ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

P.PORTO

Relatório de ALGAV
Sprint 2

Autores:
1121200 – Rui Beloto
1161096 – Pedro Águia
1211656 – José Pina
1210891 – Sérgio Cardoso

Turma 3NA
Grupo G74

1. Explicação código base:

O código base começa com definições dinâmicas.

```
:- dynamic availability/3.
:- dynamic agenda_staff/3.
:- dynamic agenda_staff1/3.
:- dynamic agenda_operation_room/3.
:- dynamic agenda_operation_room1/3.
:- dynamic better_sol/5.
```

Estas declarações tornam os factos dynamic, permitindo que sejam alterados em tempo de execução, para inserir, modificar ou apagar factos através do assertz e retract. Isso é crucial, uma vez que o planeamento altera a disponibilidade (availability) e as agendas (agenda_staff1, agenda_operation_room1) ao longo do processo.

O que se segue são factos do staff e da sala de operações, desde marcações, a horários e à atribuição das cirurgias ao staff correspondente.

```
free_agenda0([], [(0,1440)]).
free_agenda0([(0,Tfin,_)|LT], LT1) :- !, free_agenda1([(0,Tfin,_)|LT], LT1).
free_agenda0([(Tin, Tfin, _)|LT], [(0, T1)|LT1]) :-
    T1 is Tin - 1,
    free_agenda1([(Tin, Tfin, _)|LT], LT1).
```

O predicado free_agenda0 determina os intervalos livres de uma agenda.

```
adapt_timetable(D, Date, LFA, LFA2) :-
    timetable(D, Date, (InTime, FinTime)),
    treatin(InTime, LFA, LFA1),
    treatfin(FinTime, LFA1, LFA2).
```

O adapt_timetable combina os intervalos livres com o horário de trabalho do staff para determinar os tempos livres reais.

```
intersect_all_agendas([Name], Date, LA) :- !, availability(Name, Date, LA).
intersect_all_agendas([Name|LNames], Date, LI) :-
    availability(Name, Date, LA),
    intersect_all_agendas(LNames, Date, LI1),
    intersect_2_agendas(LA, LI1, LI).
```

O intersect_all_agendas usa uma lista de staff para interligar os tempos livres de cada um. Dessa forma conseguimos ter os horários disponíveis para aquela combinação de staff. Isto é importante para coincidir as marcações das cirurgias.

```
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(Room, Date, Agenda),
assert(agenda_operation_room1(Room, 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), !.

O Schedule_all_surgeries permite agendar as cirurgias na sala e dia selecionado. Este é o predicado principal que combina todos os factos e predicados.
Cada cirurgia é processada individualmente, o availability_all_surgeries é responsável por isso.
Através de recursividade cada cirurgia vai ser analisada consoante a disponibilidade de calendários da sala e dos médicos.

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).

Com availability_operation vamos analisar as possibilidades de agendamento para uma determinada operação num determinado dia e sala. Tem em conta todos os parâmetros como agendas dos médicos e sala. É devolvido uma lista de possibilidades de agendamento. De acordo com o que foi pedido, o algoritmo otimiza o planeamento de acordo com o horário mais cedo possível. Isso é feito com o Schedule_first_interval que vai escolher a possibilidade mais cedo. Depois são atualizadas as agendas antes de seguir para a próxima cirurgia.

obtain_better_sol1(Room,Day):-
asserta(better_sol(Day,Room,_,_,1441)),
findall(OpCode,surgery_id(OpCode,_),LOC),!,
permutation(LOC,LOpCode),
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.
```

Como o algoritmo deve devolver a melhor solução possível é utilizado o predicado obttain_better_sol1 para fazer todas as permutações possíveis com o número de cirurgias a marcar. Desta forma, é possível determinar quais das soluções permitem atingir um horário com marcações mais cedo e eficaz ao nível de planeamento.

O predicado evaluate_final_time e o update_better_sol comparam cada uma das soluções em relação à eficiência do algoritmo para selecionar a melhor solução.

```
obtain_better_sol(Room,Day,AgOpRoomBetter,LAgDoctorsBetter,TFinOp):-
get_time(Ti),
(obtain_better_sol1(Room,Day);true),

retract(better_sol(Day,Room,AgOpRoomBetter,LAgDoctorsBetter,TFinOp)),
write('Final Result: AgOpRoomBetter='),write(AgOpRoomBetter),nl,
write('LAgDoctorsBetter='),write(LAgDoctorsBetter),nl,
write('TFinOp='),write(TFinOp),nl,
get_time(Tf),
T is Tf-Ti,
write('Tempo de geracao da solucao:'),write(T),nl.
```

O obtain_better_sol permite correr o algoritmo e dá um feedback do tempo que demorou e os detalhes da solução ótima

2. Análise de complexidade:

Para fazer uma avaliação da complexidade adicionamos vários factos para incrementar gradualmente o número de cirurgias a agendar. Como o algoritmo funciona a avaliar todas as permutações a escala é factorial. Ou seja, conforme demonstrado na tabela, o número de soluções rapidamente atinge valores muito altos para capacidades de computação.

