P.PORTO
INSTITUTO POLITECNICO DO PORTO Portugal

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

ALGAV

SPRINT 2

3DD GRUPO 20:

Diogo Tomás Rodrigues Ferreira, 1220829

Francisco Marçal Teixeira Osório, 1220846

Sérgio Daniel Freitas Moreira, 1220890

Rafael Duarte de Oliveira Ferraz, 1221104

11/2024

Índice

Índice de Figuras

Índice de Tabelas

Explicação do código base

O código base fornecido permite o escalonamento de cirurgias, considerando as agendas dos médicos.

Inicialmente, temos o predicado "*agenda_staff/3"*, que associa os médicos aos horários que se encontram ocupados para um determinado dia. As horas de trabalho é definido pelo predicado "*timetable/3",* indicando para cada médico o tempo de trabalho para um determinado dia. Através do predicado "*surgery/4"*, definimos os tempos de anestesia, de cirurgia e limpeza para um determinado tipo de cirurgia. Para além disso, através do predicado "*surgery_id/2"*, associa-se um tipo de cirurgia a uma cirurgia. Com o predicado "*assignment_surgery/2"*, é possível definir quais médicos podem realizar quais cirurgias, sendo que o predicado "*agenda_operation_room/3"* especifica a ocupação dos quartos.

Passando para a parte da gestão das agendas, utilizamos o predicado *free_agenda0/2* para identificar os intervalos livres em uma agenda ocupada, enquanto o "*adapt_timetable/4"* ajusta esses intervalos de acordo com o horário de trabalho do médico. Além disso, o predicado "*intersect_all_agendas/3"* calcula a interseção das agendas, combinando as disponibilidades dos diferentes médicos. Desta forma é possível garantir que as cirurgias sejam agendadas apenas nos horários disponíveis.

Relativamente ao agendamento das cirurgias em si, utiliza-se o predicado "*schedule_all_surgeries/3"*, que será responsável por marcar todas as cirurgias em um quarto, num dia específico. Este predicado copia as agendas, para armazenar as horas iniciais, e de seguida obtém os intervalos livres para os médicos e para os quartos. Por sua vez, o predicado "*availability_operation/5"* permite determinar quais os horários possíveis para uma cirurgia, tendo em conta a disponibilidade dos médicos e salas, enquanto os predicados "*insert_agenda/3*" e "*insert_agenda_doctors/3"* atualizam as agendas da sala e dos médicos, respetivamente.

Para obter a melhor solução possível temos o predicado "*obtain_better_sol/5"*, que faz todas as combinações de cirurgias para encontrar uma solução que minimize o tempo total necessário para realizar todas as operações, utilizando assim o predicado "*permutation/2*," para testar todas as ordens possíveis de execução das cirurgias e avalia cada solução com o predicado "*evaluate_final_time/3"*, que mede o tempo final da última cirurgia. Caso uma solução seja melhor que a solução atual, será atualizada através do predicado "*update_better_sol/5"*, sendo assim obtida a solução.

Análise de Complexidade

Para a análise de complexidade, utilizou-se o código fornecido e executou-se para as N cirurgias disponíveis, verificando-se os seguintes resultados:

Tabela 1 - Resultados obtidos através do código base

Nº de cirurgias	Nº de soluções	Melhor escalonamento de atividades (incluindo cirurgias) da sala de operações	Tempo Final para a última Cirurgia (minutos)	Tempo para gerar as soluções
3	6	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	0.010828018
4	24	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	0.012715101
5	120	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	0.06270504
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	0.349401951
7	5040	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999,	1149	2.22795701

		so100006), (1000, ..., ...), (..., ...)]		
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	6.533469915
9	362880	[(520, 579, so100000), (580, 639, so100004), (640, 699, so100008), (700, 789, so100002), (790, 849, so100009), (850, 909, so100001), (910, 969, so100006), (1000, ..., ...), (..., ...)|...]	1299	49.07390785
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	511.4216359

De realçar que todos estes dados foram obtidos a partir da mesma máquina, para que não houvesse nenhuma discrepância entre os valores.

