ENIGMA 2023

An AI approach to Large Scale Medical Appointment Rescheduling using ASP

KR in digital healthcare

Stavros Kanias
University of Patras, Greece

Introduction

Medical appointments reamaining unclaimed after a cancelation have a significant cost in many aspects. ASP can help with that but further software integration steps are essential. Using Python, Clingo and SQL a solution to the problem was developed. Results seem promising.

What to expect

01 The problem

Why fair and optimized rescheduling of medical appointments matters

02 Technical constraints

Without the ability to provide an ASP solver with large scale data from a database no real-world solution can be developed

03 The AI solution

Description and evaluation of the proposed ASP application

04 From concept to reality

Implementing the proposed solution in a real-world healthcare system

01

The problem

It’s not **that** bad, right?

"An estimated 41% of U.S. adults had delayed medical care including urgent or emergency care (12%)."

—MMWR 2020

51.7% below 50k

Delayed treatments due to unavailable appointments mostly concern low-income households

26 days

Average waiting time for an appointment in the US for 2022

$150 billion

The monetary loss for the US healthcare system due to unclaimed medical appointments

An example

Conventional vs AI approach

The conventional approach

Maximize the individual benefit

Initial state

Timeslots

Monday	Patient - Score
9-10	Nikos – 90
	Giorgos – 87
	Kostas – 86
Tuesday	**Patient - Score**
13-14	Giorgos - 72
	Ioannis - 60
	Maria - 55
Thursday	**Patient - Score**
11-12	Ioannis - 45
	Maria - 40
Friday	**Patient - Score**
15-16	Kostas - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	Nikos				
10-11					
11-12				Ioannis	
12-13					
13-14		Giorgos			
14-15					
15-16					Kostas
16-17					

Community Benefit = 90 + 72 + 45 + 71 = 278

Nikos cancels his appointment

Timeslots

Monday	Patient - Score
9-10	~~Nikos – 90~~
	Giorgos – 87
	Kostas – 86
Tuesday	**Patient - Score**
13-14	Giorgos - 72
	Ioannis - 60
	Maria - 55
Thursday	**Patient - Score**
11-12	Ioannis - 45
	Maria - 40
Friday	**Patient - Score**
15-16	Kostas - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	~~Nikos~~				
10-11					
11-12				Ioannis	
12-13					
13-14		Giorgos			
14-15					
15-16					Kostas
16-17					

Community Benefit = ?

New timeslots become available for Giorgos and Ioannis

Timeslots

Monday	Patient - Score
9-10	Giorgos – 87
	Kostas – 86

Tuesday	Patient - Score
13-14	Ioannis - 60
	Maria - 55

Thursday	Patient - Score
11-12	~~Ioannis - 45~~
	Maria - 40

Friday	Patient - Score
15-16	Kostas - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	Giorgos				
10-11					
11-12				?	
12-13					
13-14		Ioannis			
14-15					
15-16					Kostas
16-17					

Community Benefit = ?

Now Maria can receive an appointment

Timeslots

Monday	Patient - Score
9-10	Giorgos – 87
	Kostas – 86

Tuesday	Patient - Score
13-14	Ioannis - 60
	Maria - 55

Thursday	Patient - Score
11-12	Maria - 40

Friday	Patient - Score
15-16	Kostas - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	Giorgos				
10-11					
11-12				Maria	
12-13					
13-14		Ioannis			
14-15					
15-16					Kostas
16-17					

Community Benefit = 87 + 60 + 40 + 71 = 258

The AI approach

Maximize the community benefit

The same initial state

Timeslots

Monday	Patient - Score
9-10	Nikos – 90
	Giorgos – 87
	Kostas – 86
Tuesday	**Patient - Score**
13-14	Giorgos - 72
	Ioannis - 60
	Maria - 55
Thursday	**Patient - Score**
11-12	Ioannis - 45
	Maria - 40
Friday	**Patient - Score**
15-16	Kostas - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	Nikos				
10-11					
11-12				Ioannis	
12-13					
13-14		Giorgos			
14-15					
15-16					Kostas
16-17					

Common Benefit = 90 + 70 + 45 + 75 = 280

Again, Nikos cancels his appointment but ...

