2018

A Bi-level Heuristic Solution for the Nurse Scheduling Problem Based on Shift-swapping

Ahmed Youssef
Fordham University

Samah Senbel
Sacred Heart University, senbels@sacredheart.edu

Recommended Citation
Youssef, A. & Senbel, S. (2018, Jan.). *A Bi-level heuristic solution for the nurse scheduling problem based on shift-swapping*. Paper presented at IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, Nevada. doi:10.1109/CCWC.2018.8301623

A Bi-level Heuristic Solution for the Nurse Scheduling Problem Based on Shift-swapping

Conference Paper · January 2018

DOI: 10.1109/CCWC.2018.8301623

2 authors:

Samah Senbel
Sacred Heart University
21 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Ahmed Youssef
Fordham University
1 PUBLICATION **0** CITATIONS
SEE PROFILE

A Bi-level heuristic solution for the Nurse Scheduling Problem based on Shift-swapping

Ahmed Youssef
Dept. of Computer Science
Fordham University
New York City, NY, USA
ayoussef3@fordham.edu

Samah Senbel
College of Computing
Sacred Heart University
Fairfield, CT, USA
senbels@sacredheart.edu

***Abstract*— This paper presents a new heuristic solution to the well-known Nurse Scheduling Problem (NSP). The NSP has a lot of constraints to satisfy. Some are mandatory and specified by the hospital administration, these are known as hard constraints. Some constraints are put by the nurses themselves to produce a comfortable schedule for themselves, and these are known as soft constraints. Our solution is based on the practice of shift swapping done by nurses after they receive an unsatisfactory schedule. The constraints are arranged in order of importance. Our technique works on two levels, first we generate a schedule that satisfies all the hard constraints and guarantees fairness. The next level is to attempt to satisfy as many as possible of the soft constraints, by shift-swapping while maintaining the hard constraints. The technique was implemented as a simulation and demonstrated a satisfactory outcome.**

Keywords*— *Nurse Scheduling problem, heuristic technique, hard constraints, soft constraints, shift swapping.

I. INTRODUCTION

The nurse scheduling problem is an np complete problem that deals with the problem of assigning a set of nurses to a schedule that satisfies a series of constraints, some mandatory and some preferential. It is typically done on a weekly or monthly basis, and is an extremely difficult and unpopular task to do manually.

There are a set of constraints to fulfil, some are set up by the hospital as a "hard" constraint, and some are requests by the nurses, which is considered a "soft" constraint that may or may not be satisfied. Fairness is also an important consideration, both in the number of assigned shifts to the nurses, and the degree of satisfaction of their individual soft constraints.

It is important to increase nurse satisfaction as it is a major reason for nurses who quit their position (30.4% citied it as the main reason) to help with retention and quality of work [13]. A favorable work schedule is one way to increase their satisfaction, as well as work team formation. Shortage of nurses is also a major issue in hospitals [6], which may lead to over-scheduling and inadequate rest periods. Therefore it is important to have enough nurses to cover the schedule and be able to provide some flexibility in assigning days-off.

This scheduling problem cannot be solved by exact methods in a reasonable amount of time [17], therefore most solutions to this problem involves soft computing, fuzzy systems and heuristics.

The paper is organized as follows: In section 2, we provide a literature review of the different solutions to the NSP. In section 3, we describe the NSP definition, constraints and data structures used. In Section 4, we explain our proposed bi-level solution to the problem. Section 5 presents our implementation results, and section 6 is the conclusion, and describes our future work on this problem.

II. LITERATURE REVIEW

Due to the np-completeness of the NSP, a multitude of solutions have been provided by researchers throughout the years. Most are tailored to the needs of a particular hospital, but some provide more generic solution.

Soft Computing techniques are the most popular. Several researchers used Particle Swarm optimization [18] [19] as an effective solution to the problem, as well as Genetic algorithms [9] [12]. Jan et al. [8] presents several evolutionary algorithms for solving the NSP. An interesting approach is presented in [15] that uses a fuzzy metamorphosis technique. Gonsalves et al. [4] has an interesting bi-level approach which is a mix of genetic algorithms and a local search algorithm to optimize it and get better results faster.

Mathematical modeling is also a popular technique for solving the problem [5] [7] [16]. Reference [21] has a weighted constraint optimization approach to the solution. Reference [10] uses a Bayesian optimization technique. Reference [2] has a variable neighborhood search technique for balancing the preferences satisfaction. Reference [14] worked on an automatic rotating schedule for workforce scheduling including nurses.

