Grenze International Journal of Engineering and Technology, July Issue

Performance of Simulated Annealing Variants to Solve Nurse Scheduling Problem Incorporating Preference List and Stable Marriage

Sudip Kundu[1], Suprabhat Maity[2] and Sriyankar Acharyya[3]
[1-2]B.P. Poddar Institute of Management and Technology, Kolkata, India
Email: sudip.wbut@gmail.com, suprabhatmaity@gmail.com
[3]Moulana Abul Kalam Azad University of Technology, Kolkata, India
srikalpa8@yahoo.co.in

***Abstract*—The Nurse Scheduling Problem (NSP) is a complex scheduling problem and involves a large set of constraints. Here we have incorporated the concept of Stable Marriage Problem (SMP) which has many practical applications, for example matching resident doctors to hospitals, students to colleges, or more generally to any two-sided market. In addition to the basic constraints the nurses will have their choice of shift preferences, and hospital also has own preference list of nurses on different shifts on each day. The original SMP requires all men and women to submit a complete and strictly ordered preference list - which is generally not practical, and rightly so, few relaxations have been proposed. In our problem we have allowed incomplete lists and ties –both relaxations at the same time. This enables a nurse to request only a subset of the shifts on a day and ties among preferences are also allowed. The objective is to build a high quality nurse roster satisfying all the hard constraints and minimizing the number of blocking pairs along with satisfying other soft constraints as much as possible. The problem is NP hard and we will show that Simulated Annealing incorporated with Tabu list and greedy probability has outperformed the basic Simulated Annealing in finding out near optimal solutions for real world nurse scheduling instances.**

***Index Terms*— Nurse Scheduling, Stable Marriage Problem, Constraint Satisfaction, Meta-heuristics, Simulated Annealing, Tabu list.**

I. INTRODUCTION

The Nurse Scheduling Problem (NSP) is a highly constrained Staff Scheduling Problem. According to their computational complexity, NSP has been proved to be NP-hard[6,7,8,9,10]. The objective is to generate individual rosters for all nurses consisting of different shifts and off days over a predefined period. Nurses are assigned duties into different shifts. On each different day, a nurse is given either morning shift or evening shift or night shift or given day-off. The schedule should mainly comply with nursing requirements and the rules followed by the authority and also the labor contract clauses. More precisely, nurses are given some specific shifts on specific days taking into account different issues like total work load, minimum demand coverage, day-off / on requests, consecutive shift assignments etc. [11]. The well-being of a nurse depends on