Os testes foram realizados com o seguinte comando:

obtain_better_sol(or1,20241028,_,_,_).

E um exemplo do que obtemos na consola:

...
Analysing for LOpCode=[so100004, so100003, so100001, so100002]
now: FinTime1=1149 Agenda=[(520,579,so100000), (580,639,so100004), (640,714,so100003), (791,850,so100001), (1000,1059,so099999), (1060,1149,so100002)]
Analysing for LOpCode=[so100004, so100003, so100002, so100001]
now: FinTime1=864 Agenda=[(520,579,so100000), (580,639,so100004), (640,714,so100003), (715,804,so100002), (805,864,so100001), (1000,1059,so099999)]
Final Result: AgOpRoomBetter=[(520,579,so100000), (580,654,so100003), (655,714,so100004), (715,804,so100002), (805,864,so100001), (1000,1059,so099999)]
LAgDoctorsBetter=[(d003,[(580,654,so100003), (720,790,m01), (910,980,m02)]), (d001,[(655,714,so100004), (720,790,m01), (805,864,so100001), (1080,1140,c01)]), (d002,[(655,714,so100004), (715,804,so100002), (850,900,m02), (901,960,m02), (1380,1440,c02)])]
TFinOp=864
Tempo de geracao da solucao:0.30062007904052734
true

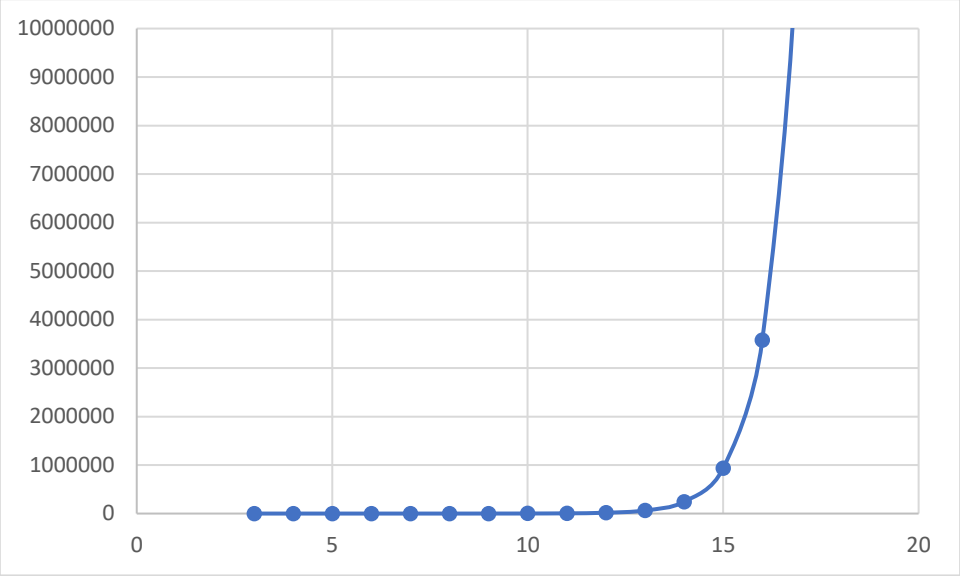
A tabela seguinte representa os resultados obtidos:

N. Of Surgeries	N. Of Solutions	Best Schedule of activities (including surgeries) of the operation room	Final Time for the last Surgery (minutes)	Time to generate the solution (s)
3	6	[(480, 539, so100001), (540, 629, so100002), (630, 704, so100003), (750, 780, mnt0001), (1080, 1110, mnt0002)]	704	0,12
4	24	[(520,579,so100000), (580,654,so100003), (655,714,so100004), (715,804,so100002),	864	0,3

		(805,864,so100001), (1000,1059,so099999)]		
5	120	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (1000,1059,so099999)]	939	1,64
6	720	[(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	5,2
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	10,8
8	40320	[(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	108,9
9	362880	[(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	345

10	3628800	n/a	n/a	excedeu o tempo limite (swi-prolog online)
11	39916800	n/a	n/a	n/a
12	479001600	n/a	n/a	n/a
13	6227020800	n/a	n/a	n/a

É possível verificar como muito rapidamente se torna quase impossível ter uma solução ideal com a avaliação de todas as permutações. Como o algoritmo tem uma complexidade factorial com 13 cirurgias para marcar já existem mais de 6 milhões de soluções. Com 20 cirurgias existem 2,4E18 soluções! Mesmo com grande poder computacional demoraria na ordem das centenas de milhões de segundos se extrapolarmos o crescimento exponencial do tempo, conforme a figura:



Isto significa que, não é viável usar este algoritmo e uma abordagem por heurísticas é a melhor opção.

