Ausgangslage

In einer Praxis mit mehr Therapeuten als Räume soll die Terminplanung optimiert werden in der Hinsicht möglichst viele Patienten zu betreuen.

Vereinfachungen:

Behandlungen erfolgen auf halbe Stunden.

Es wird für einen Tag (Montag) optimiert.

Zielfunktion:

Die drei Angestellten sollen an möglichst vielen Patienten arbeiten können.

Entscheidungsvariablen:

0800-2030Uhr je 30min, 3 Räume und 3 Angestellte

```
###########################################################################
# Zielfuntkion

einkommen = 0
for slot in range(t):
    for emp in range(e):
        einkommen += (d[slot,emp,r0] + d[slot,emp,r1] + d[slot,emp,r2]) * 50

m.setObjective(einkommen - costRoomChange, GRB.MAXIMIZE)
m.optimize()
```

```
# 08:00 - 20:30 je 30min Blöcke -> 24 Einheiten
t = 24

# 2 Patientenräume, 1 Jogaraum
r = 3
r0 = 0 # Patienten Raum 1
r1 = 1 # Patienten Raum 2
r2 = 2 # Jogaraum

# 3 Angestellte
e = 3
e0 = 0 # 1te Angestellte
e1 = 1 # 2te Angestellte
e2 = 2 # 3te Angestellte

d = m.addVars(t,e,r, vtype=GRB.BINARY)
```

Constrains: Allgemein

Hard Constrain:

- Pro Raum max 1 Angestellte
- Ein Raum kann nur von einer Angestellten belegt sein
- Es können maximal zwei Angestellte gleichzeitig arbeiten

```
################################################################################
# Allgemein

## Hard Constrain: Je Zeiteinheit darf eine Angestellte nur in einem raum sein
for slot in range(t):
    for emp in range(e):
        m.addConstr(d[slot,emp,r0] + d[slot,emp,r1] + d[slot,emp,r2] <= 1)

## Hard Constrain: Je Zeiteinheit darf ein Raum nur einmal belegt sein
for slot in range(t):
    for room in range(r):
        m.addConstr(d[slot,e0,room] + d[slot,e1,room] + d[slot,e2,room] <= 1)

## Hard Constrain: Es können maximal zwei Angestellte gleichzeitig arbeiten
for slot in range(t):
    for room in range(r):
        m.addConstr(gp.quicksum(d[slot, emp, room] for emp in range(e)) <= 2)
```

Soft Constrain:

- Wenn immer möglich sollen Therapeuten den Tag durch im gleichen Raum arbeiten können

```
for emp in range(e):
    for slot in range(t-3):
        # Wenn Therapeut in zwei aufeinanderfolgenden Zeitslots in unt
        if gp.and_(d[slot, emp, r0], d[slot+1, emp, r1]):
            costRoomChange += room_switch_cost
        elif gp.and_(d[slot, emp, r1], d[slot+1, emp, r0]):
            costRoomChange += room_switch_cost
```

Constrains: 1te Angestellte

- Arbeitet zwischen 1500-1700 im Jogaraum sonst in einem Behandlungsraum
- Max 4h ohne Pause
- Min 1h Pause
- Maximale Arbeitszeit: 9.5h
- Mindestens 4h pro Tag Arbeiten

```python
## Hard Constrain: Arbeitet zwischen 1500-1700 in Jogaraum
offset = get_keys_from_value(t_dict, "15:00")
for slot in range(4):
    m.addConstr((d[offset + slot, e0, r2] == 1))

## Hard Constrain Ausserhalb der Jogalektion wird der Jogaraum nicht von der Angestellten belegt
for slot in range(offset):
    m.addConstr((d[slot, e0, r2] == 0))

for slot in range(offset+4, t):
    m.addConstr((d[slot, e0, r2] == 0))

## Hard Constraint: Max 4h ohne Pause (240min)
# Je timeslot prüfen das nicht länger als 8 aufeinander folgende arbeiten getätigt werden
for slot in range(t-7): # Alle timeslot bis auf 3.5h vor ende
    expr = 0
    const = False
    for i in range(9):
        if slot+i < t:
            expr += d[slot+i,e0,r0] + d[slot+i,e0,r1] + d[slot+i,e0,r2]
            constr = True
    if constr:
        m.addConstr(expr <= 8) # Maximale dauer ohne Pause 8 -> 4h

## Hard Constraint: Max Arbeitszeit Total 9:30h (570min) -> 19
m.addConstr(gp.quicksum(d[slot, e0, r0]+d[slot, e0, r1]+d[slot, e0, r2] for slot in range(t)) <= 19)

## Hard Constraint: Minimale Wochenarbeitszeit am Patienten 20h (1200min) -> min 4h pro Tag
m.addConstr(gp.quicksum(d[slot, e0, r0]+d[slot, e0, r1]+d[slot, e0, r2] for slot in range(t)) >= 8)
```

