Rutarea și programarea îngrijirii medicale la domiciliu - Implementare

Popovici Marian, Cheptanariu Dorin, Tănase Alexandru-Ionuț

June 7, 2024

1 Introducere

Rutarea și programarea asistenței medicale la domiciliu implică optimizarea rutelor și a programelor pentru a permite furnizorilor de servicii medicale să ofere îngrijiri eficiente pacienților în propriile locuințe. Acest domeniu a câștigat o atenție semnificativă datorită creșterii cererii de servicii de asistență medicală la domiciliu, un fenomen determinat de îmbătrânirea populației, progresele tehnologice în medicină și o tranziție către îngrijirea centrată pe pacient. Optimizarea acestor procese poate aduce îmbunătățiri semnificative în rezultatele pacienților, reducerea costurilor și creșterea productivității furnizorilor de servicii medicale.

Această lucrare explorează metodele și abordările utilizate pentru optimizarea rutelor și programelor de îngrijire medicală la domiciliu. Utilizarea programării cu constrângeri s-a dovedit a fi o metodă eficientă pentru a maximiza numărul de pacienți care pot fi deserviți într-o anumită perioadă de timp, oferind o soluție realistă și fezabilă în contextul resurselor disponibile și al calificărilor necesare.

2 Abordare

În procesul de rezolvare a problemei rutării și programării îngrijirii medicale la domiciliu, am explorat diverse soluții și abordări. Am ales să continuăm cu programarea cu constrângeri, împărțind problema în două părți: problema principală (master) și o subproblemă care gestionează constrângerile specifice programării pacienților.

Metodologia noastră implică definirea unor variabile esențiale pentru modelarea problemei. Aceste variabile includ numărul de pacienți (n), numărul de asistente (m), durata unui serviciu pentru fiecare pacient, timpul total de lucru al fiecărei asistente și calificările necesare pentru fiecare pacient. De asemenea, am utilizat variabile de decizie pentru a determina atribuirea pacienților la asistente și programarea vizitelor.

Obiectivul principal este maximizarea numărului de pacienți acceptați într-o fereastră de timp stabilită, respectând constrângerile de resurse disponibile și calificările necesare. Aceasta abordare asigură o alocare eficientă și realistă a resurselor, îmbunătățind în același timp calitatea și accesibilitatea îngrijirii medicale la domiciliu.

2.1 Variabile

.

1. n - numărul de pacienți
2. m - numărul de asistent
3. j‘ - index-ul pacienților care au nevoie de mai multe asistente la o vizita
4. t - numărul de zile
5. sd_j - durata unui serviciu pentru pacientul j

6. et_j, lt_j - cel mai devreme și cel mai târziu srviciu pentru pacientul j
7. tt_jl - timpul de deplasare de la pacientul j la l
8. U_ik - timpul total de lucru a asistentei i în ziua k
9. $q_j, q2_j, q3_j$ - calificarea asistentei 1, 2, și 3 cerute de pacientul j
10. Q_i - calificarea asistentei i
11. N_j - numărul asistentelor necesare la o vizită a pacientului j
12. f_j - frecvența vizitei pentru un pacient j

2.2 Variabile de decizie

1. $\delta_j = 1$ dacă pacientul j este acceptat, 0 în caz contrar
2. $x_{i,j} = 1$ dacă pacientul j este atribuit asistentei i, 0 în caz contrar
3. $y_{i,j,k} = 1$ dacă pacientul j este atribuit asistentei i în ziua k, 0 în caz contrar
4. $z_{i,j,l,k} = 1$ dacă asistenta i călătorește de la pacientul j la l în ziua k, 0 în caz contrar
5. $s_{i,j,k}$ = ora începerii serviciului pentru asistenta i la pacientul j în ziua k (continuu)

2.3 Obiectiv

Obiectivul este reprezentat de funcția de mai jos și presupune maximizarea numărului de pacienți acceptați în fereastra de timp stabilită.