3. Heurísticas e comparação com solução ótima:

- **Primeira heurística**

A primeira heurística desenvolvida foi uma em que a primeira cirurgia a ser considerada para marcação é a cirurgia na qual o médico está disponível mais cedo. Vamos chamar-lhe de heurística de tempo.

Para conseguir a heurística de tempo aproveitamos grande parte da estrutura já existente e acrescentei o heuristic_by_time ao Schedule_all_surgeries:

```
schedule_all_surgeries(Room,Day):-
...
%Inicio da alteraçao de codigo. Heuristica de tempo.
heuristic_by_time(LOpCode,Room,Day,SortedLOpCode),
extract_ids(SortedLOpCode,IdList),
availability_all_surgeries(IdList,Room,Day),!,
```

O que esta heurística por tempo faz é, antes de avaliar a cirurgia a marcar, verificar qual a cirurgia com o médico disponível mais cedo.

```
heuristic_by_time([], _, _, []).
heuristic_by_time([OpCode | LOpCode], Room, Day, SortedLOpCode) :-
surgery_id(OpCode, OpType),
surgery(OpType, _, TSurgery, _),
availability_operation(OpCode, Room, Day, LPossibilities, _),
(LPossibilities \= [] ->
schedule_first_interval(TSurgery, LPossibilities, (TinS, _)),
heuristic_by_time(LOpCode, Room, Day, PartialSorted),
append([(OpCode, TinS)], PartialSorted, PartialSorted1),
inverter(PartialSorted1, NewPartialSorted),
sort_op_list(NewPartialSorted, SortedList),
SortedLOpCode = SortedList;
SortedLOpCode = []
).
```

Para isso, verifico as possibilidades de horários de todas as cirurgias de forma recursiva. Se existirem possibilidades de horário de várias cirurgias ordeno a lista de cirurgias por ordem do médico disponível mais cedo.
É essa lista que é usada para marcar a próxima cirurgia.

```
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),
heuristic_by_time(LOpCode,Room,Day,SortedLOpCode),
extract_ids(SortedLOpCode,IdList),
availability_all_surgeries(IdList,Room,Day).

Ou seja, a próxima cirurgia marcada é a primeira da lista. Após marcar a cirurgia volto a correr a heurística de tempo.

Este algoritmo segue então este fluxo, faz-se uma lista ordenada das cirurgias a marcar pelo horário do médico disponível mais cedo, tenta-se marcar a primeira cirurgia da lista, após marcar a cirurgia as cirurgias restantes são novamente avaliadas e colocadas por ordem, marca-se a primeira… e assim sucessivamente até não existirem cirurgias a marcar.

O resultado gerado quando corro Schedule_all_surgeries é o seguinte:

Resultados de agendamento para o dia 20241028: Sala de Operações or1
Agenda para sala or1 no dia 20241028:
[(480,539,so100001),(540,599,so100004),(600,674,so100005),(675,764,so100002),(791,865,so100003)]
Agendas dos Médicos: Médico d001:
[(480,539,so100001),(540,599,so100004),(720,790,m01),(1080,1140,c01)]
Médico d002:
[(540,599,so100004),(600,674,so100005),(675,764,so100002),(850,900,m02),(901,960,m02),(1380,1440,c02)]
Médico d003: [(600,674,so100005),(720,790,m01),(791,865,so100003),(910,980,m02)]
Tempo de geracao da solucao:0.001399993896484375
1true

Este output é um exemplo do funcionamento da heurística de tempo. Como não uso permutações a primeira solução obtida é a solução aceite.

Nos anexos está o algoritmo completo para consulta.

- **Segunda heurística**

A segunda heurística vai ter como objetivo tomar decisões com base no médico mais ocupado. É importante que os médicos fiquem ocupados o máximo possível para rentabilizar e gerir o tempo de forma mais adequada. É isso que esta heurística vai fazer, a primeira cirurgia a ser marcada é uma das cirurgias do médico mais ocupado.
Para conseguir isso começamos por criar alguns predicados que vão ajudar a calcular a ocupação de um funcionário.

calculate_free_time([], 0).
calculate_free_time([(In, Fin)|LT], TotalFreeTime) :-
FreeTime is Fin - In,

```
calculate_free_time(LT, RemainingTime),
TotalFreeTime is RemainingTime + FreeTime.

get_total_time_free(D,Date,TotalFreeTime):-
agenda_staff(D,Date,L),
free_agenda0(L,LFA),
adapt_timetable(D,Date,LFA,LFA2),
calculate_free_time(LFA2,TotalFreeTime).

calculate_working_hours(D, Date, R) :-
timetable(D, Date, (InTime, FinTime)),
R is FinTime - InTime.

staff_occupation_percentage(D,Date,R):-
get_total_time_free(D,Date,TFT),
calculate_working_hours(D,Date,WH),
R is (TFT/WH) * 100.
```

Estes predicados servem para calcular a percentagem de ocupação de um staff. Isto é essencial para o próximo passo que é ordenar os médicos pela ocupação.

