Nurse Rostering

Bălteanu Andrei
Băncescu Rareș-Emil
Roman Ștefan
Filip Tudor-Mihail

Table of contents

01

What is NRP

NRP = Nurse Rostering Problem

Nurse Rostering Problem

The **N**urse **S**cheduling **P**roblem **(NRP)** is an NP-Hard challenge for healthcare providers, involving the creation of feasible and **optimal** schedules that balance work-rest rules and service levels.

Objectives of NRP

1. Ensuring adequate nurse coverage to maintain high-quality patient care
2. Fair distribution of workload and shifts among nurses
3. Minimizing costs associated with overtime and agency nurses

Constraints and Considerations

Nurse qualifications, skills, and specialties

Work-rest rules and labor regulations

Nurse preferences and requests

Hospital policies and staffing requirements

Complexity and Challenges of NRP

NP-hard problem: computationally difficult to solve optimally

Large number of possible schedule combinations

Satisfying multiple constraints simultaneously

Manual scheduling: time-consuming and suboptimal solutions

Solution Approaches for NRP

Mathematical programming and optimization techniques

Heuristic and metaheuristic algorithms

Artificial intelligence and machine learning methods

Automating the scheduling process for high-quality solutions that satisfy objectives and constraints

02

Dataset

http://www.schedulingbenchmarks.org/nrp/

Dataset instances

Instance	Weeks	Employees	Shift types	Best known lower bound	Best known solution
Instance1 txt xml	2	8	1	607	**607**
Instance2 txt xml	2	14	2	828	**828**
Instance3 txt xml	2	20	3	1001	**1001**
Instance4 txt xml	4	10	2	1716	**1716**
Instance5 txt xml	4	16	2	1143	**1143**
Instance6 txt xml	4	18	3	1950	**1950**
Instance7 txt xml	4	20	3	1056	**1056**
Instance8 txt xml	4	30	4	1300	**1300**
Instance9 txt xml	4	36	4	439	**439**
Instance10 txt xml	4	40	5	4631	**4631**
Instance11 txt xml	4	50	6	3443	**3443**
Instance12 txt xml	4	60	10	4040	**4040**
Instance13 txt xml	4	120	18	1348	**1348**
Instance14 txt xml	6	32	4	1278	**1278**

Input example

```
# This is a comment. Comments start with #
SECTION_HORIZON
# All instances start on a Monday
# The horizon length in days:
14

SECTION_SHIFTS
# ShiftID, Length in mins, Shifts which cannot follow this shift | separated
D,480,

SECTION_STAFF
# ID, MaxShifts, MaxTotalMinutes, MinTotalMinutes, MaxConsecutiveShifts, MinConsecutiveShifts, MinConsecutiveDaysOff, MaxWeekends
A,D=14,4320,3360,5,2,2,1
B,D=14,4320,3360,5,2,2,1
C,D=14,4320,3360,5,2,2,1
D,D=14,4320,3360,5,2,2,1
E,D=14,4320,3360,5,2,2,1
F,D=14,4320,3360,5,2,2,1
G,D=14,4320,3360,5,2,2,1
H,D=14,4320,3360,5,2,2,1

SECTION_DAYS_OFF
# EmployeeID, DayIndexes (start at zero)
A,0
B,5
C,8
D,2
E,9
F,5
G,1
H,7
```

Input example

```
SECTION_SHIFT_ON_REQUESTS
# EmployeeID, Day, ShiftID, Weight
A,2,D,2
A,3,D,2
B,0,D,3
B,1,D,3
B,2,D,3
B,3,D,3
B,4,D,3
C,0,D,1
C,1,D,1
C,2,D,1
C,3,D,1
C,4,D,1
D,8,D,2
D,9,D,2
F,0,D,2
F,1,D,2
H,9,D,1
H,10,D,1
H,11,D,1
H,12,D,1
H,13,D,1

SECTION_SHIFT_OFF_REQUESTS
# EmployeeID, Day, ShiftID, Weight
C,12,D,1
C,13,D,1
F,8,D,3
H,2,D,3
H,3,D,3

SECTION_COVER
# Day, ShiftID, Requirement, Weight for under, Weight for over
0,D,5,100,1
1,D,7,100,1
2,D,6,100,1
3,D,4,100,1
4,D,5,100,1
5,D,5,100,1
6,D,5,100,1
7,D,6,100,1
8,D,7,100,1
9,D,4,100,1
10,D,2,100,1
11,D,5,100,1
12,D,6,100,1
13,D,4,100,1
```