Através dos resultados anteriores é possível obter os dois gráficos seguintes:

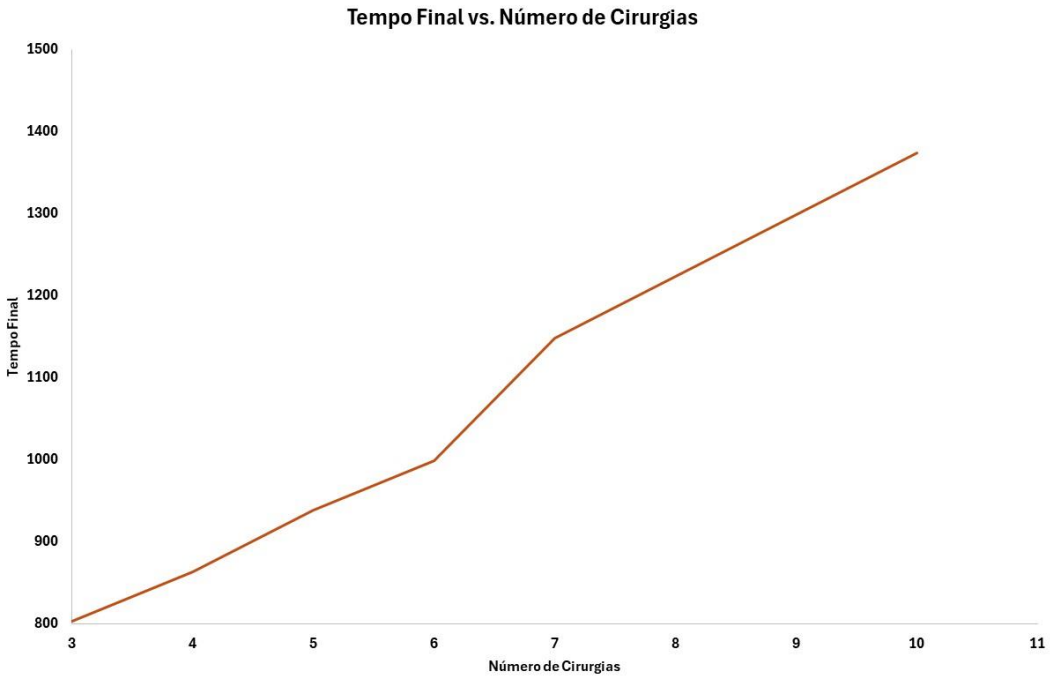

Figura 1 - Gráfico Tempo Final vs. Número de Cirurgias

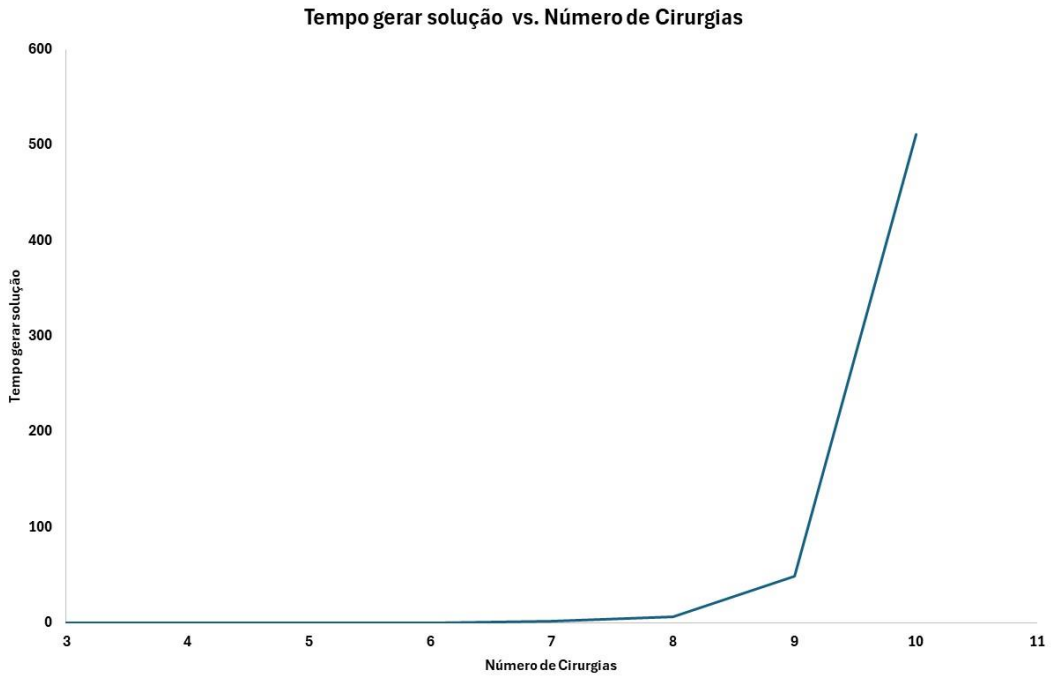

Figura 2 - Tempo gerar solução vs. Número de Cirurgias

O tempo final, reflete o desempenho do algoritmo (neste caso a melhor solução) relativamente à capacidade do mesmo, agendar as cirurgias de forma eficiente. Pela análise da *Figura 1* , é possível verificar que temos um crescimento linear, desta forma podemos dizer que temos um algoritmo eficiente, para a agendar as cirurgias.

Já o tempo de gerar solução, reflete o tempo que o algoritmo demorar para gerar a solução para o escalonamento das cirurgias. Pela análise da *Figura 2*, observa-se que um

crescimento exponencial, sendo assim pouco eficiente, neste parâmetro, mas o que é bastante normal uma vez que são realizadas todas as permutações possíveis, para encontrar a melhor solução. Desta forma, conclui-se que a complexidade do algoritmo é *O(N! x N)*, pois para as N cirurgias, para obter o número das diferentes formas de ordená-la é por N! e sendo que para cada uma é o algoritmo calcula o tempo necessário para concluir todas as cirurgias.

Para além disso, pode-se verificar que para mais de 10 cirurgias, este algoritmo não é uma boa solução, uma vez que demora bastante horas, ou mesmo dias para se puder obter a melhor solução.

Heurísticas

Heurística 1

Para o desenvolvimento desta heurística foi considerado que a próxima cirurgia é aquela que será possível para o médico que está disponível, o mais cedo possível, sendo que o necessário para estar disponível antecipadamente é ter tempo suficiente para concluir a cirurgia antecipadamente.