Timeslots

Monday	Patient - Score
9-10	~~Nikos – 90~~
	Giorgos – 87
	Kostas – 86

Tuesday	Patient - Score
13-14	Giorgos - 72
	Ioannis - 60
	Maria - 55

Thursday	Patient - Score
11-12	Ioannis - 45
	Maria - 40

Friday	Patient - Score
11-12	Kostas - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	~~Nikos~~				
10-11					
11-12				Ioannis	
12-13					
13-14		Giorgos			
14-15					
15-16					Kostas
16-17					

Common Benefit = ?

This time the system chooses Kostas

Timeslots

Monday	Patient - Score
9-10	Kostas – 86
	Giorgos – 87

Tuesday	Patient - Score
13-14	Giorgos - 72
	Ioannis - 60
	Maria - 55

Thursday	Patient - Score
11-12	Ioannis - 45
	Maria - 40

Friday	Patient - Score
15-16	Kos - 71
	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	Kostas				
10-11					
11-12				Ioannis	
12-13					
13-14		Giorgos			
14-15					
15-16					?
16-17					

Common Benefit = ?

And a greater community benefit is provided

Timeslots

Monday	Patient - Score
9-10	Kostas – 86
	Giorgos – 87

Tuesday	Patient - Score
13-14	Giorgos - 72
	Ioannis - 60
	Maria - 55

Thursday	Patient - Score
11-12	Ioannis - 45
	Maria - 40

Friday	Patient - Score
15-16	Despoina - 70

Appointment Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9-10	Kostas				
10-11					
11-12				Ioannis	
12-13					
13-14		Giorgos			
14-15					
15-16					Despoina
16-17					

Common Benefit = 80 + 70 + 45 + 70 = 273 > 258

Solution requirements

A real-world digital solution to this problem should be characterized by the following features:

- High performance (large sets of data in healthcare)
- Integration with existing digital healthcare software
- Fairness and optimality
- Pleasant user experience
- Scalability
- Ability to express complex scenarios

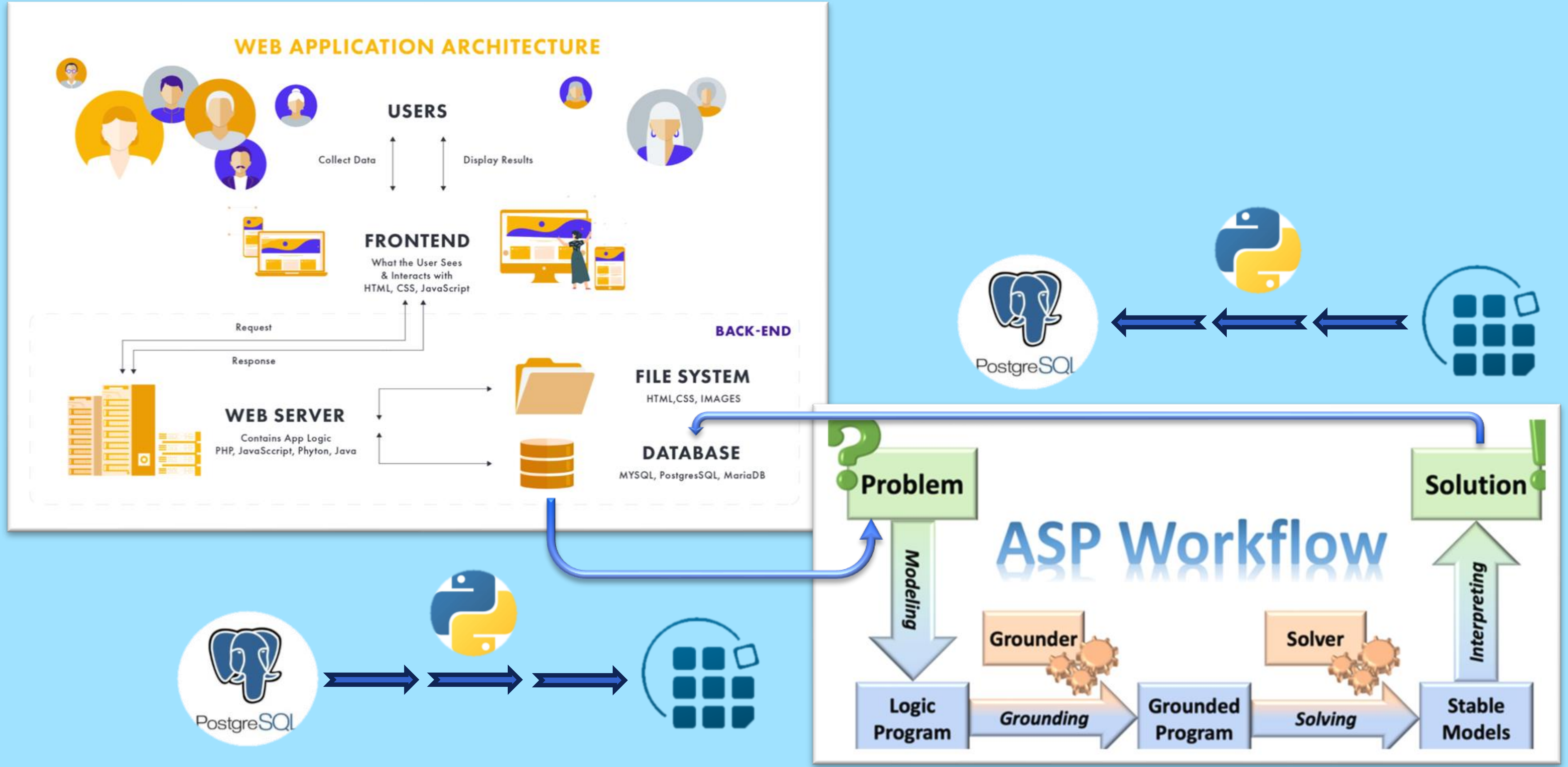
- Can ASP help?