Several researchers also worked on topics related to the NSP: Reference [22] uses a binary goal programming technique for an outpatient clinic nurse scheduling problem. Reference [1] provides a tailored schedule for the NICU using a min-max technique. Reference [3] provides an interesting solution to the problem based on the social structure for team formation to enable the nurses to work in homogenous teams. Reference [11] considered another interesting problem: the assignment of lunch breaks to nurses in operating rooms.

III. Nurse Scheduling Problem Definition

In this section, we specify the notations, data structures, and constraints used in this solution of the nurse scheduling problem.

A. Notation

- There are ***N*** nurses to Schedule, i=1, 2,N
- The Scheduling period is for ***D*** days, which is then repeated.
- Each day has ***S*** equal-time shifts to be assigned.
- Each shift needs ***C*** number of nurses to cover it. It could be the same for all shifts, or varies by shift.

B. Data Structures

We propose the use of a vector and two two-dimensional matrices for implementing our solution:

The Coverage vector (C) is a vector of length S*D, and contains the required number of nurses for each shift.

The Schedule table (Sch) is a table of size N rows (one per nurse) and S*D columns, one per shift for all D days. The value of each element is either 0 (not working) or 1 (working):

$$\forall_{i=0,1,\ldots,N-1}\ \forall_{j=0,1,\ldots,(S*D-1)}\ Sch_{i,j} \in \{0,1\} \quad (1)$$

Figure (1) illustrates the Scheduling table when S=3 (3 shifts per day).

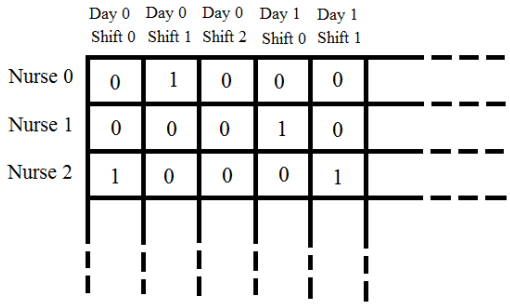

Figure 1 The Scheduling table format

The Preference table (Pref) is of similar dimensions to the Scheduling table. It contains the preference for each shift by each nurse, represented as a "penalty" point if that day is assigned. A value of 0 means that this shift is favorable, and an increasing value signals the amount of dislike for this shift. Any range of values can be used. This table is supplied by the nurses themselves to represent their desires for the scheduling period.

C. Hard Constraints:

We have four hard constraints to guarantee:

1. Guarantee hospital-required coverage for each shift. The number of nurses required per shift could be fixed for all shifts, such as in the Emergency room services, and the ICU. Or it could vary from shift to shift as in outpatient units.
   Constraint 1:
   $$\forall_{j=0,1,\ldots,(S*D-1)}\ C_j = \sum_{i=0}^{N-1} Sch_{i,j} \quad (2)$$
2. No consecutive shifts for any nurse. This is a natural constraint to guarantee a rest period for nurses and ensure they are not tired. If there is a previously generated schedule, then the last shift of it has to be taken into consideration as well.
   Constraint 2:
   $$\forall_{i=0,1,\ldots,N-1}\ \forall_{j=0,1,\ldots,(S*D-2)}\ Sch_{i,j} + Sch_{i,j+1} \in \{0,1\} \quad (3)$$
3. One shift per day for all nurses. The sum of all shift assignments is 1 (working) or 0 (day off) for all days and all nurses
   Constraint 3:
   $$\forall_{i=0,1,\ldots,N-1}\ \forall_{d=0,1,..,D-1}\ \sum_{j=0}^{S-1} Sch_{i,j+d*S} \in \{0,1\} \quad (4)$$
4. Fairness. Approx. equal number of shifts per nurse. The shift of each nurse should be equal to the average load per nurse ± Δ, where Δ is a small value, typically 0 or 1.
   Constraint 4:
   $$AverageLoad = \left(\sum_{i=0}^{N-1}\sum_{j=0}^{S*D-1} Sch_{i,j}\right) / N$$
   $$\forall_{i=0,1,\ldots,N-1}\ \sum_{i=0}^{N-1} Sch_{i,j} = AverageLoad \pm \Delta \quad (5)$$

D. Soft Constraints:

The soft constraints represent the nurses' personal preferences for days off, particular shifts off, and a preference for one or more of the different shifts. These constraints are not guaranteed to be met. The Pref table contains all theses preferences. A very high penalty is put for days and shifts off (100 or more), and a lower penalty for undesirable but acceptable shifts(1,2).

