Instituto Superior de Engenharia do Porto

# ALGAV – Relatório: Planeamento de Cirurgias

3DC, G15

Jack Pinheiro, 1120419

João Pereira, 1211503

Mariana Correia, 1211883

Vasco Sousa, 1221700

# Índice

# 1. Breve explicação do código base para agendar cirurgias

O código presente no ficheiro *schedule_operations* apresenta um sistema de agendamento de cirurgias que considera a disponibilidade dos médicos e das salas de operação, como horários de trabalho, duração das cirurgias e agendas pré-existentes, através do uso de factos e predicados dinâmicos para armazenar e atualizar as agendas dos médicos e das salas de operações durante o processo de agendamento.

Inicialmente, o sistema define dados básicos, como os horários de trabalho dos médicos (*timetable/3*), a agenda dos mesmos (*agenda_staff/3*), o tipo de cirurgia e o tempo que a cirurgia dura (*surgery/4*), existindo a possibilidade de associação entre os médicos e as cirurgias que irão realizar (*assignment_surgery/2*). Além disso, calcula-se os intervalos livres nas agendas dos médicos com os predicados *free_agenda0/2* e *free_agenda1/2* e ajusta-se esses intervalos com base nos horários de trabalho pré-definidos anteriormente através do uso do predicado *adapt_timetable/4*.

O processo principal de agendamento das cirurgias ocorre no predicado *schedule_all_surgeries/3*, onde se organiza as cirurgias numa sala específica para um determinado dia, verificando os intervalos disponíveis nas agendas dos médicos e da sala e, em seguida, aloca cada cirurgia no primeiro intervalo válido encontrado. Após cada agendamento, as agendas dos médicos e da sala de operação são atualizadas.

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

*Figura 1 - predicado schedule_all_surgeries/2*

Posteriormente, realiza-se as interseções de horários com o predicado *intersect_all_agendas/3*, para garantir que os médicos necessários para uma mesma cirurgia tenham disponibilidade simultânea.

Por fim, o predicado *obtain_better_sol/5* testa diferentes ordens de cirurgias para encontrar a solução com menor tempo de conclusão total, isto é, para apresentar o melhor agendamento de todas as cirurgias para um determinado dia, atualizando a melhor solução encontrada dinamicamente, considerando a disponibilidade dos médicos e o uso da sala de operação.

```
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.

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.
```

*Figura 2 - predicado obtain_better_sol/5*

## 2. Estudo da complexidade do agendamento de cirurgias

A gestão eficiente de uma sala de operações exige a otimização do agendamento de cirurgias, garantindo o menor tempo possível para agendar todas as operações para um determinado dia. Para abordar este problema, foi implementado o predicado *obtain_better_sol/*5, que utiliza uma abordagem exaustiva (ou força bruta) para determinar a melhor ordem de execução das cirurgias, isto é, avalia todas as permutações possíveis, garantindo a melhor solução.

Deste modo, analisámos esta abordagem ao avaliar os tempos de execução para um determinado número de cirurgias. Apesar do objetivo inicial de obter resultados para 13 cirurgias, só foi possível verificar a possibilidade de obter resultados para até 10 cirurgias.

Assim, apresentamos uma tabela que demonstra os resultados obtidos com base nos dados apresentados abaixo, seguido de uma análise da complexidade e conclusões sobre os limites do predicado.

```
%surgery(SurgeryType,TAnesthesia,TSurgery,TCleaning).
surgery(so2,45,60,45).
surgery(so3,45,90,45).
surgery(so4,45,75,45).

%surgery_id(OpCode,SurgeryType).
surgery_id(so100001,so2).
surgery_id(so100002,so3).
surgery_id(so100003,so4).
%surgery_id(so100004,so2).
%surgery_id(so100005,so4).
%surgery_id(so100006,so2).
%surgery_id(so100007,so3).
%surgery_id(so100008,so2).
%surgery_id(so100009,so2).
%surgery_id(so100010,so2).
%surgery_id(so100011,so4).
%surgery_id(so100012,so2).
%surgery_id(so100013,so2).

%assignment_surgery(OpCode,Doctor).
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).
%assignment_surgery(so100006,d001).
%assignment_surgery(so100007,d003).
%assignment_surgery(so100008,d004).
%assignment_surgery(so100008,d003).
%assignment_surgery(so100009,d002).
%assignment_surgery(so100009,d004).
%assignment_surgery(so100010,d003).
%assignment_surgery(so100011,d001).
%assignment_surgery(so100012,d001).
%assignment_surgery(so100013,d004).
```