$max : \sum_{j=1}^{n} \delta_j$

2.4 Constrângeri

1. $\sum_{i=1}^{m} x_{ij} = N_j * \delta_i, \forall j$
2. $\sum_{i,k}\ y_{ijk} = N_j v_y \delta_j, \forall$j
3. $y_{ijk} \leq x_{ij} \forall i, j, k$
4. $x_{ij} = 0 \quad \forall i, j$ with $q_{1j}, q_{2j}, q_{3j} \notin Q_i$
5. $y_{i0k} = 1 \quad \forall i, k$
6. $y_{ijk} + y_{ij,k+\tau} \leq 1 \quad \forall i, j$ with $v_j \in (2, 3)$
   $\forall \tau, k$ with $1 \leq \tau \leq 4 - v_j$, $1 \leq k \leq 5$
7. $\sum_l z_{ijkl} = y_{ijk} \quad \forall\, j,\ i,\ k$
8. $\sum_j z_{ijkl} = y_{ilk} \quad \forall\, l,\ i,\ k$
9. $s_{ijk} \leq M y_{ijk} \quad \forall\, j,\ i,\ k$
10. $s_{ijk} + (sd_j + tt_j) z_{ijkl} \leq s_{ilk} + M(1 - z_{ijkl}) \quad \forall\, j,\ l,\ i,\ k$
11. $s_{ij'k} \leq s_{ijk} + M(2 - y_{ij'k} - y_{ij'k'}) \quad \forall\, i,\ j',\ j',\ k$
12. $s_{ij'k} \geq s_{ijk} - M(2 - y_{ij'k} - y_{ij'k'}) \quad \forall\, i,\ j',\ j',\ k$
13. $z_{i0jk} tt_{0j} + s_{ilk} + (tt_{l0} + sd_l) z_{i0lk} \leq U_{ik} + M(2 - z_{i0lk} - z_{i0jk}) \quad \forall\, i,\ j,\ l,\ k$
14. $s_{ijk} + M(1 - y_{ijk}) \geq et_j \quad \forall\, i,\ j,\ k$
15. $s_{ijk} + sd_j - M(1 - y_{ijk}) \leq lt_j \quad \forall\, i,\ j,\ k$

3 Implementare

3.1 Constrângerea 1-2

Asigură că pacienții acceptați sunt repartizați la numărul de asistente solicitate de pacienți la fiecare vizită.

Matematic, aceasta este exprimată prin două constrângeri care garantează că fiecare pacient este alocat numărului necesar de asistente și că aceștia sunt repartizați în zilele corecte.

$\sum_{i=1}^{m} x_{ij} = N_j \cdot \delta_j, \forall j$ - Aceasta constrângere asigură că pentru fiecare pacient j, numărul de asistente atribuite (x_{ij}) este egal cu numărul necesar de asistente (N_j) dacă pacientul este acceptat ($\delta_j = 1$).

$\sum_{i,k} y_{ijk} = N_j \cdot v_j \cdot \delta_j, \forall j$ - Aceasta înseamnă că pentru fiecare pacient j, suma alocărilor de asistente în toate zilele (y_{ijk}) trebuie să fie egală cu numărul necesar de asistente (N_j) înmulțit cu frecvența vizitelor (v_j), doar dacă pacientul este acceptat ($\delta_j = 1$).

```
for j in range(self.number_of_patients):
    self.model.addConstr(gb.quicksum(self.decision_variables["x"][i, j] for i in range(self.number_of_nurses)) == \
                         self.decision_variables["d"][j] * self.number_of_nurses_required[j+1])
self.model.update()

for j in range(self.number_of_patients):
    self.model.addConstr(gb.quicksum(gb.quicksum(self.decision_variables["y"][i, j + 1, k] for k in range(self.number_of_days)) \
                                     for i in range(self.number_of_nurses) ) == \
                                         self.frequency[j + 1] * self.decision_variables["d"][j] * self.number_of_nurses_required[j + 1])
self.model.update()
```

Figure 1: Constrângerea 1-2

3.2 Constrângerea 3

Garantează că pacienții alocați pot fi programați pentru o zi la asistentele respective.

Aceasta înseamnă că există o compatibilitate între ziua de vizită și disponibilitatea asistentei.

$y_{ijk} \leq x_{ij}, \forall i, j, k$ - Aceasta constrângere asigură că dacă un pacient j este alocat unei asistente i într-o zi k, atunci pacientul trebuie să fie alocat acelei asistente (adică $x_{ij} = 1$).

```
for i in range(self.number_of_nurses):
    for j in range(self.number_of_patients):
        for k in range(self.number_of_days):
            self.model.addConstr(self.decision_variables["y"][i, j + 1, k] <= self.decision_variables["x"][i, j])
self.model.update()
```

Figure 2: Constrângerea 3

3.3 Constrângerea 4

Asigură că cerințele de calificare sunt îndeplinite.

Aceasta verifică dacă asistentele alocate unui pacient au calificările necesare pentru a oferi serviciile medicale necesare.

$x_{ij} = 0, \forall i, j$ with $q_{1j}, q_{2j}, q_{3j} \notin Q_i$ - Aceasta verifică dacă calificările necesare pentru pacientul j (q_{1j}, q_{2j}, q_{3j}) sunt prezente în setul de calificări al asistentei i (Q_i).

```
for i in range(self.number_of_nurses):
    for j in range(self.number_of_patients):
        if self.qualification_first_nurse[j+1] != self.qualification[i] \
                and self.qualification_second_nurse[j+1] != self.qualification[i] \
                  and self.qualification_third_nurse[j+1] != self.qualification[i]:
            self.model.addConstr(self.decision_variables["x"][i, j] == 0)
self.model.update()
```

Figure 3: Constrângerea 4

3.4 Constrângerea 5

Garantează că fiecare asistentă vizitează depozitul în fiecare zi.

Aceasta este o constrângere logistică, necesară pentru gestionarea resurselor și aprovizionarea cu echipamente medicale.

