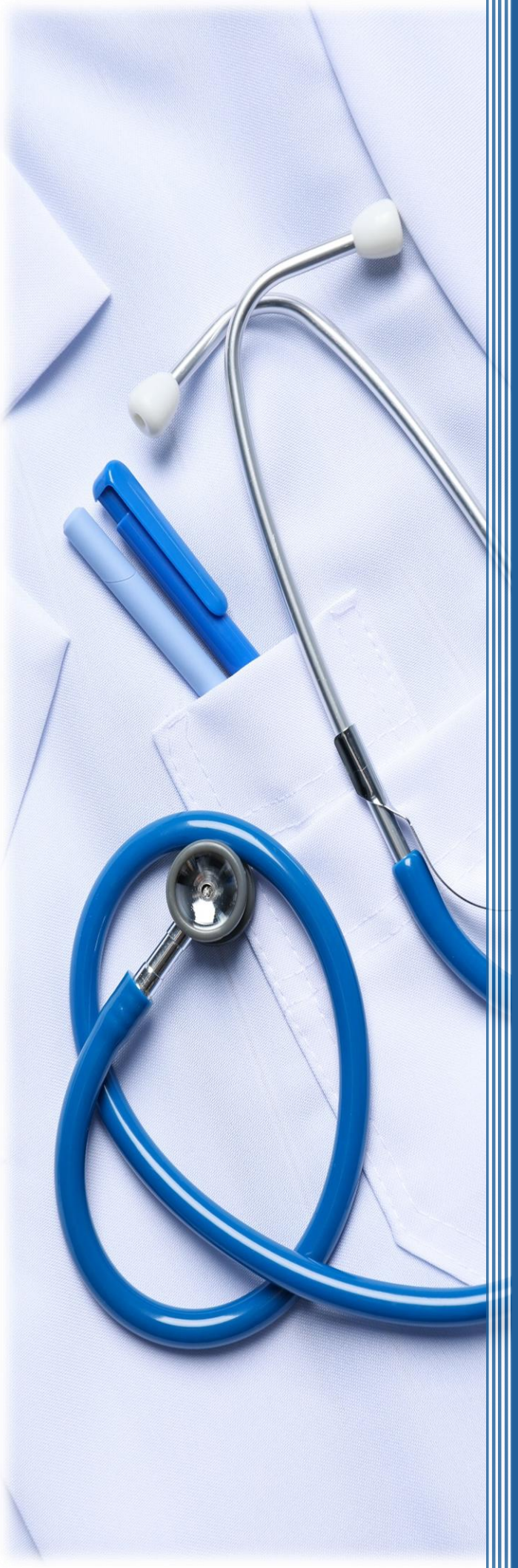


ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Relatório de *Planning of Surgeries in Operation Rooms of Hospitals*

Sprint B

1200347 Beatriz Silva

1220784 David Sousa

1220786 Guilherme Ribeiro

1221124 Tiago Carvalho

Algoritmia Avançada

24-11-2024

Índice

6.3.1 2
Definição e Disponibilidade dos Médicos 2
Adaptando os horários dos médicos 2
Cálculo das agendas livres de todos os médicos 2
Encontro de intervalos para Agendar cirurgias 3
Agendamento de Cirurgias 3
Adaptação do método para incluir todo o pessoal necessário e todas as fases da operação 4
6.3.2 Estudo da Complexidade 6
6.3.3 Duas Heurísticas e sua comparação 9
Conclusão 13

6.3.1

Como Administrador, quero obter o melhor agendamento de um conjunto de operações (cirurgias) em uma determinada sala de operação em um dia específico.

Nas primeiras três aulas de **TP** (Trabalho Prático), foi apresentado um código que lida com o agendamento de cirurgias, levando em consideração apenas os **médicos** necessários para realizar as operações. O objetivo é atribuir as cirurgias aos médicos disponíveis de forma eficiente.

Definição e Disponibilidade dos Médicos

1.1 Definição de Agendas:

Este predicado representa a agenda de cada médico num dia específico, incluindo os horários de atividades já programadas, como reuniões ou consultas.

agenda_staff(d001,20241028,[(720,840,m01),(1080,1200,c01)]).

Isto significa que o médico **d001** está ocupado com a reunião **m01** das 720 às 790, e com consultas **c01** das 1080 às 1140 no dia **20241028**.

1.2 Tempo livre

Este predicado calcula os intervalos de tempo livres a partir de uma lista de intervalos ocupados no formato **[(tin, tfin, atividade)]**.

free_agenda_staff0([(720,840,m01),(1080,1200,c01)],LFree).