5. Required Days off for all nurses.
6. Required Shifts off for all nurses.
7. Shift preference applied for all nurses.

$$\text{Minimize Total\_Penalty} = \sum_{i=0}^{N-1}\sum_{j=0}^{S*D-1} Pref_{i,j} * Sch_{i,j} \quad (6)$$

E. *Objective Function*

Based on the four hard constraints and the three soft constraints, the NSP can formulated as follows:

$$\text{Minimize Total\_Penalty} = \sum_{i=0}^{N-1} \sum_{j=0}^{S*D-1} Pref_{i,j} * \ Sch_{i,j}$$

Where

$$\forall_{j=0,1,\ldots,(S*D-1)} \ C_j = \sum_{i=0}^{N-1} Sch_{i,j}$$

$$\forall_{i=0,1,\ldots,N-1} \ \forall_{j=0,1,\ldots,(S*D-2)} \ Sch_{i,j} + Sch_{i,j+1} \in \{0,1\}$$

$$\forall_{i=0,1,\ldots,N-1} \ \forall_{d=0,1,..,D-1} \ \sum_{j=0}^{S-1} Sch_{i,j+d*S} \in \{0,1\}$$

$$\forall_{i=0,1,\ldots,N-1} \ \sum_{i=0}^{N-1} Sch_{i,j} = (\sum_{i=0}^{N-1} \sum_{j=0}^{S*D-1} Sch_{i,j}) / N \pm \Delta$$

IV. Proposed Solution to the NSP

We propose a two-level solution to this np-hard problem. The first level to find a schedule that satisfies only the hard constraints. Those constraints are in order of importance, and we satisfy the four hard constraints one by one. The second level is to try to minimize the Total_Penalty, once again one by one in order of importance. Several solutions are produced, and the solution with the minimum Total_Penalty is chosen. We use a maximum number of trials to guarantee that the algorithm eventually stops (MaxTrial). Algorithm 1 shows an overview of our solution.

Algorithm 1 Pseudo-code of the Bi-level heuristic solution.

```
1)  Set Trial = 0
2)  Set Min_Tot_Penalty = MaxInt
3)  repeat
4)      Generate a random Schedule
5)      Attempt to Satisfy the four hard constraints
6)      if constraints not satisfied then
7)          go to step 4
8)      endif
9)      Satisfy three soft constraints
10)     Calculate Total_Penalty (eqn 6)
11)     if Total_Penalty < Min_Tot_Penalty then
12)         Save Scheduling table as Optimal so far
13)         Min_tot_Penalty = Total_penalty
14)     endif
15)     Trial ++
16) until Trial = MaxTrial
17) AnnouceOptimal Scheduling table
```

A. *Phase 1: Guaranteeing the Hard Constraints*

Step 1: We start by constraint 1, the hospital requirement of guaranteeing coverage for all shifts, as this is the most important constraint. This is done by randomly choosing the required number of nurses for each shift, this data is found in the coverage vector C. Algorithm 2 shows this step.

Algorithm 2 Pseudo-code for guaranteeing constraint 1

```
1)  Initialize all the Scheduling table Sch to 0 (free)
2)  for each column j in Sch do
3)      Pick C[j] random distinct nurses and set them to 1 (busy)
4)  end for
```

Step 2: Next, we try to guarantee Constraint 2. The generated randomly-assigned Schedule table is searched, row by row, for any 2 consecutive shifts. The last shift from the previous Schedule, if it exists, is needed to guarantee no consecutive shift in column 0. If this data is unavailable, we assume all nurses were free in the time prior to the Scheduling period.

If two consecutively assigned shifts are found for nurse "x", we swap the second shift with any nurse "y" that is free on that shift. This may, of course, cause nurse "y" to break the constraint, so this step iterates until there is no consecutive shifts found for all nurses. This step assumes there are enough nurses to support the swapping. To avoid an infinite loop, a certain max number of iterations "MAXITERATIONS" is used. Algorithm 3 shows this step.

Algorithm 3 Pseudo-code for guaranteeing non-consecutive shifts

```
1)  iteration = 0
2)  do
3)      found=0
4)      iteration++
5)      for all nurses do
6)          for all shifts do
7)              if two consecutive shifts are found then
8)                  found++
9)                  set "x" to be the current nurse
10)                 Search for a free nurse "y" on the second shift
11)                 if a free nurse is found then
12)                     Set nurse x's shift to free (0)
13)                     Set nurse y's shift to busy (1)
14)                 end if
15)             end if
16)         end for
17)     end for
18) while found > 0 and iteration < MAXITERATIONS
19) if found > 0 then
20)     Dismiss solution and go back to Step 1
21) end if
```

Step 3: Guaranteeing one shift per day

The number of shifts per day is usually 2, 3, or at most 4. In case of two shifts per day, constraint 2 guarantees constraint 3 as well. In case of three shifts per day, the only sequence possible to result from step 2 and breaks constraint 3 is "101". In case of four shifts per day, the possible sequences would be "1001", "1010", or "0101". To satisfy the constraint we pick one of the two busy shifts and try to exchange it with a nurse who is free on that day, taking into consideration the shift before it or after it. Figure 2 illustrates this exchange when S=3: Find a nurse x who has the pattern 101 on a certain day j. If found, look for a nurse y with pattern 000 on day j and 0 on the first shift of day j+1. If found, exchange shift 3 on day j between them. Alternatively, look for nurse y with pattern 000

on day j and 0 on the third shift of day j-1. If found, exchange shift 1 on day j between them.

