ALGAV – SPRINT 2

Trabalho realizado por:
Ana Silva, N.º 1221133
Diogo Pereira, N.º 1221137
Luís Morais, N.º 1221148
Tiago André Areias dos Santos, N.º 1221003

Breve explicação do código fornecido considerando apenas doutores

O codigo fornecido permite obter a melhor solução para o agendamento tendo em conta apenas a fase de cirurgia e os doutores que nesta atuam, guardando apenas nos horarios do staff e do quarto o tempo destes intervalos com tempo desta fase
O funcionamento para obter a melhor solução baseia-se em volta de três principais predicados, sendo estes:

obtain_better_sol/4
Este predicado começa por registar o tempo atual para calcular a duração do processo. Em seguida, chama "obtain_better_sol1/2", onde são gerados os horários para diferentes permutações das cirurgias. Após a execução, apresenta os horários dos médicos e da sala para a melhor solução encontrada (que acabou mais cedo), o tempo final da última cirurgia realizada e o tempo de execução do processo.

obtain_better_sol1/2
Chamado pelo predicado anterior, este inicializa a melhor solução com o tempo máximo de modo a garantir que a primeira solução válida seja considerada como a melhor. Depois, guarda em LOC os códigos de todas as cirurgias e gera as suas permutações, que serão avaliadas uma a uma. Antes de cada tentativa, limpa e reinicia as variáveis dinâmicas (agenda_staff1/3, agenda_operation_room1/3 e availability/3) com os valores base. Em seguida, agendam-se todas as cirurgias da permutação atual com availability_all_surgeries/3. Caso a solução obtida seja melhor do que a atualmente registada em better_sol/5, esta é atualizada. Finalmente, é forçado um *fail* para explorar as restantes permutações.

availability_all_surgeries/3
Para cada cirurgia da permutação atual, é chamado o predicado availability_operation/5, que identifica os horários disponíveis tanto para os médicos como para a sala. Os intervalos livres nos seus horários são calculados intersetados, removendo posteriormente qualquer intervalo q não suporte o tempo de cirurgia. Após determinar-se o tempo inicial e final da cirurgia, atualizam-se as listas dinâmicas da sala de operações e dos médicos envolvidos.

Em Suma

O predicado obtain_better_sol/4 mede o tempo de execução, chama obtain_better_sol1/2 para avaliar todas as permutações possíveis das cirurgias e apresenta a melhor solução encontrada.

O obtain_better_sol1/2 inicializa a melhor solução, gera permutações das cirurgias, limpa os dados dinâmicos para cada permutação, tenta agendar as cirurgias com availability_all_surgeries/3, e atualiza a melhor solução se necessário.

O availability_all_surgeries/3 verifica os horários livres para médicos e salas, determinando os tempos de cirurgia e atualizando os dados dinâmicos correspondentes.

Estudo de Complexidade

Tendo em conta o código apresentado para o predicado "obtain_better_sol(or1,20241028,AgOpRoomBetter,LAgDoctorsBetter,TFinOp).".

N.º de cirurgias	N.º de soluções	Melhor Horário (incluindo cirurgias) da sala de operações	Tempo final da última cirurgia (minutos)	Tempo para gerar a solução (s)
3	6	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.0966
4	24	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.1739
5	120	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	0.7456
6	720	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	3.847
7	4146	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	17.425
8	40320	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1224	18.689
9	362880	[(520, 579, so100000),	1299	98.850

		(580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]		
10	3628800	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	664.532
11	39916800	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1527.854
12	479001600	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1543.328
13	6227020800	----------	----------	----------

Recorrendo à tabela acima, é possível analisar que o número de soluções corresponde a n! sendo “n” o número de cirurgias para marcar.
Desta forma, podemos concluir que esta solução, apesar de funcionar, não suporta grande volume de cirurgias.

Sendo assim, dada à complexidade da solução, não foi possível calcular a melhor solução tendo em conta 13 cirurgias.

Heurísticas

Adaptação do código para ter em conta todo o staff necessário e todas as fases da cirurgia

Adaptação do Código para Melhor Solução nas Operações Cirúrgicas

Foi solicitada uma adaptação do código fornecido, com o objetivo de melhorar a alocação de um conjunto de operações, considerando agora as várias fases de cada cirurgia e o staff envolvido em cada uma delas.

