SPRINT 2

Algoritmia Avançada

Miguel Oliveira 1211281

Rodrigo Castro 1220636

Rodrigo Cardoso 1221083

Mário Ribeiro 1221019

Instituto Superior de Engenharia do Porto

Table of Contents

Índice de Figuras

Introdução

A gestão eficiente de recursos hospitalares, especialmente no que diz respeito ao agendamento de cirurgias, é uma tarefa complexa que envolve a alocação de médicos, salas de operação e outros membros da equipa, como anestesistas e pessoal de limpeza.

O presente estudo aborda a implementação de algoritmos de agendamento de cirurgias com o objetivo de otimizar o uso dos recursos hospitalares. A partir da análise de diferentes heurísticas, procuramos entender até que ponto é possível encontrar soluções eficazes em termos de tempo e de utilização dos recursos, em comparação com métodos tradicionais que geram todas as soluções possíveis de forma exaustiva. O estudo explora também a complexidade computacional dos métodos de agendamento, demonstrando como a abordagem heurística pode ser uma solução viável para cenários em que o número de cirurgias a ser agendado aumenta significativamente.

Algoritmos Básicos de Agendamento de Cirurgias

O código abordado nas aulas implementa um sistema de agendamento de cirurgias de um hospital. O objetivo principal é gerir as agendas de médicos e salas de operação, alocando cirurgias de forma otimizada considerando os horários disponíveis e as restrições impostas. No código temos:

1. **Definição de Factos Dinâmicos**:
    a. Os predicados availability/3, agenda_staff/3, agenda_operation_room/3 e outros são definidos como dinâmicos, permitindo que sejam modificados durante a execução do programa.

2. **Agendas dos Médicos e Salas**:
    a. A agenda_staff/3 especifica horários de trabalho e alocação de médicos.
    b. A agenda_operation_room/3 representa os horários já reservados para as salas de operação.

3. **Timetable e Cirurgias**:
    a. O timetable/3 define o horário disponível de cada médico em um dia específico.
    b. O predicado surgery/4 descreve os tipos de cirurgia, incluindo duração da anestesia, da cirurgia, e do tempo de limpeza.
    c. surgery_id/2 mapeia identificadores únicos de cirurgias aos tipos de cirurgia.

4. **Alocação de Cirurgias**:
    a. assignment_surgery/2 relaciona um código de cirurgia a um médico.
    b. Predicados como availability_all_surgeries/3 e availability_operation/5 verificam horários disponíveis para encaixar cirurgias nas agendas.

5. **Funções Auxiliares para Disponibilidade**:
    a. Os predicados **free_agenda0/2 e intersect_all_agendas/3, intersect_availability/4, intersect_2_agendas** determinam os intervalos livres na agenda de médicos e salas e permitem que não haja conflitos de horários.

6. **Adaptação de Horários**:
    a. O predicado adapt_timetable/4 ajusta a disponibilidade do staff com base nos seus horários de trabalho e cirurgias já agendadas.

7. **Agendamento**:
    a. schedule_all_surgeries/2 tenta marcar todas as cirurgias planeadas para aquela dia iterando uma a uma e atribuindo um staff disponível de acordo com as restrições mas seguindo a ordem em que as mesmas estão na base de conhecimento o que a torna numa solução pouco eficiente e interessante.
    b. O predicado obtain_better_sol/5 ao contrário do schedules_all_surgeries que só testa uma ordem de cirurgias, testa todas as ordens possíveis de cirurgias e no final mostra-nos o tempo que demorou a execução do mesmo e a melhor agenda possível para aquela sala naquele dia.

8. **Atualização e Inserção de Agendas**:
    a. Funções como insert_agenda/3 e insert_agenda_doctors/3 manipulam a agenda de médicos e salas ao inserir novas cirurgias.
9. **Cálculo de Intervalos e Ajustes**:
    a. Predicados como remove_unf_intervals/3 e schedule_first_interval/3 são usados para calcular os intervalos adequados para cada cirurgia.