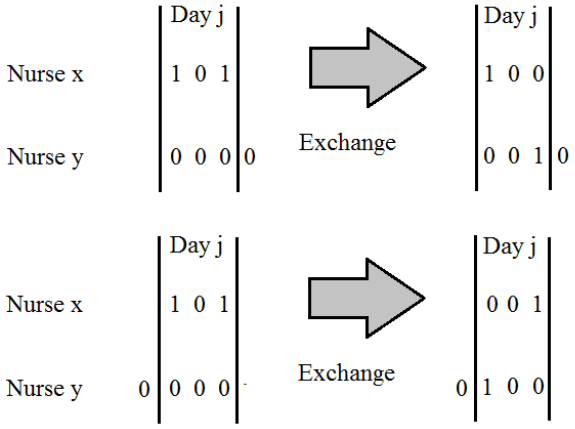

Figure 2: Satisfying Constraint 3 when S=3

In case of S=4, we have six possible exchanges. They are illustrated in figure 3, below. Algorithm 4 shows this step.

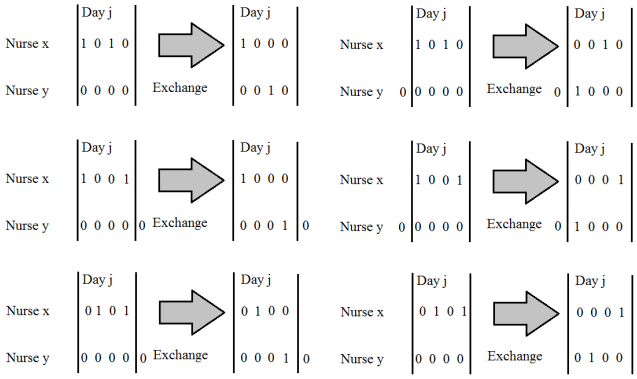

Figure 3 Satisfying Constraint 3 when S=4

Algorithm 4 Pseudo code for guaranteeing one shift a day

```
1)  iteration = 0
2)  do
3)     found=0
4)     iteration++
5)     for all nurses do
6)       for all days  do
7)          if two shifts are found on that day then
8)               found++
9)               Set "x" to be the current nurse
10)              Search for a free nurse "y" on the same day
11)               if a free nurse is found and exchange is possible then
12)                   set the shift in nurse x to free (0)
13)                   set the matching shift of nurse y to busy (1)
14)              end if
15)         end if
16)      end for
17)    end for
18) while found > 0 and iteration < MAXITERATIONS
19) if found > 0  then
20)      Dismiss solution and go back to Step 1
21) end if
```

Step 4: Guaranteed fairness for all nurses. We start by getting the average number of shifts per nurse. Then we attempt to move shifts from nurses with load > Av+1 to nurses with load < Av-1, until eventually the loads balance out. If S=3, we will have 3 possibilities as shown in Fig 4. Algorithm 5 shows this important step.