A heurística, inicia-se pelo predicado "*heuristic_1/5"* , registando o tempo inicial da execução e removendo qualquer agenda armazenada anteriormente para os médicos e quartos. Depois disso recupera-se a agenda do quarto para o dia, e organiza-se as disponibilidades dos médicos. Posteriormente, começa-se a agendar as cirurgias através do predicado "*schedule_all_surgeries_heuristic/3"*, percorrendo-se todas as cirurgias e para cada uma, encontra-se o horário disponível mais cedo para os médicos ("*find_earliest_available_doctor/4"*), agendando cada uma através do predicado "*schedule_surgery_heuristic/3"* sendo que após todas as cirurgias serem agendadas atualiza-se a disponibilidade dos médicos.

No final, calcula-se o tempo de execução, e envia-se a solução, tempo de execução e o tempo em que a última cirurgia acaba.

```
schedule_all_surgeries_heuristic([], _, _).
schedule_all_surgeries_heuristic([OpCode|Rest], Room, Day) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, _, TSurgery, _),

    % Find earliest available time slot
    availability_operation(OpCode, Room, Day, LPossibilities, _),
    schedule_first_interval(TSurgery, LPossibilities, (TinS, TfinS)),

    % Update schedules
    retract(agenda_operation_room1(Room, Day, Agenda)),
    insert_agenda((TinS, TfinS, OpCode), Agenda, Agenda1),
    assertz(agenda_operation_room1(Room, Day, Agenda1)),

    % Update doctors schedules
    findall(Doctor, assignment_surgery(OpCode, Doctor), LDoctors),
    insert_agenda_doctors((TinS, TfinS, OpCode), Day, LDoctors),

    % Update availabilities
    retractall(availability(_, Day, _)),
    findall(_, (
        agenda_staff1(D, Day, L),
        free_agenda0(L, LFA),
        adapt_timetable(D, Day, LFA, LFA2),
        assertz(availability(D, Day, LFA2))
    ), _),

    % Continue with next surgery
    schedule_all_surgeries_heuristic(Rest, Room, Day).
```

