A Utilitarian Approach to the Nurse Staffing Problem

by John C Lawrence

This is a PrePrint

j.c.lawrence@cox.net

March 2, 2025

Abstract

Hospitals must make sure they have the required number of nurses on duty 24 hours a day although that number may vary for different time slots throughout the day. The problem is filling all the time slots with nurses in a fair way taking into account the wishes and desires of the nurses themselves. A utilitarian approach assumes that the nurses will provide their inputs in terms of their preferred schedules as well as their preferences regarding compensation. The solution would maximize nurses' utility while insuring that all shifts are covered, and that this is done within the total budget of the hospital. Computational complexity has been hitherto so demanding that solutions have been difficult to attain. Presently, powerful computer chips which have been used for AI have also provided the possibility of brute force algorithmic solutions to computationally complex problems such as this one.

Introduction

The problem is filling all the time slots at a hospital with nurses in a fair way taking into account the preferences of the nurses themselves. Nurse self scheduling (Falcone, 2023) has been responsible for better work life balance and various other benefits for the nurses. For instance, some nurses might prefer to work only days for less pay per hour and some might prefer to work nights at a higher pay per hour. In our model we consider that a nurse can submit a number of programs, each program covering one week and indicating her preferences for time slots and compensation levels for that week. We consider a basic time slot of 4 hours so that a program consists of a specification of which 4 hour time slots over a week's period that a nurse would like to work. They can combine these four hour time slots in any way they wish including consecutively. We eliminate the consideration of "official" designations such as part time, full time and overtime We assume that each program can have a time slot utility (Hillinger, 2005; Smith, 2023) between 0.1 and 1 with "1" representing highest utility and ".1" indicating lowest utility. Each program can also have a range of compensation level specifications with utilities between 0.1 and 1. A nurse would not be required to work for a compensation less than her lowest designated compensation level nor in a program not on her list. Therefore, a nurse would not be required to work in a program of utility 0. Management would set a maximum and a minimum compensation level. Nurses can specify a range of compensation levels with associated utilities within those limits.

Time slot utilities are considered to be independent from compensation utilities. We assume that a nurse can submit multiple programs. If a specific program were very desirable, she might assign it a time slot utility of one, the highest utility. For each time slot program there may be associated a range of compensation levels with associated utilities. Compensation utilities may be different for different programs. Her highest and lowest submitted compensations might not be the same as the maximum or minimum compensations possible within the system. In particular the lower the compensation asked for at a utility level of 1, the more likely it would be that a particular nurse would be assigned that program, and it is assumed that she would not be assigned a program with a compensation less than the one associated with her minimum utility. The specification of a higher time slot utility and lower compensation utilities might give a nurse a better chance to gain a more popular program preference. Conversely, specification of a higher time slot utility along with higher compensation utilities might give a nurse a good chance to gain a less popular time slot preference at higher pay. Each nurse's total utility would be the sum of the time slot and compensation level utilities for her assigned program. The social utility would be the summation of both utilities over all nurses. There are more sophisticated ways to process the utility information other than simple summation (Lawrence, 2023), and also to guarantee that all nurses have at least a minimum amount of utility (Lawrence, 2024).

The idea is to maximize social utility - in this case the combined utilities of all the nurses - within the total budget and providing all time slots are filled exactly as necessary. These two constraints - total compensation over all nurses and the specification of the exact number of nurses to fill every time slot - are determined by hospital management. The problem for the social choice is to take in all this information and come up with an assignment of nurses to time slots and compensation levels in such a way as to meet the requirements of total budget and staffing requirements of the hospital. After the computations each nurse would know her compensation and time slot assignments for a particular weekly period.

The first step is to catalog all sets of combinations of nurse's programs that meet the hospital's requirements in terms of covering all time slots with the exact number of nurses. There may be several members of this set each with its unique social utility. We assume that there is at least one. Next for each member of this set we find all combinations of compensation levels with associated utilities such that the maximum budget is not exceeded. The sum of these two utilities which maximizes the total or social utility is the one which determines the assignment of programs for each nurse for the week. An algorithm for accomplishing the special case of time slot utility maximization is presented in Appendix A. An algorithm for maximizing utility for compensation preferences would be similar, but is not considered here. Due to the extremely high number of calculations per second of the most advanced AI chips, brute force algorithmic methods for providing solutions are entirely feasible whereas previously the solutions of computational problems as complex as this one were not deemed practical.