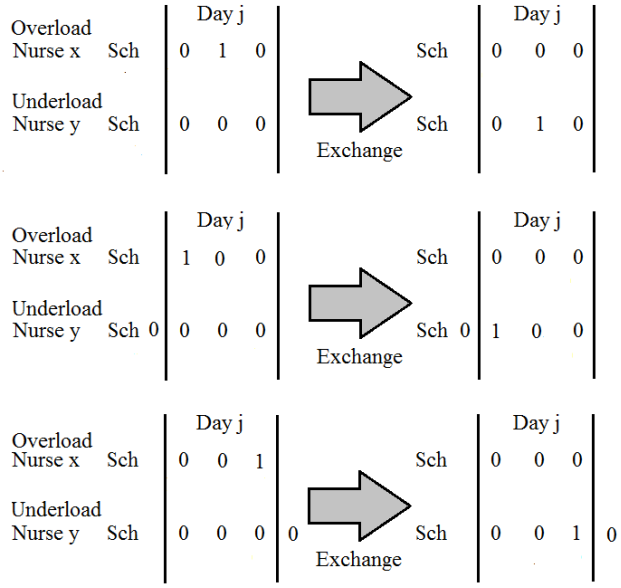

Figure 4: Exchanging loads to satisfy constraint 4 at S=3

Algorithm 5 Pseudo code for guaranteeing fairness

```
1)  let N be an array containing the number of shifts for each nurse
2)  let Av be the average of array N
3)  let overload be the number of nurses with N[i] > Av+1
4)  let underload be the number of nurses with N[i] < Av-1
5)  iteration = 0
6)  repeat
7)    if overload > 0 and underload >0 then
8)      for all nurses do
9)          if  N[i] > Av+1 then
10)             Pick a random busy  day j to exchange
11)             Look for underload nurse y (N[y]<Av-1) with the
                same day j off and an  exchange is possible without
                violating previous  constraints, as shown in figure 4.
12)             if nurse y is found then
13)                 Exchange schedules for day j
14)                 Update N for both nurses
15)             end if
16)         end if
17)     end for
18)   else if overload>0 and underload=0 then
19)     for all nurses do
20)         if  N[i] > Av+1 then
21)             Pick a random busy  day j to exchange
22)             Look for underload nurse y (N[y]=Av-1) with the
                same day j off and an  exchange is possible without
                violating previous  constraints, as shown in figure 4.
23)             if nurse y is found then
24)                 Exchange schedules for day j
25)                 Update N for both nurses
```

```
26)              end if
27)           end if
28)        end for
29)  else if  overload=0 and underload > 0 then
30)         for all nurses do
31)           if   N[i] = Av+1 then
32)              Pick a random busy  day j to exchange
33)              Look for underload nurse y (N[y]<Av-1) with the
                 same day j off and an  exchange is possible without
                 violating previous  constraints, as shown in figure 4.
34)              if nurse y is found then
35)                   Exchange schedules for day j
36)                   Update N for both nurses
37)              end if
38)           end if
39)         end for
40)   end if
41)   Re-calculate Av, overload, and underload
42)   iteration++;
43)  until (overload, underload=0) or iteration>=MAXITERATION
44)  if overload>0 or underload>0  then
45)       Dismiss solution and go back to Step 1
46)  end if
```

If a solution is found to satisfy all four constraints, we go on to the second level of satisfying the soft constraints. If no solution is found, we return to step 1.

B. Phase 2: Optimizing the Soft Constraints

Once a feasible solution to the four hard constraints has been found, phase two attempts to satisfy the three soft constraints in order, without violating any of the previously-satisfied hard constraints. Since the soft constraints are not mandatory to satisfy completely, we use a penalty system. We will calculate the total penalty for each solution, and choose the final solution to be the one with the minimum penalty. We will start using the Pref matrix in phase two. To clarify, we use a penalty of 100 for a required shift off, 1 for an un-preferred shift, and 0 for the preferred shift of the day.

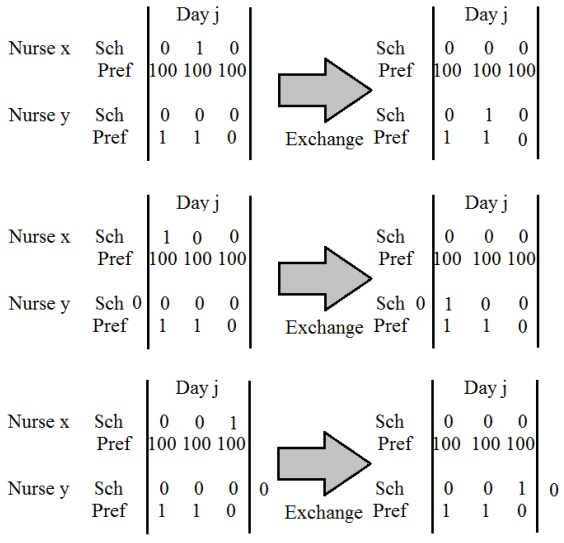

Figure 5: Satisfying constraint 5 when S=3

Step 5: Satisfying the required days off for all nurses.

We start by finding a nurse with a preference for the whole day off, but is assigned a shift on that day. If found, we find a nurse who is free on the same day, but is not preferring it as a day off. The entire day is exchanged, taking into consideration the previous shift or the following shift. To maintain the fairness constraint, we make sure a nurse only gives a work day to another nurse with an equal or less work load (Found in array N, as the fairness constraint is more important. Figure 5 shows the different possibilities when S=3. The algorithm iterates over all nurses and all days, trying to exchange with other nurses. Each day exchange decreases the penalty by 99 or 100 points, as shown in Algorithm 6.