Hard constraints

Hospital rules:

- **Employees** cannot be assigned more than one shift on a day
- **Shift rotation**: shifts which cannot follow the shift on the previous day
- **Maximum number of shifts:** The maximum number of shifts of each type that can be assigned to each employee
- **Maximum total minutes:** The maximum amount of total time in minutes that can be assigned to each employee
- **Minimum total minutes:** The minimum amount of total time in minutes that can be assigned to each employee
- **Maximum consecutive shifts:** The maximum number of consecutive shifts that can be worked before having a day off
- **Minimum consecutive shifts:** The minimum number of consecutive shifts that can be worked before having a day off
- **Minimum consecutive days off:** The minimum number of consecutive days off that must be assigned before assigning a shift
- **Maximum number of weekends:** The maximum number of weekends that can be worked
- **Days off:** Shifts must not be assigned to the specified employee on specified days

Soft constraints

Nurse rules:

- **Shift on request:** If the species shift is not assigned to the specified employee on the specified day then the solution's penalty is the specific weight value
- **Shift off request:** If the specified shift is assigned to the specified employee on the specified day then the solution's penalty is the weight value
- **Cover**: If the required number of staff on the specified day for the specified shift is not assigned then is a soft constraint violation. If the number assigned is below the required number then the solution's penalty is (requirement - x) * weight_for_under V (x - requirement) * weight_for_over

03

Genetic algorithm

First approach: Simplified instances

Genetic algorithm (Simplified instances)

Hard constraints

Hospital rules

Soft constraints

Nurse shifts preferences

Data structures

For the genetic algorithm

Hard constraints

Example of hospital rules:

- A nurse is not allowed to work two consecutive shifts
- A nurse is not allowed to work more than five shifts per week
- The number of nurses per shift in your department should fall within the following limits:

  6.00 AM - 2.00 PM: 3-4 nurses (S1)
  2.00 PM - 10.00 PM 3-4 nurses (S2)
  10:00 PM - 6.00 AM 2-3 nurses (S3)

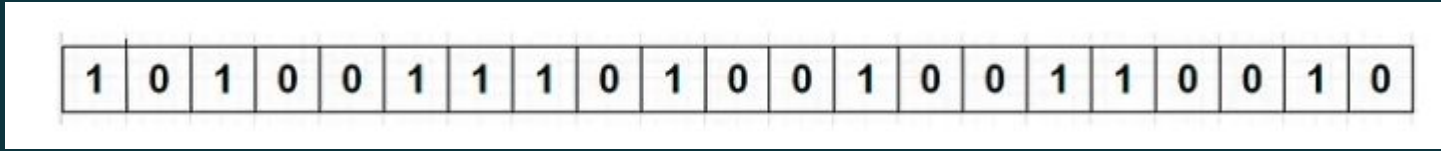
Soft constraints (Nurse preferences)

Nurse ID	Preferred Shifts
A	S1,S3
B	S2
C	S3
D	S1,S2,S3
E	S2,S3
F	S1,S3
G	S1,S2,S3
H	S3
I	S2
J	S1,S2
K	S2,S3
L	S1

Instance example

```
# list of nurses:
self.nurses = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']

# nurses' respective shift preferences - morning, evening, night:
self.shiftPreference = [[1, 0, 1], [0, 1, 0], [0, 0, 1], [1, 1, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1], [0, 0, 1], [1, 1,0],[0, 1, 1], [1, 0, 0], [1, 0, 0]]

# min and max number of nurses allowed for each shift - morning, evening, night:
self.shiftMin = [3, 3, 2]
self.shiftMax = [4, 4, 3]

# max shifts per week allowed for each nurse
self.maxShiftsPerWeek = 5

# number of weeks we create a schedule for:
self.weeks = 1
```

