Planning Module Report

Membros da equipa			
José Teixeira - 1210965	Rodrigo Almeida - 1220776	Rafael Santos - 1221267	Alexandre Costa - 1211225

User Stories		
6.3.1 As an Admin, I want to obtain the better scheduling of a set of operations (surgeries) in a certain operation room in a specific day. The better scheduling is considered as the sequence of operations that finishes early. Note that surgeries have constraints (e.g. number of doctors or other staff), namely concerning the time availability of staff during the day. The approach may be generating all surgeries' sequences and select the better, and this is possible till a certain dimension (number of surgeries). The user must have a user interface to start the process (enter any additional parameters the planning algorithm needs, e.g., room number, date). The system will then generate the plan and show it to the user on the screen. It is acceptable that the UI blocks while waiting for the planning module response.	6.3.2 As an Admin, I want to know till what dimension in terms of number of surgeries is possible to ask for the better solution. Perform a complexity analysis of the problem to understand to which dimension it is feasible to ask for the better solution. Document your results and findings.	6.3.3 As an Admin, I want to obtain a good schedule, not necessarily the better, in useful time to be adopted. The system generates a "good" (non-optimal but efficient) schedule using heuristics or informed methods (e.g., greedy algorithms, rule-based scheduling). The system prioritizes generating a schedule that is close to optimal but does so within a reasonable time frame (e.g., under 30 seconds). The user must have a user interface to start the process (enter any additional parameters the planning algorithm needs, e.g., room number, date, which heuristic to use). The system will then generate the plan and show it to the user on the screen. It is acceptable that the UI blocks while waiting for the planning module response.

Tradução das horas:

0 - 0:00 | 60 - 1:00 | 100 - 1:40 | 120 - 2:00 | 180 - 3:00 | 200 - 3:20 | 240 - 4:00 | 300 - 5:00 | 360 - 6:00 | 400 - 6:40 | 420 - 7:00 | 480 - 8:00 | 500 - 8:20 | 540 - 9:00 | 600 - 10:00 | 660 - 11:00 | 700 - 11:40 | 720 - 12:00 | 780 - 13:00 | 800 - 13:20 | 840 - 14:00 | 900 - 15:00 |

960 - 16:00 | 1000 - 16:40 | 1020 - 17:00 | 1080 - 18:00 | 1100 - 18:20 | 1140 - 19:00 | 1200 - 20:00 | 1260 - 21:00 | 1300 - 21:40 | 1320 - 22:00 | 1380 - 23:00 | 1400 - 23:20

Conteúdo

Points to cover				
1) Brief explanation of the basic code to schedule just considering doctors (the part presented in the first 3 TP classes) – 10%	**2)** Complexity Study (6.3.2; just with doctors)– 15%	**3)** Two Heuristics, including the comparison with the better solution (6.3.3)– 30%	**4)** Adaptation of the method to include all necessary staff and all phases of the operation (6.3.1). – 35%	**5)** Conclusions - 10%

1. Explanation of the base code

Predicado	Argumentos	Exemplo	Explicação
staff	(StaffID, Role, Specialization, CanPerformSurgeryType)	staff(d001, doctor, orthopaedist , [so2,so3]).	Está a ser definido um staff com ID 'd001', com a role de 'doctor', com especialização em 'orthopaedist' e que consegue realizar os tipos de cirurgia 'so2' e 'so3'
agenda_staff	(StaffID, Date, Agenda)	agenda_staff(d001, 20241028, [(720, 790, m01), (1080, 1140, c01)]).	Especifica que o staff com ID d001 tem eventos agendados no dia 20241028, sendo o primeiro das 12:00 às 12:50 com o código m01 e o segundo das 18:00 às 19:00 com o código c01.
timetable	(StaffID, Day, (StartTime,EndTime))	timetable(d001, 20241028, (480,1200)).	Especifica que o staff com ID d001 no dia 20241028, só trabalha entre as 8:00 e as 20:00
surgery	(SurgeryType, PreparationTime, Duration,	surgery(so2, 45, 60,	Define que a cirurgia do tipo so2 exige 45 minutos de