Hospitals must insure that nurse staffing is sufficient at all times although staffing needs may vary by time slot within a 24 hour day and a 7 day week. At the same time this must be accomplished in such a way that the total budget is not exceeded. While for profit hospitals will attempt to minimize the total staffing budget, a utilitarian approach accepts the total budget as a constraint and then attempts to maximize the nurses' total utility both for time slot and compensation assignments while not exceeding the hospital's total budget. Each nurse submits a number of possible programs consisting of preferred time slot specifications along with associated utilities as well as preferred compensation levels with associated utilities for each time slot program. The sum of time slot and compensation level utilities over all nurses equals the total utility for each week. A utilitarian solution attempts to maximize this value.

The Data Structures

The nurses are each assigned numbers in an array such as nurse[i]. Nurses' names can be held in this array in alphabetical order [i.e] nurse[1] = abbot, nurse[2] = baker etc. Each nurse would submit the parameters which would represent her preferences for time slots and compensation levels. We call such a submission a program. In our model a nurse can submit as many programs as she wants each with a different preference level or utility.

The hospital administration would set the template which consists of the required number of nurses for each time slot. They would also set the total budget. Within those two constraints, there are many possible time slot and compensation level assignments for each nurse. Let's also assume that the system renews every week so that these assignments can change on a weekly basis. A further assumption is that each nurse has to be assigned at least one of her submitted programs every week, and that there is at least one set of nurses' combined programs which satisfies these constraints.

We assume that the time slots are split into 4 hour segments. For example, from 8 AM to 12 PM, 12

PM to 4 PM, 4 PM to 8 PM, 8 PM to 12 AM, 12 AM to 4 AM, and 4 AM to 8 AM. There are 6 four hour time slots per day and 42 per week. We number these from 1 to 42 starting with the 8 AM to 12 PM time slot on Monday. Time slots are numbered sequentially up to time slot 42 which would be midnight to 8 AM the following Sunday. The graphical user interface could simply show a series of squares representing the different time slots, and the nurse would click on the squares she would like to work according to each program she submits. This data is stored in an array consisting of 42 computer bits. The template would specify how many nurses re needed for each time slot. If this array were to be denoted as temp[q], where q varies from 1 to 42, for instance, temp[10] = 4 signifies that 4 nurses are needed for the time slot from 8 PM to 12 AM on Tuesday. This defines the template in terms of staffing requirements.

A program consists of a series of time slots the nurse would like to work. For instance, program 1 for nurse 1 might consist of time slots 3,4,5, 8,9,10, 24,25,30. This comes to a total of 36 hours for the week. Program 2 might consist of the following time slots: 8,9,10, 30, 31, 40, 41, 42. This is a 32 hour week. A nurse can submit as many programs as she wants with associated utilities, uts[i][j], for each program where the index i designates the nurse and the index j designates the program. Let's assume that the utilities are in the following set: uts[i][j] = {0.1, ... , 1.0}. Also a nurse can't be assigned a program that has less utility for her than 0.1

Each nurse's interaction with the graphical user interface would consist of her being queried by the GUI to submit her program preferences. First she would be asked to submit her time slot preferences, followed by the time slot utility for her first program. Then she would be asked for the same information for her next program and so on until every program that she wanted to be considered for had been entered. For each program she would be asked to submit her compensation preferences with associated utilities.

The nurses' GUI might look like the following. She would just click on the appropriate squares:

Monday 12 AM - 4 AM	Monday 4 AM - 8 AM	Monday 8 AM - 12 PM	Monday 12 PM - 4 PM	Monday 4 PM - 8 PM	Monday 8 PM - 12 AM
Tuesday 12 AM - 4 AM	Tuesday 4 AM - 8 AM	Tuesday 8 AM - 12 PM	Tuesday 12 PM - 4 PM	Tuesday 4 PM - 8 PM	Tuesday 8 PM - 12 AM
Wednesday 12 AM - 4 AM	Wednesday 4 AM - 8 AM	Wednesday 8 AM - 12 PM	Wednesday 12 PM - 4 PM	Wednesday 4 PM - 8 PM	Wednesday 8 PM - 12 AM
Thursday 12 AM - 4 AM	Thursday 4 AM - 8 AM	Thursday 8 AM - 12 PM	Thursday 12 PM - 4 PM	Thursday 4 PM - 8 PM	Thursday 8 PM - 12 AM
Friday 12 AM - 4 AM	Friday 4 AM - 8 AM	Friday 8 AM - 12 PM	Friday 12 PM - 4 PM	Friday 4 PM - 8 PM	Friday 8 PM - 12 AM
Saturday 12 AM - 4 AM	Saturday 4 AM - 8 AM	Saturday 8 AM - 12 PM	Saturday 12 PM - 4 PM	Saturday 4 PM - 8 PM	Saturday 8 PM - 12 AM
Sunday 12 AM - 4 AM	Sunday 4 AM - 8 AM	Sunday 8 AM - 12 PM	Sunday 12 PM - 4 PM	Sunday 4 PM - 8 PM	Sunday 8 PM - 12 AM