*Figura 3 - Dados para o agendamento das cirurgias*

*Tabela 1 - Tabela dos resultados obtidos a partir do predicado obtain_better_sol/5*

| Nº de cirurgias | Nº de operações | Melhor agendamento da sala de operação | Tempo final da última cirurgia (minutos) | Tempo para gerar a solução (segundos) |
|---|---|---|---|---|
| **3** | 6 | [(520,579,so100000),(580,639,so100001), (640,714,so100003),(715,804,so100002), (1000,1059,so099999 )] | 804 | 0,02 |
| **4** | 24 | [(520,579,so100000),(580,654,so100003), (655,714,so100004),(715,804,so100002), (805,864,so100001),(1000,1059,so099999)] | 864 | 0,10 |
| **5** | 120 | [(520,579,so100000),(580,639,so100004), (640,714,so100005),(715,804,so100002), (805,879,so100003),(880,939,so100001), (1000,1059,so099999)] | 939 | 0,53 |
| **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 | 2,63 |
| **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 | 12,79 |
| **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 | 59,32 |
| **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 | 92,39 |
| **10** | 3628800 | [(520,579,so100000),(580,639,so100004), (640,699,so100008),(700,759,so100009), (791,865,so100003),(866,925,so100001), (926,985,so100006),(1000,1059,so099999), (1060,1134,so100005),(1135,1224,so100007), (1225,1284,so100010),(1285,1374,so100002)] | 1374 | 569,99 |

Em relação à tabela 1, é possível referir que os resultados apresentados demonstram a limitação do algoritmo devido à sua complexidade fatorial O(n!), em que até n=10 cirurgias, o tempo de execução é aceitável, mas se efetuarmos o agendamento de n≥10 cirurgias, o tempo necessário para calcular a solução ótima torna-se inviável. Isto é, a complexidade fatorial implica que quando o $n$ cresce, o número de permutações que devem ser analisadas aumenta de forma extremamente rápida, sendo que para se resolver o problema de forma exaustiva (força bruta), cada permutação precisa de ser verificada para se encontrar a melhor solução.

Assim, foi necessário encontrar soluções alternativas, como algoritmos heurísticos, para instâncias maiores, ou seja, para que seja possível apresentar um cenário com um maior número de cirurgias.

# 3. Algoritmos Heurísticos

Os algoritmos heurísticos são métodos aproximados da solução de problemas que utilizam regras práticas ou estratégias baseadas na experiência para encontrar soluções rápidas. Em vez de explorar todas as possibilidades como a força bruta, as heurísticas seguem critérios pré-definidos para selecionar as melhores opções para cada etapa, reduzindo consideravelmente a quantidade de cálculos necessários. Embora as soluções obtidas não sejam necessariamente as melhores possíveis, são frequentemente adequadas para utilizar em cenários complexos, como o apresentado neste relatório.

Deste modo, iremos implementar e analisar dois algoritmos heurísticos para o problema de agendamento de cirurgias:

## 3.1. Disponibilidade Antecipada do Médico

Esta heurística irá priorizar a alocação de cirurgias com base no médico que está disponível mais cedo, isto é, deve-se selecionar a próxima cirurgia de um médico que possa começar e concluir a cirurgia dentro do menor intervalo de tempo possível, considerando a duração da cirurgia e o horário disponível do médico.

O ficheiro '*schedule-first-doctor.pl*' implementa o algoritmo heurístico para o agendamento de cirurgias, otimizando o uso de recursos como salas de operações e agendas médicas. O processo utiliza vários predicados essenciais, destacando-se os predicados *find_earliest_available_doctor/4*, *obtain_heuristic_sol/5* e *schedule_all_surgeries_heuristic/3*, que desempenham papéis complementares na solução.

O predicado *find_earliest_available_doctor/4* identifica o médico que pode iniciar mais cedo uma cirurgia específica, tendo em conta as disponibilidades individuais e os requisitos da cirurgia. Este predicado recebe o código da cirurgia (*OpCode*) e a data (*Day*) em que se realiza a mesma, devolvendo o médico mais adequado (*EarliestDoctor*) e o horário mais cedo possível para iniciar a cirurgia (*EarliestTime*). Para isso, avalia o tempo necessário para a cirurgia com base no seu tipo, obtém os médicos atribuídos à cirurgia e verifica as suas disponibilidades, selecionando a opção que permite iniciar mais cedo, sendo crucial para o agendamento das cirurgias com eficiência.

```
find_earliest_available_doctor(OpCode, Day, EarliestDoctor, EarliestTime) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, _, TSurgery, _),
    findall(Doctor, assignment_surgery(OpCode, Doctor), Doctors),
    findall((Time, Doc), (
        member(Doc, Doctors),
        availability(Doc, Day, Availabilities),
        member((Start, End), Availabilities),
        % Check if there's enough time for the surgery
        Duration is End - Start + 1,
        Duration >= TSurgery,
        Time = Start
    ), DoctorTimes),
    sort(DoctorTimes, [(EarliestTime, EarliestDoctor) | _]).
```