	CleaningTime)	45).	preparação, 60 minutos de operação, e 45 minutos para limpeza.
surgery_id	(SurgeryID, SurgeryType)	surgery_id(so100001, so2).	Especifica que a cirurgia com ID so100001 é do tipo so2.
assignment_surgery	(SurgeryID, StaffID)	assignment_surgery(so100001, d001).	Indica que a cirurgia com ID so100001 foi atribuída ao staff com ID d001.
find_free_agendas	(Date)	find_free_agendas(20241028)	Encontra as agendas livres para cada membro da equipa para uma data específica.
free_agenda_staff0	([(Tin, Tfin, TaskID) | LT], LFA)	free_agenda_staff0([(720, 840, m01), (1080, 1200, c01)], LFA). % LFA = [(0, 719), (841, 1079), (1201, 1440)]	Este predicado calcula os intervalos livres de tempo em que um membro da equipe está disponível, com base nos horários ocupados especificados em uma lista.
free_agenda_staff1	([(Tin, Tfin, TaskID) | LT], LT1)	free_agenda_staff1([(720, 840, m01), (900, 960, m02)], LFA). % LFA = [(0, 719), (841, 899), (961, 1440)]	Continua o cálculo dos intervalos livres de tempo a partir de uma lista de intervalos ocupados. É usado internamente por free_agenda_staff0/2 para gerar os intervalos livres entre os compromissos.
adapt_timetable	(StaffID, Date, LFA, LFA2)	timetable(d001, 20241028, (480, 1200)), adapt_timetable(d001, 20241028, [(0, 719), (841, 1079), (1201, 1440)], LFA2). % LFA2 = [(480, 719), (841, 1079)]	Ajusta a lista de horários livres conforme o horário de trabalho diário de cada membro da equipe, usando o horário especificado no timetable
treatin	(InTime, LFA, LFA1)	treatin(480, [(0, 719), (841, 1079)], LFA1). % LFA1 = [(480, 719), (841, 1079)]	Ajusta a lista de intervalos livres para que comece a partir do horário de início do trabalho, descartando intervalos antes do início.

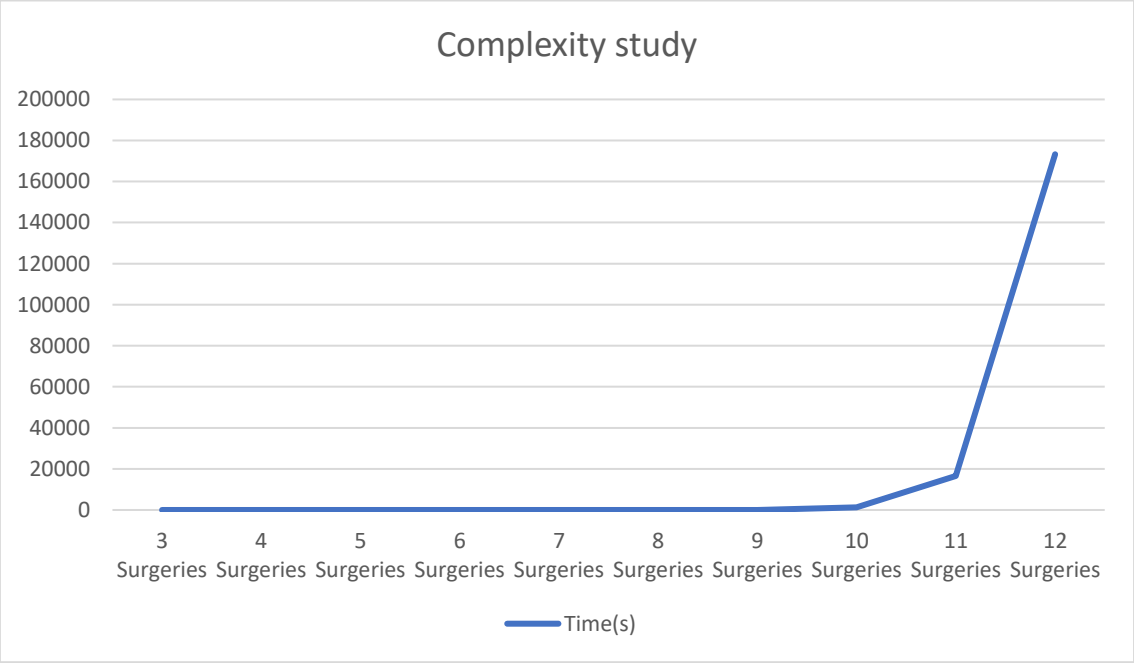
treatfin	(FinTime, LFA, LFA2)	treatfin(1200, [(480, 719), (841, 1440)], LFA2). % LFA2 = [(480, 719), (841, 1200)]	Ajusta a lista de intervalos livres para que termine no horário de fim de trabalho, removendo intervalos após o término do trabalho.
intersect_all_agendas	([Name|LNames], Date, LI)	find_free_agendas(20241028), intersect_all_agendas([d001, d002, d003], 20241028, LI). % LI = [(841, 899), (1081, 1079)]	Este predicado encontra os horários em que todos os membros da equipe especificados estão disponíveis ao mesmo tempo, em uma data específica.
availability	(StaffID, Day, FreeTime).	availability(d001, 20241028, [(500, 600), (700, 800)]).	Define que o staff com o ID d001 está disponível no dia 20241028 nos horários das 8:20 às 10:00 e das 11:40 às 13:00.
intersect_2_agendas	(LA, LB, LI)	intersect_2_agendas([(480, 719), (841, 1079)], [(600, 900), (950, 1200)], LI). % LI = [(600, 719), (841, 900), (950, 1079)]	Encontra os intervalos de tempo que se sobrepõem entre duas listas de horários disponíveis.
intersect_availability	((Início1, Fim1), LB, LI, LB1)	intersect_availability((480, 719), [(600, 900), (950, 1200)], LI, LB1). % LI = [(600, 719)], LB1 = [(950, 1200)]	Calcula a interseção (sobreposição) de um único intervalo de tempo de uma lista com todos os intervalos de outra lista.
min_max	(A, B, Min, Max)	min_max(480, 600, Min, Max). % Min = 480, Max = 600	Este predicado auxiliar calcula o valor mínimo e o máximo entre dois valores fornecidos.
agenda_operation_room	(RoomID, Date, Schedule)	agenda_operation_room(room1, 20241028, [(720, 840, s01), (900, 1020, s02)]).	Define o agendamento de uma sala de operação listando os intervalos de tempo em que ela está ocupada.
obtain_better_sol	(Solutions, BestSolution, Criteria)	obtain_better_sol([[(720, 840), (900, 1020)], [(800, 900), (920, 1100)]], BestSolution, minimize_conflicts). % BestSolution = [(720, 840), (900, 1020)]	Este predicado seleciona a melhor solução (como o agendamento mais otimizado) de uma lista de soluções possíveis, com base em critérios específicos.