Grenze ID: 01.GIJET.6.2.3

their salary, and work environment. So time available for family and leisure activities may also be considered.
The health-care institutions work round the clock. Preparing rosters involve lot of time and manpower, and it is often very difficult to generate roster that satisfy all the constraints which will vary on different shift[40]. Normally, the task of rostering is done by the head nurse manually which is very much time consuming. The objective is to find only a feasible solution, optimality is not the focus here[41]. Nurses are given shifts according to the requirement, and their skill levels. But it is also necessary to balance the workload to improve the work environment.
The scheduling of employees will be effective if the demand of daily service can be met with the minimum number of nurses assigned in a shift. Here the viewpoint of the hospital and the nurses are usually conflicting. Nurses priority will be their preferences, where hospitals priority will be minimum demand coverage. The balance between these two is very important.
For solving Nurse Scheduling Problem three methods are very common Mathematical Programming (MP), Heuristics and Artificial Intelligence (AI). MP based algorithm was proposed by Warner [12] where objective function maximizes nurses' preferences. Millar and Kiragu [13] used a network model for cyclic and non-cyclic nurse scheduling. Arthur and Ravindran[14] applied multi-objective programming to handle conflicting constraints. Musa and Saxena [15] proposed a single phase GP algorithm. Azaiez and AI Sharif [16] developed a GP model that considers nurses' preferences and different hospital objectives. Combining the strengths of Integer Programming (IP) and Variable Neighbourhood Search (VNS) algorithms, E. Rahimian et al. [39] proposed a novel hybrid algorithm for solving NSP.
Beside optimization-based approaches, heuristic methods [17] can also produce good results in reasonable time. Dowsland [18], Dowsland and Thompson [19] used tabu search and its variant. Aickelin and Dowsland[20] applied genetic algorithm to handle the conflict between objectives and constraints. Abdennadher and Schlenker [21] gave an interactive constraint based scheduler. Beligiannis et al [30] used a two-phase adaptive variable neighborhood approach for nurse scheduling. Ant Colony Optimization [33] and Particle swarm optimization [34] techniques have also been used for solving this problem.
There are several approaches [21] for solving NSP, as this is one type of Constraint Satisfaction Problem (CSP) [22]. We implemented Simulated Annealing (SA) and Genetic Algorithm (GA) to solve NSP [23]. There the result was clear- Simulated Annealing outperformed Genetic Algorithm. We also converted NSP to Satisfiability Problem (SAT)[31]. The constraints were converted to *Clauses*. Each type of constraint is represented by a set of clauses. GSAT and WalkSAT [24] [25] outperformed SA there. Then we applied other SAT techniques[32] like USAT, Novelty, RNovelty, Novelty+, RNovelty+ and got satisfactory result.
In this paper we have incorporated the concept of Stable Marriage Problem[1,2,3,4,5] i.e. preference list and blocking pair in Nurse Scheduling Problem.We are particularly interested in that version where Incomplete list and Ties are allowed at the same time. This enables a nurse to request only a subset of the shifts on a day and ties among preferences are also allowed. We have used Simulated Annealing along with two of its variants to solve the problem.
Here in Section-2 we describe the Nurse Scheduling Problem, mention the Constraints we considered, and derives the Objective function. Section-3 presents the implementation details of Simulated Annealing and its variant that we have used to solve this problem. Section-4 shows the experimental results in tabular form and their comparative assessment. Finally, conclusion is done in Section-5.

II. PROBLEM DESCRIPTION AND FORMULATION

We emphasize on real life situations when we consider the problem instances of NSP. Normally, nurse rosters can be of two types, one with two shifts and other with three shifts daily. We have considered the three-shift system [26]. Here each day is divided into three working slots: a *morning-shift*, an *evening-shift*, and a *night-shift*. Timings for these three shifts may vary a little bit, but the problem will be the same. Usually, the roster is made for one or two weeks, or one month duration.
The restrictions on the schedule come from working contracts and the preferences or requests submitted by a nurses on each day. Objective of First International Nurse Rostering Competition (INRC-2010) was to develop interest in nurse rostering by presenting more difficult problems with an increased number of real world constraints [37]. The schedule tries to satisfy as many of the nurses' requests as possible. The constraints are divided into two types: *hard* constraints and *soft constraints*. To get a feasible solution it is necessary to satisfy each of the hard constraints. Also the quality of the roster will depend on the level up to which soft constraints are satisfied.

We previously presented a mathematical model for NSP [1] . The Decision Variable $X_{i\,j\,k}$ =1 if on day-*j* nurse *i* is assigned shift-*k*. Otherwise it is =0. The parameter ***nPref[i][j]*** represent preference list of nurse i on day j. Similarly *hPref[j][k]* represents preference list of hospital for shift k on day j.

A. Nurses' preferences list

Each nurse need to submit her preference list for the entire roster. If we consider n number of nurses, then there will be n number of preference lists of length 4 for all the nurses on each day. The entries will be 'M' for morning shift, 'E' for evening shift, 'N' for night shift, and 'X' for a day-off. Nurses are allowed to give shift choices according to their wishes for all the days. Ideally the choices must be an ordered list of all the shift values on that day. But incomplete list is allowed. If a nurse doesn't want a particular shift on a particular day, she can definitely omit that shift value from her preference list on that day. For example, the preference list ***n[2][3]: M,X*** means that the shift choices for nurse-2 on day-3 is first Morning shift, and then OFF day. Evening shift or night shift is not acceptable in this case. Two or more shift-values tied in a list are also allowed. It is represented in parenthesis. So the preference list ***n[1][5]: M, (E, N), X*** means that the shift choices for nurse-1on day-5 is first Morning shift, then Evening or Night shift(tie exists), and then OFF day.