*Figura 4 - predicado find_earliest_available_doctor/4*

O predicado principal *obtain_heuristic_sol/5* coordena todo o processo de agendamento com base na heurística implementada. Este predicado recebe como entrada a sala de operações (*Room*) e a data do agendamento (*Day*), produzindo como resultados a agenda final da sala de operações (*AgOpRoomBetter*), as agendas dos médicos envolvidos (*LAgDoctorsBetter*) e o tempo final da última cirurgia realizada (*TFinOp*). O seu funcionamento inicia-se com a limpeza e preparação de fatores dinâmicos, onde são configuradas as agendas iniciais para médicos e sala de operações, calculando-se as disponibilidades iniciais que apresentam. Em seguida, processa-se todas as cirurgias utilizando o predicado *schedule_all_surgeries_heuristic/3* e, após a execução, extrai-se as agendas finais e calcula-se o tempo total de execução, obtendo-se um resumo detalhado do processo e dos resultados.

```
obtain_heuristic_sol(Room, Day, AgOpRoomBetter, LAgDoctorsBetter, TFinOp) :-
    get_time(Ti),
    % Initialize schedules
    retractall(agenda_staff1(_, _, _)),retractall(agenda_operation_room1(_, _, _)),retractall(availability(_, _, _)),
    % Set up 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)),
    % Calculate 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 using the heuristic
    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),
    % Store results
    AgOpRoomBetter = FinalAgenda,LAgDoctorsBetter = LAgendas,TFinOp = FinalTime,get_time(Tf),T is Tf - Ti,
    % Output results
    format('~nFinal Result with Heuristic:~n', []),
    format('~nOperation Room Schedule: ~w~n', [AgOpRoomBetter]),
    format('~nDoctors Schedules: ~w~n', [LAgDoctorsBetter]),
    format('~nFinal Time: ~w~n', [TFinOp]),
    format('~nTime to generate solution: ~2f seconds~n~n', [T]).
```

*Figura 5 - predicado obtain_heuristic_sol/5*

O predicado s*chedule_all_surgeries_heuristic/3* implementa a lógica central da heurística, processando cada cirurgia de forma recursiva até que todas sejam agendadas. Este predicado recebe como entrada a lista de cirurgias a agendar (*ListOfOperations*), a sala de operações (*Room*) e a data (*Day*) em que se realiza a cirurgia. Para cada cirurgia, identifica-se os intervalos de tempo disponíveis na sala de operações e nas agendas dos médicos atribuídos, selecionando-se o primeiro intervalo viável através do predicado *schedule_first_interval/3*. Após agendar a cirurgia, atualiza-se as agendas da sala de operações e dos médicos, bem como a disponibilidade que apresentem. Este processo é repetido até que todas as cirurgias tenham sido alocadas, assegurando uma utilização eficiente dos recursos.

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

    % Find earliest available time slot using the heuristic
    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 6 – predicado schedule_all_surgeries_heuristic/3*

Deste modo, o algoritmo implementado no ficheiro organiza e distribui as cirurgias de forma heurística, garantindo o cumprimento dos requisitos de tempo e assegurando que o sistema consegue lidar com múltiplas restrições e prioridades de maneira eficaz.

### 3.1.1. Estudo da Complexidade

O estudo da complexidade do algoritmo heurístico implementado no ficheiro *schedule-first-doctor.pl* foi realizado com o objetivo de avaliar a sua eficiência e a qualidade das soluções geradas em comparação com a solução ótima. Deste modo, analisou-se para um determinado número de cirurgias a serem agendadas diferentes cenários, comparando o tempo final da última cirurgia e o tempo necessário para calcular a solução.

A tabela abaixo apresenta os resultados obtidos, os tempos de execução e as soluções encontradas pelo algoritmo heurístico, incluindo a solução ótima.