availability_all_surgeries	([RoomID | Rooms], Date, Availability)	availability_all_surgeries([room1, room2], 20241028, Availability). % Availability = [(480, 719), (841, 899), (1081, 1200)]	Calcula a disponibilidade para todas as cirurgias em uma data específica em uma lista de salas.

2. Complexity Study

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
3	6	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.055301904678344 73
4	24	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.252915143966674 8
5	120	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.182440996170043 95
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	4.766103029251099
7	5040	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	10.57641005516052 2
8	40320	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	40.66297888755798 3
9	362880	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (790, 849, so100009), (850, 909, so100001),	1299	152.3664948940277 1

		(910, 969, so100006), (1000, ..., ...), (..., ...)|...]		≈ 2,5 minutos
10	3,628,800	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1316.9835941791534 ≈ 21 minutos
11	39,916,800	Não apresentou (devido a um bug)	1441	16639.72731685638 ≈ 4 horas e 40 minutos
12	479,001,600	Não apresentou (devido a um bug)	1441	173276.0112609863 ≈ 48 horas



O gráfico serve apenas para auxiliar visualmente, mostrando que estamos diante de um acréscimo fatorial neste tipo de função (f(x) = t), pois o tempo (t) aumenta de forma fatorial com o crescimento do número de cirurgias (x).

3. Heuristics and comparison

Para lidar com os desafios de escalabilidade causados pela complexidade fatorial do algoritmo exato de permutação, foram desenvolvidas duas heurísticas distintas. Com essa abordagem, busca-se criar soluções de agendamento de forma mais ágil, ainda que não sejam as ideais. As heurísticas permitem obter boas alternativas em um período aceitável, o que é crucial em situações onde encontrar uma solução exata seria impraticável devido a restrições de tempo ou ao grande volume de cirurgias a serem programadas.

Breakdown of the heuristics:

Heurstica 1:

“The next surgery to consider is one for the doctor that is available early. What isto be available early is to have the suffivcient time to complete the surgery early(for exemple if the doctor start the timetable at 8:00am and has a meeting at8:30 he/she will not be available at 8:00am for a surgery of type so2 because ittakes 60 minutes)”

1. **Função Principal (obtain_better_sol_heuristica1)**:

- Inicia a contagem de tempo.
- Chama obtain_better_sol1_heuristica1 para encontrar uma melhor solução.
- Recupera e exibe a melhor solução armazenada.
- Calcula o tempo total para gerar a solução.

2. **Heurística de Otimização (obtain_better_sol1_heuristica1)**:

- Define uma solução inicial pessimista usando asserta(better_sol(..., 1441)), onde 1441 indica que não há cirurgia agendada (um valor alto de "tempo final").

