not yet been found. We are thus forced to use heuristic methods *[17]*. There are several approaches *[1]* to the nurse scheduling based on the framework of *Constraint Satisfaction Problem* (CSP) *[13]*. We have tried to solve randomly generated problem instances using the two randomized CSP methods. The solutions obtained by us are not always complete, in the sense that all the soft constraints are not always fully satisfied.

Number of variables in our problem, N = *no_of_nurses*number_of_days*. The main objective is to assign shift values to all these variables such that the total cost becomes minimum. The implementations given below consider several problems consisting of i) the hard constraints and ii) soft-constraints mentioned earlier.

A. SIMULATED ANNEALING

The initial trial solution S in Procedure SA [19] is obtained by randomly assigning each nurse to one of the three shifts or day-off on each day. As a result, all the variables in our problem (*x[nurse_no] [day_no]*) are assigned random shift values. So, now, a subset of the constraints are unsatisfied. The initial cost corresponding to S is calculated using the cost function mentioned earlier. Now this cost is taken as current cost *c*. Then we randomly choose a variable, i.e., a combination of nurse_no and day_no, and change its shift value randomly. In this way we move to a neighbouring solution S' of S. Then we calculate the cost corresponding to S', taken as new cost *c'*. Now we have to take a decision, whether we will accept this movement permanently or not. This is discussed in details in **Procedure SA** given below. This process is repeated and the algorithm outputs the feasible solution of lowest cost.

Procedure SA makes use of a number of parameters. The values of these parameters must be finely tuned; otherwise, inferior results are obtained frequently. The most important issue is the initialization of the temperature and the

determination of the rate at which it should decrease [14]. In our problem a high temperature such as 2000 is initially chosen. Whenever, *changes/trials* > *tcent*, the temperature is reduced fast using a parameter called *fastfactor* having a typical value of 0.5; if *changes/trials* < *tcent*, the temperature is reduced slowly. For the reducing factor, known as *tempfactor* we choose a value of 0.98. The variable of interest is *c**, which stores the cost of the trial solution of minimum cost among all feasible solutions found so far, up to the current instant. *c** is initialized to the initial cost *c*.

```
Procedure SA     /* detailed procedure*/
{
  input a trial solution S; c = cost(S);
  c* = c;  freezecount = 0;  initialize temp;
  initialize frzlim, sizefactor, fastfactor, tempfactor,
  minpercent, tcent;
  while ( freezecount < frzlim ) {
    changes = trials = 0;
    while ( trials < sizefactor * N ) {
            /* N is determined by the size of the problem */
      trials = trials + 1; generate a random neighbour S' of S;
      c' = cost(S'); Δ = c'- c;
      if (S' is feasible and cost(S') < c* )
            { S* = S'; c* = cost(S'); }
                /* save best feasible solution found so far */
      if (Δ ≤ 0) { changes = changes + 1;  c = c'; S = S'; }
                          /* downhill move */
      else {                    /* possible uphill move */
        choose a random number r in [0,1];
        if ( r <= exp(-Δ/temp) )
              { changes = changes+1; c = c'; S = S'; }
      }
    }
    if  (changes/trials ≥ tcent ) temp = fastfactor * temp;
                /* reduce temperature quickly */
    else temp = tempfactor * temp;
        /* reduce temperature slowly */
    if ( changes/trials < minpercent )
        freezecount = freezecount+1;
    else freezecount = 0;
  }
  output the final solution S*; /* S* is a feasible solution of
minimum cost */
}
```

B. GENETIC ALGORITHM

Canonical GAs [15] were not intended for function optimization [3], but slightly modified versions proved to be successful [2]. In our experiments, the initial population consisted of WP random trial solutions, where WP was chosen to be around 10. Each trial solution S is defined as a chromosome string of length N (number of variables, defined earlier) where the elements were shift values, that is, integers between 1 and 4. Each chromosome represents one complete solution. First element of any chromosome represents the shift assigned to $nurse_1$ on day_1; second element represents the shift assigned to $nurse_1$ on day_2, and so on. The fitness function was essentially identical to the cost function used in Simulated Annealing, and nGen had a typical value of 10,000. Larger values of nGen gave solutions of better quality at the expense of higher runtimes. Cross-over and Mutation operators are used to generate new chromosomes[2]. Among the chromosomes generated, the one that had the highest rating among all feasible solutions is finally outputted.