Desta forma, o staff só terá a sua agenda ocupada com os tempos referentes às fases específicas em que participou, enquanto a sala de cirurgia terá de considerar o horário total da operação. Para este caso de uso, foi escolhido um processo automático de distribuição do staff pelas operações, levando em conta o número de operações já realizadas por cada membro da equipe, sendo prioritariamente escolhidos os staffs com menos operações atribuídas.

Principais alterações e adaptações:

Novas Variáveis Dinâmicas:

n_staff_op/2: Guarda o número de operações realizadas por cada staff, bem como o seu identificador.

agenda_operation_room2/3: Usada para verificar a disponibilidade tanto dos funcionários como da sala de cirurgia.

Predicado obtain_better_sol1/2

Este predicado é semelhante ao original, mas agora inclui a limpeza das novas variáveis dinâmicas. Além disso, inicializa n_staff_op/2 com o número de operações realizadas por cada staff e o seu identificador, através do predicado numberOfOperation/2. O restante do método continua a realizar as permutações possíveis das operações e a executar o predicado availability_all_surgeries/3.

Predicado availability_all_surgeries/3

Este predicado foi adaptado para que, para cada operação da permutação, sejam obtidos os tempos de cada fase da operação, bem como a lista de staff que participará em cada fase, através do predicado findALLNecessaryStaff/4.

Após isso, é copiado o horário atual da sala para agenda_operation_room2/3, e tenta-se encontrar um intervalo de tempo disponível para a cirurgia, onde todos os membros do staff estejam livres durante as suas fases, utilizando o predicado findAvailableTimeForStaffAndRoom/9.

Finalmente, as agendas tanto da sala de cirurgia quanto do staff são atualizadas, bem como o número de operações atribuídas a cada membro do staff, antes de passar para a próxima operação da permutação.

Predicado findALLNecessaryStaff/4

Este predicado tem como objetivo identificar e retornar as listas de staff necessárias para cada fase da operação. Para cada fase da operação (anestesia, limpeza e cirurgia), são feitas as seguintes consultas:

- O predicado findNecessaryStaffByType/3 é utilizado para obter a lista de staff necessário para a fase de anestesia (com especialização de anestesista).

- De forma similar, é usada uma consulta para encontrar o staff necessário para a fase de limpeza (com especialização de assistente).
- Para a fase de cirurgia, são obtidas todas as especializações necessárias através de surgery_Required_Staff/4, removendo as especialidades de anestesista e assistente, e evitando a repetição de outras especializações. Para cada especialidade restante, o staff necessário é identificado e adicionado a uma lista de staff para a fase de cirurgia.

Predicado findAvailableTimeForStaffAndRoom/9:

Este predicado tem como objetivo encontrar um intervalo de tempo que seja simultaneamente disponível para todos os membros da staff necessários para uma operação e para a sala de cirurgia.

O processo inicia-se com a identificação de horários livres para a sala de cirurgia, removendo os intervalos onde não seria possível realizar a cirurgia completa. De seguida, é agendada uma hora inicial e final provisória para a cirurgia com base na disponibilidade da sala. Após esta etapa, verifica-se a disponibilidade de cada tipo de staff para o horário agendado da sala utilizando o predicado availability_staff/5. Este predicado retorna o tempo inicial e final da fase correspondente caso o staff esteja disponível; caso contrário, devolve horários vazios.

A verificação prossegue analisando os tempos retornados para cada fase. Se todos os horários forem válidos (ou seja, nenhum está vazio), significa que todos os membros do staff estão disponíveis naquele período, permitindo assim validar o agendamento. Caso contrário, o horário da tentativa é marcado como ocupado em agenda_operation_room2/3, e o processo reinicia-se para encontrar outro intervalo válido, repetindo o ciclo até que um agendamento compatível seja encontrado.