O resultado é uma lista de intervalos no formato **[(tin, tfin)]**.

LFree = [(0, 719), (841, 1079), (1201, 1440)].

Adaptando os horários dos médicos

Este predicado ajusta os intervalos disponíveis de um médico com base no horário de trabalho, que define os limites de início e fim do seu horário laboral.

timetable(d001,20241028,(480,1200))

Isto significa que o médico **d001** trabalha das 480 às 1200 minutos no dia **20241028**.

O predicado **adapt_timetable** garante que os intervalos livres do médico estejam dentro do seu horário de trabalho, ajustando-os.

adapt_timetable(D, Date, LFA, LFA2).

Cálculo das agendas livres de todos os médicos

Este predicado calcula os horários livres de todos os médicos para um dia específico, chamando as funções necessárias para calcular os intervalos livres (**free_agenda_staff0** e **adapt_timetable**).

find_free_agendas(20241028).

Ao chamar esse predicado, o Prolog calculará os intervalos livres de todos os médicos para o dia especificado.

O resultado é armazenado no predicado dinâmico **availability/3**, que representa os intervalos livres de cada médico no dia:

availability(d001, 20241028, [(480, 719), (791, 1079), (1141, 1200)]).

Encontro de intervalos para Agendar cirurgias

Este predicado calcula a interseção dos intervalos livres de todos os médicos envolvidos em uma cirurgia. Este garante que a cirurgia seja agendada apenas em horários em que todos os médicos necessários estejam disponíveis.

intersect_all_agendas([d001, d002, d003], 20241028, LA).

Isso retornará uma lista de intervalos de tempo em que todos os médicos (**d001, d002 e d003**) estão livres, e, portanto, a cirurgia pode ser agendada nesse período:

LA = [(520, 719), (791, 849), (981, 1079), (1141, 1200)].

Agendamento de Cirurgias

A probabilidade, agora que os tempos livres estão disponíveis através do predicado **intersect_all_agendas/3**, é de agendar as cirurgias alocando intervalos dessa lista em horários disponíveis. O agendamento real pode ser modelado com outros predicados (como **agenda_operation_room** ou **schedule_all_surgeries**), mas a principal preocupação aqui é garantir que os médicos estejam disponíveis e identificar a sobreposição nos tempos livres deles.

?- find_free_agendas(20241028).

true.

?- availability(d001, 20241028, LA).

D = d001,

DAte = 20241028,

LA = [(480, 719), (791, 1079), (1141, 1200)].