Figura 3 - Predicado schedule_all_sugeries_heuristic

```
heuristic_1(Room, Day, AgOpRoomBetter, LAgDoctorsBetter, TFinOp) :-
    get_time(Ti),
    retractall(agenda_staff1(_, _, _)),
    retractall(agenda_operation_room1(_, _, _)),
    retractall(availability(_, _, _)),

    % Setup initial agendas
    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)),

    % Setup initial availabilities
    findall(_, (
        agenda_staff1(D, Day, L),
        free_agenda0(L, LFA),
        adapt_timetable(D, Day, LFA, LFA2),
        assertz(availability(D, Day, LFA2))
    ), _),

    % Get all surgeries
    findall(OpCode, surgery_id(OpCode, _), LOC),

    % Schedule each surgery
    schedule_all_surgeries_heuristic(LOC, Room, Day),

    % Get final schedule
    agenda_operation_room1(Room, Day, FinalAgenda),
    findall(Doctor, assignment_surgery(_, Doctor), LDoctors1),
    remove_equals(LDoctors1, LDoctors),
    list_doctors_agenda(Day, LDoctors, LAgendas),

    % Calculate final time
    reverse(FinalAgenda, ReversedAgenda),
    evaluate_final_time(ReversedAgenda, LOC, FinalTime),

    % Save the results
    AgOpRoomBetter = FinalAgenda,
    LAgDoctorsBetter = LAgendas,
    TFinOp = FinalTime,

    get_time(Tf),
    T is Tf-Ti,
    write('Final Result with Heuristic:'), nl,
    write('AgOpRoomBetter= '), write(AgOpRoomBetter), nl,
    write('LAgDoctorsBetter= '), write(LAgDoctorsBetter), nl,
    write('TFinOp com a heuristica= '), write(TFinOp), nl,
    write('Tempo de geracao da solucao= '), write(T), write(' seconds'), nl.
```

Figura 4 - Predicado heuristic_1

Para obter os resultados utilizou-se os mesmos dados que foram fornecidos no código base, sendo que foram obtidos os seguintes resultados, comparando com o código base:

Tabela 2 - Resultados obtidos heurística 1 e comparação com código base

Nº de cirurgias	Melhor solução	Tempo Final para a última Cirurgia (minutos)	Tempo Final para a última Cirurgia Usando a Heurística (minutos)	Tempo para gerar as soluções	Tempo para geração a solução da heurística	Solução com a Heurística
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	865	0.010828018	0.000262975692749023	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (1000, 1059, so099999)]
4	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805,	864	1200	0.012715101	0.000266790390014648	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791,

	864, so100001), (1000, 1059, so099999)]					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	0.06270504	0.000366926193237304	[(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,	999	1200	0.349401951	0.000396013259887695	[(520, 579, so100000), (580, 639, so100001),

	714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...)]					(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)	1149	1290	2.22795701	0.000555038452148437	[(520, 579, so100000), (580, 639, so100001), (640, 729, so100002), (791, 865, so100003), (866, 925, so100006), (1000, 1059, so099999), (1060,

, (1000, ..., ...), (..., ...)]	1134, so100005), (1141, ..., ...), (..., ...)]