B. Hospital's preference list

Finally we construct hospital's preference lists. We construct hospital's preference list automatically from the nurses' preference lists. Preference list for every shift on different days are created only from the options given by the nurses in their list. This means, normally, that the hospital's preference list on a shift- s on a day d does not include a nurse-n if the shift-s doesn't appear on the preference list of nurse-n on that day-d. But the minimum demand coverage must also be maintained. On the other hand, if, on a public holiday most of the nurses request day-of, the hospital needs to prioritize the nurses requests. The sequence of nurses in each preference list is rearranged according to the choice of hospital. This may include the factors related to work-experience, technical expertise etc.

C. Blocking pair

Now we will discuss blocking pair. Suppose Nurse I on day J is assigned shift k. now ([i^{th} nurse on day J], and [shift $K^{/}$]) where k != $K^{/}$, will form a blocking pair if any of the following three conditions is satisfied:

i> nurse i is not assigned shift $k^{/}$ on day j. but $k^{/}$ appears before k in the preference list of I on that day. And another nurse is assigned shift $k^{/}$ whose rank is lower than Nurse I on the preference list of hospital for shift $k^{/}$ on day j.

ii> nurse i strictly prefers shift $k^{/}$ to shift k on day j, and shift $k^{/}$ is under- subscribed in M.

iii> $k^{/}$ appears in the preference list of nurse I for day j, but nurse I is assigned k which is not part of her preference list on that day.

D. Hard Constraints

Hard Constraints are same here as we considered previously [1].
HC1: This will ensure only one shift can be assigned to a nurse in a day. Otherwise off day is given.

HC2: The number of nurses assigned to a shift must be within a given range. For example 6-8 nurses needed in morning shift, but 5-7 nurses needed in night shift.

HC3: *I*t must avoid some prohibited working patterns like:*Three consecutive night-shifts, Morning-shift after night-shift, Evening-shift after night-shift,Morning -shift after evening-shift*

The violation of HC1, HC2, HC3 contributes to the cost component *cost_HC1, cost_HC2,* and *cost_HC3* respectively.

E. Soft Constraints

Soft constraints are the desirable constraints. We have considered the following two soft constraints:
SC1: For each nurse, number of morning, evening, night shift or day-off should be within a given range.

SC2: Total number of blocking pair will be minimized.

The violation of SC1, SC2 contributes to the cost component *cost_SC1,cost_SC2.* So, for each presence of a blocking pair cost component *cost_SC2* is incremented.

F. Objective Function

Considering different hard and soft constraints, the expression for the total cost of any random solution instance becomes:

Cost = w1**cost_HC1* + w2**cost_HC2* + w3**cost_HC3* + w4**cost_SC1*+ w5**cost_SC2*. Here w1, w2, w3, w4, w5 are the non-zero positive weights assigned to different type of constraints. The values are chosen according to the priority of the respective constraints.

In order to get any feasible solution it is required that the hard constraints must be satisfied, and then we need to minimize the cost associated with soft constraints. Obviously the first three terms will result in zero. Thus the Objective function becomes:

Minimize (w4**cost_SC1*+ w5**cost_SC2)* subject to the condition: *cost_HC1=0*, *cost_HC2=0*, and *cost_HC3=0*.

III. SOLUTION METHODS

As the deterministic methods are very slow in solving this kind of problem, we are applying the stochastic local search methods to solve the problem instances. Here we will discuss Simulated Annealing approach along with two variants and their implementation details.

A. Simulated Annealing (SA)

Simulated Annealing [23] starts with one initial solution which is generated by randomly assigning values to all variables. Since the trail solution is randomly generated, quality of that solution will be not be good at all. The initial cost is calculated taking into account the costs for violating hard constraints, soft constraints and the number of blocking pairs. Then we move to a neighbourhood solution. This is done by randomly choosing a variable and changing its value. Now the cost is calculated for the neighbourhood solution as well.