Adaptação do método para incluir todo o pessoal necessário e todas as fases da operação

```
        Ini>Fim1,!,
        intersect_availability((Ini,Fim),LD,LI,LA).

intersect_availability((Ini,Fim),[(Ini1,Fim1)|LD],[(Imax,Fmin)],[(Fim,Fim1)|LD]):-
        Fim1>Fim,!,
        min_max(Ini,Ini1,_,Imax),
        min_max(Fim,Fim1,Fmin,_).

intersect_availability((Ini,Fim),[(Ini1,Fim1)|LD],[(Imax,Fmin)|LI],LA):-
        Fim>=Fim1,!,
        min_max(Ini,Ini1,_,Imax),
        min_max(Fim,Fim1,Fmin,_),
        intersect_availability((Fim1,Fim),LD,LI,LA).


min_max(I,I1,I,I1):- I<I1,!.
min_max(I,I1,I1,I).
```

```
schedule_all_surgeries(Room,Day):-
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),
    findall(_,(agenda_staff(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda))),_),
    agenda_operation_room(Or,Date,Agenda),assert(agenda_operation_room1(Or,Date,Agenda)),
    findall(_,(agenda_staff1(D,Date,L),free_agenda0(L,LFA),adapt_timetable(D,Date,LFA,LFA2),assertz(availability(D,Date,LFA2))),_),
    findall(OpCode,surgery_id(OpCode,_),LOpCode),

    availability_all_surgeries(LOpCode,Room,Day),!.

availability_all_surgeries([],_,_).
availability_all_surgeries([OpCode|LOpCode],Room,Day):-
    surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
    availability_operation(OpCode,Room,Day,LPossibilities,LDoctors),
    schedule_first_interval(TSurgery,LPossibilities,(TinS,TfinS)),
    retract(agenda_operation_room1(Room,Day,Agenda)),
    insert_agenda((TinS,TfinS,OpCode),Agenda,Agenda1),
    assertz(agenda_operation_room1(Room,Day,Agenda1)),
    insert_agenda_doctors((TinS,TfinS,OpCode),Day,LDoctors),
    availability_all_surgeries(LOpCode,Room,Day).
```

```
availability_operation(OpCode,Room,Day,LPossibilities,LDoctors):-surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
    findall(Doctor,assignment_surgery(OpCode,Doctor),LDoctors),
    intersect_all_agendas(LDoctors,Day,LA),
    agenda_operation_room1(Room,Day,LAgenda),
    free_agenda0(LAgenda,LFAgRoom),
    intersect_2_agendas(LA,LFAgRoom,LIntAgDoctorsRoom),
    remove_unf_intervals(TSurgery,LIntAgDoctorsRoom,LPossibilities).


remove_unf_intervals(_,[],[]).
remove_unf_intervals(TSurgery,[(Tin,Tfin)|LA],[(Tin,Tfin)|LA1]):-DT is Tfin-Tin+1,TSurgery=<DT,!,
    remove_unf_intervals(TSurgery,LA,LA1).
remove_unf_intervals(TSurgery,[_|LA],LA1):- remove_unf_intervals(TSurgery,LA,LA1).


schedule_first_interval(TSurgery,[(Tin,_)|_],(Tin,TfinS)):-
    TfinS is Tin + TSurgery - 1.

insert_agenda((TinS,TfinS,OpCode),[],[(TinS,TfinS,OpCode)]).
insert_agenda((TinS,TfinS,OpCode),[(Tin,Tfin,OpCode1)|LA],[(TinS,TfinS,OpCode),(Tin,Tfin,OpCode1)|LA]):-TfinS<Tin,!.
insert_agenda((TinS,TfinS,OpCode),[(Tin,Tfin,OpCode1)|LA],[(Tin,Tfin,OpCode1)|LA1]):-insert_agenda((TinS,TfinS,OpCode),LA,LA1).
```

O objetivo é percorrer todas as entradas de **requiredstaff(Role, Specialization, N)** associadas a um determinado tipo de operação e alocar N membros da equipa com a função (Role) e especialização (Specialization) necessárias, desde que estejam disponíveis conforme indicado no *timetable*.

Esses membros alocados devem ser adicionados ao **assignedstaff** da cirurgia. Além disso, as agendas devem ser atualizadas de acordo com as fases da cirurgia nas quais eles estarão envolvidos.

6.3.2 Estudo da Complexidade

Como Administrador, quero saber até que dimensão, em termos de número de cirurgias, é possível solicitar a melhor solução.

Para completar esta tabela utilizamos os seguintes predicados:

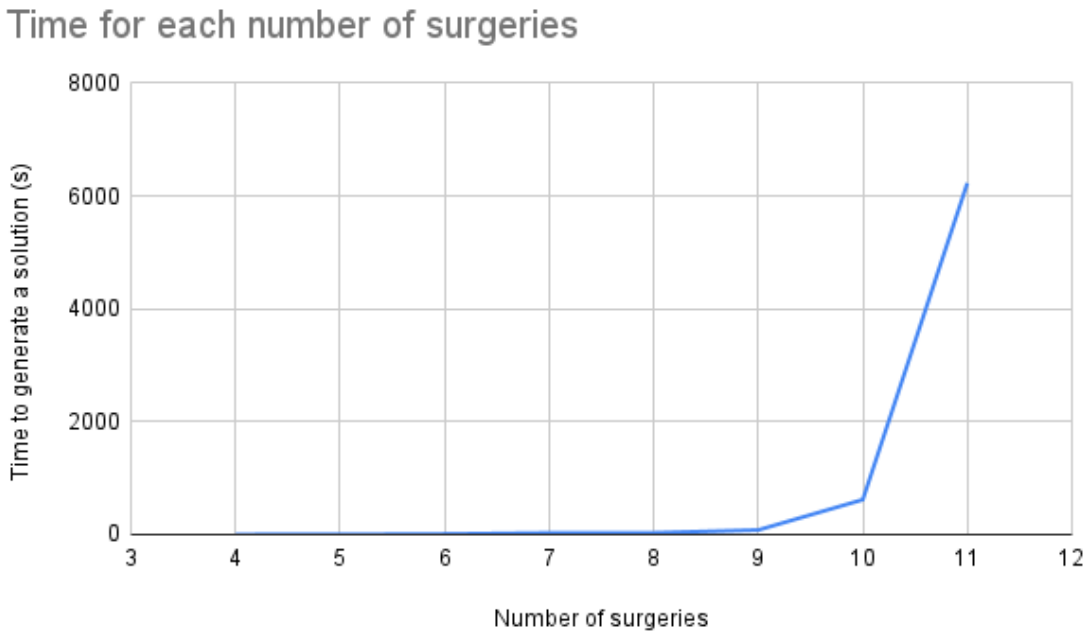
schedule_all_surgeries(or1,20241028).

obtain_better_sol(or1,20241028,AgOpRoomBetter,LAgDoctorsBetter,TFinOp).