*Tabela 2 - Estudo complexidade do algoritmo heurístico*

| Nº cirurgias | Solução ótima | Tempo final da última cirurgia (min) | Tempo final da última cirurgia utilizando heurística (min) | Tempo para gerar a solução (s) | Tempo para gerar a solução utilizando Heurística (s) | Solução usando Heurística |
|---|---|---|---|---|---|---|
| 3 | [(520,579,so100000),(580,639,so100001),(640,714,so100003),(715,804,so100002),(1000,1059,so099999 )] | 804 | 865 | 0,02 | 0,03 | [(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,so100001),(1000,1059,so099999)] | 864 | 1200 | 0,10 | 0,04 | [(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 | 0,53 | 0,05 | [(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),(1000,1 | 999 | 1200 | 2,63 | 0,07 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,865,so100003),(866,925,so100006),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004)] |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 059,so099999)] | | | | | |
| 7 | [(520,579,so100000),(580,639,so100004),(640,714,so100005),(715,804,so100002),(805,879,so100003),(880,939,so100001),(940,999,so100006),(1000,1059,so099999),(1060,1149,so100007)] | 1149 | 1290 | 12,79 | 0,07 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,865,so100003),(866,925,so100006),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004),(1201,1290,so100007)] |
| 8 | [(520,579,so100000),(580,639,so100004),(640,699,so100008),(700,789,so100002),(791,865,so100003),(866,925,so100001),(926,985,so100006),(1000,1059,so099999),(1060,1134,so100005),(1135,1224,so100007)] | 1224 | 1290 | 59,32 | 0,07 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(730,789,so100008),(791,865,so100003),(866,925,so100006),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004),(1201,1290,so100007)] |
| 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 | 1350 | 92,39 | 0,08 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(730,789,so100008),(791,865,so100003),(866,925,so100006),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004),(1201,1290,so100007),(1291,1350,so100009)] |

| | | | | | | |
|---|---|---|---|---|---|---|
| | ,<br>(1060,1134,so100005),(1135,1209,so100003),<br>(1210,1299,so100007)] | | | | | |
| **10** | [(520,579,so100000),(580,639,so100004),<br>(640,699,so100008),(700,759,so100009),<br>(791,865,so100003),(866,925,so100001),<br>(926,985,so100006),(1000,1059,so099999)<br>,<br>(1060,1134,so100005),(1135,1224,so100007),<br>(1225,1284,so100010),(1285,1374,so100002)] | 1374 | 1350 | 569,99 | 0,12 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(730,789,so100008),(791,865,so100003),(866,925,so100006),(926,985,so100012),(1000,1059,so099999),(1060,1134,so100005),(1141,1200,so100004),(1201,1290,so100007),(1291,1350,so100009)] |

A partir da tabela 2, pode-se verificar que o algoritmo heurístico demonstrou ser altamente eficiente, especialmente em problemas com maior número de cirurgias, visto que à medida que o número de cirurgias aumenta, o tempo de execução da solução ótima cresce exponencialmente, enquanto a heurística mantém tempos de execução consistentes, mesmo para casos mais complexos, como 10 cirurgias (0,12 segundos).

Adicionalmente, a heurística frequentemente apresenta resultados muito próximos à solução ótima, especialmente para um número menor de cirurgias. Enquanto para números maiores, os tempos de execução permanecem aceitáveis e demonstram uma escalabilidade robusta.

## 3.2. Taxa de Ocupação do Médico

Este algoritmo considera a ocupação relativa do médico ao longo do tempo restante do cronograma, selecionando-se como a próxima cirurgia aquela que envolve o médico com a maior taxa de ocupação projetada, ou seja, o médico que já apresenta a agenda mais preenchida.

O ficheiro *"schedule-more-booked.pl"* contém a implementação do algoritmo heurístico para agendamento de cirurgias que visa otimizar a alocação de salas de operações e médicos disponíveis, priorizando a cirurgia para o médico com agenda com maior taxa de ocupação.

O algoritmo seleciona e organiza as cirurgias utilizando o predicado principal *obtain_heuristic_sol/5*, que coordena as etapas de configuração e execução da heurística. Após inicializar as estruturas de dados, o algoritmo lista todas as cirurgias a agendar, utilizando o predicado *surgery_id*. Em seguida, cada cirurgia é processada através de *schedule_all_surgeries_heuristic/3*, que invoca métodos auxiliares para calcular intervalos possíveis e realizar o agendamento. Quando um intervalo viável é encontrado, a cirurgia é agendada na sala de operações e nas agendas dos médicos.

```
obtain_heuristic_sol(Room, Day, AgOpRoomBetter, LAgDoctorsBetter, TFinOp) :-
    get_time(Ti),
    % Initialize schedules
    retractall(agenda_staff1(_, _, _)),
    retractall(agenda_operation_room1(_, _, _)),
    retractall(availability(_, _, _)),
    % Set up 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)),
    % Calculate 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 using the heuristic
    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),
    % Store results
    AgOpRoomBetter = FinalAgenda,LAgDoctorsBetter = LAgendas,TFinOp = FinalTime,get_time(Tf),T is Tf - Ti,
    % Output results
    format('~n----------------------------------------------------------------------------------------~n', []),
    format('~nFinal Result with Heuristic:~n', []),
    format('~nOperation Room Schedule: ~w~n', [AgOpRoomBetter]),
    format('~nDoctors Schedules: ~w~n', [LAgDoctorsBetter]),
    format('~nFinal Time: ~w~n', [TFinOp]),
    format('~nTime to generate solution: ~2f seconds~n~n', [T]).
```