Data structures




```
A : [0 0 1 1 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0 1 0]
B : [1 1 0 0 1 0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1]
C : [1 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 0]
D : [1 0 1 1 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1]
E : [1 1 1 1 0 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0 0]
F : [1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1 1]
G : [1 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 1 1 0 1 0]
H : [0 1 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0]
I : [0 1 1 1 0 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0]
J : [0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 1 1 1]
K : [0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1]
L : [1 0 0 0 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0]

consecutive shift violations = 65
weekly Shifts = [10, 13, 10, 13, 11, 15, 11, 9, 12, 10, 7, 10]
Shift Preference Violations = 38
Total Cost = 1938
```

```
A : [1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0]
B : [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
C : [0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
D : [0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0]
E : [0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
F : [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
G : [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0]
H : [0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1]
I : [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0]
J : [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]
K : [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0]
L : [0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]
consecutive shift violations =  0

weekly Shifts =  [5, 4, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5]
Shifts Per Week Violations =  0

Nurses Per Shift =  [3, 4, 2, 3, 3, 2, 3, 3, 3, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2]
Nurses Per Shift Violations =  0

Shift Preference Violations =  13
```

Results

Genetic algorithm (Complex instances)

Hard constraints

Hospital rules

Soft constraints

Nurse shifts preferences

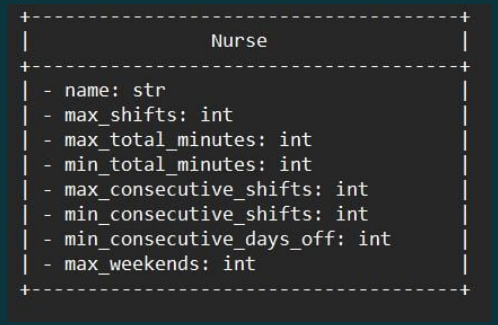
Data structures

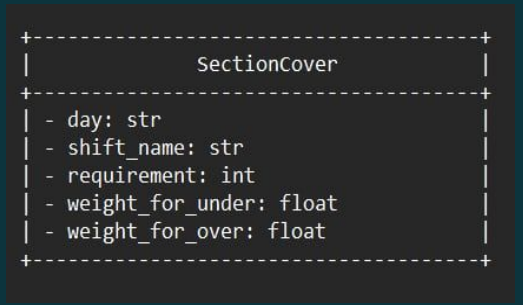
For the genetic algorithm

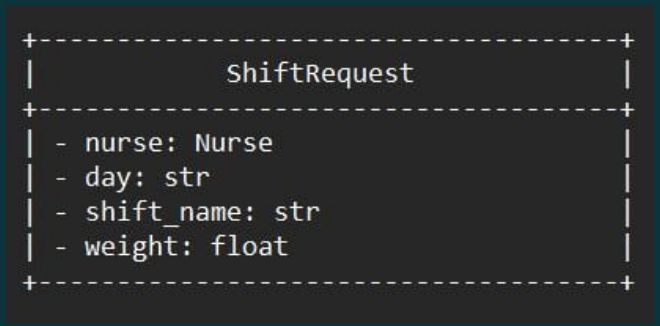
Data structures

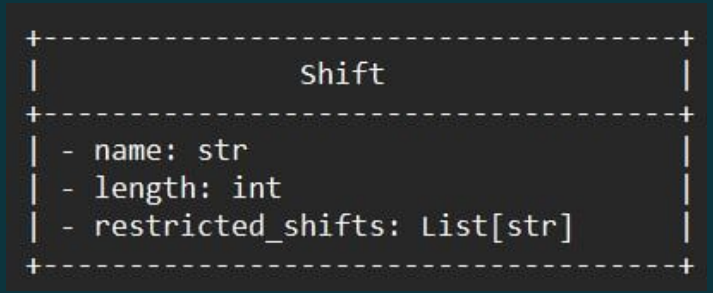
```
A:['L', 'free', 'E', 'L', 'free', 'L', 'free', 'E', 'free', 'D', 'free', 'free', 'E', 'L']
B:['free', 'free', 'D', 'D', 'E', 'free', 'E', 'E', 'D', 'E', 'D', 'D', 'free', 'D']
C:['L', 'L', 'D', 'D', 'L', 'free', 'free', 'E', 'D', 'D', 'E', 'D', 'L', 'L']
D:['free', 'L', 'E', 'free', 'E', 'free', 'free', 'L', 'L', 'L', 'D', 'E', 'free', 'L']
E:['E', 'D', 'E', 'L', 'L', 'L', 'D', 'D', 'free', 'L', 'D', 'L', 'free', 'free']
F:['D', 'free', 'E', 'E', 'free', 'E', 'D', 'free', 'E', 'D', 'D', 'E', 'free', 'free']