- Coleta todos os códigos de cirurgias e os ordena de acordo com a disponibilidade dos médicos (ordenar_cirurgias_por_disponibilidade).
- Limpa e atualiza as agendas dos médicos e salas de operações, estabelecendo novas disponibilidades.
- Atualiza a melhor solução com a agenda resultante.

3. **Ordenação de Cirurgias pela Disponibilidade dos Médicos (ordenar_cirurgias_por_disponibilidade)**:

- Para cada cirurgia, calcula o menor tempo disponível de cada médico associado usando calcular_disponibilidade_inicial.
- Gera uma lista de cirurgias organizada pela menor disponibilidade dos médicos para priorizar as cirurgias que podem ser agendadas mais cedo.

4. **Cálculo da Disponibilidade Inicial do Médico (calcular_disponibilidade_inicial)**:

- Consulta a agenda do médico para o dia específico e obtém o primeiro horário disponível.
- Calcula o horário de início da cirurgia, considerando a agenda existente do médico.

5. **Cálculo do Primeiro Horário Disponível (calcular_primeiro_horario_disponivel)**:

- Avalia cada compromisso na agenda do médico.
- Define o primeiro horário em que o médico estará livre para realizar uma cirurgia.

Código da heurística:

```
obtain_better_sol_heuristical(Room, Day, AgOpRoomBetter, LAgDoctorsBetter, TFinOp):-
    get_time(Ti),
    (obtain_better_sol1_heuristical(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.

obtain_better_sol1_heuristical(Room, Day):-
    asserta(better_sol(Day, Room, _, _, 1441)),
    findall(OpCode, surgery_id(OpCode, _), LOC), !,
    ordenar_cirurgias_por_disponibilidade(LOC, Day, LOpCode), % Ordenar cirurgias pela disponibilidade do medico
    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.
```

```
ordenar_cirurgias_por_disponibilidade(LOpCode, Day, LOpCodeOrdenada):-
    findall((TempoDisponivel, OpCode),
            (
                member(OpCode, LOpCode),
                findall(TempoDisponivelMedico,
                        (
                            assignment_surgery(OpCode, Medico),
                            calcular_disponibilidade_inicial(Medico, Day, TempoDisponivelMedico)
                        ),
                        ListaTemposDisponiveis),
                min_list(ListaTemposDisponiveis, TempoDisponivel)
            ),
            ListaOpCodesComDisponibilidade),
    sort(ListaOpCodesComDisponibilidade, ListaOpCodesComDisponibilidadeOrdenada),
    findall(OpCode, member((_, OpCode), ListaOpCodesComDisponibilidadeOrdenada), LOpCodeOrdenada).

calcular_disponibilidade_inicial(Medico, Dia, TempoDisponivel):-
    timetable(Medico, Dia, (Inicio, _)),
    agenda_staff(Medico, Dia, Agenda),
    calcular_primeiro_horario_disponivel(Agenda, Inicio, TempoDisponivel).

calcular_primeiro_horario_disponivel([], Inicio, Inicio).
calcular_primeiro_horario_disponivel([(InicioCompromisso, _, _) | _], Inicio, TempoDisponivel):-
    InicioCompromisso > Inicio,
    TempoDisponivel is Inicio.
calcular_primeiro_horario_disponivel([(_, FimCompromisso, _) | Resto], Inicio, TempoDisponivel):-
    FimCompromisso < Inicio,
    calcular_primeiro_horario_disponivel(Resto, Inicio, TempoDisponivel).
```

Heurstica 2:

“The next surgery is for doctor that will need to be occupied the higher percentageof time. For exemple, if for a certain doctor it remains 2 surgeries of 60 minutes and 1 surgery of 90 minutes, a total of 210 minutes, and if we sum the free times of that Doctor and this results in 420 minutes, this means that he will need to beoccupied 50% of the remaining time. Let’s suppose that other doctors have lessthan this percentagem, the better is to start with a surgery involving the mostoccupied doctor. Note that these calculations need to be done step by step (after the prévios Schedule, let us suppose of the surgery of 90 minutes, theoccupation of the doctor will be 120/330 = ~36,4%, and then other doctor with40% of occupation can be more critical”

1. **Função Principal**: Inicia a contagem de tempo, chama a heurística para encontrar a melhor solução (obtain_better_sol2), e calcula o tempo total da última cirurgia realizada. Em seguida, exibe o tempo total de execução.