Através dos resultados anteriores é possível obter os seguintes gráficos:

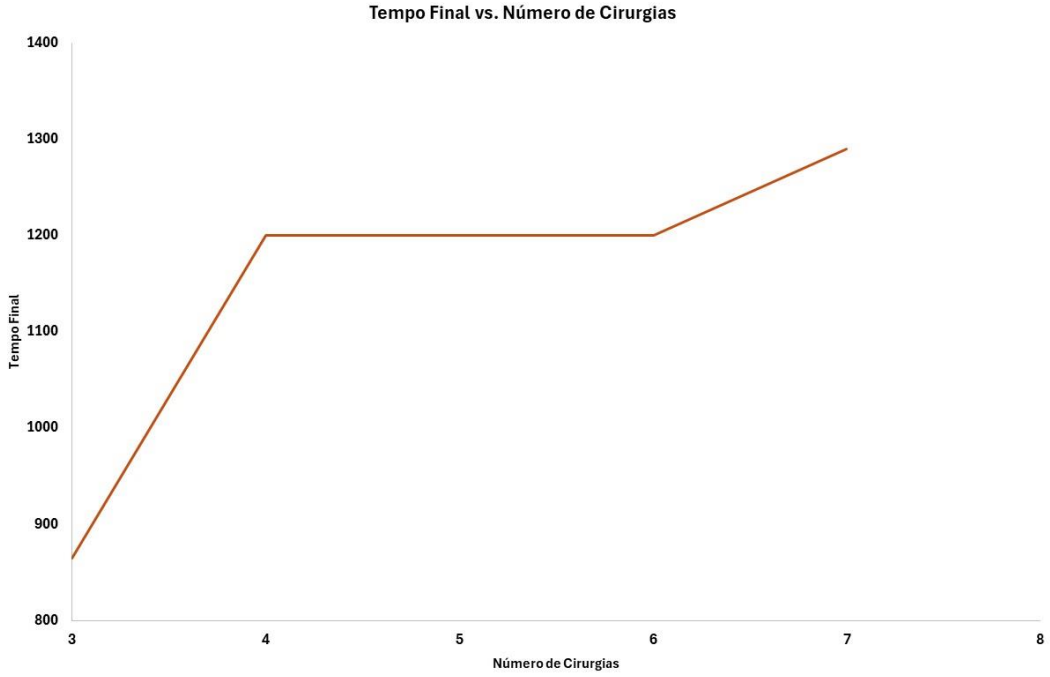

Figura 5 - Gráfico Tempo Final vs. Número de Cirurgias (Heurística 1)