G:['D', 'D', 'L', 'E', 'free', 'L', 'free', 'free', 'D', 'free', 'E', 'L', 'free', 'E']
H:['E', 'free', 'D', 'E', 'free', 'L', 'free', 'D', 'free', 'D', 'D', 'D', 'D', 'D']
I:['E', 'D', 'L', 'free', 'free', 'free', 'free', 'free', 'D', 'free', 'E', 'E', 'L', 'D']
J:['L', 'L', 'E', 'L', 'free', 'E', 'D', 'D', 'D', 'E', 'D', 'L', 'L', 'L']
K:['D', 'D', 'E', 'D', 'D', 'L', 'D', 'free', 'E', 'L', 'L', 'D', 'E', 'L']
L:['E', 'L', 'L', 'D', 'free', 'free', 'D', 'L', 'E', 'L', 'free', 'free', 'L', 'E']
M:['L', 'free', 'D', 'D', 'D', 'D', 'D', 'free', 'free', 'E', 'free', 'free', 'D', 'E']
N:['L', 'D', 'free', 'E', 'free', 'free', 'free', 'free', 'L', 'free', 'free', 'D', 'E', 'D']
O:['L', 'free', 'free', 'E', 'L', 'L', 'free', 'free', 'L', 'L', 'E', 'E', 'L', 'free']
P:['free', 'D', 'free', 'D', 'L', 'free', 'free', 'free', 'E', 'free', 'D', 'free', 'E', 'D']
Q:['E', 'E', 'free', 'E', 'free', 'free', 'D', 'E', 'L', 'free', 'L', 'free', 'E', 'D']
R:['D', 'free', 'D', 'L', 'free', 'E', 'L', 'E', 'E', 'D', 'free', 'D', 'E', 'free']
S:['free', 'free', 'D', 'free', 'E', 'E', 'D', 'L', 'free', 'L', 'E', 'free', 'E', 'free']
T:['E', 'E', 'L', 'L', 'E', 'free', 'L', 'E', 'L', 'free', 'L', 'free', 'L', 'L']
```

Data structures









Genetic algorithm code

```python
def init_algoritm(values, days, number_of_nurses):

    random.seed(RANDOM_SEED)

    toolbox = base.Toolbox()

    # define a single objective, maximizing fitness strategy:
    creator.create("FitnessMin", base.Fitness, weights=(-1.0,))

    # create the Individual class based on list:
    creator.create("Individual", list, fitness=creator.FitnessMin)

    toolbox.register("ShiftForNurse", create_individual, values, days)

    # create the individual operator to fill up an Individual instance:
    toolbox.register("IndividualCreator", tools.initRepeat, creator.Individual, toolbox.ShiftForNurse, number_of_nurses)

    # create the population operator to generate a list of individuals:
    toolbox.register("populationCreator", tools.initRepeat, list, toolbox.IndividualCreator)

    toolbox.register("evaluate", calculate_cost)

    # genetic operators:
    toolbox.register("select", tools.selTournament, tournsize=tournsize)
    toolbox.register("mate", tools.cxTwoPoint)
    toolbox.register("mutate", mutate_custom, values=values, indpb=1.0/days)

    return toolbox
```

Results

Instance 1

Population size	Generations	Mutation	HCF	Score
200	50	1%	2	201021
200	100	1%	2	200924
200	200	1%	1	100822
200	500	1%	1	100618
500	50	1%	0	1118
500	100	1%	0	1018
500	200	1%	0	914
500	500	1%	0	709
500	50	10%	0	916
500	100	10%	0	513
500	200	10%	0	408
500	500	10%	0	404
500	1000	10%	0	404

Selection type: tournament (1%)

HOF size: 10% of population size

Crossover: Two point crossover

Results

Instance 3

Population size	Generations	Mutation	HCF	Score
500	50	10%	42	4203406
500	100	10%	27	2702603
500	200	10%	17	1702493
500	500	10%	11	1101769
1000	500	10%	7	702476

Selection type: tournament (1%)