2. **Heurística de Otimização (obtain_better_sol2)**:

- Define uma solução inicial pessimista com alta ocupação (1441%).
- Gera permutações das cirurgias e limpa as agendas anteriores.
- Atualiza a disponibilidade dos médicos, verificando horários livres.
- Calcula e atualiza a melhor solução com base na ocupação máxima dos médicos.

3. **Atualização de Melhor Solução (update_better_sol2)**:

- Verifica se a nova alocação tem menor ocupação máxima dos médicos do que a solução anterior.

- Se a ocupação for menor, salva a nova agenda como a "melhor solução" e exibe a ocupação máxima atualizada.

4. **Cálculo de Ocupação (calcular_ocupacao)**:

- Avalia a percentagem de tempo ocupado de cada médico no dia, considerando o total de horas trabalhadas e o tempo agendado para cirurgias.

Código da heurística:

```
obtain_better_sol_heuristica2(Room, Day, AgOpRoomBetter, LAgDoctorsBetter, TFinOp):-
    get_time(Ti),
    (obtain_better_sol2(Room, Day); true),
    retract(better_sol(Day, Room, AgOpRoomBetter, LAgDoctorsBetter, _)), % Recupera os valores de melhor solução
    calcular_tempo_final_realizado(AgOpRoomBetter, LAgDoctorsBetter, TFinOp), % Calcula o tempo da última cirurgia realizada
    get_time(Tf),
    T is Tf - Ti,
    write('Tempo de geracao da solucao:'), write(T), nl.

calcular_tempo_final_realizado(AgOpRoom, LAgDoctors, TFinOp):-
  findall(Tfin,
    (
      member((_, Tfin, OpCode), AgOpRoom), % Para cada operação na sala
      member((_, AgendaDoctor), LAgDoctors), % Verifica a agenda de cada doutor
      member((_, _, OpCode), AgendaDoctor) % Confirma se a operação está na agenda de algum doutor, **ignorando os tempos de inicio e fim**
    ), ListaFins), % Coleta os tempos finais das cirurgias realizadas
  (ListaFins = [] -> TFinOp = 0 ; max_list(ListaFins, TFinOp)). % Retorna o maior tempo final ou 0 se não houver cirurgias realizadas.


obtain_better_sol2(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_sol2(Day,Room,AgendaR,LOpCode),
                fail.


update_better_sol2(Day, Room, Agenda, LOpCode):-
    better_sol(Day, Room, _, _, MelhorPercentagem),
    calcular_percentagem_maxima(Day, LOpCode, PercentagemMaxima),
    (PercentagemMaxima < MelhorPercentagem ->
        retract(better_sol(_, _, _, _, _)),
        findall(Doctor, assignment_surgery(_, Doctor), LDoctors1),
        remove_equals(LDoctors1, LDoctors),
        list_doctors_agenda(Day, LDoctors, LDAgendas),
        asserta(better_sol(Day, Room, Agenda, LDAgendas, PercentagemMaxima)),
        write('Melhor solução atualizada: '),nl, write('Percentagem máxima = '), write(PercentagemMaxima), nl
    ; true).

calcular_percentagem_maxima(Day, LOpCode, PercentagemMaxima):-
    findall(Percentagem, (member(OpCode, LOpCode), assignment_surgery(OpCode, Medico), calcular_ocupacao(Medico, Day, Percentagem)), ListaPercentagem)
,
    max_list(ListaPercentagem, PercentagemMaxima).

calcular_ocupacao(Medico, Dia, Percentagem):-
    agenda_staff1(Medico, Dia, Agenda),
    calcular_tempo_ocupado(Agenda, TempoTotalOcupado),
    timetable(Medico, Dia, (Inicio, Fim)),
    TempoDisponivel is Fim - Inicio,
    Percentagem is (TempoTotalOcupado / TempoDisponivel) * 100.

calcular_tempo_ocupado([], 0).
calcular_tempo_ocupado([(Tin, Tfin, _) | Resto], TempoTotalOcupado):-
    Duracao is Tfin - Tin,
    calcular_tempo_ocupado(Resto, TempoResto),
    TempoTotalOcupado is Duracao + TempoResto.
```

Heuristics complexity study:

Heuristic 1)