Imagens dos Predicados Identificados

```
obtain_better_sol1(Room,Day):-

    % Pegar nas operaÃ§Ãµes
    asserta(better_sol(Day,Room,_,_,1441)),
    findall(OpCode,surgery_id(OpCode,_),LOC),!,

    % Fazer permutaÃ§Ãµes
    permutation(LOC,LOpCode),

    % Limpar dados dinamicos
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(agenda_operation_room2(_,_,_)),
    retractall(availability(_,_,_)),
    retractall(n_staff_op(_,_)),

    % Criar copia dos dados base para as variaveis dinamicas e define o numero de cirurgias de cada staff
    findall(_,(agenda_staff(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda)),numberOfOperation(Agenda,N),assertz(n_staff_op(N,D))),_),
    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))),_),

    % Agendar cirurgias
    availability_all_surgeries(LOpCode,Room,Day),

    % Update da melhor soluÃ§Ã£o
    agenda_operation_room1(Room,Day,AgendaR),
    update_better_sol(Day,Room,AgendaR,LOpCode),

    % Proxima permutaÃ§Ã£o
    fail.
```

Figura 1. obtain_better_sol1/2

```
availability_all_surgeries([],_,_).
availability_all_surgeries([OpCode|LOpCode],Room,Day):-

    % Encontrar os tempos das cirurgias
    surgery_id(OpCode,OpType),surgery(OpType,TAnesthesia,TSurgery,TCleaning),
    TotalTime is TAnesthesia+TSurgery+TCleaning,
    % Pegar no Staff necessario para uma operaÃ§ao do tipo OpType
    findALLNecessaryStaff(OpType,ListOfStaffsAnesthesia,ListOfStaffsAssistant,ListOfStaffsSurgery),
    agenda_operation_room1(Room,Day,LAgenda),
    % copiar agenda para agenda temporaria 2
    asserta(agenda_operation_room2(Room,Day,LAgenda)),
    % Tenta encontrar um intervalo para a cirurgia onde todos os staffs estÃ£o disponiveis para as suas fases
    findAvailableTimeForStaffAndRoom(TotalTime,OpType,Room,Day,ListOfStaffsAnesthesia,ListOfStaffsAssistant,ListOfStaffsSurgery,TinS,TfinS),

    retract(agenda_operation_room2(Room,Day,_)),

    TimeFinalForAnaesthesia is TinS + TAnesthesia + TSurgery - 1,
    SurgeryIntialTIme is TinS + TAnesthesia,
    TInitialForCleaning is TimeFinalForAnaesthesia + 1,
    TimeFinalForCleaning is TinS + TSurgery + TAnesthesia + TCleaning - 1,

    % Dar update ao horario de quartos
    retract(agenda_operation_room1(Room,Day,Agenda)),
    insert_agenda((TinS,TfinS,OpCode),Agenda,Agenda1),
    assertz(agenda_operation_room1(Room,Day,Agenda1)),
```

```
% Retorna a lista de staffs necessarios para um SurgeryType
findALLNecessaryStaff(SurgeryType,ListOfStaffsAnesthesia,ListOfStaffsCleaning,ListOfStaffsSurgery):-

    % Econtrar os funcionarios da fase Aneasthesia
    findNecessaryStaffByType(SurgeryType,anaesthetist,ListOfStaffsAnesthesia),

    % Econtrar os funcionarios da fase Cleaning
    findNecessaryStaffByType(SurgeryType,assistant,ListOfStaffsCleaning),

    % Pegar em todas as especialidades da fase Surgery
    findall(DocType,(surgery_Required_Staff(SurgeryType,_,_,DocType)),ListOfDocTypes),
    delete(ListOfDocTypes,anaesthetist,ListOfDocTypes1),
    delete(ListOfDocTypes1,assistant,ListOfDocTypes2),

    % Remover especialidades repetidas
    remove_equals(ListOfDocTypes2,ListOfDocTypes3),


    % Encontrar os funcionarios da fase Surgery
    findall(Res,(member(StaffType,ListOfDocTypes3),findNecessaryStaffByType(SurgeryType,StaffType,Res)),ListOfStaffsSurgery1),
    append(ListOfStaffsSurgery1,ListOfStaffsSurgery).

% Coloca os staff necessarios com StaffType em FinalList
findNecessaryStaffByType(SurgeryType,StaffType,FinalList):-


    findall(Res,(surgery_Required_Staff(SurgeryType,NumberOfStaffs,Role,StaffType) ,
    findByStaff_Role_and_Type(Role,StaffType,Res,NumberOfStaffs)),ListOFStaffs),
    append(ListOFStaffs,FinalList).
 % Encontrar NumberLeft staff com menor numero de operaÃ§Ãµes e com Role e Type
 findByStaff_Role_and_Type(Role,Type,NeededDoctors,NumberLeftOfDoctors):-

     % Pega em todos os staff com Role e Type e mete em LOfDoctors
     findall(D,staff(D,Role,Type),LOfDoctors),

     % Coloca o numero de staff preciso e com menos operaÃ§Ãµes em NeededDoctors
     findByStaff_Role_and_Type(LOfDoctors,NeededDoctors,NumberLeftOfDoctors).

 % CondiÃ§Ã£o de paragem (ja nÃ£o Ã© preciso mais deste tipo de staff)
 findByStaff_Role_and_Type(_,[],0).

 % Pega no numero de staff necessario com menos operaÃ§Ãµes
 findByStaff_Role_and_Type(LOfDoctors,[LowDoc|NeededDoctors],NumberLeftOfDoctors):-
     findLowestSurgeryNumber(LOfDoctors,LowDoc,_),
     NumberLeftOfDoctors1 is NumberLeftOfDoctors - 1,
     delete(LOfDoctors,LowDoc,NewList),
     findByStaff_Role_and_Type(NewList,NeededDoctors,NumberLeftOfDoctors1).

 % CondiÃ§Ã£o de paragem (ultimo staff da lista)
 findLowestSurgeryNumber([D],D,Number):- n_staff_op(Number,D).

 % Encontrar o staff com o menor numero de operaÃ§Ãµes
 findLowestSurgeryNumber([D|LOfDoctors],Res,Number):-
     n_staff_op(N,D),
     findLowestSurgeryNumber(LOfDoctors,Res1,Number1),
     (N<Number1 -> Res = D, Number = N; Res = Res1, Number = Number1).
```