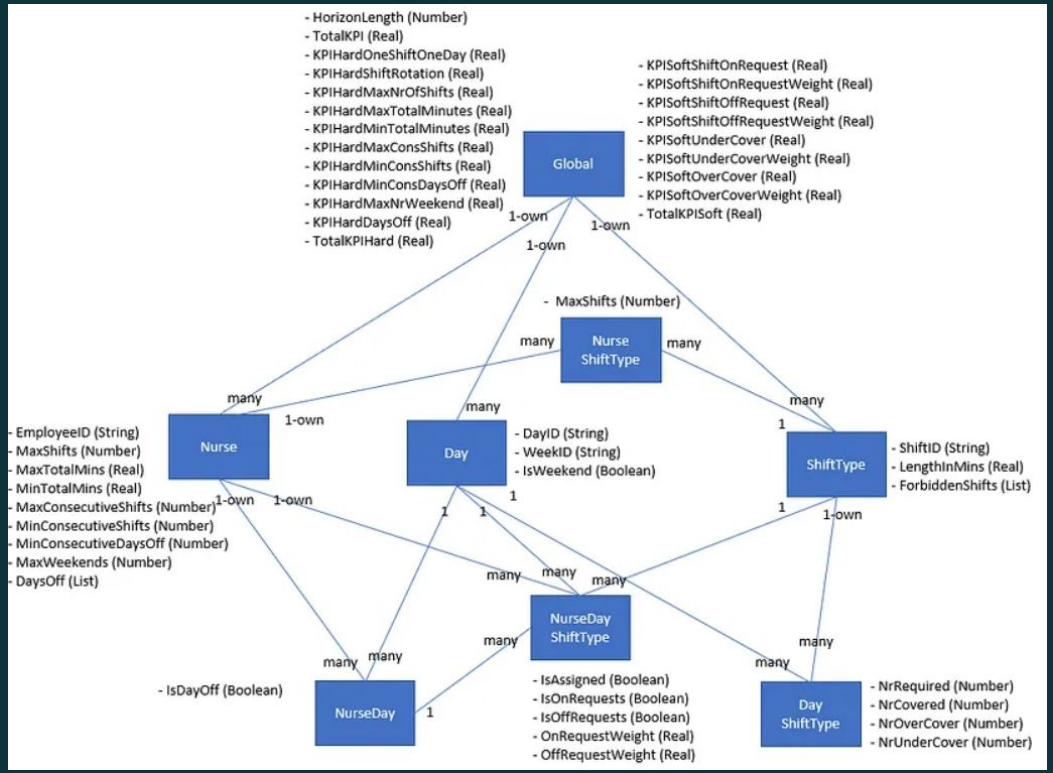
HOF size: 10% of population size

Crossover: Two point crossover

04

Constraint programming

Modeling 1


- HorizonLength (Number)
- TotalKPI (Real)
- KPIHardOneShiftOneDay (Real)
- KPIHardShiftRotation (Real)
- KPIHardMaxNrOfShifts (Real)
- KPIHardMaxTotalMinutes (Real)
- KPIHardMinTotalMinutes (Real)
- KPIHardMaxConsShifts (Real)
- KPIHardMinConsShifts (Real)
- KPIHardMinConsDaysOff (Real)
- KPIHardMaxNrWeekend (Real)
- KPIHardDaysOff (Real)
- TotalKPIHard (Real)
Global
- KPISoftShiftOnRequest (Real)
- KPISoftShiftOnRequestWeight (Real)
- KPISoftShiftOffRequest (Real)
- KPISoftShiftOffRequestWeight (Real)
- KPISoftUnderCover (Real)
- KPISoftUnderCoverWeight (Real)
- KPISoftOverCover (Real)
- KPISoftOverCoverWeight (Real)
- TotalKPISoft (Real)
1-own
- MaxShifts (Number)
Nurse ShiftType
many
Nurse
- EmployeeID (String)
- MaxShifts (Number)
- MaxTotalMins (Real)
- MinTotalMins (Real)
- MaxConsecutiveShifts (Number)
- MinConsecutiveShifts (Number)
- MinConsecutiveDaysOff (Number)
- MaxWeekends (Number)
- DaysOff (List)
Day
- DayID (String)
- WeekID (String)
- IsWeekend (Boolean)
ShiftType
- ShiftID (String)
- LengthInMins (Real)
- ForbiddenShifts (List)
1
NurseDay ShiftType
- IsAssigned (Boolean)
- IsOnRequests (Boolean)
- IsOffRequests (Boolean)
- OnRequestWeight (Real)
- OffRequestWeight (Real)
NurseDay
- IsDayOff (Boolean)
Day ShiftType
- NrRequired (Number)
- NrCovered (Number)
- NrOverCover (Number)
- NrUnderCover (Number)