If the neighborhood solution has lower cost, it is taken immediately. Otherwise, if the neighborhood solution has higher cost then we accept this only with some probability. The process is repeated until some stopping criteria is met.

The values of different parameters need to be fine tuned to get good results. The most important is the initialization of the temperature (*T*), and the rate (*tempfactor*) at which it should decrease. Whenever *changes*/*trials* become higher than *tcent*, the temperature is halved; otherwise, the temperature is reduced slowly using tempfactor having a value greater than 95%. This process continues until we get the desired result or temperature reaches zero value or *freeze-counter* equals the *freeze limit*. In the third case, we assume that there is no chance of finding better result and the procedure stops. In other cases, we gradually move to an optimal solution.

To improve the the performance of simulated annealing we have used two variants. First, simulated annealing with tabu list, and then, simulated annealing with tabu list and greedy probability.

B. Simulated Annealing (SA) with Tabu list

Here a tabu search strategy is added to increase the performance of SA. Tabu List is implemented as a first in first out queue which is initially empty. It prevents the same variable from being selected for neighborhood movement [35] repeatedly. The most recently selected variable is inserted into the list. Next variable is selected randomly [36] from among those variables which are not present in the tabu list. This is how some variables are restricted from being selected repeatedly, and local mimima can be avoided.

C. Simulated Annealing (SA) with Tabu list and Greedy Probability

We have also used greedy probability which will impose a condition that only a percentage of inferior solutions will be allowed. A random number is generated and depending on the greedy probability value the movement will be allowed. In SA temperature plays the most important role when we consider the inferior moves. More the temperature, higher the chance of inferior moves, resulting more increase in execution time. At the same time if we start with lower initial temperature, we cannot explore larger portion of search space. This greedy probability helps us to balance these two. We can set higher initial temperature, and also can control the inferior moves using greedy probability value.

Algorithm: Template of basic simulated annealing

procedure `SimulatedAnnealing()`

Generate an initial solution S_0 and set $S \leftarrow S_0$;
Compute the initial temperature T_0 and set $T \leftarrow T_0$;
while *stopping criterion* is not reached **do**
  while *thermal equilibrium* is not reached **do**
    Obtain a neighbor solution $S' \in N(S)$ at random;
    Compute $\Delta E = f(S') - f(S)$;
    if $\Delta E < 0$ **then** $S \leftarrow S'$;
    else if $e^{-\Delta E/(K_B T)} > random[0,1)$ **then** $S \leftarrow S'$;
    end_if
  end_while
  Decrease T according with the *annealing schedule*;
end_while;
return S;
end.

IV. Experimental Results

We used simulated annealing and two of its variants for solving NSP. Programs were run on Intel 1.8 GHz PC with 4 GB RAM. Identical problem instances were taken so that comparative assessment can be done. Success rate here denotes number of problems solved (in percentages). We recorded the Cost of the best solution excluding blocking pair, Average Cost of Feasible Solutions, average number of blocking pair and average time taken by the different methods (in seconds).

The average run time was taken over the solved instances. The comparative performance of SA, and its two variants in solving different problem instances of NSP is shown in *Table-I.* M represent the duration (in days) and N is the number of nurses.

TABLE I. Comparative Performance of different methods in various instances of NSP

M	N	Implementation	Success Rate	Cost of Best Solution	Avg Cost of Feasible Solutions	Avg blocking pair	Avg Time(Sec)
7	10	SA	100	0	2.8	48	6.8
		SA Tabu	100	0	3.0	46	5.4
		SA Tabu Greedy	100	0	2.4	42	4.5
7	15	SA	96	2	4.8	58	10.5
		SA Tabu	100	2	3.0	54	9.6
		SA Tabu Greedy	100	0	3.0	54	8.5
7	20	SA	85	1	3.6	57	14.8
		SA Tabu	90	2	3.2	59	14.2
		SA Tabu Greedy	94	0	2.8	42	12.5
14	10	SA	96	1	3.0	45	18.5
		SA Tabu	95	1	2.8	52	16.5
		SA Tabu Greedy	100	0	2.8	42	15.9
14	15	SA	100	1	4.6	57	22.2