Em resumo, o código gerência a logística do agendamento de cirurgias, alocando médicos e salas com base nas suas disponibilidades de forma a evitar conflitos de horários.

Estudo de Complexidade

No âmbito da gestão de cirurgias e recursos hospitalares, é fundamental compreender os limites práticos do sistema atual para gerar todas as soluções possíveis e selecionar a melhor. Esta análise de complexidade surge com o objetivo de determinar até que ponto, em termos de número de cirurgias, o método de geração exaustiva de soluções permanece viável dentro de prazos razoáveis.

O estudo permite avaliar o desempenho do sistema em cenários progressivamente mais complexos, ajudando a identificar o limite prático para o número de cirurgias que podem ser processadas em tempo útil. Adicionalmente, os resultados deste estudo fundamentam a necessidade de explorar alternativas mais eficientes, como heurísticas ou algoritmos genéticos, quando o número de cirurgias ultrapassa esse limite.

Para realizar esta análise, partimos de um cenário inicial com 3 cirurgias, aumentando gradualmente o número de cirurgias envolvidas. Para cada dimensão do problema, medimos o tempo necessário para gerar e avaliar todas as soluções, documentando os resultados.

Este processo ajuda a determinar o ponto em que o tempo de execução cresce de forma impraticável, estabelecendo diretrizes para futuras abordagens e melhorias no sistema.

Número de cirurgias	**Número de soluções**	**Melhor marcação de atividades da sala**	**Último tempo da última cirurgia(m)**	**Tempo para gerar solução(s)**
3	6	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (1000, 1059, so099999)]	804	~0.014
4	24	[(520, 579, so100000), (580, 639, so100001), (640, 714, so100003), (715, 804, so100002), (805, 864, so100004), (1000, 1059, so099999)]	864	~0.35
4 cirurgias com entrelaçamento de 2 médicos na mesma cirurgia numa cirurgia	24	[(520, 579, so100000), (580, 654, so100003), (655, 714, so100004), (715, 804, so100002), (805, 864, so100001), (1000, 1059, so099999)]	864	~0.35
5	120	[(520, 579, so100000), (580, 639, so100004), (640, 729, so100002), (730, 804, so100005),	939	~0.96

		(805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]		
5 cirurgias com entrelaçamento de 2 médicos na mesma cirurgia em duas cirurgias	120	[(520, 579, so100000), (580, 639, so100004), (640, 714, so100005), (715, 804, so100002), (805, 879, so100003), (880, 939, so100001), (1000, 1059, so099999)]	939	~1.01
6	720	[(520,579,so100000),),(580,639,so100004),(640,714,so100005),(715,804,so100002),(805,879,so100003), (880,939,so100001),(940,999,so100006),(1000,1059,so099999)]	999	~5.05
7	5040	[(520,579,so100000),(580,669,so100007), (670,759,so100002),(791,850,so100006),(851,910,so100001),(1000,1059,so099999) ,(1060,1134,so100005),(1141,1200,so100004),(1201,1275,so100003)]	1149	~22.26
8	40320	[(520,579,so100000),(580,639,so100004), (640,729,so100002),(730,789,so100008), (791,865,so100003),(866,925,so100001) ,(926,985,so100006),(1000,1059,so099999), (1060,1134,so100005),(1135,1224,so100007)]	1224	~28.12
8 cirurgias com entrelaçamento de 2 médicos na mesma cirurgia em três cirurgias	40320	[(520,579,so100000),(580,639,so100004), (640,729,so100002),(730,789,so100008), (791,865,so100003),(866,925,so100001) ,(926,985,so100006),(1000,1059,so099999), (1060,1134,so100005),(1135,1224,so100007)]	1224	~30.52
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	~35.05
9 cirurgias com entrelaçamento de 2 médicos na	362880	[(520,579,so100000),(580,639,so100004), (640,699,so100008),(700,789,so100002), (790,849,so100009),(850,909,so100001),	1299	~40.05

mesma cirurgia em quatro cirurgias		(910,969,so100006),(1000,1059,so099999), (1060,1134,so100005),(1135,1209,so100003), (1210,1299,so100007)]		
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	~352.04
11	39916800			
12	479001600			
13	6227020800			