Modeling 2

Sets	Description
N	Set of nurses
D	Set of days
S	Set of shift types
R_S	Set of shifts which cannot follow shift *s*
W	Set of week

Variables	Description
$IsAssign_{s,n,d}$	1 if shift *s* is assigned to nurse *n* on day *d*, 0 if otherwise
$IsRest_{n,d}$	1 if nurse *n* on day *d* is a day off, 0 if otherwise
$IsEndOfWorkBlock_{n,d}$	1 if nurse *n* on day *d* is an end of working block, 0 if otherwise
$IsEndOfRestBlock_{n,d}$	1 if nurse *n* on day *d* is an end of rest block, 0 if otherwise
$HasWeekendWork_{n,w}$	1 if in week *w*, nurse *n* has weekend work, 0 if otherwise
$PenaltyShiftOnReq_{s,n,d}$	Penalty for not assigning shift *s* on nurse *n* on day *d*
$PenaltyShiftOffReq_{s,n,d}$	Penalty for assigning shift *s* on nurse *n* on day *d*
$UnderCover_{s,d}$	Number of under assignment of shift *s* on day *d*
$OverCover_{s,d}$	Number of over assignment of shift *s* on day *d*
$PenaltyUnder_{s,d}$	Penalty for under assignment of shift *s* on day *d*
$PenaltyOver_{s,d}$	Penalty for over assignment of shift *s* on day *d*

Modeling 3

Parameters	Description
$maxshift_{n,s}$	Maximum planned shift of shift type *s* on nurse *n*
$maxtotalmins_{n}$	Maximum planned shift duration on nurse *n*
$mintotalmins_{n}$	Minimum planned shift duration on nurse *n*
$duration_{s}$	Duration of shift type *s*
$maxconsshift_{n}$	Maximum number of consecutive shifts planned on nurse *n*
$minconsshift_{n}$	Minimum number of consecutive shifts planned on nurse *n*
$minconsdayoff_{n}$	Minimum number of consecutive day-offs planned on nurse *n*
$maxweekends_{n}$	Maximum number of weekend work planned on nurse *n*
$reqnr_{s,d}$	Required number of shift type *s* on day *d*
$penaltyon_{s,n,d}$	Penalty for not following shift-on-request for shift type *s* on day *d* for nurse *n*
$penaltyoff_{s,n,d}$	Penalty for not following shift-off-request for shift type *s* on day *d* for nurse *n*
$penaltyover_{s,d}$	Penalty for planning shift type *s* on day *d* more than required
$penaltyunder_{s,d}$	Penalty for planning shift type *s* on day *d* less than required

Hard constraints

OneShiftOneDay:

$$\sum_{s \in S} IsAssign_{s,n,d} + IsRest_{n,d} = 1 \qquad \forall n \in N, \forall d \in D$$

ShiftRotation:

$$\sum_{r \in Rs} IsAssign_{r,n,d+1} + IsAssign_{s,n,d} \leq 1 \qquad \forall n \in N, \forall d \in D, \forall s \in S$$

MaxNrOfShifts:

$$\sum_{d \in D} IsAssign_{s,n,d} \leq maxshift_{n,s} \qquad \forall n \in N, \forall s \in S$$

MaxTotalMinutes:

$$\sum_{d \in D} \sum_{s \in S} duration_s \,.\, IsAssign_{s,n,d} \geq maxtotalmins_n \qquad \forall n \in N$$

MinTotalMinutes:

$$\sum_{d \in D} \sum_{s \in S} duration_s \,.\, IsAssign_{s,n,d} \geq mintotalmins_n \qquad \forall n \in N$$

Hard constraints

MaxConsShifts:

$$\sum_{prevd=d-maxconsshift_n}^{d} IsRest_{n,prevd} \geq RHS \qquad \forall n \in N, \forall d \in D$$