$y_{i0k} = 1, \forall i, k$ - Garantează că fiecare asistentă vizitează depozitul (notat 0) în fiecare zi.

```
for i in range(self.number_of_nurses):
    for k in range(self.number_of_days):
        self.model.addConstr(self.decision_variables["y"][i, 0, k] == 1)
self.model.update()
```

Figure 4: Constrângerea 5

3.5 Constrângerea 6

Asigură că numărul de zile dintre vizitele multiple pentru același pacient într-o săptămână este respectat. Aceasta ajută la menținerea unui program regulat și coerent de vizite.

$y_{ijk} + y_{ij,k+\tau} \leq 1, \forall i, j$ with $v_j \in (2, 3)$ și $\forall \tau, k$ with $1 \leq \tau \leq 4 - v_j, 1 \leq k \leq 5$ - Asigură că numărul de zile dintre vizitele multiple pentru același pacient într-o săptămână este respectat, în funcție de frecvența vizitelor necesare (v_j).

Explicație:

- y_{ijk}: Variabilă binară care indică dacă asistenta i vizitează pacientul j în ziua k.
- v_j: Frecvența vizitelor pentru pacientul j în timpul săptămânii (de exemplu, 2 sau 3 vizite pe săptămână).

Detalii ale constrângerii:

1. **Frecvența vizitelor (v_j):**
   - Dacă un pacient are nevoie de 2 vizite pe săptămână ($v_j = 2$), vizitele trebuie să fie distribuite astfel încât să nu fie în zile consecutive.
   - Dacă un pacient are nevoie de 3 vizite pe săptămână ($v_j = 3$), vizitele trebuie să fie distribuite astfel încât să nu fie mai mult de o zi consecutivă liberă între vizite.
2. **Intervalul dintre vizite (τ):**
   - τ reprezintă numărul de zile dintre vizitele succesive.
   - Pentru $v_j = 2$, τ poate fi între 1 și 3 zile.
   - Pentru $v_j = 3$, τ poate fi între 1 și 2 zile.
3. **Constrângerea propriu-zisă:**

- $y_{ijk} + y_{ij,k+\tau} \leq 1$: Aceasta înseamnă că, dacă asistenta i vizitează pacientul j în ziua k ($y_{ijk} = 1$), nu ar trebui să mai fie o altă vizită în ziua $k+\tau$ ($y_{ij,k+\tau} = 0$) pentru τ specificat.
- Aceasta previne programarea vizitelor prea apropiate una de cealaltă și respectă cerințele de frecvență ale pacientului.

Exemplu concret: Să presupunem că un pacient j are nevoie de 2 vizite pe săptămână ($v_j = 2$):

- Dacă asistenta i vizitează pacientul j în ziua 1 ($y_{ij1} = 1$), atunci y_{ij2}, y_{ij3} și y_{ij4} trebuie să fie 0, asigurând astfel un interval adecvat de cel puțin 1 zi liberă între vizite.
- Dacă prima vizită este în ziua 2 ($y_{ij2} = 1$), atunci a doua vizită poate fi programată în ziua 4 sau 5 (y_{ij4} sau y_{ij5}).

```
for i in range(self.number_of_nurses):
    for j in range(self.number_of_patients):
        for k in range(self.number_of_days):
            for tou in range(1, (4 - (self.frequency[j+1]) + 1)):
                if (k + tou) <= 4:
                    self.model.addConstr(self.decision_variables["y"][i, j + 1, k] + \
                                         self.decision_variables["y"][i, j + 1, k + tou] <= 1)
self.model.update()
```

Figure 5: Constrângerea 6

3.6 Constrângerea 7-8

Garantează că sosirea și plecarea de la fiecare pacient pentru o asistentă într-o zi este conformă cu decizia de atribuire a pacientului.

Aceasta se asigură că programul asistentei este sincronizat cu atribuirea pacienților.

$\sum_l z_{ijkl} = y_{ijk}, \forall j, i, k$ - Asigură că pentru fiecare pacient j, asistenta i în ziua k călătorește de la j la un alt pacient l dacă pacientul j este programat pentru ziua respectivă.

$\sum_j z_{ijkl} = y_{ilk}, \forall l, i, k$ - Similar, asigură că asistenta i în ziua k călătorește către pacientul l de la un alt pacient j dacă pacientul l este programat pentru ziua respectivă.

```
for i in range(self.number_of_nurses):
    for j in range(self.number_of_patients + 1):
        for k in range(self.number_of_days):

            self.model.addConstr(gb.quicksum(self.decision_variables["z"][i, j, l, k] for l in range(self.number_of_patients + 1)) == \
                                 self.decision_variables["y"][i, j, k])
            self.model.addConstr(gb.quicksum(self.decision_variables["z"][i, l, j, k] for l in range(self.number_of_patients + 1)) == \
                                 self.decision_variables["y"][i, j, k])
```

Figure 6: Constrângerea 7-8

3.7 Constrângerea 9-10

Determină ora de începere a serviciului pentru fiecare pacient într-o zi. Aceasta ajută la stabilirea unui program precis și predictibil pentru fiecare pacient.