```
all_doctor_occupation(Date,[(Doctor, OccupationPercent) | R]) :-
findall((Doctor, OccupationPercent), (staff_occupation_percentage(Doctor, Date,
OccupationPercent)), R).

% Ordena os médicos pela ocupação em ordem decrescente
sort_doctors_by_occupation(DoctorOccupations, SortedDoctors) :-
% Ordena a lista de médicos pela ocupação (em ordem decrescente)
sort(2, @>=, DoctorOccupations, SortedDoctorsWithOccupation),
% Extrai apenas os médicos da lista ordenada
findall(Doctor, member((Doctor, _), SortedDoctorsWithOccupation),
SortedDoctors).

%Bloco para encontrar medico mais ocupado.
% Encontra o médico com maior ocupação a partir da lista de ocupações
find_most_occupied_doctor(DoctorOccupations, SortedDoctors) :-
sort_doctors_by_occupation(DoctorOccupations, SortedDoctors).
```

Este bloco de código permite obter uma listagem de ocupação de médicos.

```
%Verifica se existem cirurgias para o medico selecionado e seleciona todas.
search_surgery_for_doctor([], _, []) :-
!.  % Caso base: se a lista de médicos está vazia, não há cirurgias para marcar.

search_surgery_for_doctor(_, [], []) :-
!.  % Caso base: se a lista de cirurgias pendentes está vazia, não há mais cirurgias
para marcar.
```

```
search_surgery_for_doctor(LOpCode, [Doctor | Rest], OpCode) :-
% Filtra as cirurgias da lista LOpCode que são associadas ao médico atual (Doctor)
findall(OpCode, (member(OpCode, LOpCode), assignment_surgery(OpCode,
Doctor)), LOpCodeForDoctor),

% Se encontrar cirurgias, retorna a primeira OpCode encontrada
(
LOpCodeForDoctor \= []
-> select(OpCode, LOpCodeForDoctor, _), % Seleciona uma cirurgia da lista
true % Sai e utiliza a cirurgia encontrada
;
% Se não encontrou cirurgias, tenta o próximo médico
search_surgery_for_doctor(LOpCode, Rest, OpCode)
).
```

Como na listagem de médicos incluímos todos os médicos podemos escolher um médico que não tem uma cirurgia marcada. Então o bloco em cima permite avaliar essas condições e atuar em conformidade. Ao encontrar uma cirurgia no médico mais ocupado é devolvida para o algoritmo de marcação.