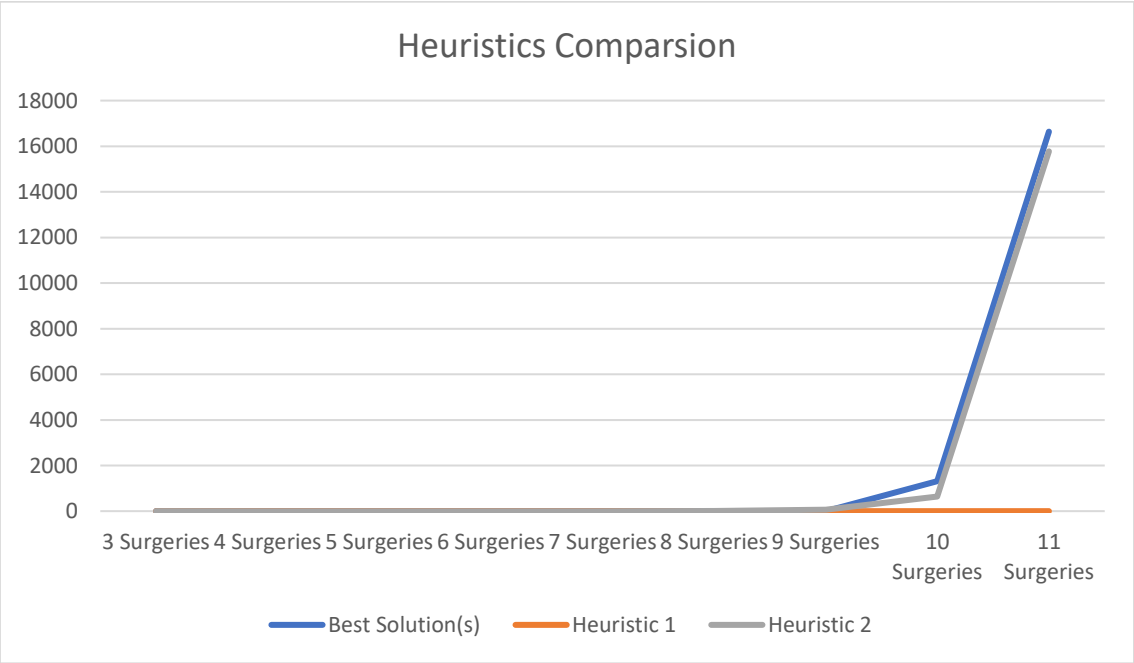
N. of Surg.	Optimal Solution	Final Time for the last Surgery in generate all select better (minutes)	Time to generate the best solution (s)	Solution with the first Heuristic	Final Time for the last Surgery Using first Heuristic (minutes)	Time to generate the first heuristic solution (s)
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.055301904678344 73	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999)]	865	0.03045797348022461
4	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.2529151 4396667 48	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100004), (700, 789, so100002), (791, 865, so100003), (1000, 1059, so099999)]	865	0.020440101623535156
5	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.182440996170043 95	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100004), (700, 789, so100002), (791, 865, so100003), (1000, 1059, so099999), (1060, 1134, so100005)]	1134	0.024897098541259766
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	4.7661030 2925109 9	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100004), (700, 789, so100002), (791, 850, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1135, ..., ...)]	1209	0.032131195068359375
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	10.576410 055160522	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100004), (700, 789, so100002), (791, 850, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1135, ..., ...), (..., ...)]	1299	0.02721714973449707
8	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	40.662978 887557983	Não apresentou (devido a um bug)	1441	0.006838083267211914
9	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (790, 849, so100009), (850, 909, so100001),	1299	152.366494 894027 71	Não apresentou (devido a um bug)	1441	0.006727933883666992

	(910, 969, so100006), (1000, ..., ...), (..., ...)|...]		≈ 2,5 minutos			
10	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1316.98359 4179153 4 ≈ 21 minutos	Não apresentou (devido a um bug)	1441	0.012515068054199219
11	Não apresentou (devido a um bug)	1441	16639.727 3168563 84 ≈ 4 horas e 40 minutos	Não apresentou (devido a um bug)	1441	0.006591081619262695
12	Não apresentou (devido a um bug)	1441	173276.01126098 63≈ 48 horas	Não apresentou (devido a um bug)	1441	0.009422779083251953

Heuristic 2)

N. of Surg.	Optimal Solution	Final Time for the last Surgery in generate all select better (minutes)	Time to generate the best solution (s)	Solution with the second Heuristic	Final Time for the last Surgery Using second Heuristic (minutes)	Time to generate the second heuristic solution (s)
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.05530190 467834473	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999)]	704	0.0017969608306884766
4	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.2529151 439666748	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999), (1141, 1200, so100004)]	1020	0.006867885589599609
5	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.18244099 617004395	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999), (1060, 1134, so100005), (1141, 1200, so100004)]	1185	0.028509855270385742
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]	999	4.7661030 29251099	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060, 1134, so100005), (1141, ..., ...)]	1185	0.14905786514282227
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	10.576410 055160522	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060, 1134,	1275	1.101658821105957