Conclusão

A viabilidade em substituir o método de geração exaustiva de soluções ou não depende de alguns fatores externos, como por exemplo, em quanto tempo é necessário obter o resultado da solução? No contexto clinico/hospitalar em que nos encontramos a urgência em obter um bom resultado o mais rápido possível é providencial. Para além disso a ocorrência de imprevistos, como cancelamento de consultas/cirurgias é uma realidade para a qual devemos estar preparados. Sendo que o cancelamento de um agendamento pode gerar a possibilidade de reformular por completo o que chamamos de melhor solução convém que o sistema consiga reagir rapidamente a estes imprevistos. Dito isto e através da análise realizada acima, chegamos à conclusão que a partir das 11 cirurgias tornar-se-ia mais viável a utilização de alternativas mais eficientes, como heurísticas ou algoritmos genéticos, que apesar de não nos fornecerem "a melhor solução" irão fornecer uma solução otimizada o suficiente em que o custo tempo/otimização será benéfico.

Explicação da Heurística 1

Para o desenvolvimento da heurística primeiro definimos o critério que iríamos utilizar, optámos por realizar o proposto pelo professor

> *("The next surgery to consider is one for the doctor that is available early. What is to 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 at 8:30 he/she will not be available at 8:00am for a surgery of type so2 because it takes 60 minutes)"),*

mas adaptámos para a utilização não só de médicos, mas também do outro staff presente nas diversas fases da cirurgia. No desenvolvimento desse critério a nossa ideia passou por desenvolver um predicado que recebesse como parâmetro uma lista de operações da qual queremos selecionar e retornasse a mais indicada. Para encontrar a mais indicada utilizarmos o predicado find_valid_interval(é explicado no relatório na parte do obtain_better_sol). A nossa ideia passou por percorrer todas as operações recebidas como parâmetro e para cada uma chamar o predicado find_valid_interval que retorna o intervalo válido para a realização da cirurgia. No final do predicado pegamos na lista que contém todos os intervalos de cada cirurgia e ordenamos pelo valor do início do intervalo, ou seja a cirurgia que pudesse ser marcada mais cedo. Se houverem duas cirurgias que comecem ao mesmo tempo ele escolhe como a mais indicada a que acabar mais cedo entre elas. No predicado availability_all_surgeries2 apenas remove a que foi escolhido da lista, e volta a chamar o critério para escolher a próxima, e faz isto até não haverem cirurgias para analisar.

```
% Predicado que escolhe a próxima cirurgia com base no primeiro tempo disponível
select_next_surgeryCriteria1(LOpCode, BestOpCode, Interval, LDoctorsSurgery, LStaffAnesthesy,LStaffCleaning) :-
    write('Lista de operações a ser analisada: '), write(LOpCode), nl,
    findall(
        (OpCode, Interval, LDoctorsSurgery, LStaffAnesthesy,LStaffCleaning),
        (
            member(OpCode, LOpCode),
            surgery_id(OpCode, OpType),
            %write('A analisar a cirurgia: '), write(OpCode), nl,
            surgery(OpType, _, _, _),
            availability_operation(OpCode, _, _, Interval, LDoctorsSurgery, LStaffAnesthesy,LStaffCleaning)
        ),
        Candidates
    ),
    % Selecionar a cirurgia com o menor intervalo de início
    sort(2, @=<, Candidates, SortedCandidates),
    SortedCandidates = [(BestOpCode, Interval, LDoctorsSurgery, LStaffAnesthesy,LStaffCleaning) | _],
    write('BestOpCode='), write(BestOpCode), nl.
```