```
mark_surgery_if_possible(OpCode,Room,Date) :-
surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
findall(Doctor,assignment_surgery(OpCode,Doctor),LDoctors),
intersect_all_agendas(LDoctors, Date, LA),
agenda_operation_room1(Room, Date, LAgenda),
free_agenda0(LAgenda, LFAgRoom),
intersect_2_agendas(LA, LFAgRoom, LIntAgDoctorsRoom),
remove_unf_intervals(TSurgery, LIntAgDoctorsRoom, LPossibilities),
(
LPossibilities \= []
-> % Se houver horários disponíveis, marca a cirurgia
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),
write('Cirurgia marcada com sucesso!')
;
% Se não houver horários disponíveis, apenas avisa e continua sem interromper
write('Não há horários disponíveis para marcar a cirurgia.'), nl
).
```

Com o mark_surgery_if_possible tentamos verificar a disponibilidade de horário para marcar a cirurgia selecionada, se não for possível tentamos com a seguinte cirurgia da lista, se for possível a cirurgia fica marcada.

Com a chamada do algoritmo principal o Schedule_all_surgeries usamos todas as cirurgias que têm que ser marcadas e entramos no predicado principal da heurística attempt_to_mark_surgeries.

attempt_to_mark_surgeries([],_,_) :-
write('Todas as cirurgias foram marcadas com sucesso!').
attempt_to_mark_surgeries(LOpCode,Room,Day) :-
all_doctor_occupation(Day, DoctorOccupations),
find_most_occupied_doctor(DoctorOccupations, SortedDoctors),
search_surgery_for_doctor(LOpCode,SortedDoctors, OpCode),
mark_surgery_if_possible(OpCode,Room,Day),
remove_surgery_from_list(OpCode,LOpCode,NewLOpCode),
attempt_to_mark_surgeries(NewLOpCode,Room,Day).

Como indica o nome, é o compilar dos predicados criados que lidam com a logica da heurística. O fluxo é, primeiro, verifica-se a ocupação dos médicos, seleciona-se o medico mais ocupado, seleciona-se uma das cirurgias associadas a esse medico, faz-se a marcação se possível, voltamos a verificar a ocupação dos médicos e assim sucessivamente até não existirem cirurgias a marcar.

O resultado final pode ser visualizado na consola, por exemplo:

Resultados de agendamento para o dia 20241028: Sala de Operações or1
Agenda para sala or1 no dia 20241028:
[(520,594,so100003),(595,669,so100005),(670,759,so100002),(791,850,so100001),(961,1020,so100004)]
Agendas dos Médicos: Médico d003:
[(520,594,so100003),(595,669,so100005),(720,790,m01),(910,980,m02)]
Médico d002:
[(595,669,so100005),(670,759,so100002),(850,900,m02),(901,960,m02),(961,1020,so100004),(1380,1440,c02)]
Médico d001:
[(720,790,m01),(791,850,so100001),(961,1020,so100004),(1080,1140,c01)]
Tempo de geracao da solucao:0.0016200542449951172
true

- **Comparação das heurísticas com a melhor solução**

Para conseguirmos avaliar a eficiência das heurísticas vamos analisar quanto aos objetivos propostos anteriormente e comparativamente à melhor solução.

A seguinte tabela permite comparar os resultados obtidos:

N.	Best solution	Final Time	Generate Time	Heuristic One	Final Time	Generate Time	Heuristic Two	Final Time	Generate
3	[(480, 539, so100001), (540, 629, so100002), (630, 704, so100003), (750, 780,	704	0,12	[(520,579,so100000),(580,669,so100002),(791,850,so100001),(1000,1059,so099999),(1060,1134,so100003)]	1134	0,001	[(520,579,so100000),(580,654,so100003),(655,744,so100002),(791,850,so100001),(1000,1059,so099999)]	850	0,0013
4	[(520,579,so100000), (580,654,so100003), (655,714,so100004), (715,804,so100002), (805,864,so100001),	864	0,3	[(520,579,so100000),(580,669,so100002),(791,865,so100003),(866,925,so100001),(1000,1059,so099999),(1141,1200,so100004)]	1200	0,001	[(520,579,so100000),(580,654,so100003),(655,744,so100002),(791,850,so100001),(1000,1059,so099999),(1141,1200,so100004)]	1200	0,0014
5	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001),	939	1,64	[(520,579,so100000),(580,669,so100002),(791,850,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1209,so100003)]	1209	0,0013	[(520,579,so100000),(580,654,so100003),(655,744,so100002),(791,850,so100001),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004)]	1200	0,0017
6	[(520,579,so100000), (580,639,so100004), (640,714,so100005), (715,804,so100002), (805,879,so100003), (880,939,so100001), (940,999,so100006),	999	5,2	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999),(1060,1134,so100003),(1135,1209,so100005)]	1209	0,0023	[(520,579,so100000),(580,654,so100003),(655,744,so100002),(791,850,so100001),(851,910,so100006),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004)]	1200	0,0019
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),	1149	10,8	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1209,so100003),(1210,1299,so100007)]	1299	0,0019	[(520,579,so100000),(580,654,so100003),(655,744,so100002),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004)]	1200	0,0022