*Figura 7 - predicado obtain_heuristic_sol/5*

O predicado *schedule_surgery_heuristic/3* determina o médico mais ocupado e aloca o início da cirurgia com base no primeiro intervalo disponível. Em seguida, os intervalos são atualizados utilizando os predicados *insert_agenda* e *insert_agenda_doctors*, garantindo a consistência dos agendamentos.

```
schedule_surgery_heuristic(OpCode, Room, Day) :-
    find_busiest_available_doctor(OpCode, Day, EarliestDoctor, StartTime),
    surgery_id(OpCode, OpType),
    surgery(OpType, _, TSurgery, _),
    EndTime is StartTime + TSurgery - 1,

    % Update operation room agenda
    retract(agenda_operation_room1(Room, Day, Agenda)),
    insert_agenda((StartTime, EndTime, OpCode), Agenda, NewAgenda),
    assertz(agenda_operation_room1(Room, Day, NewAgenda)),
    format('~nSurgery ~w inserted into room ~w agenda: (~w, ~w)~n', [OpCode, Room, StartTime, EndTime]),

    % Update only the earliest doctor's agenda first
    insert_agenda_doctors((StartTime, EndTime, OpCode), Day, [EarliestDoctor]),

    % Then update other assigned doctors' agendas if any
    findall(Doc, (assignment_surgery(OpCode, Doc), Doc \= EarliestDoctor), OtherDocs),
    insert_agenda_doctors((StartTime, EndTime, OpCode), Day, OtherDocs).
```

*Figura 8 - predicado schedule_surgery_heuristic/3*

Por fim, o algoritmo avalia os resultados, incluindo a agenda completa da sala de operações (*AgOpRoomBetter*), as agendas dos médicos (*LAgDoctorsBetter*) e o tempo final necessário para concluir todas as cirurgias (*TFinOp*).

Assim, a heurística dá prioridade a médicos mais ocupados, contribuindo para o objetivo de distribuir de forma eficiente os tempos de trabalho e minimizar o impacto em horários futuros e, deste modo, maximizar a utilização de tempos livres, garantindo que as cirurgias sejam agendadas de forma otimizada dentro das restrições estabelecidas.

### 3.2.1. Estudo da Complexidade

O estudo da complexidade do algoritmo heurístico implementado no ficheiro *schedule-more-booked.pl* foi realizado com o objetivo de avaliar a sua eficiência e a qualidade das soluções geradas em comparação com a solução ótima. Deste modo, analisou-se para um determinado número de cirurgias a serem agendadas diferentes cenários, comparando o tempo final da última cirurgia e o tempo necessário para calcular a solução.

A tabela abaixo apresenta os resultados obtidos, incluindo o tempo final da última cirurgia na solução ótima e heurística, o tempo de execução de cada abordagem e as cirurgias alocadas pela heurística.

*Tabela 3 - Estudo da complexidade do algoritmo heurístico*

| Nº cirurgias | Solução ótima | Tempo final da última cirurgia (min) | Tempo final da última cirurgia utilizando heurística (min) | Tempo para gerar a solução (s) | Tempo para gerar a solução utilizando Heurística (s) | Solução usando Heurística |
|---|---|---|---|---|---|---|
| 3 | [(520,579,so100000),(580,639,so100001),(640,714,so100003),(715,804,so100002),(1000,1059,so099999 )] | 804 | 1125 | 0,02 | 0,03 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,850,so100001),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003)] |
| 4 | [(520,579,so100000),(580,654,so100003),(655,714,so100004),(715,804,so100002),(805,864,so100001),(1000,1059,so099999)] | 864 | 1200 | 0,10 | 0,34 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,850,so100001),(851,910,so100004),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(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 | 1275 | 0,53 | 0,30 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,850,so100001),(851,910,so100004),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)] |
| 6 | [(520,579,so100000),(580,639,so100004),(640,714,so100005),(715,804,so100002),(805,879,so100003),(880,939,so100001),(940,999,so10 | 999 | 1275 | 2,63 | 0,26 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 0006),(1000,1059,so099999)] | | | | | 125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)] |
| **7** | [(520,579,so100000),(580,639,so100004),(640,714,so100005),(715,804,so100002),(805,879,so100003),(880,939,so100001),(940,999,so100006),(1000,1059,so099999),(1060,1149,so100007)] | 1149 | 1275 | 12,79 | 0,33 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem slots disponíveis para a cirurgia 7 |
| **8** | [(520,579,so100000),(580,639,so100004),(640,699,so100008),(700,789,so100002),(791,865,so100003),(866,925,so100001),(926,985,so100006),(1000,1059,so099999),(1060,1134,so100005),(1135,1224,so100007)] | 1224 | 1275 | 59,32 | 0,35 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem slots disponíveis para a cirurgia 7 e 8 |
| **9** | [(520,579,so100000),(580,639,so100004),(640,699,so100008),(700,789,so100002),(790,849,so100009),(850,909,so100001),(910,969,so10 | 1299 | 1275 | 92,39 | 0,38 | [(520,579,so100000),(580,639,so100001),(640,729,so1000020,50,(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(10 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 0006),(1000,1059,so099999),<br>(1060,1134,so100005),(1135,1209,so100003),<br>(1210,1299,so100007)] | | | | | 51,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem slots disponíveis para a cirurgia 7 e 8 |
| **10** | [(520,579,so100000),(580,639,so100004),<br>(640,699,so100008),(700,759,so100009),<br>(791,865,so100003),(866,925,so100001),<br>(926,985,so100006),(1000,1059,so099999),<br>(1060,1134,so100005),(1135,1224,so100007),<br>(1225,1284,so100010),(1285,1374,so100002)] | 1374 | 1275 | 569,99 | 0,45 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(620,679,so100009),(730,789,so100009),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem slots disponíveis para a cirurgia 7, 8 e 10 |
| **11** | - | - | 1275 | - | 0,52 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(620,679,so100009),(730,789,so100009),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | slots disponíveis para a cirurgia 7, 8, 10 e 11 |
| **12** | - | - | 1275 | - | 0,46 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(620,679,so100009),(730,789,so100009),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem slots disponíveis para a cirurgia 7, 8, 10, 11 e 12 |
| **13** | - | - | 1275 | - | 0,50 | [(520,579,so100000),(580,639,so100001),(640,729,so100002),(620,679,so100009),(730,789,so100009),(791,850,so100001),(851,910,so100004),(911,970,so100006),(911,970,so100006),(1000,1059,so099999),(961,1050,so100002),(981,1055,so100003),(1051,1125,so100003),(1141,1200,so100004),(1126,1200,so100005),(1201,1275,so100005)]; sem slots disponíveis para a cirurgia 7, 8, 10, 11, 12 e 13 |