The corresponding data structure in the computer would consist of 42 bits where "1" indicates that a particular time slot is part of the current program and "0" indicates that a particular time slot is not part of the current program for a particular nurse. The programs are stored sequentially for each nurse. For instance nurse x could submit a program such as the following: Monday though Friday 8 AM to 4 PM which results in a 40 hour week. The data structure would look like the following:

0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Another example might be a nurse who wanted to work 8 AM to 12 PM Monday through Friday and then 8 PM Saturday to 8 AM Sunday for a total of 28 hours for the week. Presumably, she might be able to make a lot more money per hour on her overnight shift than on her morning shifts.

0	0	1	0	0	0
0	0	1	0	0	0
0	0	1	0	0	0
0	0	1	0	0	0
0	0	1	0	0	0
0	0	0	0	0	1
1	0	0	0	0	0

The Variables

The problem is to assign nurses to programs and to determine corresponding compensations levels for each nurse for the week. Following are the variables.

numnur	**total number of available nurses
i	**nurse identification index
j	**program identification index
prg[i]	**current program decision for nurse i
numprg[i]	**number of submitted programs of nurse i
uts[i][j]	**time slot utility for nurse i and program j
ucm[i][j][k]	**compensation level utilities for program j of nurse i
cmp[i][j][k]	**k^{th} compensation level of program j for nurse i
numcmp[i][j]	**number of compensation levels for program j of nurse i
tset[q][i]	**set of combinations of programs that meet time slot requirements, one for each **nurse

cset[q][i][k]	**set of compensations for each member of set q that meet total budget **requirements
ts[i][j][k]	**time slot information for nurse i, program j, time slot k. ts[i][j][k] = {0,1}

The Algorithm

First we want to identify all the possible combinations of work programs in which all time slots are filled with the exact number of nurses needed as specified by the template. It is assumed that there is at least one combination of nurses' programs such that all time slots are exactly covered and that every nurse is included in the final work assignments. If this were not the case, some adjustments might have to be made in nurses' assigned programs. This will not be considered here.

All of the information regarding time slot and compensation requests must be combined in such a way that each nurse is given her preferences as much as possible regarding the times that she works and the compensation she receives. This must be done subject to the constraints of total budget and staffing requirements 24 hours a day. We assume that there is a minimum and maximum compensation set by the hospital, but a nurse is free to specify her minimum and maximum acceptable compensations within those limits. The goal is to maximize the nurses' collective social utility both with respect to time slot preferences and with regard to compensation. For our model there are separate utilities for time slot and compensation, and each nurse's total individual utility is the sum of the two. We first find all those combinations of work assignments that satisfy time slot requirements. Then for those we find the ones that maximize utility for individual compensation assignments.

The algorithm proceeds as follows. First we identify the combined nurses' programs that exactly fill the hospital's requirements for staffing for the week. We check time requirements first to make sure all time slots are covered with the correct number of nurses. The set of combined nurses' programs which fulfill this requirement are saved in tset[q][i] with q being the set number and i being the nurse number. Nurses are considered in order. The programs of each nurse are also considered in order so that we can specify, for instance, a particular time slot of a particular program of a particular nurse in a 3-dimensional array e.g. ts[i][j][k] which represents the k^{th} time slot of the j^{th} program of the i^{th} nurse. ts[i][j][k] = {1,0}