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),	1224	108,9	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1209,so100003),(1210,1299,so100007)]	1299	0,002	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1224,so100002)]	1224	0,0024
9	[(520,579,so100000), (580,639,so100004), (640,699,so100008), (700,789,so100002), (790,849,so100009), (850,909,so100001), (910,969,so100006)	1299	345	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1209,so100003),(1210,1299,so100007)]	1299	0,0021	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(715,774,so100009),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1224,so100002)]	1224	0,0028

N.	Best solution	Final Time	Generate Time	Heuristic One	Final Time	Generate Time	Heuristic Two	Final Time	Generate
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),	1299	345	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1209,so100003),(1210,1299,so100007)]	1299	0,0021	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(715,774,so100009),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1224,so100002)]	1224	0,0028
10	n/a	n/a	n/a	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999),(1060,1119,so100010),(1120,1194,so100005),(1195,1284,so100007)]	1284	0,0023	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(715,774,so100009),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1194,so100010),(1195,1284,so100002)]	1284	0,0029
11	n/a	n/a	n/a	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(911,985,so100011),(1000,1059,so099999),(1060,1119,so100010),(1120,1194,so100003),(1195,1269,so100005)]	1269	0,0023	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(715,774,so100009),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1194,so100010),(1195,1284,so100002)]	1284	0,0037
12	n/a	n/a	n/a	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(911,970,so100012),(1000,1059,so099999)]	970	0,002	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(715,774,so100009),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1194,so100010),(1195,1284,so100002)]	1284	0,0037
13	n/a	n/a	n/a	[(520,579,so100000),(580,669,so100002),(791,850,so100006),(851,910,so100001),(911,970,so100012),(1000,1059,so099999)]	970	0,0021	[(520,579,so100000),(580,654,so100003),(655,714,so100008),(715,774,so100009),(791,880,so100007),(881,940,so100001),(1000,1059,so099999),(1060,1134,so100005),(1135,1194,so100010),(1195,1284,so100002)]	1284	0,0037

Verifica-se que tanto a primeira heurística como a segunda têm velocidade muito rápida. São capazes de lidar com um número de cirurgias muito elevado. Quanto ao tempo final das marcações, de facto, a solução melhor permite obter um tempo final reduzido até 9 cirurgias, a partir de 9 cirurgias para marcar, o tempo final é idêntico em todas as soluções. Outro aspecto a comentar é o facto de, com um número elevado de cirurgias, o calendário fica cheio para aquela sala para aquele dia e sucessivas cirurgias não são marcada dai o estabilizar do tempo de processamento e da solução obtida.

4. Adaptação dos métodos para incluir staff completo e fases de operação:

No sentido de colocar o algoritmo desenvolvido funcional a um contexto de mundo real foram adaptados para poder incluir vários membros do staff e fases de operação distintas.

- **Heurística um**

O processo de incluir o staff na totalidade faz com que seja necessário adaptar um pouco o algoritmo. Foram adicionados os seguintes factos para servirem de exemplo:

agenda_staff(d001,20241028,[(720,790,m01),(1080,1140,c01)]).
agenda_staff(d002,20241028,[(850,900,m02),(901,960,m02),(1380,1440,c02)]).
agenda_staff(d003,20241028,[(720,790,m01),(910,980,m02)]).
agenda_staff(da001,20241028,[]).
agenda_staff(da002,20241028,[]).
agenda_staff(in001,20241028,[]).
agenda_staff(cn001,20241028,[]).
agenda_staff(an001,20241028,[]).
agenda_staff(maa001,20241028,[]).
agenda_staff(in002,20241028,[]).
agenda_staff(cn002,20241028,[]).
agenda_staff(an002,20241028,[]).
agenda_staff(maa002,20241028,[]).

timetable(d001,20241028,(480,1200)).
timetable(d002,20241028,(500,1440)).
timetable(d003,20241028,(520,1320)).
timetable(da001,20241028,(480,1200)).
timetable(da002,20241028,(500,1440)).
timetable(in001,20241028,(520,1320)).
timetable(cn001,20241028,(480,1200)).
timetable(an001,20241028,(500,1440)).
timetable(maa001,20241028,(520,1320)).
timetable(in002,20241028,(480,1200)).
timetable(cn002,20241028,(500,1440)).
timetable(an002,20241028,(520,1320)).
timetable(maa002,20241028,(520,1320)).

staff(d001,doctor,orthopaedist,[so2,so3,so4]).
staff(d002,doctor,orthopaedist,[so2,so3,so4]).
staff(d003,doctor,orthopaedist,[so2,so3,so4]).
staff(da001,doctor,anaesthetist,[so2,so3,so4]).
staff(da002,doctor,anaesthetist,[so2,so3,so4]).
staff(in001,staff,instrumentingnurse,[so2,so3,so4]).