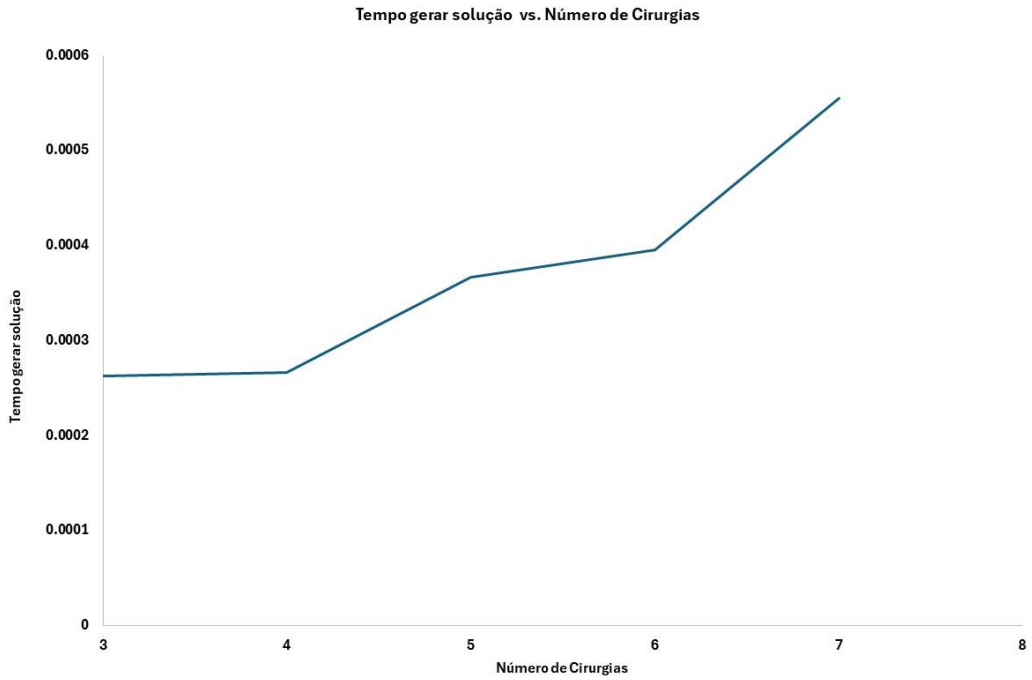

Figura 6 - Tempo gerar solução vs. Número de Cirurgias (Heurística 1)

Os resultados foram obtidos até 7 cirurgias, uma vez que para a oitava cirurgia, não havia disponibilidade para realizar a operação pelos médicos respetivos, não sendo possível agendar mais.

Esta heurística apresenta uma complexidade de *O(n)*, onde *n* é o número de cirurgias, essencialmente devido ao predicado "*schedule_all_surgeries_heuristic/3"* , onde é realizado um *loop* por todas as cirurgias.

Heurística 2

Na segunda heurística, vai ser usado o algoritmo "*longest first*", ou seja, vai ser priorizada a cirurgia que mais tempo demora a ser concluída, e depois a segunda mais longa, e sempre assim.

Para isso, desenvolveu-se o predicado "*heuristic_2/4*". Este predicado começa por marcar o tempo de início de execução e por limpar quaisquer marcações feitas e por ordenar todas as cirurgias, da mais longa para a mais curta, assim temos as cirurgias com maior prioridade no início da fila. Em seguida, começando pela primeira da lista, vê-se se há disponibilidade dos médicos responsáveis pela tal. Verifica-se se estes estão disponíveis e se isso se verificar, marca-se a cirurgia, senão, repete-se o processo para a próxima cirurgia na lista.

Este processo repete-se até acabar o dia, até marcar todas as cirurgias, ou não houver mais médicos disponíveis.

Por fim, calcula-se o tempo de execução, e envia-se a solução, tempo de execução e o tempo em que a última cirurgia acaba.

```
heuristic_2(Room,Day,AgOpRoomBetter,LAgDoctorsBetter,TFinOp):-
    get_time(Ti),
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),

    % Get all surgeries
    findall(OpCode-Duration,(surgery_id(OpCode,Type),surgery(Type,_,Duration,_)),Pairs),
    % Sort by duration
    keysort(Pairs,Sorted),
    % Longest First
    reverse(Sorted,LongestFirst),
    pairs_keys(LongestFirst,SortedOpCodes),

    % Setup initial agendas and 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))),_),

    % Schedule surgeries
    availability_all_surgeries(SortedOpCodes,Room,Day),

    % Get final schedules
    agenda_operation_room1(Room,Day,AgOpRoomBetter),
    findall(Doctor,assignment_surgery(_,Doctor),LDoctors1),
    remove_equals(LDoctors1,LDoctors),
    list_doctors_agenda(Day,LDoctors,LAgDoctorsBetter),

    % Calculate final time
    reverse(AgOpRoomBetter,ReversedAgenda),
    evaluate_final_time(ReversedAgenda,SortedOpCodes,TFinOp),

    % Calculate execution time
    get_time(Tf),
    T is Tf-Ti,
    write('TFinOp com a heurística: '),write(TFinOp),nl,
    write('Solucao com a heuristica: '),write(AgOpRoomBetter),nl,
    write('Tempo de geracao da solucao com a heurística: '),write(T),write(' seconds'),nl.
```

Figura 7 - Predicado heuristic_2

De seguida, estão representados os dados obtidos na mesma máquina e para os mesmos dados:

Tabela 3 - Resultados obtidos heurística 2 e comparação com código base

Nº de cirurgias	Melhor solução	Tempo Final para a última	Tempo Final para a última	Tempo para gerar as soluções	Tempo para geração a	Soluçã o com a