Figura 3. findALLNecessaryStaff/4 e predicados auxiliares a este

```
findAvailableTimeForStaffAndRoom(TotalTime,OpType,Room,Day,ListOfStaffsAnesthesia,ListOfStaffsAssistant,ListOfStaffsSurgery,TinSF,TfinSF):-

    % Pegar no tempo livre da sala
    surgery(OpType,TAnesthesia,TSurgery,TCleaning),
    agenda_operation_room2(Room,Day,Agenda),
    free_agenda0(Agenda,LFAgRoom),
    remove_unf_intervals(TotalTime,LFAgRoom,LRoomAvailability),
    schedule_first_interval(TotalTime,LRoomAvailability,(TinS,TfinS)),
    % Retorna o tempo livre dentro do horario da cirurgia para Aneasthesia
    TimeFinalForAnaesthesia is TinS + TAnesthesia + TSurgery - 1,
    SurgeryTotalTime is TSurgery + TAnesthesia,
    availability_staff(Day,AneasthesiaTime,ListOfStaffsAnesthesia,(TinS,TimeFinalForAnaesthesia),SurgeryTotalTime),
    % Retorna o tempo livre dentro do horario da cirurgia para Surgery
    SurgeryIntialTIme is TinS + TAnesthesia,
    availability_staff(Day,SurgeryTime,ListOfStaffsSurgery,(SurgeryIntialTIme,TimeFinalForAnaesthesia),TSurgery),

% Retorna o tempo livre dentro do horario da cirurgia para Cleaning
    TInitialForCleaning is TimeFinalForAnaesthesia + 1,
    TimeFinalForCleaning is TinS + TSurgery + TAnesthesia + TCleaning - 1,
    availability_staff(Day,CleaningTime,ListOfStaffsAssistant,(TInitialForCleaning,TimeFinalForCleaning),TCleaning),

    % Verificar se os staff conseguem fazer a cirurgia na hora marcada. Caso nÃ£o consiga volta a repetir-se o processo de marcaÃ§Ã£o todo

    % VÃª se algum dos tempos Ã© vazio (staffs nÃ£o disponiveis para aquela hora de cirurgia)
    ( (is_list_empty(AneasthesiaTime) ; is_list_empty(SurgeryTime) ; is_list_empty(CleaningTime)) ->

    % Coloca o inervalo de cirurgia ocupado para tentar o seguinte intervalo possivel
    retract(agenda_operation_room2(Room,Day,Agenda1)),
    insert_agenda((TinS,TfinS,TotalTime),Agenda1,Agenda2),
    assertz(agenda_operation_room2(Room,Day,Agenda2)),

    % Volta a tentar encontrar um intervalo para a cirurgia mas com agenda_operation_room2 atualizada
    findAvailableTimeForStaffAndRoom(TotalTime,OpType,Room,Day,ListOfStaffsAnesthesia,ListOfStaffsAssistant,ListOfStaffsSurgery,TinSF1,TfinSF1),

    % Staff consegue fazer a cirurgia
    TinSF is TinSF1,TfinSF is TfinSF1;
    % Staff consegue fazer a cirurgia
    TinSF is TinS, TfinSF is TfinS,!).
```