				so100005), (1141, ..., ...), (..., ...)]		
8	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	40.662978887557983	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100008), (700, 789, so100002), (791, 880, so100007), (881, 940, so100006), (1000, 1059, so099999), (1060, ..., ...), (..., ...)|...]	1275	7.745025873184204
9	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (790, 849, so100009), (850, 909, so100001), (910, 969, so100006), (1000, ..., ...), (..., ...)|...]	1299	152.36649489402771 ≈ 2,5 minutos	[(520, 579, so100000), (580, 639, so100001), (640, 699, so100008), (700, 759, so100009), (791, 880, so100007), (881, 940, so100006), (1000, 1059, so099999), (1060, ..., ...), (..., ...)|...]	1275	63.09666895866394 ≈ 1 minuto
10	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1374	1316.9835941791534 ≈ 21 minutos	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1320	642.4712359905243 ≈ 11 minutos
11	Não apresentou (devido a um bug)	1441	16639.727316856384 ≈ 4 horas e 40 minutos	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 759, so100009), (791, 865, so100003), (866, 925, so100001), (926, 985, so100006), (1000, ..., ...), (..., ...)|...]	1320	15776.1679426289 ≈ 4 horas e 30 minutos



4. Adaptation of the methods

Para a realização deste ponto, alguns dos predicados disponibilizados no moodle foram alterados:

1. **schedule_all_surgeries/2** - É o predicado principal para o agendamento de cirurgias. Limpa os dados dinâmicos, cria novos dados com base no dia especificado, define a disponibilidade do staff e, em seguida, agenda todas as cirurgias para o dia e sala especificados, chamando availability_all_surgeries/3 para cada operação.

```
schedule_all_surgeries(Room, Day):-
clean_dynamic_data,
create_dynamic_data(Day),
define_staff_availability,
findall(OpCode, surgery_id(OpCode,_), LOpCode),
availability_all_surgeries(LOpCode, Room, Day),!.
```

2. **availability_all_surgeries/3** - Itera sobre cada cirurgia, obtém os tempos das fases da cirurgia (anestesia, cirurgia e limpeza), e chama predicados auxiliares (**availability_operation/7** e **calculate_intervals/8**) para verificar a disponibilidade do staff e calcular os intervalos de tempo. Atualiza a agenda da sala e dos profissionais de acordo com os tempos calculados para cada fase.

```
availability_all_surgeries([],_,_).
availability_all_surgeries([OpCode|LOpCode], Room, Day):-
surgery_id(OpCode, OpType), surgery(OpType, TAnesthesia, TSurgery, TCleaning),
availability_operation(OpCode, Room, Day, Interval, LDoctorsSurgery, LStaffAnesthesia, LStaffCleaning),
calculate_intervals(Interval, TAnesthesia, TSurgery, TCleaning, MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess),
retract(agenda_operation_room1(Room, Day, Agenda)),
insert_agenda((MinuteStartAnesthesia, MinuteEndProcess, OpCode), Agenda, Agenda1), assertz(agenda_operation_room1(Room, Day, Agenda1)),
insert_agenda_staff((MinuteStartSurgery,MinuteStartCleaning, OpCode), Day, LDoctorsSurgery),
insert_agenda_staff((MinuteStartAnesthesia, MinuteStartCleaning, OpCode), Day, LStaffAnesthesia),
insert_agenda_staff((MinuteStartCleaning, MinuteEndProcess, OpCode), Day, LStaffCleaning),
availability_all_surgeries(LOpCode, Room, Day).
```

3. **availability_operation/7** - Define a lista de profissionais necessários para cada etapa da operação e usa **intersect_all_agendas/3** para encontrar horários comuns entre o staff. Verifica a disponibilidade da sala e chama **find_first_interval/8** para encontrar o

primeiro intervalo adequado para a cirurgia.

```
availability_operation(OpCode, Room, Day, Interval, LDoctorsSurgery, LStaffAnesthesia, LStaffCleaning):-
surgery_id(OpCode, OpType), surgery(OpType, TAnesthesia, TSurgery, TCleaning),
findall(Staff, (assignment_surgery(OpCode, Staff), staff(Staff,__, surgeon,_)), LDoctorsSurgery),
findall(Staff, (assignment_surgery(OpCode, Staff), staff(Staff,_,anesthesia,_)), LStaffAnesthesia),
findall(Staff, (assignment_surgery(OpCode, Staff), staff(Staff,_,cleaning,_)), LStaffcleaning),
intersect_all_agendas(LDoctorsSurgery, Day, LSurgery), intersect_all_agendas(LStaffAnesthesia, Day, LAnesthesia),
intersect_all_agendas(LStaffCleaning, Day, LCleaning),
agenda_operation_room1(Room, Day, LAgenda),
free_agenda0(LAgenda, LFAgRoom),
find_first_interval(LAnesthesia, LSurgery, LCleaning, LFAgRoom, TAnesthesia, TSurgery, TCleaning, Interval).
```