		SA Tabu	98	2	5.4	56	20.4
		SA Tabu Greedy	100	0	2.4	54	18.9
14	20	SA	95	1	3.8	58	20.5
		SA Tabu	100	1	2.8	55	20.2
		SA Tabu Greedy	100	1	2.4	56	19.5
21	10	SA	92	1	2.2	84	26.3
		SA Tabu	94	0	1.0	75	27.1
		SA Tabu Greedy	95	0	0.8	64	24.6
21	15	SA	80	2	5.0	78	25.6
		SA Tabu	75	0	2.4	72	24.8
		SA Tabu Greedy	89	0	2.0	54	22.5
Overall Performance		SA	93	1.12	3.72	60.62	18.15
		SA Tabu	94	1.00	2.95	58.62	17.27
		SA Tabu Greedy	97.25	0.12	2.32	51	15.86

From the result we can see that Simulated Annealing with tabu list and greedy probability has outperformed the basic simulated annealing and simulated annealing with tabu list. Success rate, Average cost and average time has improved in almost all the instances. Average no of blocking pair which was initially close to 900 was reduced to 51 by the SA Tabu Greedy version where the basic SA version was able to reduce the blocking pair to 60 only. Most interesting thing is that average min cost for SA Tabu greedy is close to zero which is far better than other two.

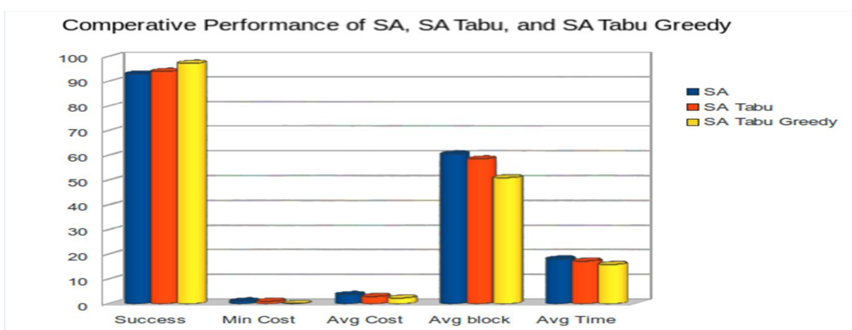
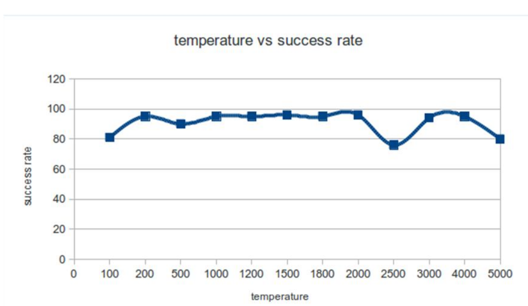
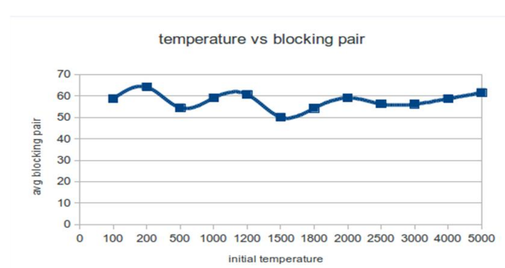


Fig 1. Overall Performance of SA, SA Tabu, and SA Tabu Greedy approach

It is known that to get near optimal result using simulated annealing the parameters must be fine-tuned to their appropriate values. So we checked our experimental observations assigning large set of values for initial temperature, tabu list length, and greedy probability.