Algorithm 6 Pseudo code for scheduling days off

```
1)  for all nurses (x) do
2)    for all days  (j) do
3)       if  nurse x busy and prefers day off then
4)            Search for a nurse y with day j off
              and it is not her preferred day off
              and load(nurse y ) <= load(nurse x)
5)            if nurse y is found then
6)               Exchange the schedule for nurses x and y
7)               Update the nurse load data for nurses x and y
8)            end if
9)       end if
10)   end for
11) end for
```

Step 6: Guarantee required shift off

This step is similar to the previous one, except that we search for a nurse where a shift falls into the time slot with a penalty of 100. If found, the entire day is exchanged with a nurse who can take that shift with a penalty of 0 or 1, and provide the original nurse with a shift with penalty 0 or 1 instead of 100. The loads of both nurses are taken into consideration to maintain fairness.

Step 7: Application of shift preference

At this point, the number of shifts with a penalty of 100 would have reached a minimum for this solution, and we can now minimize the number of shifts with a penalty of 1. We use the same shift trading technique used in step 5. We demonstrate the technique in Table 1 when S=3.

Table 1: Shift Trading when S=3

Preferred Shift	Assigned Shift	Action
0 1 1 (Morning)	1 0 0 (Morning)	None
	0 1 0 (Afternoon)	Try to exchange to 1 0 0
	0 0 1 (Night)	Try to exchange to 1 0 0
1 0 1 (Afternoon)	1 0 0 (Morning)	Try to exchange to 0 1 0
	0 1 0 (Afternoon)	None
	0 0 1 (Night)	Try to exchange to 0 1 0
0 0 1 (Night)	1 0 0 (Morning)	Try to exchange to 0 0 1
	0 1 0 (Afternoon)	Try to exchange to 0 0 1
	0 0 1 (Night)	None

When an exchange is attempted, we search for a nurse that can do the exchange without violating any previous constraint and

not increase the total penalty. We may need to check the shift before or after the required day to exchange so as not to violate constraint 2. If found, the exchange will decrease the total penalty by 1 or 2. Figure 5 illustrates the six possible exchanges when S= 3. If S=4, then there will be 12 possible exchanges. The algorithm is similar to that of step 5, except for the penalties. Figure 6 shows the possible exchanges when S=3.

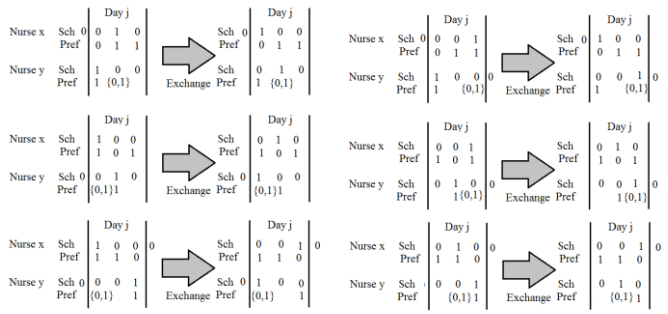

Figure 6: Satisfying Constraint 7 when s=3

After several iterations looping for possible exchanges, we calculate the total penalty, and repeat the whole seven steps several times and we keep the solution with the minimum total penalty.

V. Experimental Results

Our technique was implemented in the Java Programming language on an IBM Lenovo ultrabook with an intel core i5 processor, and 8 GB of memory. We tested the program using several values for N, S, P, Maxtrails, and different nurse preferences. In this section, we will display our experimental results for the parameters shown in table 2, which are based on the NSPL benchmark.

Table 2: Experimental parameters

Parameter	Values Used
N (number of Available nurses)	26,28,30,32,34,36
Coverage per day	Fixed at 4, 6, 8, or 10
S (Number of shifts per day)	3
D (Scheduling period)	14 and 28 days
MaxTrials	1000
MAXITERATIONS	1000
Required Days off per nurse	1 in 14 days, or 2 in 28 days
Required shifts off per nurse	1 in 14 days, or 2 in 28 days
Penalty for a day off	100x3
Penalty for a shift off	100
Penalty for a non-preferred shift	1
Penalty for a preferred shift	0

The choice of coverage, number of available nurses, scheduling period, and number of shifts per day demonstrated in this section are based on an interview in [20] based on the needs of the Emergency department at Kobry Elkobba Hospital (C=8, N=32, S=3, and D=28). However, several different values were tested as well.

The preference table was automatically generated by giving each nurse one random required day off per 14 days, in addition to her other days off. And, one required shift of per 14 days. The shift preference was also set to be at random. We are currently developing a web application to enable nurses to select their preference online, and have it reflect automatically in the preference table. Figure 7 shows an automatically generated preferences table (N=32, S=3, D=14), and Figure 8 shows a sample generated Schedule when C=8 for all days. All nurses had their days off and shifts off, and the average penalty was 2.68. The nurse loads were 11, 12 or 13 days out of 14 days.