List all the programs of all nurses in the following order. First list all the programs of nurse 1. Second list all the programs of nurse 2 and so on. This resembles a tree like structure as illustrated in Figure 1 where the nurses represent the trunk, the programs represent the branches and compensation levels are represented by the twigs. There are three possibilities: at each step, for the nurse's program under consideration and considering all previous nurses in the tree, there will be for each time slot, an exact fill, an underfill or an overfill with respect to staffing requirements as determined by the template. Start with the first program of nurse 1. Consider the first program of the next nurse in numerical order and check that their combined requested time slots don't exceed staffing requirements for any time slot. If there is an overfill, consider the next program in order of the nurse under consideration. If all her programs have been considered without any one of them causing an exact fill or an underfill, go back to the previous nurse and consider her next program. If this program does not result in an overfill, proceed to the next nurse in order and consider her programs in order starting with her first program until an exact fill or an underfill is found. Then go to the next nurse in order. Proceed in this way going forward when all previous nurse's programs in order have resulted in an exact fill or an underfill and proceeding backward when an overfill is found and all programs of the nurse under consideration have

been considered.

A record is kept of progress through the chain of nurses and programs in order. This record can be kept in the form prg[i] where i indicates the nurse number. For instance, prg[10] = 2 would mean that, as we proceed through the tree, the current program decision for the 10^{th} nurse is 2. This may change if no path through the tree is found in which this program for this nurse is included. It is assumed that all nurses prior to nurse 10 and including nurse 10 have been assigned programs which do not overfill any time slots. At any stage when a program is found that underfills or exactly fills shift requirements, record that in prg[i] and proceed to the next nurse. When all nurses have been considered and the process is completed with an exact fill, proceed to find other possible exact fills. Go to the next program of the last nurse and see if this results in an exact fill. If it does, record this in tset[q][i] and consider the next program of the last nurse. If it does not, proceed backward by one nurse and go to that nurse's next program. If that program does not overfill any time slot, proceed forward again. At any point if an overfill is detected, proceed backward by one nurse and consider her next program in sequence and then proceed forward to the next nurse.

Proceed until either more shifts are exactly filled or a shift is overfilled. For the last nurse sequentially there will be either an exact fill, an underfill or an overfill. If an exact fill, mark this as a possible solution in tset[q,j]. Proceed to the next program of the last nurse in order to find all of her programs that result in exact fills, and result, therefore, in acceptable sets. When all of the last nurse's programs have been considered, proceed backward and repeat the same process. Eventually, by proceeding backward we will get all the way back to nurse 1. After all of her programs have been considered, the process will terminate, and all of the acceptable sets of programs will have been found. We designate the total number of acceptable sets by the variable m. Then compute the total utilities for each acceptable set. These utilities will then be added to the compensation utilities for each acceptable time slot set, and the sum of the utilities for time slot and compensation assignments over all nurses which is a maximum will determine the nurses' assignments for the week.

The next step is to compute the compensation levels for each nurse in tset[q][j] which result in total budgets within the constraint of total compensation as defined by the template. We proceed in a similar fashion as we did with the computations for the time slot part of the program. First consider the highest requested compensation of nurse 1 in tset[1][j]. Then add the highest requested compensation of nurse 2 etc.. Continue in this way until the maximum compensation level is exceeded. Then go back one nurse and add in her next highest compensation level and then proceed forward. Proceed backward and forward in this tree like structure until an acceptable set of compensation levels is found which does not exceed the total budget. There may be several of these sets of acceptable compensation levels for each acceptable time slot program. Then do the same for tset[2]j] etc. The algorithm proceeds in a tree like structure of nurses' compensation levels as shown in Figure 1. Continue until all acceptable combinations have been found and stored in cset[q][j][k]. Then compute the social utility of each member of this set which is the sum of individual utilities. The member of the set with maximum utility is not necessarily the one which most closely approaches the total budget constraint. Finally, for each member of tset[q][j] choose that member of cset[q][j][k] whose social utility when added to the social utility of tset[q][j] is a maximum. This determines the individual assignments of time slot and compensation level programs for the week.

