Problem

This solution approach addresses the Decision Management Community April 2020 challenge. The problem comprises a staff rostering problem where doctors must be assigned to shifts, subject to the given constraints.

Approach

The approach taken in this solution is to treat the problem as a constraint satisfaction problem and solve the model using integer programming techniques. The ZIMPL mathematical modelling language is used to encode the problem suitable for solving using any number of linear programming solvers.

Model Description

The dimensions of the problem in terms of shifts and doctors as a resource are modelled as problem parameters;

```
set PROFESSIONALS := {"Fleming","Freud","Heimlich","Eustachi","Golgi"};

set DAYS := {1..7};

set SHIFTS := {"Early", "Late", "Night"};
```

The key decision variables comprise a set of binary variables which have a value of 1 if that shift, on that day is resourced by that doctor, otherwise the value is 0. In addition, some indicator variables indicating that the doctor is allocated to a shift on the weekend are used to facilitate implementation of the weekend constraints.

```
var ASSIGNED[DAYS*SHIFTS*PROFESSIONALS] binary;

var SATURDAY_INDICATOR[PROFESSIONALS] binary;

var SUNDAY_INDICATOR[PROFESSIONALS] binary;
```

A doctor's availability is captured as a matrix representing the days and shifts where a 1 indicates the doctor is available and a 0 indicates the doctor is not available. As shown in the sample below, Hemlich is available for all shifts on Friday, however unavailable for the night shift on Saturday or Sunday.

```
|"Heimlich",5| 1, 1, 1 |

|"Heimlich",6| 1, 1, 0 |

|"Heimlich",7| 1, 1, 0 |
```

One of the features of the problem is to allow doctors to specify the maximum number of shifts of a given type they are prepared to work each week. This is captured as a table. As shown in the extract below, Golgi is prepared to work a maximum of 7 early or late shifts, but no more than 2 night shifts in a given week.

|"Golgi"| 7, 7, 2 |

Objective

As stated, the problem is more oriented towards a constraint satisfaction problem rather than an optimisation problem. Hence the objective in this approach is somewhat arbitrary and just maximises the sum of assignments. This is naturally constrained by the problem constraints, however the output has value in terms of proving the model correct. Given there are 21 shifts, if all shifts are covered by 1 doctor, then the objective should equal 21.

Alternate objectives are possible. Variations could include mini/max objectives to minimise the utilisation variation amongst doctors. Another option could be maximise the roster satisfaction of the doctors by including a preference matrix as well as availability. Yet another variation could be to soften the constraints such that constraints are treated as penalties rather than hard constraints. This can often be useful in large scale problems where a solution may not be feasible given the vagaries of human choice.

Results

Using LPSolve on an Intel i7 laptop, the problem is readily solved in 0.030 seconds. ZIMPL can be translated into numerous formats including LP (Cplex) format, MPS etc. Hence many solvers culd be used to solve this model. The output using LPSolve is shown below;

```
ASSIGNED#1$Early$Eustachi                        1
ASSIGNED#1$Late$Golgi                            1
ASSIGNED#1$Night$Heimlich                        1
ASSIGNED#2$Early$Eustachi                        1
ASSIGNED#2$Late$Golgi                            1
ASSIGNED#2$Night$Heimlich                        1
ASSIGNED#3$Early$Eustachi                        1
ASSIGNED#3$Late$Golgi                            1
ASSIGNED#3$Night$Heimlich                        1
ASSIGNED#4$Early$Eustachi                        1
ASSIGNED#4$Late$Golgi                            1
ASSIGNED#4$Night$Heimlich                        1
ASSIGNED#5$Early$Eustachi                        1
ASSIGNED#5$Late$Golgi                            1
ASSIGNED#5$Night$Fleming                         1
ASSIGNED#6$Early$Golgi                           1
ASSIGNED#6$Late$Freud                            1
ASSIGNED#6$Night$Fleming                         1
ASSIGNED#7$Early$Golgi                           1
```

ASSIGNED#7$Late$Freud	1
ASSIGNED#7$Night$Fleming	1
SATURDAY_INDICATOR$Fleming	1
SATURDAY_INDICATOR$Freud	1
SATURDAY_INDICATOR$Golgi	1
SUNDAY_INDICATOR$Fleming	1
SUNDAY_INDICATOR$Freud	1
SUNDAY_INDICATOR$Golgi	1