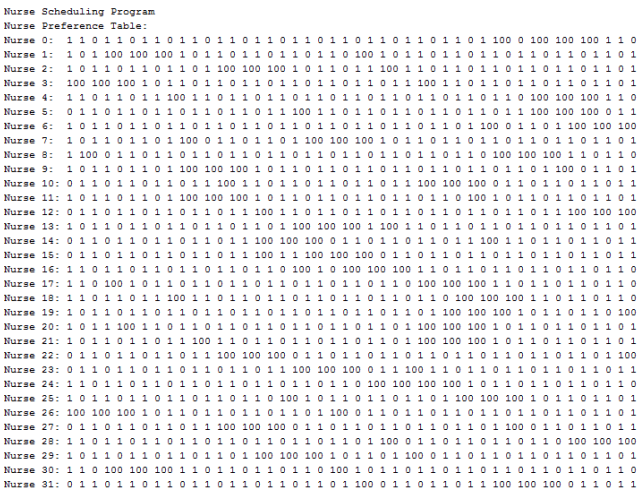

Figure 7: Nurse Preferences Table (N=32, S=3, D=14)

Figure 8 A sample Generated Schedule.

Figure 9 shows the results of running the technique 1000 times with C=8, S=3, N=32 and D=28, and observing the resulting total penalty for all nurses. All 1000 trials satisfied the four hard constraints. 8 trials had a total penalty of about 400, meaning there were 4 shifts among all 32 nurses that did not satisfy constraints 5 or 6. A penalty of 200 or 300 means that 2 or 3 shifts among all 32 nurses were not satisfying constraints 5 or 6. A reasonable acceptable total penalty would be under 100 for 32 nurses (Average penalty of about 3 per nurse for 28 days). Table 3 summarizes the results for this experiment.

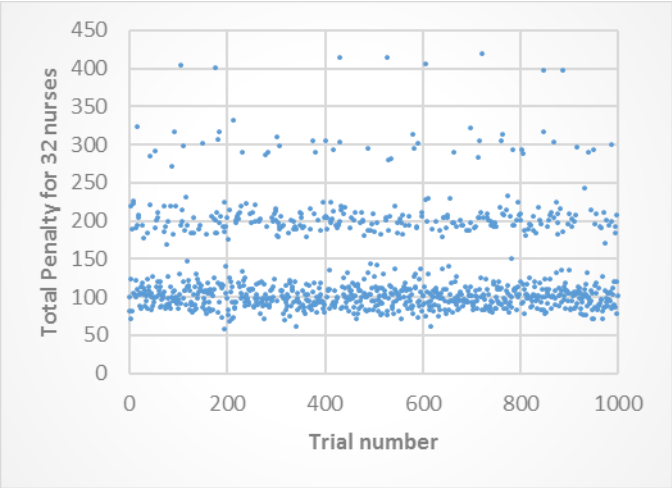

Figure 9: Total Penalty observed for multiple runs of our technique

Table 3 Performance results at C=8, S=3, N=32, and D=28

Total Penalty	Number	Percentage
Around 100	706	70.6%
Around 200	245	24.5%
Around 300	41	4.1%
Around 400	8	0.8%
Minimum	57	
Maximum	419	

After an optimal Schedule has been found, the next step would be to output the Schedule table, and deliver it to the nurses, as well as an overall view to the head nurse and hospital administration for billing. We are currently working on implementing a more visually-appealing output format.

VI. Conclusion and Future Work

In this paper, we describe a new technique for solving the nurse scheduling problem using a simple two-level system. We start by guaranteeing the hard constraints one by one using simple shift trading between the nurses. Then, the solution is refined by attempting to satisfy as much of the soft constraints as possible, using shift trading as well. Our technique provided a penalty of less than 2 per nurse, when the number of nurses is about 30% more that the needed coverage. In the future, we will develop an interactive interface for the nurses to enter their shift preferences, and an interface for the nurse manager to run the system and distribute the resulting scheduling to the nurses. We would also like to measure the nurses' satisfaction with the system, and analyze its performance over several cycles.

Reference

[1] Capan, M., Hoover, S., Jackson, E., Paul, D., Locke, R., "Integrating Nurse Preferences and Organizational Priorities into Nurse Schedules-Application to the Neonatal Intensive Care Unit", 2017 Industrial and Systems Engineering Conference, 2017.

[2] Constantiono, A., Tozzo, E., Pinheiro, R., Landa-Silva, D., Ramao, W., " A Variable Neighbourhood Search for Nurse Scheduling with Balanced Preference Satisfaction", 17th Int. conf. on Enterprise Information Systems, pp. 462-470, 2015.