		Cirurgia (minutos)	Cirurgia Usando a Heurística (minutos)		solução da heurística	Heurística
3	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	850	0.010828018	0.000187873840332031	[(520, 579, so100000), (580, 654, so100003), (655, 744, so100002), (791, 850, so100001), (1000, 1059, so099999)]
4	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	864	0.012715101	0.000236988067626953	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100003), (715, 804, so100002), (805, 864, so1000

						01), (1000, 1059, so099999)]
5	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	1149	0.06270504	[(520, 579, so100000), (580, 654, so100005), (655, 714, so100004), (791, 865, so100003), (866, 925, so100001), (1000, 1059, so099999), (1060, 1149, so100002)]	0.0002489089965820 31
6	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939,	999	1290	0.349401951	[(520, 579, so100000), (580, 639, so100006), (640, 714, so100005), (791, 865, so100003), (866, 925, so100001), (1000, 1059, so099999), (1141, 1200, so100004), (1201, ..., ...)]	0.0003688335418701 17

	so100001), (940, 999, so100006), (1000, ..., ...)]					
7	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (940, 999, so100006), (1000, ..., ...), (..., ...)]	1149	1275	2.22795701	[(520, 579, so100000), (580, 669, so100007), (670, 759, so100002), (791, 850, so100006), (851, 910, so100001), (1000, 1059, so099999), (1060, 1134, so100005), (1141, ..., ...), (..., ...)]	0.0002930164337158 2

Mais uma vez, com os resultados anteriores contruiu-se os gráficos seguintes:

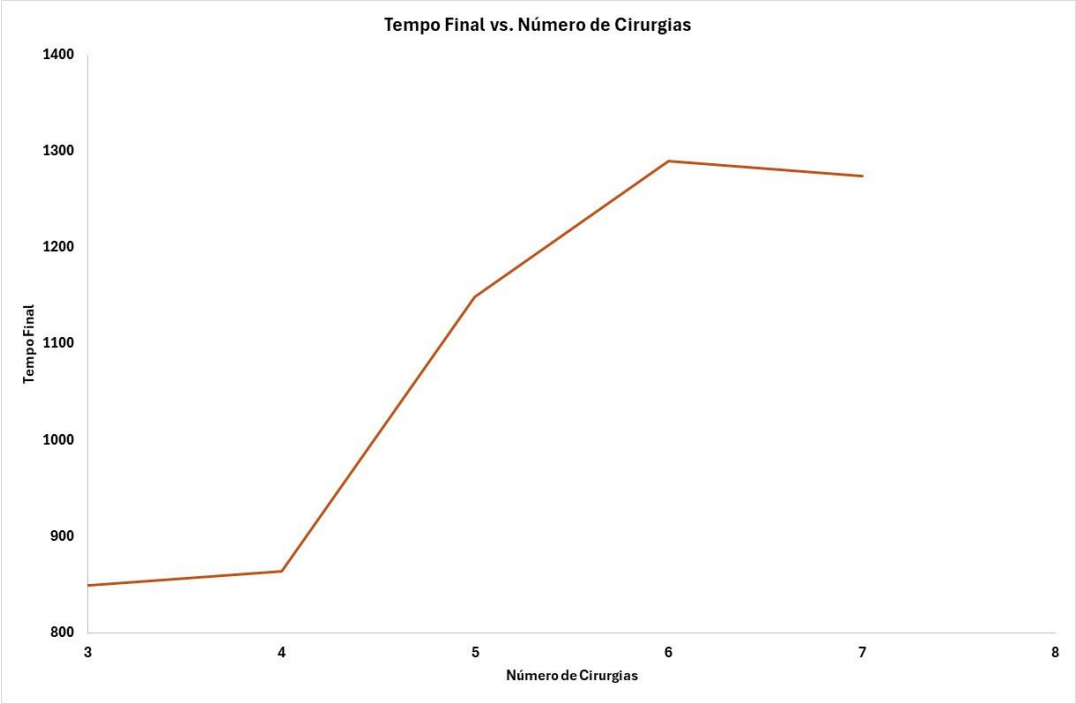

Figura 8 - Gráfico Tempo Final vs. Número de Cirurgias (Heurística 2)