N. of Surgeries	N. of Solutions	Best Schedule of activities (including surgeries) of the operation room N. of solutions	Final Time for the last Surgery (minutes)	Time to generate the solution (s)
3	6	[(520,579,so100000), (580,639,so100001), (640,714,so100003), (715,804,so100002), (1000,1059,so099999)]	804	0.0740420818328 8574
4	24	[(520,579,so100000), (580,654,so100003), (655,714,so100004), (715,804,so100002), (805,864,so100001), (1000,1059,so099999)]	864	0.18855118751525 88
5	120	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (1000,1059,so099999)]	939	1.43095612525939 94
6	720	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (940,999,so100006),	999	8.90377807617187 5

		(1000,1059,so099999)]		
7	5040	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (940,999,so100006), (1000,1059,so099999), (1060,1149,so100007)]	1149	27.3507559299469
8	40,320	[(520,579,so100000), (580,639,so100004), (640,699,so100008), (700,789,so100002), (791,865,so100003), (866,925,so100001), (926,985,so100006), (1000,1059,so099999), (1060,1134,so100005), (1135,1224,so100007)]	1224	29.0301268100738 53
9	362,880	[(520,579,so100000), (580,639,so100004), (640,699,so100008), (700,789,so100002), (790,849,so100009), (850,909,so100001), (910,969,so100006), (1000,1059,so099999), (1060,1134,so100005), (1135,1209,so100003), (1210,1299,so100007)]	1299	79.3147311210632 3
10	3,628,800	[(520,579,so100000), (580,639,so100004) ,(640,699,so100008),	1374	616.857285976409 9

		(700,759,so100009), (791,865,so100003), (866,925,so100001), (926,985,so100006) ,(1000,1059,so099999), (1060,1134,so100005), (1135,1224,so100007), (1225,1284,so100010), (1285,1374,so100002)]		
11	39,916,800		1441	6228.80145406723



6.3.3 Duas Heurísticas e sua comparação

Como Administrador, quero obter um bom agendamento, não necessariamente o melhor, em tempo útil para ser adotado.

N. of Surgeries	Optimal solution	Final Time for the last Surgery In generate all select better (minutes)	Final Time for the last Surgery Using Heuristic (minutes)	Time to generate the best solution (s)	Time to generate the heuristic solution (s)	Solution with the Heuristic
3	[(520,579,so100000), (580,639,so100001), (640,714,so100003), (715,804,so100002), (1000,1059,so099999)]	804	804	0.07404208183288574	0.0449340 3434753418	[(520,579,so100000), (580,639,so100001), (640,714,so100003), (715,804,so100002), (1000,1059,so099999)]
4	[(520,579,so100000), (580,654,so100003), (655,714,so100004), (715,804,so100002), (805,864,so100001), (1000,1059,so099999)]	864	865	0.1885511875152588	0.0642070 7702636719	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100004), (700, 789, so100002), (791, 865, so100003), (1000, 1059, so099999)]
5	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (1000,1059,so099999)]	939	1134	1.4309561252593994	0.05446505 546569824	[(520,579,so100000), (580,639,so100001), (640,699,so100004), (700,789,so100002), (791,865,so100003), (1000,1059,so099999), (1060,1134,so100005)]
6	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (940,999,so100006), (1000,1059,so099999)]	999	1200	8.903778076171875	0.0599598 8845825195	[(520,579,so100000), (580,639,so100006), (640,699,so100001), (700,789,so100002), (791,865,so100003), (1000,1059,so099999), (1060,1134,so100005), (1141,1200,so100004)]