[3] Farasat, A., Nikolaev, A., "Signed Social Structure Optimization for shift Assignment in the Nurse Scheduling Problem", Socio-Economic Planning Sciences, Vol. 56, pp. 3-13, 2016.

[4] Gonsalves, T., Kuwata, K., "Memetic Algorithm for the nurse Scheduling Problem", Int. Jornal of Artificial Intelligence and Applications, Vol. 6, No. 4, 2015.

[5] Hakim, L., Bakhtiar, T., Jaharuddin, " The nurse Scheduling problem: A goal programming and nonlinear optimization approaches", IOP Conf.series:Material Science and Engineering 166, 2017.

[6] Halfer, D., Graf, E., "Graduate nurse perceptions of the work experience.," Nurs. Econ., vol. 24, no. 3, pp. 150–5, 123, 2006.

[7] Jafari, H., Bateni, S., Daneshvar, P., Bateni, S., Mahdioun, H., "Fuzzy Mathematical Modeling approach for the Nurse Scheduling Problem: A case study", Int. Journal of Fuzzy Systems, June 2015.

[8] Jan, A., Yamamoto, M., Ohuchi,A., "Evolutionary algorithms for nurse scheduling problem", 2000 Congress om Evolutionary Computation, 2000.

[9] Leksakul, K., Phetsawat, S., "Nurse Scheduling Using Genetic Algorithm", Mathematical Problems in Engineering, Art. 246543, 2014.

[10] Li, J., Aickelin, U., "Bayesian Optimization Algorithm for Nurse Scheduling", Ssalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Chapter 17, pp. 315-332, 2006.

[11] Lim, G., Mobasher, A., Bard, J., Najjarbashi, A., "Nurse Scheduling with lunch break assignments in operating suites", Operations Research for Health care, 2016.

[12] Lin, C., Kang, J., Chiang, D., Chen, C., "Nurse Scheduling with Joint Normalized Shift and Day-Off Preference Satisfaction Using a Genetic Alogorithm with Immigrant Scheme", Int. Journal of Distributed Sensor Networks, Art. 595419, 2015.

[13] Lu, K., Lin, P., Wu, C., Hsieh, Y.: The relationships amongst turnover intentions, professional commitment and job satisfaction of hospital nurses. J. Prof. Nurs. 18(4), 214–219 (2002)

[14] Musliu, N., "Heuristic Methods for Automatic Rotating Workforce Scheduling", Int. Journal of Computational Intelligence Research, Vol. 2, No. 4, 2006.

[15] Mutingi, M., Mbohwa, C., "A multi-criteria approach for nurse scheduling fuzzy simulated metamorphosis algorithm approach", 2015 Int. Conf. on Industrial Engineering and Operations Management, 2015.

[16] Nasiri, M., Rahvar, M., " A two-step multi-objective mathematical model for nurse scheduling problem considering nurse preferences and consecutive shifts", Int. Journal of Services and Operations Management, March 2017.

[17] Osogami, T., Imai, H.: Classification of Various Neighborhood Operations for the Nurse Scheduling Problem. In Goos, Gerhard and Hartmanis, Juris and van Leeuwen, Jan and Lee, D.T. and Teng, Shang-Hua (ed.), Algorithms and Computation, Lecture Notes in Computer Science, Vol. 1969. Springer Berlin Heidelberg (2000)

[18] Ramli, M., Hussin, B., Abas, Z., Ibrahim, N., "Solving Complex Nurse Scheduling problems using Particle Swarm Optimization", International Review on Computers and Software, Vol 11, No 8, 2007.

[19] Rasip, N., Basari, A., Hussin, B., Ibrahim, N., "A Guided Particle Swarm Optimization for Nurse Scheduling Problem", Applied Mathematical Sciences, Vol. 8, No. 113, 5625-5632, 2014.

[20] Refat, R., Taha, A., Senbel, S., " A Heuristic Quality-based Nurse Scheduling Algorithm for Emergency Centers", 24th IEEE Int. Conference on Computer Theory and Applications (ICCTA), Egypt, October 2014.

[21] Santos, D., Fernandes, P., Cardoso, H., Oliveira, E., " A Weighted Constraint Optimization Approach to the Nurse Scheduling Problem", IEEE 18th Int. Conf. on Computational Science and Engineering, 2015.

[22] Wang, S., Yu-Khang, H., Zhuang, Z., Ou, N., " Solving an outpatient nurse scheduling problem by binary goal programming", Journal of Industrial and Production Engineering, Vol. 31, No. 1, pp. 41-50, 2014.