RHS = 0 for the first [maxconsshift$_n$] days (because the last day of previous planning horizon is a day off). Otherwise, RHS = 1

MinConsShifts:

$$IsEndOfWorkBlock_{n,d} \geq IsRest_{n,nextd} - IsRest_{n,d} \qquad \forall n \in N, \forall d \in D$$

MinConsDaysOff:

$$IsEndOfRestBlock_{n,d} \geq IsRest_{n,d} - IsRest_{n,nextd} \qquad \forall n \in N, \forall d \in D$$

MaxNrWeekend:

$$HasWeekendWork_{n,w} \geq \sum_{s \in S} \sum_{d \in weekend\ on\ w} IsAssign_{s,n,d} \qquad \forall n \in N,\ \forall w \in W$$

DaysOff:

$$IsRest_{n,d} = 1 \qquad \forall n \in N,\ \forall d \in D\ off\ for\ n$$

Soft constraints

ShiftOnRequest:

$$PenaltyShiftOnReq_{s,n,d} = penaltyon_{s,n,d} \cdot (1 - IsAssign_{s,n,d}) \qquad \forall s \in S, \forall n \in N \; \forall d \in D$$

ShiftOffRequest:

$$PenaltyShiftOffReq_{s,n,d} = penaltyoff_{s,n,d} \cdot IsAssign_{s,n,d} \qquad \forall s \in S, \forall n \in N \; \forall d \in D$$

UnderCover:

$$\sum_{n \epsilon N} IsAssign_{s,n,d} + UnderCover_{s,d} \geq reqnr_{s,d} \qquad \forall s \in S, \forall d \in D$$

$$PenaltyUnder_{s,d} = penaltyunder_{s,d} \cdot UnderCover_{s,d} \qquad \forall s \in S, \forall d \in D$$

OverCover:

$$\sum_{n \epsilon N} IsAssign_{s,n,d} - OverCover_{s,d} \leq reqnr_{s,d} \qquad \forall s \in S, \forall d \in D$$

$$PenaltyOver_{s,d} = penaltyover_{s,d} \cdot OverCover_{s,d} \qquad \forall s \in S, \forall d \in D$$

Results

We are using Python Optimization Modeling Object (**Pyomo**) as an optimizer with a **CPLEX** solver, and as an instance for the problem he uses the instances listed here. In theory his solution gave a very good solution for the first 7 instances, the others taking just too much time to reach an answer. However, our implementation succeeded only for the first instance. The CPLEX solver needed a full usage license for more than 1000 constraints and variables, thus we were unable to continue compute the answer for the remaining instances.

Instance 1:

- Score: 607
- Duration: 0.971 seconds

```
607.0
init --- 0.0029816627502441406 seconds ---
solve --- 0.9714789390563965 seconds ---
```

*All solutions were written in Python, on a Windows machine, with 16 GB memory and Intel Core i5-9300HF 2.40GHz CPU

05

Conclusions

Conclusions

Genetic algorithm:

- In the case of medium and small instances, it works very well, it finds optimal solutions
- In the case of large instances, the algorithm fails to find solutions within an acceptable timeframe

Constraint programming:

- The algorithm finds solutions for most of the instances in an efficient time (most of the results on the instances were obtained with this algorithm)
- In the case of small and medium courts, the algorithm does not find the optimal solution

06

Bibliography

1. Ziyi Chen, Yajie Dou, Patrick De Causmaecker: “Neural networked-assisted method for the nurse rostering problem”, 2022, Link
2. Tim Tjhay: “Using Machine Learning predictions in Operations Research techniques to solve the Nurse Rostering Problem”, 2023, Link
3. Sanja Petrovic, Greet Vanden Berghe: “Comparison of Algorithms for Nurse Rostering Problems”, Link
4. Chong Man Ngoo, Say Leng Goh, Nasser R. Sabar, San Nah Sze, Salwani Abdullah, Graham Kendall: “A Survey of the Nurse Rostering Solution Methodologies: The State-of-the-Art and Emerging Trends”, 2022, Link
5. “Modeling Nurse Scheduling Problem in Python (Part 1-Object Model)”, Medium, 2023, Link
6. “Solving the Nurse Rostering Problem using Google OR-Tools”, Medium, 2022, Link
7. Nurse rostering instances, Link
8. Solving Nurse Scheduling/Rostering Problems in Python, Link