Procedure GA

1. Start with a randomly generated population of 'n', 'l' bit chromosomes.
   Number of bits in each chromosome= number of nurse * number of days;
2. Calculate the fitness f(y) of each chromosome y in the population.
3. Move the best chromosome to the new population.
4. Repeat the following steps until (n-1) more offsprings are created.
   (a) Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done "with replacement", meaning that the same chromosome can be selected more than once to become a parent.
   (b) With probability P_c (crossover probability), cross over the pair between two randomly chosen points (chosen with uniform probability) to form two offsprings. If no cross over takes place, form two offspring that are exact copies of their respective parents. (Here the cross over rate is defined to be the probability that two parents will cross over between two break points).
   (c) Mutate the two offspring at a randomly chosen bit with probability P_n (the mutation probability or mutation rate) and place the resulting chromosomes in the new population.

If n is even, one new population member can be discarded at random.

5. Replace the current population with new population.
6. Go to step -2 until a desirable solution is found or maximum number of generations is completed.

IV. EXPERIMENTAL OBSERVATIONS

Our experimental result will show the best roster in tabular form. A sample roster is given below. This roster is for 15 nurses and for one week.

- Each row in the table 1 shows the shift sequence of a particular nurse.
- Each column in the table 1 shows shift assignment of each nurse on that particular day.
- The marks 'M', 'E', 'N' and '*****' represent a morning shift, an evening shift, a night shift, and a day off respectively.
- The number of nurses assigned to each shift is given at the bottom of the table 1.
- The number of the shifts that each nurse works in the scheduling period appears at the right of the table 1.
- The total cost will show the cost penalty for

unsatisfied constraints. If all constraints are satisfied, then this cost is zero.

TABLE 1: A SAMPLE ROSTER

	M	T	W	T	F	S	S	MOR	EVE	NGT	OFF
n01:	N	N	*	M	M	E	E	2	2	2	1
n02:	E	E	N	N	*	M	M	2	2	2	1
n03:	M	M	E	E	N	N	*	2	2	2	1
n04:	*	M	M	E	E	N	N	2	2	2	1
n05:	M	M	E	E	N	N	*	2	2	2	1
n06:	E	N	N	*	M	M	E	2	2	2	1
n07:	N	*	M	M	E	E	N	2	2	2	1
n08:	E	N	N	*	M	M	E	2	2	2	1
n09:	N	*	M	M	E	E	N	2	2	2	1
n10:	M	E	E	N	N	*	M	2	2	2	1
n11:	M	M	E	E	N	N	*	2	2	2	1
n12:	E	E	N	N	*	M	M	2	2	2	1
n13:	N	N	*	M	M	E	E	2	2	2	1
n14:	N	*	M	M	E	E	N	2	2	2	1
n15:	E	E	N	N	*	M	M	2	2	2	1
MOR:	4	4	4	5	4	5	4				
EVE:	5	4	4	4	4	5	4				
NGT:	5	4	5	4	4	4	4				
OFF:	1	3	2	2	3	1	3	Total Cost= 0			

We now summarize our experimental observations on the Nurse Scheduling Problem (in Table 2). The two CSP methods were programmed in C and run on the WINDOWS-based Pentium 4 machine. The random number generators random(int) and rand() were used for generating random numbers. Identical problem instances were run for both the methods. We wanted to create random instances of the NSP that were realistic and indicative of real life situations. For this purpose, we collected data about nurse rosters from well-known Peerless Hospital in Kolkata. Duration of roster varies in different hospitals, so we decided to keep the duration between 7 to 30 days. Hard constraints are same for all the problems. Here Soft constraints depend on the roster period. 100 problems were generated for each set. We determined the number of problems solved in a set and the average runtime in seconds. We also computed, for each feasible solution, the number of unsatisfied soft-constraints. The averages were taken over solved instances.

The methods were compared on the basis of three criteria: i) the number of problems solved (denoted by *Solved* in the table 2) in each set of 100; ii) the runtime in second averaged over solved problems (denoted by *Time* in the table 2), iii) the average cost of the solution obtained (denoted by *Cost* in the table 2), cost being determined by the number of soft-constraints which were not satisfied.

Simulated Annealing (SA) is the better method, judged on the basis of the three criteria mentioned above. The quality of solutions improves markedly as parameters are fine tuned to their optimal values. Problems become harder to solve if the constraints are made stricter.

To solve the same problems, Genetic Algorithm (GA) was not very effective compared to SA. It took more time and the quality of the solution was not very impressive. This is because no constraints checking is done while choosing cross-over points. And often, after cross-over, the child chromosome may not be as good as their parent.