4. **find_first_interval/8 e find_first0/8** - **find_first_interval/8** chama **find_first0/8** para localizar o primeiro intervalo possível de operação. **find_first0/8** verifica se o staff e a sala estão disponíveis para cada fase da cirurgia, garantindo que o agendamento respeite as condições para anestesistas, cirurgiões e técnicos de limpeza.

```
find_first_interval(LAnesthesia, LSurgery, LCleaning, LFAgRoom, TAnesthesia, TSurgery, TCleaning, Interval) :-
    once(find_first0(LAnesthesia, LSurgery, LCleaning, LFAgRoom, TAnesthesia, TSurgery, Tcleaning, Interval)).

find_first0(LAnesthesia, LSurgery, LCleaning, RoomsAvailable, TAnesthesia, TSurgery, TCleaning, Interval):-
member((RoomStart, RoomEnd), RoomsAvailable),
TotalTime is TAnesthesia + TSurgery + TCleaning,
MaxStart is RoomEnd - TotalTime,
between(RoomStart, MaxStart, StartAnesthesia),
StartAnesthesia + TAnesthesia + TSurgery + TCleaning =<1400,
StartSurgery is StartAnesthesia + TAnesthesia,
StartCleaning is StartSurgery + TSurgery,
member((AnesthesiaStart, AnesthesiaEnd), LAnesthesia),
AnesthesiaStart =< StartAnesthesia,
AnesthesiaEnd >= (StartAnesthesia + TAnesthesia + TSurgery),
member((SurgeryStart, SurgeryEnd), LSurgery),
SurgeryStart =< StartSurgery,
SurgeryEnd >= (StartSurgery + TSurgery),
member((CleaningStart, CleaningEnd), LCleaning),
CleaningStart =< StartCleaning,
CleaningEnd >= (StartCleaning + TCleaning),
RoomStart =< StartAnesthesia,
RoomEnd >= (StartAnesthesia + TAnesthesia + TSurgery + TCleaning),
EndProcess is StartAnesthesia + TAnesthesia + TSurgery + TCleaning, Interval = (StartAnesthesia, EndProcess).
```

5. **calculate_intervals/8** - Calcula os tempos de início e fim de cada fase da operação (anestesia, cirurgia, limpeza), retornando esses valores para uso na atualização das agendas.

```
calculate_intervals((Start, End), TAnesthesia, TSurgery, TCleaning,
MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess):-
MinuteStartAnesthesia = Start,
MinuteStartSurgery is MinuteStartAnesthesia + TAnesthesia, MinuteStartCleaning is MinuteStartSurgery + TSurgery,
MinuteEndProcess is MinuteStartCleaning + TCleaning,
MinuteEndProcess =< End.
```

5. Conclusions

Ao analisarmos a complexidade do problema de agendamento, observamos que o tempo necessário para encontrar a melhor solução aumenta drasticamente com o número de cirurgias. Esse crescimento fatorial, um dos mais abruptos em matemática, reflete-se na necessidade de considerar todas as possíveis combinações para obter o agendamento ideal. Nossos testes mostraram que, ao atingir 12 cirurgias, o tempo para calcular a melhor solução ultrapassa 24 horas, tornando inviável a obtenção antecipada do horário de trabalho necessário para o staff.

No contexto estudado, onde temos até 56 horas para gerar a solução, concluímos que 12 é o número máximo de cirurgias para o qual ainda conseguimos um agendamento ideal. Para um número maior de cirurgias, a complexidade temporal impede que a solução esteja pronta antes do início das atividades. Assim, ao ultrapassarmos o limite de 12 cirurgias, torna-se mais eficaz optar por heurísticas, que, conforme nossos testes, oferecem uma alternativa viável com tempos de execução significativamente inferiores.

Entre as heurísticas testadas, a primeira mostrou-se particularmente eficaz, gerando agendamentos em tempos razoáveis sem comprometer demasiadamente a qualidade. Dessa forma, as heurísticas serão preferíveis em contextos onde o tempo é um fator crítico, oferecendo uma solução prática e eficiente para o desafio de agendamento de múltiplas cirurgias.