Figura 2. findAvailableTimeForStaffAndRoom/9

```
% Encontrar o tempo disponivel para um conjunto de staffs
availability_staff(Day,Result,ListOfStaffs,(TinS,TfinS),TotalTime) :-
    intersect_all_agendas(ListOfStaffs,Day,StaffFreeTime),
    intersect_2_agendas(StaffFreeTime,[(TinS,TfinS)],Result1),
    remove_unf_intervals(TotalTime,Result1,Result).
```

Figura 3. availability_staff/5

Heurística 1 - Priorizar os médicos que começam mais cedo e possuem mais tempo livre

Para este efeito foi necessário acrescentar alguns métodos que não constavam na solução inicial, tais como:

Método inicial: **heuristic_one** - Define as listas dinâmicas iniciais e começa o processamento delas. No fim, escreve no ecrã o tempo de processamento e a hora final da solução.

```
heuristic_one(Room,Day):-
    get_time(Ti),
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),
    retractall(assignment_surgery1(_,_)),

    findall(_,(assignment_surgery(Surgery,Staff),assertz(assignment_surgery1(Surgery,Staff))),_),
    findall(assignment_surgery1(Surgery,Staff),assignment_surgery1(Surgery,Staff),_),
    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),
        purge_unschedulables(LOpCode,LOpCode2,Room,Day),
        heuristic_one_body(LOpCode2,Room,Day),
    get_time(Tf),
    T is Tf-Ti,
    write('Tempo de geracao da solucao:'),write(T),nl,
    agenda_operation_room1(Room,Day,Agendab),
    reverse(Agendab,AgendaR),
    evaluate_final_time(AgendaR,LOpCode2,FinTime1),
    write('Tempo da ultima operaÃ§Ã£o:'),write(FinTime1),nl.
```

Método **cycle_surgery** – Percorre a lista de Ids de Cirurgia com o intuito de obter a duração da Cirurgia e simular o escalonamento de uma Cirurgia.

Método **mock_surgery** – Simula o escalonamento de uma Cirurgia usando os mesmos métodos que um escolonamento real, porém sem guardar nos horários.

```
cycle_surgery([],_,_,[]):-!.
cycle_surgery([H|T],Room,Day,[(H,TinS)|R]):-
    surgery_id(H,OpType),surgery(OpType,_,TSurgery,_),
        mock_surgery(H,Room,Day,TSurgery,TinS,_),
        cycle_surgery(T,Room,Day,R).

mock_surgery(H,Room,Day,TSurgery,TinS,TfinS):-
    availability_operation(H,Room,Day,LPossibilities,_),
    schedule_first_interval(TSurgery,LPossibilities,(TinS,TfinS)).
mock_surgery(_,_,_,_,1400,1400):-!.
```

Método **purge_unschedulables** – Percorre a lista de Ids de Cirurgia com o intuito de remover da lista a considerar todos os que não podem ser escalonados de todo (exemplo, Ids que existem, mas não estão associados a nenhum médico).

```
purge_unschedulables([],[],_,_).
purge_unschedulables([OpCode|L],[OpCode|L2],Room,Day):-
    surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
        mock_surgery(OpCode,Room,Day,TSurgery,TinS,TfinS),
        \+ (TinS = 1400, TfinS = 1400),
        purge_unschedulables(L,L2,Room,Day).
purge_unschedulables([OpCode|L],L2,Room,Day):-
    surgery_id(OpCode,OpType),surgery(OpType,_,TSurgery,_),
        mock_surgery(OpCode,Room,Day,TSurgery,_,_),
        purge_unschedulables(L,L2,Room,Day).
```

Sendo assim, realizamos a análise de complexidade desta heurística, comparando -a com a solução inicial (analisada acima, no ponto Estudo de Complexidade).