The Tree Structure

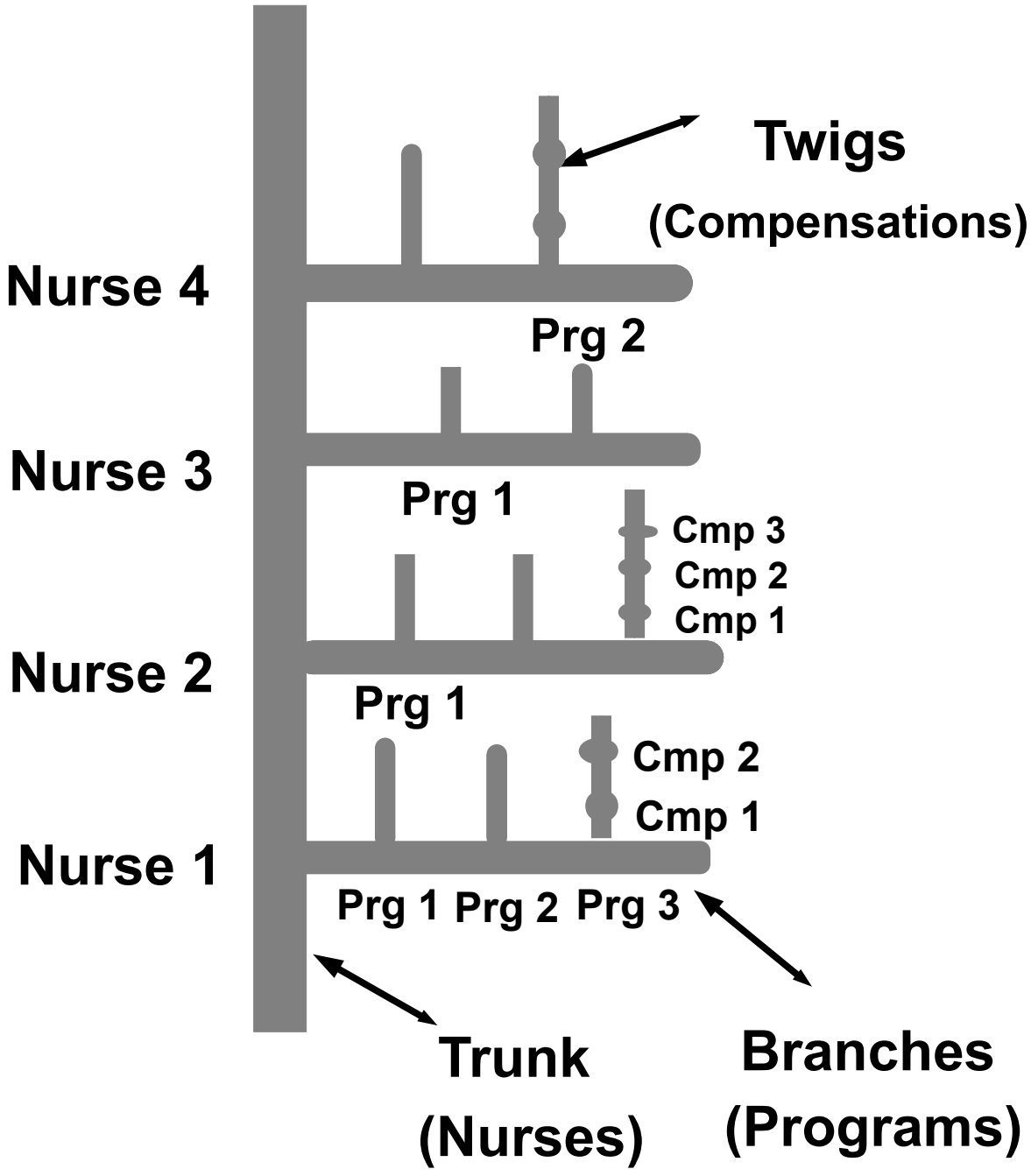

Figure 1

Summary and Conclusion

We have considered a solution to the nurse staffing problem which assigns nurses to time slots and compensation levels according to the preferences of the nurses themselves. Each nurse submits her preferences for a number of programs she would like to be considered for where a program is a perfectly general selection of 4 hour time slots over a week's period, and also a specification of preferred compensation levels with associated utilities. The utilities are chosen from the set {0.1, ..., 1} where a utility of 1 means that that time slot or compensation level is most preferred. Time slot utilities are independent from compensation utilities. For all the acceptable combinations of nurses' time slot programs which meet staffing requirements, we compute the total utility which is the sum of all the nurses' individual utilities. We also compute for each acceptable time slot program that set of individual compensation levels such that the budget set by management is not exceeded. The sum of total time slot utilities and total compensation utilities which results in a maximum then determines the assignment of programs for individual nurses for the week.