$s_{ijk} \leq M \cdot y_{ijk}, \forall j, i, k$ - Asigură că ora de începere a serviciului pentru pacientul j în ziua k este definită doar dacă pacientul este programat pentru acea zi (unde M este o constantă mare).

$s_{ijk} + (sd_j + tt_{jl}) \cdot z_{ijkl} \leq s_{ilk} + M \cdot (1 - z_{ijkl}), \forall j, l, i, k$ - Asigură că timpul de începere a serviciului pentru următorul pacient l este după timpul necesar pentru a finaliza serviciul la pacientul j și a călători de la j la l.

```
def add_sub_constraints(self):
    for i in range(self.number_of_nurses):
        for j in range(self.number_of_patients + 1):
            for k in range(self.number_of_days):

                if j <= self.number_of_patients - 1:
                    self.model.addConstr(self.decision_variables["s"][i, j + 1, k] + \
                                         self.bigM * (1 - self.decision_variables["y"][i, j + 1, k]) >= \
                                           self.earliest_service_start_time[j + 1])
                    self.model.addConstr(self.decision_variables["s"][i, j + 1, k] + \
                                        self.service_duration[j + 1] - self.bigM * (1 - self.decision_variables["y"][i, j + 1, k]) <= \
                                            self.latest_service_start_time[j + 1])
```

Figure 7: Constrângerea 9-10

3.8 Constrângerea 11-12

Asigură că ora de începere a serviciului pentru mai multe asistente care lucrează împreună este aceeași. Aceasta este esențială pentru coordonarea echipei de asistente care oferă servicii simultan.

$s_{ij'k} \leq s_{ijk} + M \cdot (2 - y_{ij'k} - y_{ij'k'}), \forall i, j', j, k$ - Asigură că ora de începere a serviciului pentru mai multe asistente care lucrează împreună este aceeași. M este o constantă mare care face ca această constrângere să fie relevantă doar atunci când ambele asistente sunt programate în același timp.

$s_{ij'k} \geq s_{ijk} - M \cdot (2 - y_{ij'k} - y_{ij'k'}), \forall i, j', j, k$ - Similar cu constrângerea anterioară, dar pentru limite inferioare.

```
    for i in range(self.number_of_nurses):
        for j in range(self.number_of_patients + 1):
            for k in range(self.number_of_days):

                if j <= self.number_of_patients - 1:
                    for p in range(self.number_of_nurses):
                        if self.number_of_nurses_required[j + 1] != 1 and i != p:
                            self.model.addConstr(self.decision_variables["s"][i, j + 1, k] +\
                                                 self.bigM * (2 - self.decision_variables["y"][i, j + 1, k] -
                                                              self.decision_variables["y"][p, j + 1, k]) >= \
                                                                  self.decision_variables["s"][p, j + 1, k])
                            self.model.addConstr(self.decision_variables["s"][i, j + 1, k] <=\
                                                 self.decision_variables["s"][p, j + 1, k] + self.bigM *\
                                                     (2 - self.decision_variables["y"][i, j + 1, k] -\
                                                       self.decision_variables["y"][p, j + 1, k]))
    self.model.update()
```

Figure 8: Constrângerea 11-12

3.9 Constrângerea 13

Garantează că timpul total de lucru al asistentelor într-o zi nu depășește programul lor de lucru. Aceasta protejează asistentele de suprasolicitare și respectă reglementările de muncă.

$z_{i0jk} \cdot tt_{0j} + s_{ilk} + (tt_{l0} + sd_l) \cdot z_{i0lk} \leq U_{ik} + M \cdot (2 - z_{i0lk} - z_{i0jk}), \forall i, j, l, k$

Această constrângere asigură că timpul total de lucru al unei asistente i într-o zi k nu depășește durata maximă de lucru permisă (U_i). Termenii specifici ai formulei sunt:

- $z_{i0jk} \cdot tt_{0j}$: Timpul de călătorie de la depozit (notat 0) la pacientul j.
- s_{ijk}: Timpul de începere a serviciului la pacientul j.
- $sd_j \cdot y_{ijk}$: Durata serviciului efectuat la pacientul j, luată în considerare doar dacă pacientul este programat ($y_{ijk} = 1$).
- $z_{ij0k} \cdot tt_{j0}$: Timpul de călătorie de la pacientul j înapoi la depozit.