N.º de cirurgias	Solução ideal	Tempo final da última cirurgia (minutos)	Tempo final da última cirurgia usando a Heurística (minutos)	Tempo para gerar a melhor solução (s)	Tempo para gerar a solução com a heurística (s)	Solução com a heurística
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)	804	865	0.0966	0.001	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100001), (1000,1059,so099999), (1060, ..., ...), (..., ...)|...]
4	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000,1059,so099999)]	864	1200	0.1739	0.0016	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000,1059 so099999), (1141,1200,so100004)]
5	[(520, 579,so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000,1059,so099999)]	939	1200	0.7456	0.00174	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000,1059,so099999), (1060,1134,so100005), (1141,1200,so100004)].
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	1200	3.847	0.0016	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059,so099999), (1060, 1134,so100005), (1141, ..., ...)]

7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	1290	17.425	0.002	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059,so099999), (1060, 1134,so100005), (1141, ..., ...), (..., ...)]
8	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1224	1441	18.689	0.0226	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (926, 985, so100006), (1000, 1059,so099999), (1060, ..., ...), (..., ...)|...]
9	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1299	1441	98.850	0.0357	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (730, 789, so100009), (791, 865, so100003), (866, 925, so100006), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]
10	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1441	664.532	0.0666	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (730, 789, so100009), (791, 865, so100003), (866, 925, so100006), (926, 985, so100006), (1000, ..., ...), (..., ...)|...].
11	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1441	1527.854	0.1245	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (730, 789, so100009), (791, 865, so100003), (866, 925, so100006), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]
12	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001),	1374	1441	1543.328	0.239	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (730, 789, so100009), (791, 865, so100003), (866, 925, so100006),

	(926, 985, so100006), (1000, ..., ...), (..., ...)|...]					(926, 985, so100012), (1000, ..., ...), (..., ...)|...].

Heurística 2- Priorizar o médico com maior tempo de ocupação

Para implementar esta heurísticas recorremos ao desenvolvimento de outros métodos, como por exemplo:

Método inicial – **heuristic_two** – Define as listas dinâmicas iniciais e começa o processamento delas. No fim, escreve no ecrã o tempo de processamento e a hora final da solução

```
heuristic_two(Room,Day):-
    get_time(Ti),
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),
    retractall(assignment_surgery1(_,_)),

    findall(_,(assignment_surgery(Surgery,Staff),assertz(assignment_surgery1(Surgery,Staff))),_),
    findall(assignment_surgery1(Surgery,Staff),assignment_surgery1(Surgery,Staff),Assignments),
    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
))),_),

        all_staff_occupancy_percentage_header(Room,Day, Assignments),
    get_time(Tf),
    T is Tf-Ti,
    write('Tempo de geracao da solucao:'),write(T),nl,
    agenda_operation_room1(Room,Day,Agendab),
    reverse(Agendab,AgendaR),
    evaluate_final_time(AgendaR,LOpCode2,FinTime1),
    write('Tempo da ultima operaÃ§Ã£o:'),write(FinTime1),nl.
```

Método **all_staff_occupancy_percentage_header** – Processa recursivamente, até a lista de Ids de Cirurgia estar vazia: Chama métodos para receber a lista das percentagens de ocupação do staff e processar a maior percentagem. De seguida obtém a primeira cirurgia da lista desse staff e escalona-a.

Método **all_staff_occupancy_percentage** – Processa a lista de staff e cria a lista das suas percentagens de ocupação

Método **largest_percentage** – Percorre a lista de percentagens de ocupação e retorna a maior

```
all_staff_occupancy_percentage_header(_,_,[]):-!.
all_staff_occupancy_percentage_header(Room,Day,Assignments):-
        findall(agenda_staff1(StaffId,Day,L),agenda_staff1(StaffId,Day,L),StaffList),
        all_staff_occupancy_percentage(StaffList,PL,Assignments),
        largest_percentage(PL,(Staff,_)),

        get_first_surgery(Assignments, Staff, OpCode),
        availability_one_surgery(OpCode,Room,Day),
        remove_surgery(Assignments, OpCode, Assignments2),
        %remove_surgery(Assignments, OpCode, Assignments2),
        all_staff_occupancy_percentage_header(Room,Day,Assignments2).

all_staff_occupancy_percentage([],[], _):-!.
all_staff_occupancy_percentage([agenda_staff1(X,_,Y)|L],[(X,Percentage)|PL],Assignments):-
        staff_occupancy_percentage_header((X,_,Y), Percentage,Assignments),
        all_staff_occupancy_percentage(L,PL,Assignments).

largest_percentage([Last],Last):-!.
largest_percentage([(Xa,Ya)|L],(Xb,Yb)):-
        largest_percentage(L,(Xc,Yc)),
    (   (Ya > Yc, Xb = Xa, Yb = Ya)
    ;   (Ya =< Yc, Xb = Xc, Yb = Yc)
    ).
```

Método **availability_one_surgery** – Escalona uma única cirurgia usando os mesmos métodos que a base.