02

Technical constraints

Yes, but first...

ASP must be integrated with standard software



The three points of data flow

PostgreSQL DB

By being the software component where data is stored, a solid database is indeed the base of every modern application

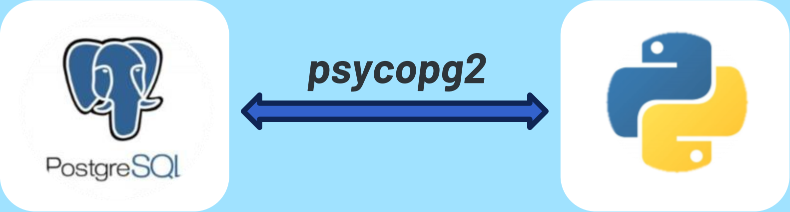
Python

The Python programming language with its clorm, clingo and psycopg2 modules can be used to enable the communication between the batabase and the ASP solver

Clingo Solver

The clingo solver created by Potassco offers outstanding performance and rule formulation capabilities

Python and postgreSQL



DataModel
dbName:str dbKey:str schema:dict con:connection object (psycopg2)
connect() close() isEmpty():boolean create(tablename,tabledict):creation query as str createTables() loadTestData() getTables() getAttributes(table) dropTables() dropData() executeSQL(query,values,show,txtFile,fetch):list values(val):list conditions(cond,sep):str select(table,attributes,conditions,joins):list insert(table,val):bool update(table,new,conditions,joins):bool delete(table,conditions,joins):bool

Python and clingo

KnowledgeBase

name:str
schema:dict
kb:factbase object(clorm)
type2field:dict
splitPreds:dict
mergedPreds:dict
foreignPaths:dict

showPredContent(p)
isPrimary(entity,attribute)
getPrimary(entity)
getPrimaryData(entity,attributes,data)
getSplitPredName(schemaName)
createPrimaryPredicate(entity, attributes, predicates)
createSplitPreds()
createSplitPred(name,attributes,primary)
createMergedPreds()
createMergedPred(entity,attributes)
split(mergedPred)
merge(splitPreds)
isForeign(entity,attribute)
getForeign(e1,e2)
clear2dDict(dict)
getForeignPath(jent,entity,attribute)
getForeignPaths()
in2out(inPaths)
getInwardForeigns(entity,attribute)
getOutwardForeigns(entity)
getAllDeps(ent,cond)
getDepChain(e1,e2)
getJoinPreds(e1,e2)
getJoinEntities(entities,conditions)
getJoins()
getJoinedConditions(entity, joins)
getDbConditions()
bind2db(dbInfo)
db2kb()
getCompExp(entity,conditions)
select(entities,conditions,order,pOut,getQuery)
insert(entity,data,toDb)
update(upd,conditions,cascade,toDb)
delete(entities,conditions,getData,cascade,fromDb)
extract(ent,split,cond,order)
run(asp,outPreds,searchduration,show,limit,subKB,subKBCond,merged,symbOut