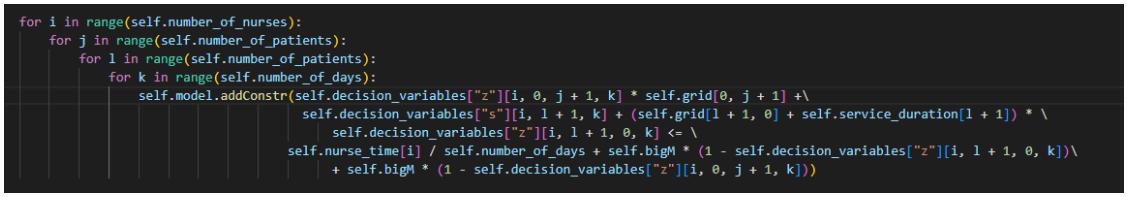

Figure 9: Constrângerea 13

3.10 Constrângerea 14-15

Asigură că fereastra de vizită a fiecărui pacient este respectată. Aceasta înseamnă că serviciile medicale sunt furnizate în intervalul orar promis pacienților.

$s_{ijk} + M \cdot (1 - y_{ijk}) \geq e_t j, \forall i, j, k$ - Asigură că serviciile sunt furnizate după cel mai devreme timp posibil pentru pacientul j $(e_t j)$.

$s_{ijk} + sd_j - M \cdot (1 - y_{ijk}) \leq l_t j, \forall i, j, k$ - Asigură că serviciile sunt furnizate înainte de cel mai târziu timp posibil pentru pacientul j $(l_t j)$.

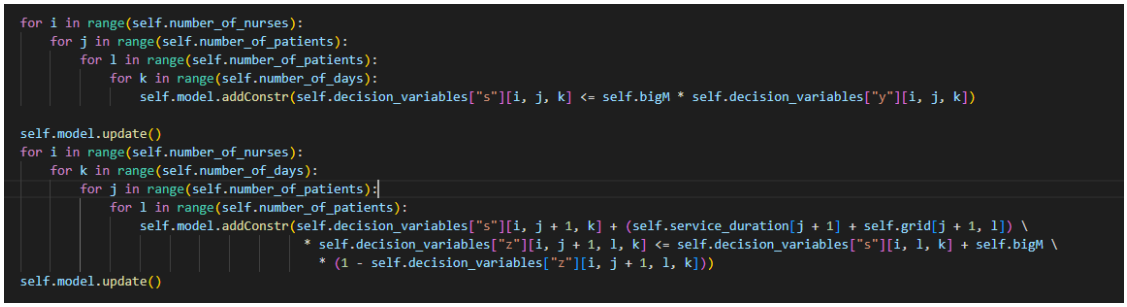

Figure 10: Constrângerea 14-15

4 Rezultate

Seturile de date pe care le-am folosit erau formate din 30, 35, 50 și 100 de pacienți.

Dacă foloseam doar primele 6 constrângeri obțineam o rata de asignare de 100%. Rezultatele după adăugarea celorlalte constrângeri sunt:

Sample	Nr. Pacienți	Nr. Asistente	Asignați	Time
1	30	3	17	42s
2	35	4	32	200s (timeout)
3	50	8	42	566s (timeout)
4	100	15	89	2887s (timeout)

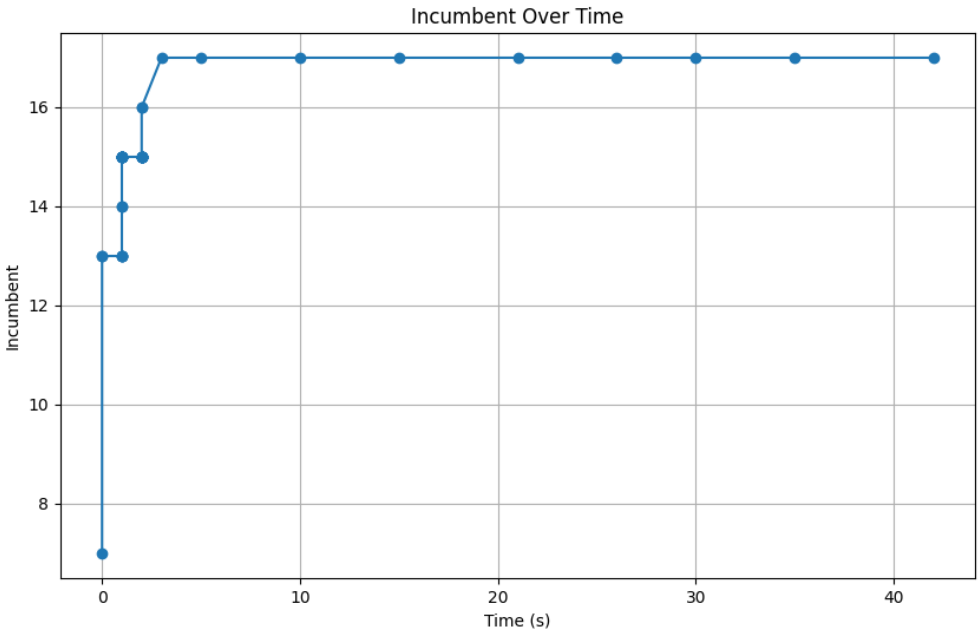

Figure 11: Numărul de pacienți acceptați pentru setul de date de 30.