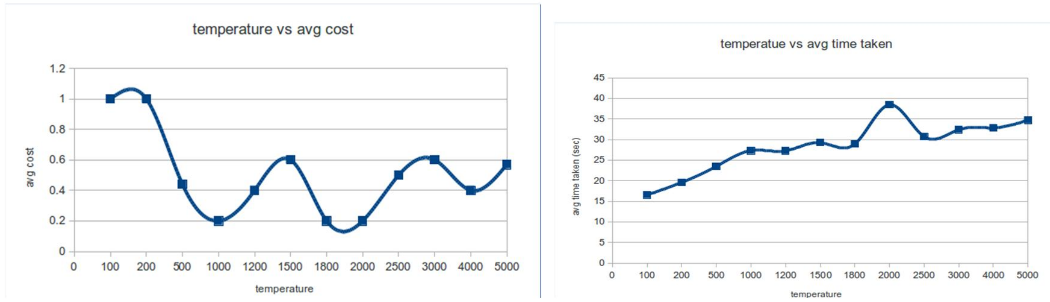


Fig 2. Effect of Initial Temperature on output parameters

We can see from Figure2, if the temperature is set very low (below 500) then we'll get a feasible solution very fast. But the quality of solution will not be that good. The reason is that the search space will be smaller. It might lead to local minima. On the other hand, if we set the temperature very high, then it will take much time to converge to a good result. In our problem we have found initial temperature close to 1800 effective. We did not found any relation between temperature and the final number of blocking pairs.

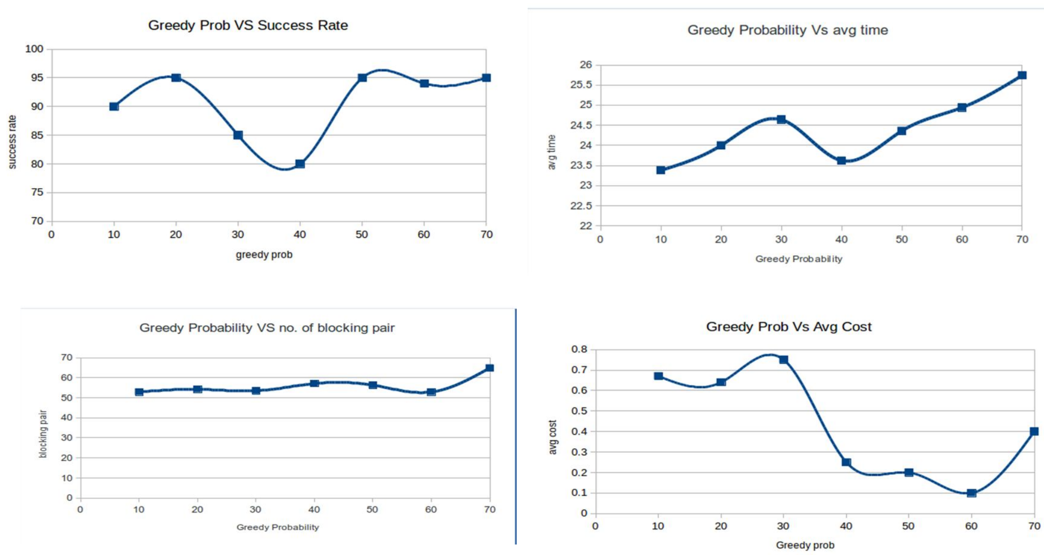


Fig 3. Effect of Greedy Probability value on output parameters

Figure3 above shows how the greedy probability affects the output parameters. We can see that if we set the greedy probability value is below 40 percent, either the success rate becomes lower or the average cost increases. We found greedy probability value near 60 % very effective in our program. If we set greedy probability value above 60, time taken and number of blocking pair will increase.

V. Conclusion

The Nurse Scheduling Problem (NSP) is modelled as a combinatorial optimization problem. In addition to the basic constraints we have incorporated the preference list and Stable Marriage Problem. This will help to satisfy the nurses' preferences while maintaining minimum demand coverage. We implemented Simulated Annealing and two of its variant to solve it. The tabu list was used to prevent same neighbourhood movement repeatedly. This also lower the chance of getting stuck to a local minima. For our problem instances, initial temperature of 1800, tabu list length of 4, and greedy probability of 0.06 proved appropriate. In most of the instances, Simulated Annealing with tabu list greedy probability outperformed other two.