From record to predicate

Two types of encoding were tested for the purpose of translating an SQL record to an ASP predicate

	id [PK] integer	patient_id text	timeslot_id integer	preference integer	score integer	status integer
1	2	26057784758	1251	1	37	0

1

Split

```
request(2)
patient_id(2,26057784758)
timeslot_id(2,1251)
preference(2,1)
score(2,37)
status(2,0)
```

Used for solving

+

- Code resembles natural language
- Easier to express more complex rules

-

- More predicates in KB
- Much higher complexity of SQL type transactions in the KB

2

Merged

```
request(2,26057784758,1251,1,37)
```

Used for KBMS

+

- Smaller KB
- Much easier to use with SQL type transactions in the KB

-

- Less readable code
- Can cause a slight drop in performance due to more complex rule expression and the many anonymous variables

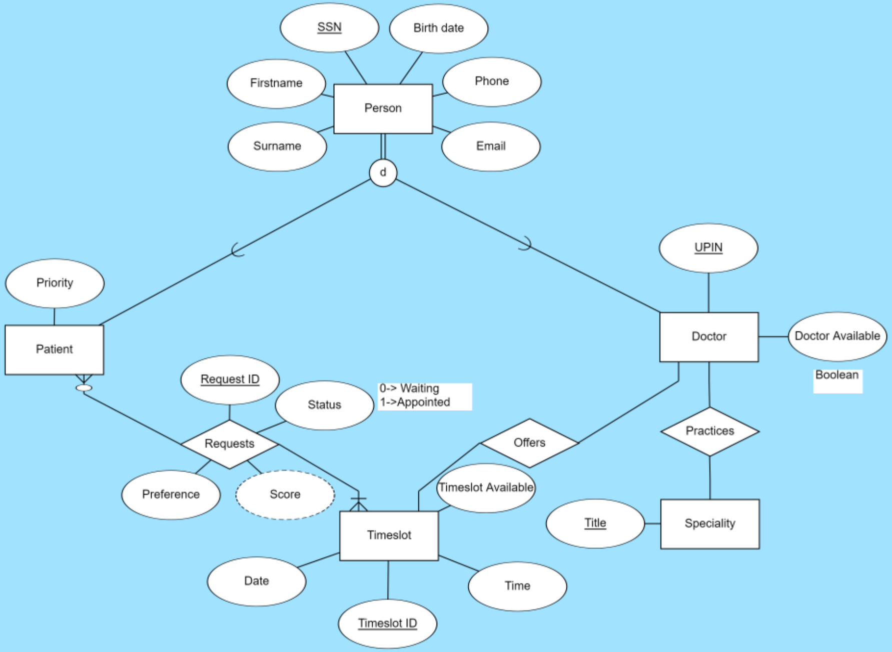
03

The AI solution

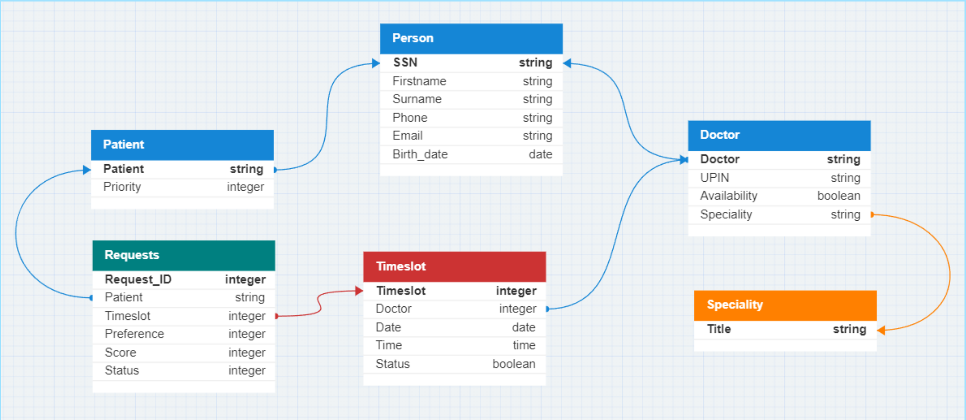
Solving the problem with PostgreSQL, Python and ASP

Database design

PostgreSQL, ideal for modern web applications



Entity Relation Diagram (ERD)