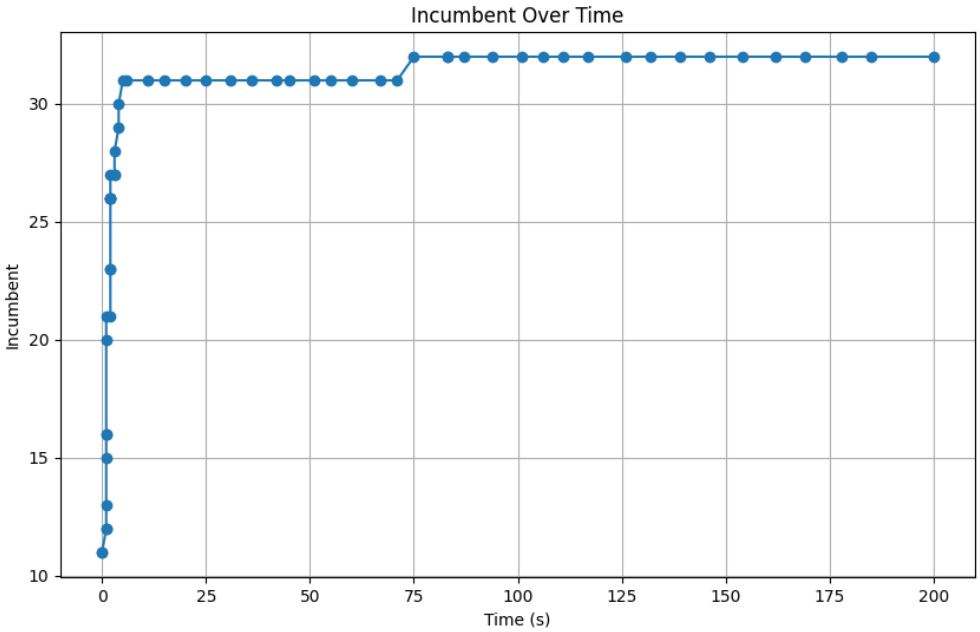

Figure 12: Numărul de pacienți acceptați pentru setul de date de 35.

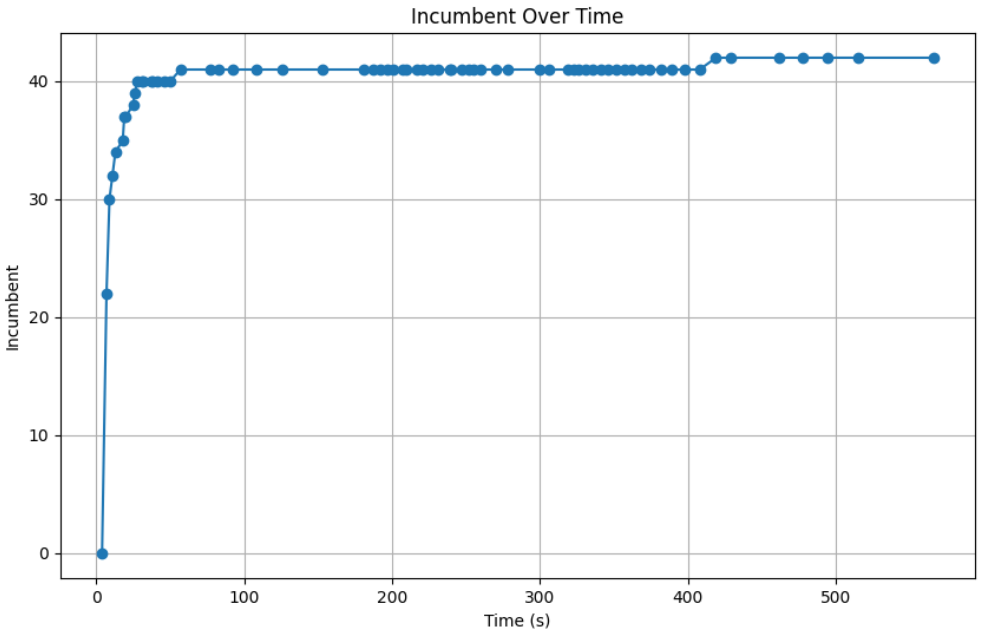

Figure 13: Numărul de pacienți acceptați pentru setul de date de 50.