Com base na tabela 3, é possível determinar que os dados revelam uma diferença significativa entre a abordagem ótima e a heurística em termos de tempo de execução e escalabilidade.

A solução ótima, embora demonstre agendamentos de maior qualidade, apresenta um crescimento exponencial no tempo de execução, tornando-se inviável para problemas com mais de 6 ou 7 cirurgias, uma vez que se considera todas as combinações possíveis.

Por outro lado, a heurística, com tempo de execução quase constante, é adequada para problemas maiores. Embora apresente desvios no tempo final da última cirurgia e não consiga alocar todas as cirurgias em cenários complexos, a heurística demonstra boa eficiência prática e escalabilidade, sendo uma abordagem mais viável para cenários reais com grande número de cirurgias.

# 4. Explicação do código base adaptado

Os principais métodos que passaram por alterações para se incluir o resto do staff para a realização das cirurgias foram o *schedule_operations*, o *schedule-first-doctors* e o *schedule-more-booked*.

Primeiramente, introduziu-se os novos dados, como as agendas individuais de cada membro do staff, os horários de disponibilidade e a associação desses profissionais às cirurgias.

No ficheiro *schedule_operations-staff* (adaptação do ficheiro *schedule_operations*), a principal mudança foi a inclusão das agendas dos profissionais, o que permitiu que, ao agendar uma cirurgia, o sistema verifica-se se todos os membros do staff necessários estavam disponíveis no mesmo intervalo de tempo. Dessa forma, o agendamento foi realizado conforme a disponibilidade de todos os envolvidos, evitando conflitos de horário e garantindo que as cirurgias podiam ser realizadas conforme o planeado.

```
availability_all_surgeries([],_,_):-
    write('\nNo more surgeries to schedule'), nl.
availability_all_surgeries([OpCode|LOpCode],Room,Day):-
    write('-----------------------------------------------'), nl,
    write('Scheduling surgery: '), write(OpCode), nl,
    surgery_id(OpCode,OpType),
    surgery(OpType,_,TSurgery,_),
    availability_operation(OpCode,Room,Day,LPossibilities,LStaff),
    write('\nLPossibilities: '), write(LPossibilities), nl,
    schedule_first_interval(TSurgery,LPossibilities,(TinS,TfinS)),
    write('\nScheduled interval: '), write((TinS,TfinS)), nl,
    % Atualizar sala de operação
    retract(agenda_operation_room1(Room,Day,Agenda)),
    insert_agenda((TinS,TfinS,OpCode),Agenda,Agenda1),
    assertz(agenda_operation_room1(Room,Day,Agenda1)),
    % Atualizar agendas do staff
    write('\nUpdating staff agendas'), nl,
    insert_agenda((TinS,TfinS,OpCode),Agenda,Agenda1),
    write('\nAgenda1: '), write(Agenda1), nl,
    availability_all_surgeries(LOpCode,Room,Day).
availability_operation(OpCode,Room,Day,LPossibilities,LStaff):-
    write('\nChecking availability for operation: '), write(OpCode), nl,
    surgery_id(OpCode,OpType),
    surgery(OpType,_,TSurgery,_),
    % Identificar doutores atribuidos a uma determinada cirurgia
    findall(Doctor, assignment_surgery(OpCode,Doctor), LDoctors),
    %write('LDoctors: '), write(LDoctors), nl,
    findall(Anaesthetist, staff(Anaesthetist, anaesthetist, anaesthetist, _), LAnaesthetists),
    %write('LAnaesthetists: '), write(LAnaesthetists), nl,
    findall(InstrumentingNurse, staff(InstrumentingNurse, nurse, instrumenting_nurse, _), LInstrumentingNurses),