Database schema

Data fabrication

The DataFabricator class was developed for testing purposes

- Dynamically create pseudo realistic healthcare systems
- Assumes 2.5 appointments per patient
- Assumes an 9 to 5, five days a week schedule for doctors
- Schedules the appointments by choosing the highest (best case) or the lowest (worst case) score

DataFabricator
schema:dict tSpan:int quantities:dict tAvailability:bool
write2csv(entity,entityDiction,listOfDicts) handleIntPrimaries(primaries,primaryKey,tempDict,attribute) loadNonForeign() loadForeign(entityDiction, attribute) chooseForeign(foreign_lists, attribute,remove, getIndex) calculateScore(request,patientInfo) **handleStatus(requests, patientInfo)** **fabricatePerson(quantity)** **fabricateSpecialty(quantity)** **fabricateDoctor(quantity)** **fabricatePatient(quantity)** **fabricateTimeslot(quantity)** **fabricateRequest(quantity)** fabricate(entity)

The clingo code

Appointment states and auxiliary predicates

Granted

The appointment is currently appointed to a patient

Claimed

The appointment is granted and is the first choice of a patient

Appointed

The patient has booked an appointment

Best

The appointment is the first choice of a patient

OnlyOption

The appointment is a patient's only option

SingleRequest

This request is the only one for a specific appointment

BestSingleRequest

This is the highest scoring of the patient's single requests

The clingo code

Set generation rules

01

0 {grant(R)} 1 :- granted(R), not best(R)

Include the **granted** but **not best** requests as a patient can retain their currently owned appointment if it appears in the answer set that maximizes the common benefit.

02

0 {grant(R)} 1 :- patient_id(R, P), score(R, S), not granted(R), patient_id(X,P), score(X, SX), granted(X), S > SX, R != X.

Include the **not granted** requests of a patient **already having a granted request**, that **present a higher score** than the one **currently appointed** to the patient. This enables a patient to receive a request with higher priority if it helps to maximize the common benefit.

03

0 {grant(R)} 1 :- patient_id(R, P), not appointed(P).

Include all the requests that belong to a patient who **has not been appointed** a timeslot if this leads to the optimal answer set.

The clingo code

Logic constraints

01

```
:- grant(R1), timeslot_id(R1, T), grant(R2), timeslot_id(R2, T), R1 != R2.
```

Each **timeslot** can be appointed to **only one patient**.

02

```
:- grant(R1), patient_id(R1, P), timeslot_id(R1, T1), doctor_id(T1, D1), specialty_title(D1, S), grant(R2), patient_id(R2, P),
timeslot_id(R2, T2), doctor_id(T2,D2), specialty_title(D2, S), R1 != R2.
```

Each **patient** can only **receive one timeslot** (from a specific specialty if the general scope is used).

The clingo code

Fairness constraints

01

```
:- granted(R), patient_id(R, P), not claimed(R), 0 { grant(X) : patient_id(X, P) } 0.
```

If a **patient had an appointment** in the **previous schedule** a **timeslot must** also **be granted** to that patient **after the rescheduling**.

02

```
:- timeslot_id(R, T), patient_id(R, P), score(R,S), grant(R), not granted(R), appointed(P), timeslot_id(X, T), score(X, SX), onlyOption(X), S < SX, R != X.
```

If a request is a patient's **only option**, it **cannot be dismissed** for the sake of a lower scoring request even if it leads to a chain reaction that maximizes common benefit. If we don't apply this constraint a patient with only one request will most probably never receive an appointment.

The clingo code

Set exclusion to increase performance

01

```
:- grant(R), timeslot_id(R,T), claimed(X), timeslot_id(X,T), R != X.
```

All requests that **claim** an **already claimed** timeslot **will not be taken into consideration**.

02

```
:- timeslot_id(R, T), patient_id(R, P), score(R,S), grant(R), not granted(R), appointed(P), timeslot_id(X, T), score(X, SX),
onlyOption(X), S < SX, R != X.
```

All the **single** requests that are **not the** patient's **best single** request **will not be taken into consideration**.

The clingo code

Automatic assignments to increase the number of served patients

01

```
:- patient_id(R, P), score(R, S), bestSingleRequest(R), patient_id(X, P), score(X, SX), granted(X),
R != X, S > SX, patient_id(Y, P), Y != R, grant(Y), not grant(R).
```

If a request is a patient's **best single** request and it has a better score than the one already granted to the patient **it will automatically be assigned to the patient** blocking all the other requests made by the patient.