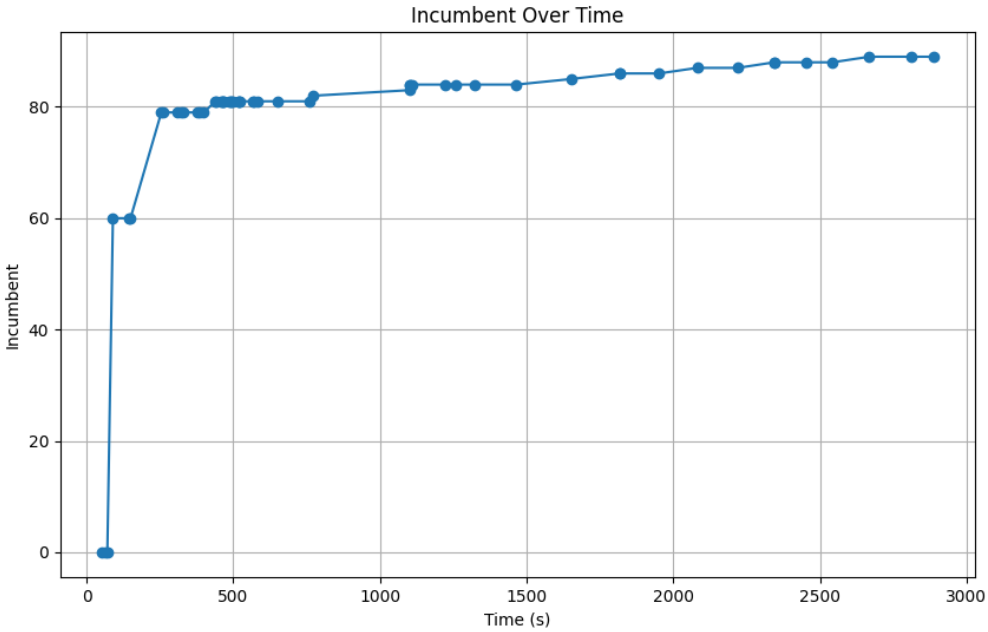

Figure 14: Numărul de pacienți acceptați pentru setul de date de 100.

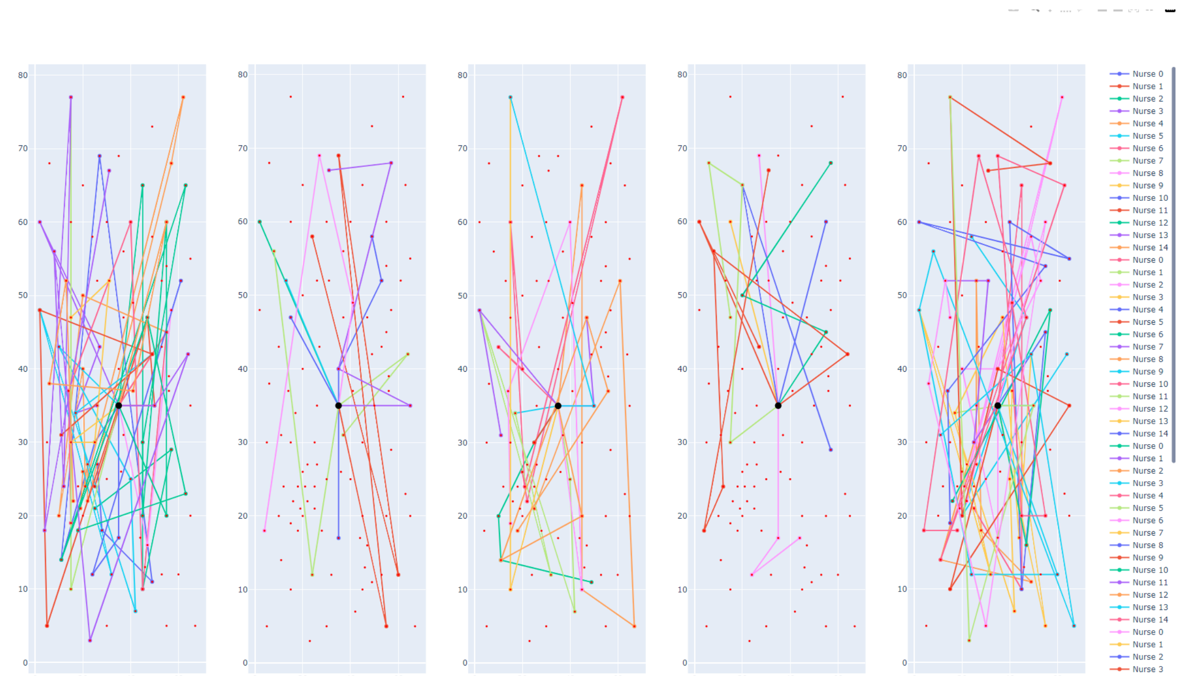

Figure 15: Drumul parcurs de fiecare asistentă în fiecare zi. Punctul din centru reprezintă depozitul. Setul de date folosit este cel cu 100 de pacienți.