    %write('LInstrumentingNurses: '), write(LInstrumentingNurses), nl,
    findall(CirculatingNurse, staff(CirculatingNurse, nurse, circulating_nurse, _), LCirculatingNurses),
    %write('LCirculatingNurses: '), write(LCirculatingNurses), nl,
    findall(NurseAnaesthetist, staff(NurseAnaesthetist, nurse, nurse_anaesthetist, _), LNurseAnaesthetists),
    %write('LNurseAnaesthetists: '), write(LNurseAnaesthetists), nl,
    findall(MedicalActionAssistant, staff(MedicalActionAssistant, medical, medical_action_assistant, _), LMedicalActionAssistants),
    %write('LMedicalActionAssistants: '), write(LMedicalActionAssistants), nl,
    % Interseção das agendas dos doutores e staff adicional
    intersect_all_agendas(LDoctors,Day,LA_Doctors),
    intersect_all_agendas(LAnaesthetists,Day,LA_Anaesthetists),
    intersect_all_agendas(LInstrumentingNurses,Day,LA_InstrumentingNurses),
    intersect_all_agendas(LCirculatingNurses,Day,LA_CirculatingNurses),
    intersect_all_agendas(LNurseAnaesthetists,Day,LA_NurseAnaesthetists),
    intersect_all_agendas(LMedicalActionAssistants,Day,LA_MedicalActionAssistants),
    intersect_2_agendas(LA_Doctors,LA_Anaesthetists,LStaff),
    %intersect_2_agendas(LA_All,LA_InstrumentingNurses,LA_All),
    %intersect_2_agendas(LA_All,LA_CirculatingNurses,LA_All),
    %intersect_2_agendas(LA_All,LA_NurseAnaesthetists,LA_All),
    %intersect_2_agendas(LA_All,LA_MedicalActionAssistants,LA_All),
    agenda_operation_room1(Room,Day,LAgenda),
    free_agenda0(LAgenda,LFAgRoom),
    intersect_2_agendas(LStaff,LFAgRoom,LIntAgDoctorsRoom),
    remove_unf_intervals(TSurgery,LIntAgDoctorsRoom,LPossibilities),
    write('\nLPossibilities after filtering: '), write(LPossibilities), nl.
```

*Figura 9 - predicados alterados para o agendamento das cirurgias com o restante staff*

No ficheiro *schedule-first-staff* (adaptação do ficheiro *schedule-first-doctors*), que utiliza uma abordagem heurística para agendar as cirurgias nos primeiros intervalos disponíveis, a principal alteração foi a consideração da disponibilidade do staff completo. Deste modo, ao se encontrar o primeiro intervalo disponível para a cirurgia, o sistema determinou não só os horários dos médicos, mas também os dos outros profissionais necessários, priorizando a alocação dos recursos de forma eficiente.

```
schedule_all_surgeries_heuristic([], _, _).
schedule_all_surgeries_heuristic([OpCode | Rest], Room, Day) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, _, TSurgery, _),
    % Find earliest available time slot using the heuristic
    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 Staff schedules
    findall(Anaesthetist, assignment_surgery(OpCode, Anaesthetist), LAnaesthetists),
    insert_agenda_doctors((TinS, TfinS, OpCode), Day, LAnaesthetists),
    findall(InstrumentingNurse, assignment_surgery(OpCode, InstrumentingNurse), LInstrumentingNurses),
    insert_agenda_doctors((TinS, TfinS, OpCode), Day, LInstrumentingNurses),
    findall(CirculatingNurse, assignment_surgery(OpCode, CirculatingNurse), LCirculatingNurses),
    insert_agenda_doctors((TinS, TfinS, OpCode), Day, LCirculatingNurses),
    findall(NurseAnaesthetist, assignment_surgery(OpCode, NurseAnaesthetist), LNurseAnaesthetists),
    insert_agenda_doctors((TinS, TfinS, OpCode), Day, LNurseAnaesthetists),
    findall(Medical, assignment_surgery(OpCode, Medical), LMedicals),
    insert_agenda_doctors((TinS, TfinS, OpCode), Day, LMedicals).
```