02

```
:- patient_id(R, P), bestSingleRequest(R), not appointed(P), patient_id(Y, P), Y != R, grant(Y), not grant(R).
```

If a **single request** is attributed to an **unappointed patient**, the system should **grant it automatically** and block all the other requests made by the patient.

Evaluation

Execution times for 3 different datasets

Timeslots	Patients	Requests	Execution Time (s)	
			Best	Worst
400	500	1000	0.06	2.79
960	1200	2400	0.18	152.07
2000	2500	5000	421.4	>1800

Split encoding

Timeslots	Patients	Requests	Execution Time (s)	
			Best	Worst
400	500	1000	0.09	4.19
960	1200	2400	0.15	448.9
2000	2500	5000	516.9	> 1800

Merged encoding

Evaluation

Parameters that affect the performance

- The previous state of the canceled appointment (granted or not granted) ✓
- The type of prior scheduling (close or far from optimality) ✓
- The total number of timeslots ✓
- The patients to request and the request to timeslot ratio (demand) ✓
- The path followed to reach optimality ✓
- The difference in the number of predicates in the KB caused by the choice of encoding ✗

04

From concept to reality

Facing the challenges of a real-world implementation

Performance limitations

Although very powerful, ASP solvers have their limitations. Possible paths to further scalability are:

Further code optimization

More constraints can be added and less set can be created with further code optimization

Set time limit

Sacrificing optimality an interrupt signal can be given to the solver if a model occurs after a certain time limit

Limit models

Again, at the cost of optimality a limit for the number of examined models can be set

Batching

The most practical path to scalability is to search for a more localized optimality through batching according to a specific unit. The lowest level of clustering will always be the fundamental resource, in this case the doctors and their time. The total optimality will be sacrificed but the level of specialization can be tuned to get a high enough benefit for a low enough execution time

Applying batching for 2000 timeslots

	Time(s)	Benefit
General	315,64	8161
Specialty	3,51	7339
Doctor	1,73	5831

Web App UI

1

Log in

2

Display the current schedule

3

Store the requests and their score in the database

4

Actively listen for a cancelation

In the event of a cancelation

1

Delete the request from DB

2

Inform the KB

3

Run the rescheduler

4

Contact the patiens of the action chain

Managing action chains

The explainability of ASP provides us with an action chain of request to grant at the end of the optimization process. To realize the chain, patients should give their consent. The process will use only automated contact methods with binary (yes/no) answers.

The system finds the patient behind the first request of the solver's output

An automated text is sent asking for permission to reschedule the patient to a better request

The chain breaks

The process continues

Managing action chains

What if in the meantime another cancelation happens?

01

Immediately break the chain sacrificing part of the previous rescheduling

Take into consideration only the confirmed part of the previous chain. Update the database and then reschedule with the new cancelation.

02

Wait for the confirmation process of the previous chain to finish

Wait for the previous rescheduling to be fully realized before the new rescheduling starts.

03

Set a timeout for the patient's response and reschedule after it expires for the first time breaking the previous action chain

A happy medium between solutions 1 and 2, the previous action chain will be taken into consideration only if its confirmation process progresses at a reasonable rate

Thanks!

Do you have any questions?

stavroskanias@gmail.com - up1066563@upnet.gr

https://github.com/StavrosKanias/Medical_Appointment_Rescheduling_App
+30 694 755 3976

Resources

Bibliography

- M. Hawkins, "Survey of Physician Appointment Wait Times and Medicare and Medicaid Acceptance Rates," AMN Healthcare, 2022.

- A. H. Gertz, C. C.Pollack, M. D.Schultheiss and J. S. Brownstein, "Delayed medical care and underlying health in the United States during the COVID-19 pandemic: A cross-sectional study," Preventive Medicine Reports, 2022.

- S. Kanias, "An AI approach to Large Scale Medical Appointment (Re)Scheduling Using ASP," University of Patras, Patra, 2023.

Icons

- Icon Pack: Medicine | Lineal

Online

- J. Gier, "Healthcare Innovation," SCI Solutions, [Online]. Available: https://www.hcinnovationgroup.com/clinical-it/article/13008175/missed-appointments-cost-the-us-healthcare-system-150b-each-year. [Accessed August 2023].