Método **get_first_surgery** – Obtém a primeira cirurgia por escalonar de um dado staff.

Método **remove_surgery** – Remove uma dada cirurgia da lista de cirurgias.

```
availability_one_surgery(OpCode,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).


get_first_surgery([assignment_surgery1(OpCode, Sid)|_], Sid, OpCode) :- !.
get_first_surgery([_|L], Sid, OpCode) :- get_first_surgery(L, Sid, OpCode).

remove_surgery([], _, []).
remove_surgery([assignment_surgery1(OpCode, _)|L], OpCode, Assignments) :-
    remove_surgery(L, OpCode, Assignments).
remove_surgery([X|L], OpCode, [X|Assignments]):- remove_surgery(L, OpCode, Assignments).
```

Sendo assim, realizamos a análise de complexidade desta heurística, comparando -a com a solução inicial (analisada acima, no ponto Estudo de Complexidade).

N.º de cirurgias	Solução ideal	Tempo final da última cirurgia (minutos)	Tempo final da última cirurgia usando a Heurística (minutos)	Tempo para gerar a melhor solução (s)	Tempo para gerar a solução com a heurística (s)	Solução com a heurística
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000,1059,so099999)	804	1059	0.0966	0.011	[(520, 579, so100000), (580, 669, so100002), (791, 865, so100003), (866, 925, so100001), (1000,1059,so099999)].
4	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000,1059,so099999)]	864	1200	0.1739	0.0113	[(520, 579, so100000), (580, 669, so100002), (791, 850, so100001), (1000, 1059,so099999), (1060, 1134,so100003), (1141,1200,so100004)].
5	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001),	939	1200	0.7456	0.00079	[(520, 579, so100000), (580, 669, so100002), (791, 865, so100003), (866, 925, so100001), (1000, 1059,so099999), (1060, 1134,so100005), (1141,1200,so100004)]

	(1000, 1059, so099999)]					
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	1275	3.847	0.0106	[(520, 579, so100000), (580, 669, so100002), (791, 850, so100001), (851, 910, so100006), (1000, 1059,so099999), (1060, 1134,so100003), (1141, 1200,so100004), (1201, ..., ...)]
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	1290	17.425	0.0009	[(520, 579, so100000), (580, 654, so100003), (655, 744, so100002), (791, 850, so100001), (851, 910, so100006), (1000, 1059,so099999), (1060, 1134,so100005), (1141, ..., ...), (..., ...)]
8	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1224	-----	18.689	-----	-----
9	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1299	-----	98.850	-----	-----
10	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	-----	664.532	-----	-----
11	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003),	1374	-----	1527.854	-----	-----

	(866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]					
12	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	-----	1543.328	-----	-----

Analisando a tabela, podemos ver que, inicialmente, esta heurística consegue obter um tempo de execução muito inferior à solução inicial.

Para os exemplos testados, a partir de 8 cirurgias, o programa não encontra mais horários validos. Se os as durações das cirurgias fossem menores, seria possível obter mais resultados. Da mesma forma, aumentando a sua duração, o número de cirurgias que seria possível escalonar seria menor.

Conclusão

Em suma, comparando as Heurísticas apresentadas e a solução inicial, apresentada nas aulas de ALGAV, é possível concluir ambas tempos de processamento bastante inferiores à solução ideal. Desta forma, demonstram ser soluções mais eficientes.

Em relação à heurística 1 apresenta maior flexibilidade devido ao baixo tempo de processamento mesmo para 12 cirurgias, como é possível observar pela analise da tabela. Assim, vemos que esta solução é eficiente para casos onde o objetivo é escalonar bastantes cirurgias rapidamente.

Por outro lado, a heurística 2 apresenta algumas restrições relativamente à capacidade, já que depende do tempo de cada cirurgia. Ademais, é útil quando o objetivo é priorizar a ocupação de médicos com alta carga de trabalho.