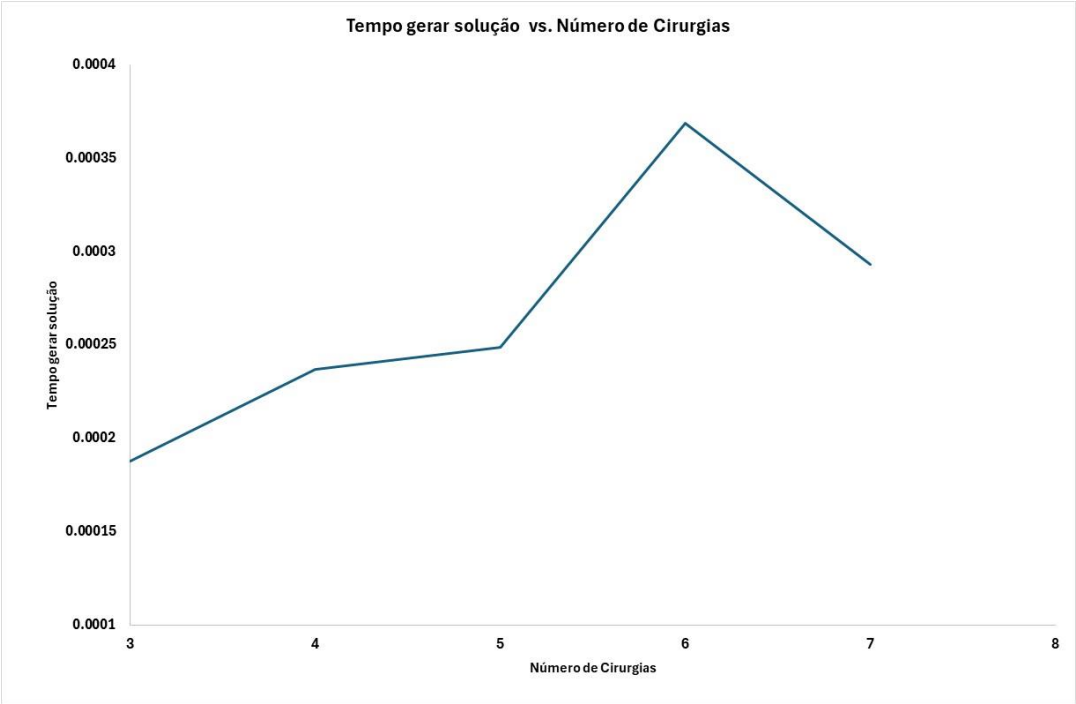

Figura 9 - Tempo gerar solução vs. Número de Cirurgias (Heurística 2)

Também nesta heurística, os resultados foram obtidos até 7 cirurgias, uma vez que para a oitava cirurgia, não havia disponibilidade para realizar a operação pelos médicos respetivos, não sendo possível agendar mais.

A complexidade desta heurística é refletida essencialmente pela ordenação das cirurgias pela sua duração, da mais demorada para a menos demorada. Sendo assim, o

algoritmo apresenta uma complexidade *O(n * log(n))*, sendo também algo percetível pelo *Figura 6*.

Conclusões

Pelas análises anteriores, podem se tirar algumas conclusões. Desta forma, é possível verificar que as duas heurísticas se destacam em relação á *better solution* a nível de tempo de execução, tornando-as ideias para situações onde o tempo de processamento é essencial. Porém, as soluções obtidas pelas duas heurísticas são menos precisas, ou seja, tempos finais superiores para a última cirurgia, relativamente ao *better solution*.

Desta forma, as heurísticas são ideais para cenários com restrições de tempo e grande número de cirurgias, apesar de ter menos precisão, sendo a segunda heurística parece mais eficiente. No entanto, para cenários onde a precisão é prioritária, o algoritmo do *better solution* é preferível, apesar de ter um custo superior de tempo.