REFERENCES

[1] Gale and L. S. Shapley, “College admissions and the stability of marriage,” American Mathematical Monthly, Vol.69, pp.9-15, 1962.

[2] D. Gusfield and R. W. Irving, “The Stable Marriage Problem: Structure and Algorithms,” MIT Press, Boston, MA, 1989.

[3] R. W. Irving, “Stable marriage and indifference,” Discrete Applied Mathematics, Vol.48, pp.261-272, 1994.

[4] D. Gale and M. Sotomayor, “Some remarks on the stable matching problem,” Discrete Applied Mathematics, Vol.11, pp.223-232, 1985.

[5] D. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theor. Comput. Sci.*, 276(1-2):261–279, 2002.

[6] AwadallahMA, KhaderAT,Al-BetarMA,BolajiAL.Hybridharmonysearchfor nurse rosteringproblems. In: Proceedings of the 2013 IEEE symposiumon computational intelligence in scheduling(SCIS);2013.p.60–7.

[7] MesselisT, DeCausmaeckerP,VandenBergheG .Algorithmperformance prediction for nurse rostering. In: Proceedings of the 6th multi disciplinary international schedulingconference:theoryandapplications(MISTA);2013.

[8] Osogami T,ImaiH.Classification of various neighborhood operations for the nurse scheduling problem. Technical report 135 . The Institute of Statistical Mathematics; 2000.

[9] De CausmaeckerP,VandenBergheG.Acategorisationofnurserostering problems. JSched2011;14(1):3–16.

[10] Nemhauser G,WolseyLA.Integerandcombinatorialoptimization.NewYork: John Wiley&Sons;1988.

[11] HaspeslaghS, DeCausmaeckerP, StølevikM, SchaerfA. The first international nurse rostering competition 2010 .In:Proceedings of the 8th international conference on the practice and theory of automated timetabling (PATAT); 2010.

[12] Warner H. W. , “Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Approach”. *Operations Research*, Vol. 24, 1976 , pp. 842-856

[13] Millar and Kiragu, *European Journal of Operational Research*, Vol. 104, Issue 3, 1 February 1998, pp. 582-592

[14] Arthur J. L., A. Ravindran, "A Multiple Objective Nurse Scheduling Model," *AIIE Transactions*, Vol. 13, March 1981, pp. 55-60

[15] Musa A., Saxena U., "Scheduling Nurses Using Goal-Programming Techniques," *IIE Transactions*, Vol. 16, September 1984, pp. 216-221

[16] Azaiez M. N., S. S. Al Sharif, “A 0-1 goal programming model for nurse scheduling”, *Computers and Operations Research*, Vol. 32, Issue 3, March 2005, pp. 491-507

[17] Reeves C. R. ,”Modern Heuristic Techniques for Combinatorial Problems", *Orient Longman*, 1993

[18] Dowsland, “Nurse scheduling with tabu search and strategic oscillation”, *European Journal of Operational Research*. Vol. 106, Issue 2-3, 1998, pp.393-407

[19] Dowsland, Thompson, “Solving a nurse scheduling problem with knapsacks”, networks and tabu search, *Journal of the Operational Research Society*, Vol. 51, Issue 7, 2000, pp. 825-833

[20] Aickelin U., A. Kathryn, Dowsland, “An indirect genetic algorithm for a nurse-scheduling problem”, *Computers and Operations Research*, Vol. 31, Issue 5, April 2004, pp. 761-778

[21] Abdennadher S., H. Schlenker, “Nurse scheduling using constraint logic programming”, *Proc. of the 11th Annual Conference on Innovative Applications of Artificial Intelligence* (IAAI-99), Orlando, Florida, United States, July 18-22, 1999, pp. 838–843

[22] Kumar V., “Algorithms for Constraint Satisfaction Problems: A Survey”, *A I Magazine*, Vol. 13, Issue 1, 1992, pp. 32-44