7	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (940,999,so100006), (1000,1059,so099999), (1060,1149,so100007)]	1149	1275	27.3507559299469	0.045076847076416016	[(520,579,so100000), (580,669,so100007), (670,759,so100002), (791,850,so100001), (851,910,so100006), (1000,1059,so099999), (1060,1134,so100003), (1141,1200,so100004), (1201,1275,so100005)]
8	[(520,579,so100000), (580,639,so100004), (640,699,so100008), (700,789,so100002), (791,865,so100003), (866,925,so100001), (926,985,so100006), (1000,1059,so099999), (1060,1134,so100005), (1135,1224,so100007)]	1224		29.030126810073853		
9	[(520,579,so100000), (580,639,so100004), (640,699,so100008), (700,789,so100002), (790,849,so100009), (850,909,so100001), (910,969,so100006), (1000,1059,so099999), (1060,1134,so100005), (1135,1209,so100003), (1210,1299,so100007)]	1299		79.31473112106323		
10	[(520,579,so100000), (580,639,so100004) ,(640,699,so100008), (700,759,so100009), (791,865,so100003),	1374		616.8572859764099		

	(866,925,so100001), (926,985,so100006) ,(1000,1059,so099999), (1060,1134,so100005), (1135,1224,so100007), (1225,1284,so100010), (1285,1374,so100002)]					
11		1441		6228.80145406723		

Heurísticas são métodos muito práticos para resolver problemas complexos de forma eficiente quando uma solução exata pode ser mais demorada em termos de tempo ou poder computacional. Estas não garantem encontrar a melhor solução, mas procuram uma **solução aceitável ou suficientemente eficiente** dentro de um prazo razoavelmente rápido. Para isso, fizemos heurística:

1. Agendar cirurgias em um determinado **dia** e **sala** (Room, Day), considerando a disponibilidade da equipa, salas de operação e tempo necessário para as cirurgias.

```
obtain_better_sol1(Room, Day) :-
    asserta(better_sol(Day, Room, _, _, 1441)),
    findall(OpCode, surgery_id(OpCode, _), LOC),
    generate_n_lists(LOC, LOpCodes),
    member(LOpCode, LOpCodes),
    retractall(agenda_staff1(_, _, _)),
    retractall(agenda_operation_room1(_, _, _)),
    retractall(availability(_, _, _)),
    findall(_, (agenda_staff(D, Day, Agenda), assertz(agenda_staff1(D, Day, Agenda))), _),
    agenda_operation_room(Room, Day, Agenda), assert(agenda_operation_room1(Room, Day, Agenda)),
    findall(_, (agenda_staff1(D, Day, L), free_agenda0(L, LFA), adapt_timetable(D, Day, LFA, LFA2), assertz(availability(D, Day, LFA2))), _),
    availability_all_surgeries(LOpCode, Room, Day),
    agenda_operation_room1(Room, Day, AgendaR),
    update_better_sol(Day, Room, AgendaR, LOpCode),
    fail.

generate_n_lists(AllReqs, NLists) :-
    findall(List,
        (member(StartReq, AllReqs), generate_ordered_list(StartReq, AllReqs, List)),
        NLists).

generate_ordered_list(StartReq, AllReqs, OrderedList) :-
    select(StartReq, AllReqs, RemainingReqs), % Remove o StartReq da lista
    sort_requests_by_time(RemainingReqs, SortedRemainingReqs),
    Or
```

```
sort_requests_by_time(RequestIDs, SortedRequests) :-
    maplist(find_surgery_time, RequestIDs, Times),
    pairs_keys_values(Pairs, Times, RequestIDs),
    keysort(Pairs, SortedPairs),
    pairs_values(SortedPairs, SortedRequests).

find_surgery_time(RequestID, TotalTime) :-
    surgery_id(RequestID, SurgeryID),
    calculate_total_time(SurgeryID, TotalTime).

calculate_total_time(SurgeryID, TotalTime) :-
    surgery(SurgeryID, PrepTime, SurgeryTime, RecoveryTime),
    TotalTime is PrepTime + SurgeryTime + RecoveryTime.
```

Esta heurística cria listas ordenadas de cirurgias com base nos tempos totais, testando diferentes permutações. Em seguida, ajusta as disponibilidades e verifica a viabilidade de cada lista. Quando uma configuração é melhor que a atual, ela **substitui a solução armazenada**. O processo explora sistematicamente todas as combinações, equilibrando eficiência e viabilidade, com o objetivo de encontrar uma solução próxima do ideal.

Conclusão

- É possível verificar que quando utilizamos heurística, o tempo necessário para encontrar uma solução é significativamente reduzido em comparação com métodos exatos que exploram todas as possibilidades possíveis.