```
staff(cn001,staff,circulatingnurse,[so2,so3,so4]).
staff(an001,staff,anaesthetistnurse,[so2,so3,so4]).
staff(maa001,staff,medicalactionassistant,[so2,so3,so4]).
staff(in002,staff,instrumentingnurse,[so2,so3,so4]).
staff(cn002,staff,circulatingnurse,[so2,so3,so4]).
staff(an002,staff,anaesthetistnurse,[so2,so3,so4]).
staff(maa002,staff,medicalactionassistant,[so2,so3,so4]).

assignment_surgery(so100001,d001).
assignment_surgery(so100002,d002).
assignment_surgery(so100003,d003).
assignment_surgery(so100004,d001).
assignment_surgery(so100004,d002).
assignment_surgery(so100005,d002).
assignment_surgery(so100005,d003).
```

Para avaliar, num conjunto de staff disponível, quais serão selecionados para cada cirurgia, criamos um conjunto de predicados que permitem verificar a ocupação do staff e organizar por ordem.

```
staff_occupation_percentage(D,Date,R):-
    get_total_time_free(D,Date,TFT),
    calculate_working_hours(D,Date,WH),
    R is (TFT/WH) * 100.

% Caso base: lista vazia retorna resultado vazio.
all_staff_occupation(_, [], []).
% Caso recursivo: calcula a ocupação para o primeiro membro e continua com o
restante.
all_staff_occupation(Date, [Staff | RestStaff], [(Staff, OccupationPercent) | RestResult])
:-
    staff_occupation_percentage(Staff, Date, OccupationPercent),
    all_staff_occupation(Date, RestStaff, RestResult).

% Ordena os médicos pela ocupação em ordem decrescente
sort_staff_by_occupation(RestResult, SortedStaff) :-
% Ordena a lista de médicos pela ocupação (em ordem decrescente)
    sort(2, @=<, RestResult, SortedStaffWithOccupation),
    % Extrai apenas os médicos da lista ordenada
    findall(Staff, member((Staff, _), SortedStaffWithOccupation), SortedStaff).
```

Para fazer este tipo de ordenação também é importante obter uma lista de staff para avaliar a ocupação de todos.