*Figura 10 - predicado alterado para agendar o resto do staff para as cirurgias*

Por último, no ficheiro *schedule-more-booked-staff* (adaptação do ficheiro *schedule-more-booked*), que prioriza os profissionais mais ocupados, foi realizada uma adaptação para garantir que o membro do staff mais solicitado seja alocado primeiro, seguido da verificação da disponibilidade dos outros profissionais. Esta abordagem permitiu otimizar a utilização do staff, minimizando o tempo de espera para agendamento.

```
schedule_surgery_heuristic(OpCode, Room, Day) :-
    find_busiest_available_doctor(OpCode, Day, EarliestDoctor, StartTime),
    surgery_id(OpCode, OpType),
    surgery(OpType, _, TSurgery, _),
    EndTime is StartTime + TSurgery - 1,
    % Update operation room agenda
    retract(agenda_operation_room1(Room, Day, Agenda)),
    insert_agenda((StartTime, EndTime, OpCode), Agenda, NewAgenda),
    assertz(agenda_operation_room1(Room, Day, NewAgenda)),
    format('~nSurgery ~w inserted into room ~w agenda: (~w, ~w)~n', [OpCode, Room, StartTime, EndTime]),
    % Update only the earliest doctor's agenda first
    insert_agenda_doctors((StartTime, EndTime, OpCode), Day, [EarliestDoctor]),
    % Then update other assigned doctors' agendas if any
    findall(Doc, (assignment_surgery(OpCode, Doc), Doc \= EarliestDoctor), OtherDocs),
    insert_agenda_doctors((StartTime, EndTime, OpCode), Day, OtherDocs).
    % Update schedules for other staff
    findall(Anaesthetist, assignment_surgery(OpCode, Anaesthetist), LAnaesthetists),
    insert_agenda_staff((StartTime, EndTime, OpCode), Day, LAnaesthetists),
    findall(InstrumentingNurse, assignment_surgery(OpCode, InstrumentingNurse), LInstrumentingNurses),
    insert_agenda_staff((StartTime, EndTime, OpCode), Day, LInstrumentingNurses),
    findall(CirculatingNurse, assignment_surgery(OpCode, CirculatingNurse), LCirculatingNurses),
    insert_agenda_staff((StartTime, EndTime, OpCode), Day, LCirculatingNurses),
    findall(NurseAnaesthetist, assignment_surgery(OpCode, NurseAnaesthetist), LNurseAnaesthetists),
    insert_agenda_staff((StartTime, EndTime, OpCode), Day, LNurseAnaesthetists),
    findall(Medical, assignment_surgery(OpCode, Medical), LMedicals),
    insert_agenda_staff((StartTime, EndTime, OpCode), Day, LMedicals).
```

*Figura 11 - predicado alterado para agendar o resto do staff para as cirurgias*

Estas adaptações garantiram que o sistema tivesse em consideração todos os recursos necessários para realizar uma cirurgia, desde os médicos até os enfermeiros e outros profissionais envolvidos. O resultado é um processo de agendamento mais robusto.

# 5. Conclusão

O presente relatório abordou o problema de agendamento de cirurgias em salas de operações, detalhando os desafios, soluções implementadas e melhorias realizadas no sistema base.

Inicialmente, foi analisado o código base, que se concentra na disponibilidade de médicos e salas, oferecendo uma solução eficiente para casos simples e de menor escala. Contudo, ao aplicar abordagens exaustivas para obter a solução ótima, verificou-se a limitação da complexidade fatorial (O(n!)), que impossibilita a apresentação de resultados em casos com mais de 10 cirurgias.

Diante dessas limitações, foram explorados algoritmos heurísticos que, embora não garantam soluções ótimas, apresentam resultados satisfatórios com tempos de execução significativamente menores. As heurísticas implementadas, baseadas na disponibilidade antecipada do médico e na taxa de ocupação, demonstraram-se eficientes e escaláveis, sendo capazes de lidar com um maior número de cirurgias em cenários reais. A comparação com a solução ótima evidenciou que os desvios nas soluções geradas são aceitáveis.

Além disso, foi realizada uma extensão do código para incluir o restante do staff envolvido na realização das cirurgias, como anestesistas, enfermeiros e outros profissionais essenciais. As adaptações realizadas nos algoritmos garantiram que o agendamento levasse em conta a disponibilidade de todos os membros do staff, evitando conflitos de horários e maximizando a eficiência.

Assim, a utilização de algoritmos heurísticos e a inclusão do restante do staff no processo de agendamento oferecem uma solução prática e eficaz para a gestão de salas de operações em contextos reais.