Constrains: 2te & 3te Angestellte

2te Angestellte:

- Arbeitet von 0800-1200 ohne Pause
- Besetzt einen Behandlungsraum

```
for slot in range(8):
    m.addConstr(d[slot, e1, r0] + d[slot, e1, r1] == 1)
# Den rest nicht
for slot in range(9,t):
    m.addConstr(d[slot, e1, r0] + d[slot, e1, r1] == 0)

## Hard Constrain: Darf den Jogaraum nicht belegen
for slot in range(t):
    m.addConstr(d[slot, e1, r2] == 0)
```

3te Angestellte:

- Arbeitet zwischen 0800-2000
- Maximal 4h ohne Pause
- Mindestens 2h Arbeit am Stück ohne Pause
- Belegt einen Behandlungsraum
- Nicht mehr als 8.5h pro Tag

```
## Hard Constrain: Max 4h ohne Pause (240min)
# Je timeslot prüfen das nicht länger als 8 aufeinander folgende arbeiten getätigt we
for slot in range(t-7): # Alle timeslot bis auf 3.5h vor ende
    expr = 0
    const = False
    for i in range(9):
        if slot+i < t:
            expr += d[slot+i,e2,r0] + d[slot+i,e2,r1]
            constr = True
    if constr:
        m.addConstr(expr <= 8) # Maximale dauer ohne Pause 8 -> 4h

## Hard Constrain: Darf den Jogaraum nicht belegen
for slot in range(t):
    m.addConstr(d[slot, e2, r2] == 0)

## Hard Constrain: Die dritte Angestellte sollte nicht mehr als 8.5h Arbeiten
m.addConstr(gp.quicksum(d[slot, e2, r0]+d[slot, e2, r1] for slot in range(t)) <= 17)
```

Zusammenfassung

Ausgang:

Die Total geleisteten Stunden beträgt: **22.5h**

Rechenzeit:

23.1ms

Ungelöst:

- Pausen erkennen
- Wechsel zwischen Pause und Arbeit

TIME	A1: R1	R2	R3	A2: R1	R2	R3	A3: R1	R2	R3
08:00	A1: 0	0	0	A2: 1	0	0	A3: 0	1	0
08:30	A1: 0	0	0	A2: 1	0	0	A3: 0	1	0
09:00	A1: 0	0	0	A2: 1	0	0	A3: 0	1	0
09:30	A1: 1	0	0	A2: 0	1	0	A3: 0	0	0
10:00	A1: 1	0	0	A2: 0	1	0	A3: 0	0	0
10:30	A1: 0	1	0	A2: 1	0	0	A3: 0	0	0
11:00	A1: 1	0	0	A2: 0	1	0	A3: 0	0	0
11:30	A1: 0	1	0	A2: 1	0	0	A3: 0	0	0
12:00	A1: 1	0	0	A2: 0	1	0	A3: 0	0	0
12:30	A1: 0	0	0	A2: 0	0	0	A3: 1	0	0
13:00	A1: 0	1	0	A2: 0	0	0	A3: 1	0	0
13:30	A1: 0	1	0	A2: 0	0	0	A3: 1	0	0
14:00	A1: 0	1	0	A2: 0	0	0	A3: 1	0	0
14:30	A1: 0	1	0	A2: 0	0	0	A3: 1	0	0
15:00	A1: 0	0	1	A2: 0	0	0	A3: 0	1	0
15:30	A1: 0	0	1	A2: 0	0	0	A3: 1	0	0
16:00	A1: 0	0	1	A2: 0	0	0	A3: 1	0	0
16:30	A1: 0	0	1	A2: 0	0	0	A3: 0	0	0
17:00	A1: 0	0	0	A2: 0	0	0	A3: 1	0	0
17:30	A1: 1	0	0	A2: 0	0	0	A3: 0	1	0
18:00	A1: 1	0	0	A2: 0	0	0	A3: 0	1	0
18:30	A1: 1	0	0	A2: 0	0	0	A3: 0	1	0
19:00	A1: 1	0	0	A2: 0	0	0	A3: 0	1	0
19:30	A1: 1	0	0	A2: 0	0	0	A3: 0	1	0