```
get_all_anaesthologist_doctors(LADoctors):-
    findall(ADoctor,staff(ADoctor,doctor,anaesthetist,_),LADoctors).
get_all_anaesthologist_nurses(LANurses):-
```

findall(ANurse,staff(ANurse,staff,anaesthetistnurse,_),LANurses).
get_all_instrumenting_nurses(LINurses):-
findall(INurse,staff(INurse,staff,instrumentingnurse,_),LINurses).
get_all_circulating_nurses(LCNurses):-
findall(CNurse,staff(CNurse,staff,circulatingnurse,_),LCNurses).
get_all_medicalactionassistant_staff(LMAAs):-
findall(MAA,staff(MAA,staff,medicalactionassistant,_),LMAAs).

Agora através de availability_all_surgeries é feita a introdução de cada staff e das fases das cirurgias.

surgery_id(OpCode,OpType),
surgery(OpType,TAnaes,TSurgery,TClean),
total_time(TAnaes,TSurgery,TATotal),
total_time(TATotal,TClean,TTotal),
get_all_anaesthologist_doctors(LADoctors),
all_staff_occupation(Day,LADoctors,Result1),
sort_staff_by_occupation(Result1,SortedStaff1),
get_all_anaesthologist_nurses(LANurses),
all_staff_occupation(Day,LANurses,Result2),
sort_staff_by_occupation(Result2,SortedStaff2),
get_all_medicalactionassistant_staff(LMAAs),
all_staff_occupation(Day,LMAAs,Result3),
sort_staff_by_occupation(Result3,SortedStaff3),
get_all_instrumenting_nurses(LINurses),
all_staff_occupation(Day,LINurses,Result4),
sort_staff_by_occupation(Result4,SortedStaff4),
get_all_circulating_nurses(LCNurses),
all_staff_occupation(Day,LCNurses,Result5),
sort_staff_by_occupation(Result5,SortedStaff5),

É feita analise das ocupações de cada staff para decidir quais serão escolhidos na próxima cirurgia a marcar.

append([[AD],[AN], [IN], [CN], LDoctors, [MAA]], TotalStaff),
intersect_all_agendas(TotalStaff, Day, ATotalStaff),

São intersectadas todas as agendas de todo o staff escolhido incluindo médicos. O objetivo aqui é arranjar o horário disponível em comum com todos.

intersect_2_agendas(LFAgRoom,ATotalStaff,GlobalAgenda),

Depois é intersectada a agenda do staff com a agenda da sala.
Por fim, é marcada a cirurgia se existir um horário possível.

LPossibilities \= [] ->
(

schedule_first_interval(TTotal, LPossibilities, (TinS, TfinS)),
TfinAnae is TinS + TATotal,
TinDoc is TinS + TAnaes,
TfinDoc is TinDoc + TSurgery,
TinMAA is TinS + TATotal,

retract(agenda_operation_room1(Room, Day, Agenda)),
insert_agenda((TinS, TfinS, OpCode), Agenda, Agenda1),
assertz(agenda_operation_room1(Room, Day, Agenda1)),
insert_agenda_doctors((TinDoc, TfinDoc, OpCode), Day, LDoctors),
insert_agenda_staff((TinS, TfinAnae, OpCode), Day, AD),
insert_agenda_staff((TinS, TfinAnae, OpCode), Day, AN),
insert_agenda_staff((TinMAA, TfinS, OpCode), Day, MAA),
insert_agenda_staff((TinDoc, TfinDoc, OpCode), Day, IN),
insert_agenda_staff((TinDoc, TfinDoc, OpCode), Day, CN)
%write('Cirurgia marcada.')
)
;
Nl

Os horários possíveis contemplam todas as fases de cirurgia numa só, no entanto, a marcação dos horários é individual a cada staff e apenas na fase onde devem atuar. Assim, temos uma agenda completa marcada na sala e parciais a cada staff segundo a especialidade.

Obviamente, a cirurgia a ser marcada é sempre definida pela heurística de tempo referente aos médicos. A cirurgia escolhida pela heurística é depois avaliada e marcada tendo em conta o staff inteiro.

- **Heurística dois**

Para a segunda heurística a heurística de ocupação seguimos exatamente a mesma lógica. Primeiro é determinada a cirurgia a marcar segundo a heurística de ocupação referente aos médicos e depois passa pela avaliação de cada staff e disponibilidade de horário. O código para a inclusão do staff inteiro e fases de cirurgia é igual nesta heurística.

- **Resultados**

Um exemplo dos resultados obtidos para a primeira heurística:

Resultados de agendamento para o dia 20241028: Sala de Operações or1
Agenda para sala or1 no dia 20241028:
[(520,669,so100001),(670,849,so100002),(981,1145,so100003)]
Agendas dos Médicos: Médico da002: []
Médico in002: []
Médico cn002: []

Médico an002: []
Médico maa002: []
Médico d001: [(565,625,so100001),(720,790,m01),(1080,1140,c01)]
Médico d002: [(715,805,so100002),(850,900,m02),(901,960,m02),(1380,1440,c02)]
Médico d003: [(720,790,m01),(910,980,m02),(1026,1101,so100003)]
Médico da001: [(520,625,so100001),(670,805,so100002),(981,1101,so100003)]
Médico an001: [(520,625,so100001),(670,805,so100002),(981,1101,so100003)]
Médico maa001: [(625,669,so100001),(805,849,so100002),(1101,1145,so100003)]
Médico in001: [(565,625,so100001),(715,805,so100002),(1026,1101,so100003)]
Médico cn001: [(565,625,so100001),(715,805,so100002),(1026,1101,so100003)]
Tempo de geracao da solucao:0.0027799606323242188
1true

E para a segunda heurística:

Resultados de agendamento para o dia 20241028: Sala de Operações or1
Agenda para sala or1 no dia 20241028:
[(520,684,so100003),(791,940,so100001),(981,1145,so100005)]
Agendas dos Médicos: Médico da002: []
Médico in002: []
Médico cn002: []
Médico an002: []
Médico maa002: []
Médico d002: [(850,900,m02),(901,960,m02),(1026,1101,so100005),(1380,1440,c02)]
Médico d003:
[(565,640,so100003),(720,790,m01),(910,980,m02),(1026,1101,so100005)]
Médico d001: [(720,790,m01),(836,896,so100001),(1080,1140,c01)]
Médico da001: [(520,640,so100003),(791,896,so100001),(981,1101,so100005)]
Médico an001: [(520,640,so100003),(791,896,so100001),(981,1101,so100005)]
Médico maa001: [(640,684,so100003),(896,940,so100001),(1101,1145,so100005)]
Médico in001: [(565,640,so100003),(836,896,so100001),(1026,1101,so100005)]
Médico cn001: [(565,640,so100003),(836,896,so100001),(1026,1101,so100005)]
Tempo de geracao da solucao:0.0012400150299072266
1true

5. Conclusão

Para concluir, um dos pontos importantes a reter é a capacidade um problema de otimização escalar ao nível de tempo e complexidade conforme as soluções possíveis.

Esse facto faz com que seja quase impossível obter uma solução ideal, através da avaliação, de todas as soluções.

Para um problema da escala factorial como o que enfrentamos aqui, fica claro a necessidade de implementar heurísticas para contornar as limitações físicas do hardware e software.

Verificou-se que a qualidade das heurísticas implementadas não fica aquém da solução ideal, tendo em conta que a diferença da solução final é mínima ao nível de tempo da última cirurgia e a possibilidade de realizar a otimização de grande número de cirurgias é crucial.

Num contexto real, podemos ter dezenas de cirurgias para marcar com várias salas, médicos e staff. Neste caso, escolher a melhor heurística é a melhor forma de otimizar o processo.

Comparando as duas heurísticas desenvolvidas foi possível observar que a velocidade de resposta vai ser quase constante independente do número de cirurgias, isto acontece porque cada sala está limitada ao horário diário e o número de cirurgias possíveis de marcar fica reduzido para uma quantidade limitada.

Por fim, a inclusão do staff não causa implicações ao nível de complexidade nas heurísticas pois continuam a funcionar segundo o critério implementado, sendo necessário apenas avaliar os tempos de cada staff e cruzar com as restantes disponibilidades.