Management sets the template which consists of the number of nurses needed for each 4 hour time slot, and also the total budget. A utilitarian solution comprises finding the set of nurses' programs which fulills the time slot and budget requirements in such a way as to maximize nurses' total utility. The social utility is the sum of all the nurses' individual utilities for the assigned programs. There is a separate social utility for time slot programs and for compensation levels. The assignment of programs maximizes the sum of time slot utilities and compensation level utilities over all nurses.

Recently, the application of AI to nursing has been considered (Clancy, 2020) although primarily from the point of view of disease diagnosis and other clinical applications. For the purposes of this paper, we take advantage of the processing power of advanced AI chips, but use it for an algorithmic solution. Brute force solutions for problems such as the one considered here have heretofore been considered intractable because of the large number of computations involved. For instance, consider n nurses each having m programs. There are m^n possible paths through the tree of Figure 1. With n = 100 and m = 4, this would compute to $3^{100} = 5.15 \times 10^{47}$ calculations. According to The Verge (2024) "Nvidia says the new B200 GPU offers up to 20 *petaflops* of FP4 horsepower from its 208 billion transistors." Petaflop is a unit of computing speed equal to one thousand million million (10^{15}) floating point operations per second. FP4 means four bits of floating point precision per operation. However, for the problem considered here not every path through the tree needs to be followed. This makes it likely that chips such as the Nvidia B200 would be able to provide a solution in a reasonable amount of time. We do not ask the AI to come up with the algorithm for finding the solution to this problem. We provide an algorithmic programmed solution and then ask the AI chip to

come up with the results in a reasonable amount of time.