[23] Kundu S., M. Mahato, B. Mahanty and S. Acharyya, “Comparative Performance of Simulated Annealing and Genetic Algorithm in Solving Nurse Scheduling Problem”, *Lecture Notes in Engineering and Computer Science*, Editors: S.I. Ao, O. Castillo, C. Douglas, D.D. Feng, Jrong-A Lee, *(IMECS-2008, Proceedings of International MultiConference of Engineers and Computer Scientists)*, Hong Kong, 2008, pp. 96-100

[24] Kundu S., S. Acharyya, "A SAT Approach for Solving The Nurse Scheduling Problem", *Proc IEEE TENCON-08, CD, Session P16(Algorithms and Architecture)*(6 Pages), Hyderabad, India, Nov 18-21, 2008

[25] Kundu S., S. Acharyya, “Performance of WalkSAT in Solving The Nurse Scheduling Problem”, *International Journal of Information Processing*, Vol. 4, Issue 1, 2010, pp. 46-53

[26] Ikegami A., A. Niwa, M. Ohkura, “Nurse scheduling in Japan. Commun”, *Oper. Res. Society of Japan*, Vol. 41, 1996, pp. 436–442 (in Japanese)

[27] Mitchell M., “An Introduction to Genetic Algorithms”, *MIT Press*, 1996

[28] Aickelin U., Dowsland, “Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem”, *Journal of Scheduling*, Vol. 3, 2000, pp. 139–153

[29] Holland J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[30]]Ioannis X. Tassopoulos, Ioannis P. Solos, Grigorios N. Beligiannis: A two-phase adaptive variable neighborhood approach for nurse rostering. Computers & OR 60: 150-169 (2015)

[31] Acharyya S. “The Satisfiability Problem: A Constraint Satisfaction Approach”, *PhD Thesis*, Computer Science & Engg., Calcutta University, 2001

[32] Kundu S., S. Acharyya, “Stochastic Local Search Approaches in Solving the Nurse Scheduling Problem”, in the 10th International Conference on Computer Information Systems and Industrial Management Applications (CISIM 2011), Kolkata, India, December 14-16, 2011, pp. 202-211
[33] Gutjahr WJ, RaunerMS. An ACO algorithm for a dynamic regionalnurse- scheduling problem in Austria. Comput Oper Res 2007;34(3):642–66.
[34] Gunther M, Nissen V. Particle swarm optimization and the agent-based algorithm for a problem of staff scheduling. Lecture notes in computer science, vol. 6025.Berlin, Heidelberg: Springer;2010.p.451–61.
[35] Mazure, B., Sais, L., Gregoire, E.: Tabu Search for SAT. In: Proc. AAAI-1997, pp. 281–285 (1997)
[36] Fukunaga, A.S.: Variable Selection Heuristics in Local Search for SAT. In: Proc. AAAI-1997, pp. 275–280 (1997)
[37] Haspeslagh S, De Causmaecker P, Stølevik M, Schaerf A. The first international nurse rostering competition 2010. In: Proceedings of the 8th international conference on the practice and theory of automated timetabling (PATAT); 2010.
[38] Ioannis X. Tassopoulos, Ioannis P. Solos, Grigorios N. Beligiannis ,“A two-phase adaptive variable neighborhood approach for nurse rostering ”,Computers & Operations Research 60 (2015) 150–169
[39] Erfan Rahimian , Kerem Akartunalı , John Levine, “ A hybrid Integer Programming and Variable Neighbourhood Search algorithm to solve Nurse Rostering Problems ”, European Journal of Operational Research 000 (2016) 1–13
[40] Semra Ağralı , Z. Caner Taşkın , A. Tamer Ünal, “Employee scheduling in service industries with flexible employee availability and demand ” Omega 66 (2017) 159–169
[41] H. Jafari et al., “Fuzzy Mathematical Modeling Appquroach for the Nurse Scheduling Problem: A Case Study”, International Journal of Fuzzy Systems ,April 2016 , Volume 18, Issue 2, pp 320–332