Zimpl Model

The full model is shown below;

```
# Set of Doctors

set PROFESSIONALS := {"Fleming","Freud","Heimlich","Eustachi","Golgi"};

# Set of days - Monday is 1...

set DAYS := {1..7};

# Set of shifts

set SHIFTS := {"Early", "Late", "Night"};

# Availability matrix - 1 if available, 0 otherwise...

param AVAILABILITY[PROFESSIONALS*DAYS*SHIFTS] := | "Early", "Late", "Night"
|

|"Fleming",1| 0, 0, 0 |

|"Fleming",2| 0, 0, 0 |

|"Fleming",3| 0, 0, 0 |

|"Fleming",4| 0, 0, 0 |

|"Fleming",5| 1, 1, 1 |

|"Fleming",6| 1, 1, 1 |

|"Fleming",7| 1, 1, 1 |

|"Freud",1| 1, 1, 0 |

|"Freud",2| 1, 1, 0 |

|"Freud",3| 1, 1, 0 |
|"Freud",4| 1, 1, 0 |
|"Freud",5| 1, 1, 0 |
|"Freud",6| 1, 1, 0 |
|"Freud",7| 1, 1, 0 |
|"Heimlich",1| 1, 1, 1 |
|"Heimlich",2| 1, 1, 1 |
|"Heimlich",3| 1, 1, 1 |
|"Heimlich",4| 1, 1, 1 |
|"Heimlich",5| 1, 1, 1 |
|"Heimlich",6| 1, 1, 0 |
|"Heimlich",7| 1, 1, 0 |
|"Eustachi",1| 1, 1, 1 |
|"Eustachi",2| 1, 1, 1 |
|"Eustachi",3| 1, 1, 1 |
|"Eustachi",4| 1, 1, 1 |
|"Eustachi",5| 1, 1, 1 |
|"Eustachi",6| 1, 1, 1 |
|"Eustachi",7| 1, 1, 1 |
|"Golgi",1| 1, 1, 1 |
|"Golgi",2| 1, 1, 1 |
|"Golgi",3| 1, 1, 1 |
|"Golgi",4| 1, 1, 1 |
|"Golgi",5| 1, 1, 1 |
|"Golgi",6| 1, 1, 1 |
|"Golgi",7| 1, 1, 1 |;
```

```
# Max weekly shift type constraints – desired maximum count of shifts per shift type

param MAX_WEEKLY_SHIFTS[PROFESSIONALS*SHIFTS] := | "Early", "Late", "Night" |

|"Fleming"| 7, 7, 7 |

|"Freud"| 7, 7, 7 |

|"Heimlich"| 7, 7, 7 |

|"Eustachi"| 7, 7, 7 |

|"Golgi"| 7, 7, 2 |;

# Decision variables - the values the solver needs to find

var ASSIGNED[DAYS*SHIFTS*PROFESSIONALS] binary;

# Indicator variable to indicate the doctor works Saturday

var SATURDAY_INDICATOR[PROFESSIONALS] binary;

# Indicator variable to indicate the doctor works Sunday

var SUNDAY_INDICATOR[PROFESSIONALS] binary;

# Objective function - arbitrary maximisation for constraint satisfaction problems

maximize assigned:

sum <d,s,p> in DAYS*SHIFTS*PROFESSIONALS : ASSIGNED[d,s,p];


# A doctor works at most one shift per day...

subto One_Shift_Per_Day:

forall <p> in PROFESSIONALS :

  forall <d> in DAYS :

    sum <s> in SHIFTS : ASSIGNED[d,s,p] <= 1;


# Each shift must be covered by exactly one doctor...

subto Each_Shift_Covered:
```

```
forall <d> in DAYS :
  forall <s> in SHIFTS :
      sum <p> in PROFESSIONALS : ASSIGNED[d,s,p] == 1;

# If a doctor works night shift, they cannot work early or late the next day...
subto Night_Shift_Continuity1:
forall <p> in PROFESSIONALS :
  forall <d> in DAYS with d < 7:
    vif (ASSIGNED[d,"Night",p] >= 1) then
        ASSIGNED[d+1,"Early",p] <=0
    end;

subto Night_Shift_Continuity2:
forall <p> in PROFESSIONALS :
  forall <d> in DAYS with d < 7:
    vif (ASSIGNED[d,"Night",p] >= 1) then
        ASSIGNED[d+1,"Late",p] <=0
    end;
# If doctor works night Sunday, they can't work early or late Monday...
subto periodicity1:
forall <p> in PROFESSIONALS :
    vif (ASSIGNED[7,"Night",p] >= 1) then
        ASSIGNED[1,"Early",p] <=0
    end;

subto periodicity2:
forall <p> in PROFESSIONALS :
    vif (ASSIGNED[7,"Night",p] >= 1) then
        ASSIGNED[1,"Late",p] <=0
end;

# Indicates if doctor works Saturday
subto Sat_Indicator:
forall <p> in PROFESSIONALS :
  sum <s> in SHIFTS : ASSIGNED[6,s,p] - SATURDAY_INDICATOR[p] == 0;

# Indicates if doctor works Sunday
subto Sun_Indicator:
forall <p> in PROFESSIONALS :
  sum <s> in SHIFTS : ASSIGNED[7,s,p] - SUNDAY_INDICATOR[p] == 0;

# If a doctor works Saturday, they must work Sunday and vice versa
subto Weekend_Constraint:
forall <p> in PROFESSIONALS :
  SATURDAY_INDICATOR[p] - SUNDAY_INDICATOR[p] == 0;

# A doctor cannot work a shift they are not available for...
subto Availability:
forall <p> in PROFESSIONALS :
  forall <d> in DAYS :
    forall <s> in SHIFTS : ASSIGNED[d,s,p] <= AVAILABILITY[p,d,s];

# A doctor can only work preferred maximum type of shifts per week
subto Shift_Tolerance:
```

```
forall <p> in PROFESSIONALS :
  forall <s> in SHIFTS :
    sum <d> in DAYS : ASSIGNED[d,s,p] <= MAX_WEEKLY_SHIFTS[p,s];
```