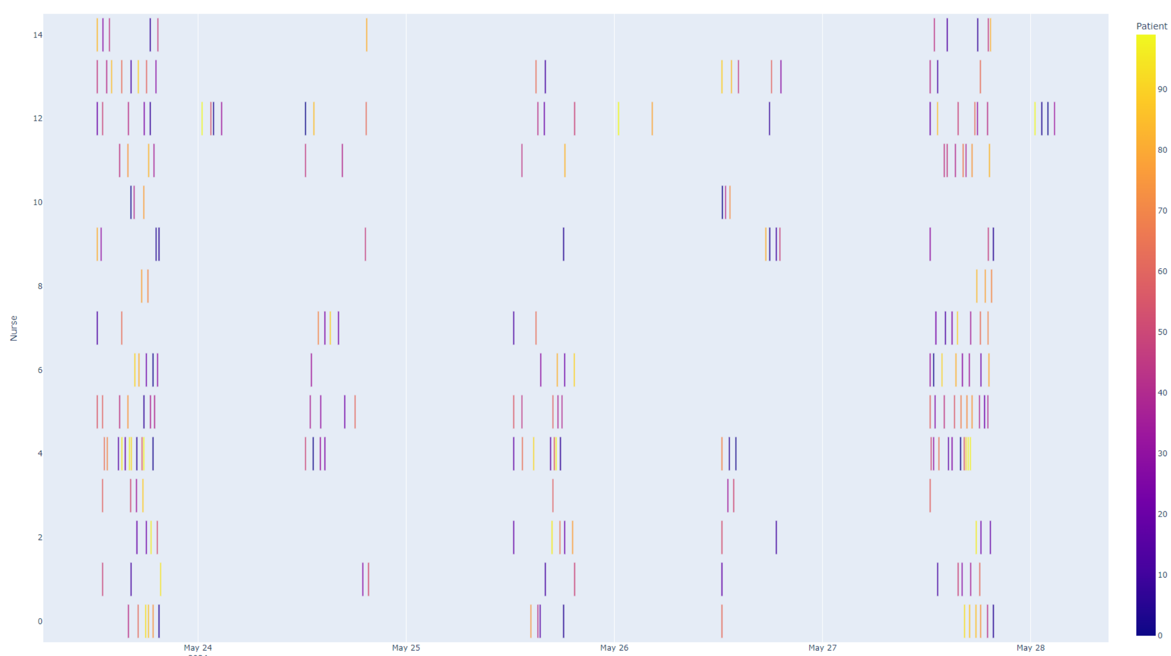

Figure 16: Repartizarea pacienților pe zile pentru fiecare asistentă. Setul de date folosit este cel cu 100 de pacienți.

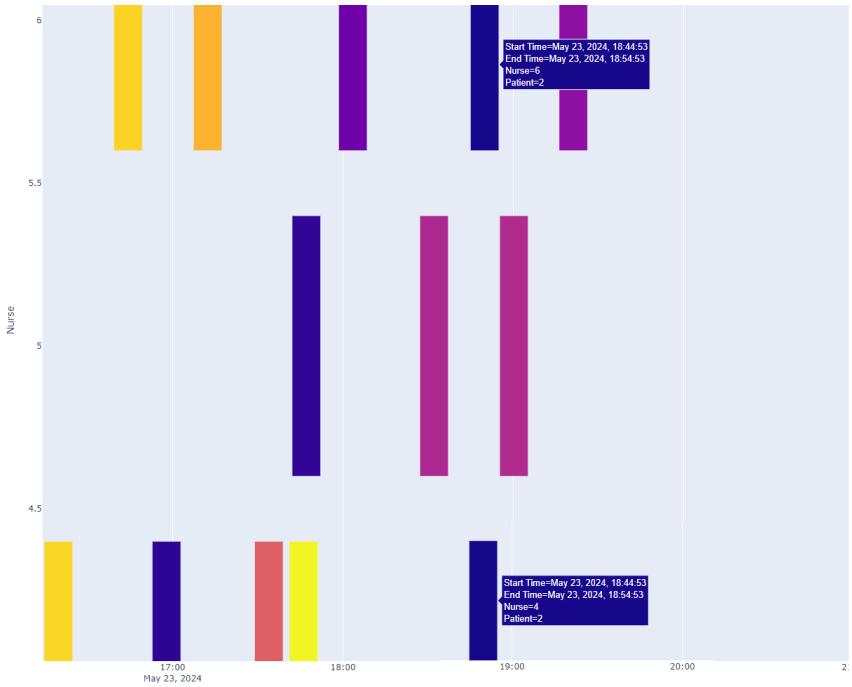

Figure 17: în această imagine se poate observa că pacientul 2 este atribuit atât asistentei 4, cât și asistentei 6 în același interval de timp.

5 Concluzie

În concluzie, lucrarea despre rutarea și programarea îngrijirii medicale la domiciliu a demonstrat că utilizarea programării cu constrângeri este eficientă pentru maximizarea numărului de pacienți acceptați în fereastra de timp stabilită.

Studiul a abordat problema prin împărțirea acesteia într-o problemă master și o subproblemă pentru constrângerile de programare a pacienților. Rezultatele au arătat că, deși inițial obținem o rată de asignare de 100% folosind primele șase constrângeri, adăugarea celorlalte constrângeri a redus numărul pacienților asignați, însă a permis o programare mai realistă și fezabilă în termeni de resurse disponibile și calificări necesare