Appendix A

The following is a generic computer program which identifies the acceptable combination of nurses' programs which maximizes time slot utility. We leave the calculation of compensation utilities for another time. We assume that there is at least one acceptable set i.e. one set for which every nurse is assigned a time slot program that exactly fills time slot requirements. The following is a bare bones computer program which uses generic language such as for statements, go to statements, if-then-else statements, continue statements and end statements and shows the logical flow without assigning data types etc. A basic knowledge of computer arrays is also necessary. Copious comments explain the logic.

```
i                   **number of current nurse under consideration. 1 ≤ i ≤ numnur
j                   **number of current program under consideration. 1 ≤ j ≤ numprg[i]
prg[i]              **jth program of nurse i. Current program decision.
numprg[i]           **total number of programs of nurse i
numnur              **total number of nurses
uts[i][j]           **time slot utility of  program j for nurse i
k                   **time slot indicator 1 ≤ k ≤ 42
ts[i][j][k]         **time slot information for nurse i, program j, time slot k.
                    **ts[i][j][k] = {1,0}
count[k]            **running count of nurses who want time slot k
max[k]              **maximum number of nurses needed for time slot k
tset[q][i]          **acceptable program decisions - one such that every nurse is
                    **assigned a program and the combination of programs exactly fills
                    **all time slot requirements. Program decision for nurse i is prg[i]
m                   **number of acceptable combinations in set[q][i]
utstotal[m]         **total time slot utility over all nurses for mth acceptable set

for( k=1, 42)       ** initializations

        count[k] = 0
end k

m = 0
i = 1
j = 1

b: continue

        for (k=1, 42)                   **Consider all time slots

                if (ts[i][j][k] = 1)    **a 1 means that that nurse i wants time slot k in
                                        **program j. Otherwise, ts[i][j][k] = 0
```

```
            count[k] = count[k] + 1          **there are 3 cases: exact fill, underfill and overfill
                                             **if exact fill or underfill, keep going to nurse i+1,
                                             **program j=1
                                             **if overfill, go to next program of current nurse i

            if (count[k] > max[k])           **an overfill. consider next program of current
                                             **nurse i
            then
                  for (k1=1, k)

                  count[k] = count[k] - ts[i][j][k]    **subtract off k count for current
                                                       **program which has been disqualified
                  end k1

                  j = j + 1                            **consider next program of current nurse

                  if ( j = numprg[i]+ 1)       **if all programs of current nurse have been
                                               **considered

                  go to c                      **go back to previous nurse

                  else

                  go to b                      **check next program of current nurse
            else

                  continue                     **continue with for loop if k<42
                                               **if k=42, for loop ended without overfilling
                                               **any time slot. time slots are underfilled or exactly
                                               **filled. go to next nurse
      end k


      prg[i] = j                         **current program decision for nurse i


      if (i = numnur)                    **all nurses have been considered

            go to d                      **is this an acceptable combination or not
                                         **are all time slots exactly filled or not
            else
                  i = i + 1              **consider first program of next nurse
                  j = 1
            go to b                      **go to first program of next nurse

c:          i = i - 1                    **go back to previous nurse

            if (i  =  0)                 **all acceptable combinations have been found
```

```
            go to e                     **go to end

            j = prg[i] + 1              **go to next program of previous nurse


            if (j = numprg[i] + 1)          **if yes, all programs of this nurse have been
                                            **considered

                  go to c                   **go back to previous nurse in order

            else

                  go to b                   **start checking time slot availability with next
                                            **program of this nurse

d:    for (k = 1, 42)                       **have all time slots been exactly filled?
                                            **or are some still underfilled
            count[k] = 0                    **initialize count[k]
      end k

            for( i = 1, numnur)

                  j = prg[i]          **program decisions for each nurse for this combination

                  for (k = 1, 42)     **check to see if each time slot is exactly filled with
                                      **required number of nurses

                        count[k] = count[k] + ts[i][j][k]

                  end k
            end i

      for (k = 1,. 42}

            if (count[k] = max[k])    **is time slot k exactly filled?

                  continue
            else
                  go to g             **consider next program of last nurse
      end k

      m = m + 1                       **an acceptable combination has been found

      for (i = 1, numnur)             **store each acceptable set

            tset[m][i] = prg[i]       **store each nurse's program for this acceptable set


      end i                           **acceptable combination info stored in set[m][i]
```

```
        i = i+ 1                ** a "1" will be subtracted off at c

        go to c                 **consider next program of last nurse

g:                              **current program of last nurse did not result in
                                **acceptable set
                                **consider next program of last nurse

        for (k=1, 42)           **subtract k count from last program considered of last nurse

                count[k] = count[k] - ts[numnur][prg[i]][k]

        end k

        i = i + 1               **at c "1" will be subtracted off

        go to c                 **check out remaining programs of  last nurse

e:      end                     **all acceptable combinations have been found


        for (q=1, m)

            utstotal[q] = 0     **computation of total time slot utilities. sum over
                                **all nurses for time slot utilities for each
                                **acceptable set of programs. An acceptable
                                **program is one in which all time slots are exactly
        end q                   **filled and each nurse is included


        for (q=1, m)

            for ( i=1, numnur)

                j  = set[q][i]          **set[m][i] = program decision for nurse i in
                                        **acceptable set m

                utstotal[q] = utstotal[q] + uts[i][j]       **utstotal equals sum of utilities for
                                                            **acceptable set q
            end i

            utstotal[q] = utstotal[q] / numnur      **normalized value of time slot utility
                                                    **a value between zero and one.

f:      end q                                       **time slot utilities for each acceptable set
                                                    **have been computed
```

References

1. Clancy, Thomas R. PhD, MBA, RN, FAAN. Artificial Intelligence and Nursing: The Future Is Now. JONA: The Journal of Nursing Administration 50(3):p 125-127, March 2020. | DOI: 10.1097/NNA.0000000000000855

2. Falcone, Sarah S. (2023) Why Self Scheduling is the Future of Nursing, ConnectRN blog, https://www.connectrn.com/blog/why-self-scheduling-is-the-future-of-nursing

3. Hillinger, Claude (2005) The Case for Utilitarian Voting. *Homo Oeconomicus* 22(3).

4. Lawrence, John C (2023) *Proving Social Choice Possible*, Preprint: https://www.socialchoiceandbeyond.com/proving.pdf. Awaiting reviewer selection @ Theory and Decision.

5. Lawrence, John C (2024) Utilitarian Social Choice With a Minimax Provision, Preprint: https://www.socialchoiceandbeyond.com/utilitariansocialchoice.pdf. Awaiting reviewer selection @ Journal of Theoretical Politics

6. Warren D. Smith (2023). "The case for score voting," Constitutional Political Economy, Springer, vol. 34(3), pages 297-309, September.

7. The Verge (2024), https://www.theverge.com/2024/3/18/24105157/nvidia-blackwell-gpu-b200-ai