TABLE 2: THE NURSE SCHEDULING PROBLEM
PERFORMANCE OF SIMULATED ANNEALING AND GENETIC ALGORITHM

Period (days)	Probs	Method	Solved	Cost	Time (sec)
7	100	SA	88	0.27	0.77
		GA	80	26.85	2.5
14	100	SA	92	0.11	2.85
		GA	73	4.72	7.21
21	100	SA	100	2.46	3.48
		GA	86	22.16	8.26
30	100	SA	65	2.52	11.77
		GA	24	30.00	11.28

V. CONCLUSION

The Nurse Scheduling Problem is a complex scheduling problem. The runtime increases as the number of variable becomes higher. Assigning proper weight to each constraints helps to get feasible solution faster. But if we assign too much weight to the hard constraints, then the solutions of good quality are hard to find. Of course, not every randomly generated problem instance has a feasible solution. When no weight it assigned to the constraints it is quite possible that some of the problems do not have feasible solutions.

In this paper, we applied Simulated Annealing and Genetic Algorithm for solving Nurse Scheduling Problem which was modeled as weighted CSP. In most of the cases our programs were able to return a feasible solution satisfying the hard constraints. But the SA implementation proved to be more useful than GA. What is more interesting is that the resulting roster is cyclic, i.e. the same roster can be repeated after the given duration.

REFERENCES

[1] S. Abdennadher, Schlenker, "Nurse scheduling using constraint logic programming.", In Proc. of the 11th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-99). 1999 pp. 838–843.

[2] U. Aickelin, Dowsland,"Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem", Journal of Scheduling, vol. 3, 2000, pp 139-153.

[3] K. De Jong, "Genetic Algorithms are NOT Function Optimisers." In D. Whitley (ed.), Foundations of Genetic Algorithms, vol. 2, San Mateo, CA: Morgan Kaufmann, 1993, pp. 5–17.

[4] J. Frank, "Local Search for NP-Hard Problems", Ph. D. Thesis, University of California, Davis, California, 1997.

[5] G. Post, B. Veltman, "Harmonious Personnel Scheduling", Proceedings of the 5th International Conference on the Practice and Automated Timetabling (PATAT) 2004, pp. 557-559.

[6] D. E Goldberg, "Using time efficiently: Genetic-evolutionary algorithms and the continuation problem", in: Proc. Genetic and Evolutionary Computation Conf., 1999, pp. 212–219.

[7] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, 1989.

[8] H. Meyer, auf'm Hofe, "Solving Rostering Tasks as Constraint Optimization", E. K. Burke, W. Erben (Eds.): Practice and Theory of

Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, pp. 191-212.
[9] A. Ikegami, A. Niwa, M. Ohkura, “Nurse scheduling in Japan”, Commun. Oper. Res. Society of Japan, vol. 41, 1996, pp. 436–442 (in Japanese)
[10] D. S. Johnson, C. R. Aragon, L. A.McGeoch & C.Schevon,” Optimization by Simulated Annealing”: An Experimental Evaluation, Part I, Graph Partitioning, Operations Research, vol 37, 1989, pp 865-892.
[11] H. Kawanaka, K. Yamamoto, T. Yoshikawa, T. Shinogi, and S Tsuruoka, “Genetic algorithm with constraints for the nurse scheduling problem”, Proceedings of Congress on Evolutionary Computation, IEEE Press, Seoul, South Korea , vol. 2, 2001, pp. 1123–1130.
[12] S. Kirkpatrick, C. D. Gelatt & M.P.Vecchi, “Optimization by Simulated Annealing”, Science, vol 220, 1983, pp 671-680.
[13] V. Kumar, “Algorithms for Constraint Satisfaction Problems: A Survey”, A I Magazine, vol. 13, no 1, 1992, pp 32-44.
[14] C. Liu, R.Kao & A.Wang, “Solving Location-allocation Problems with Rectilinear Distances by Simulated Annealing”, J Opl Res Soc, vol .45, no 11, 1994, pp 1304-1315.
[15] M.Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.
[16] Parkes & Walser, "Tuning Local Search for Satisfiability Testing”, Proc AAAI-96, vol. 1, 1996, pp 356-362.
[17] C. R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, Orient Longman, 1993.
[18] Saxena & Anupam, ”Synthesis of Compliant Mechanisms for Path Generation using Genetic Algorithm.”, Journal of Mechanical Design 127(4). 2005.
[19] W. M. Spears, Simulated Annealing for Hard Satisfiability Problems, Technical Report, Naval Research Academy, Wasington D C, 1993.
[20] H. W. Warner, “Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Approach.”, Operations Research, vol. 24, 1976, pp. 842-856.