Figure 1 - Critério 1

```
availability_all_surgeries2([], _, _, MaxEndTime) :-
    write('Tempo final de todas as cirurgias: '), write(MaxEndTime), nl.

% Caso recursivo: Selecionar e agendar cirurgias
availability_all_surgeries2(LOpCode, Room, Day, CurrentMaxEndTime) :-
    % Selecionar a próxima cirurgia com base no critério
    (   select_next_surgeryCriteria1(LOpCode, OpCode, Interval, LDoctorsSurgery, LStaffAnesthesy,LStaffCleaning)
    ->  % Se uma cirurgia for encontrada, continuar o agendamento
        surgery_id(OpCode, OpType),
        surgery(OpType, TAnesthesy, TSurgery, TCleaning),
        calculate_intervals(Interval, TAnesthesy, 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, LStaffAnesthesy),
        insert_agenda_staff((MinuteStartCleaning, MinuteEndProcess, OpCode), Day, LStaffCleaning),
        retractall(availability(_, _, _)),
        findall(_,
            (
                agenda_staff1(D, Day, L),
                free_agenda0(L, LFA),
                adapt_timetable(D, Day, LFA, LFA2),
                assertz(availability(D, Day, LFA2))
            ), _),
        MaxEndTime is max(CurrentMaxEndTime, MinuteEndProcess),
        delete(LOpCode, OpCode, RemainingLOpCode),
        availability_all_surgeries2(RemainingLOpCode, Room, Day, MaxEndTime)
    ;   % Se nenhuma cirurgia puder ser selecionada, falhar explicitamente
        write('Nenhuma cirurgia pode ser agendada com os critérios atuais.'), nl,
        fail
    ).
```

Figure 2 - Predicado que chama utiliza o critério para escolher a próxima cirurgia a ser marcada

```
schedule_all_surgeriesHeuristic(Room, Day) :-
    get_time(Ti),
    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),
    assertz(agenda_operation_room1(Or, Date, Agenda)),

    % Atualizar a disponibilidade de staff
    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_surgeries2(LOpCode, Room, Day, 0),
    !,
        % Registrar o tempo de fim
    get_time(EndTime),

    % Calcular e mostrar o tempo total de execução
    TotalTime is EndTime - Ti,
    write('Tempo total de execução: '), write(TotalTime), nl.
```

Figure 3 - Predicado inicial de setup e cálculo de tempo de execução

```
?- schedule_all_surgeriesHeuristic(or1,20241130).
Lista de operaÃ§Ãµes a ser analisada: [so100001,so100002]
Cirurgia: so100001 tem Intervalo disponivel : 100,130
Cirurgia: so100002 tem Intervalo disponivel : 100,129
BestOpCode=so100002
Lista de operaÃ§Ãµes a ser analisada: [so100001]
Cirurgia: so100001 tem Intervalo disponivel : 130,160
BestOpCode=so100001
Tempo final de todas as cirurgias: 160
Tempo total de execuÃ§Ã£o: 0.014530897140502930
true.
```

Figure 4 - Exemplo para cirurgias que começam ao mesmo minuto

```
?- schedule_all_surgeriesHeuristic(or1,20241130).
Lista de operaÃ§Ãµes a ser analisada: [so100001,so100002]
Cirurgia: so100001 tem Intervalo disponivel : 100,130
Cirurgia: so100002 tem Intervalo disponivel : 100,131
BestOpCode=so100001
Lista de operaÃ§Ãµes a ser analisada: [so100002]
Cirurgia: so100002 tem Intervalo disponivel : 131,162
BestOpCode=so100002
Tempo final de todas as cirurgias: 162
Tempo total de execuÃ§Ã£o: 0.019124031066894531
true.
```

Figure 5 - Exemplo 2 para cirurgias que começam ao mesmo minuto

```
?- schedule_all_surgeriesHeuristic(or1,20241130).
Lista de operaÃ§Ãµes a ser analisada: [so100001,so100002]
Cirurgia: so100001 tem Intervalo disponivel : 100,130
Cirurgia: so100002 tem Intervalo disponivel : 99,129
BestOpCode=so100002
Lista de operaÃ§Ãµes a ser analisada: [so100001]
Cirurgia: so100001 tem Intervalo disponivel : 130,160
BestOpCode=so100001
Tempo final de todas as cirurgias: 160
Tempo total de execuÃ§Ã£o: 0.015612125396728516
true.
```

Figure 6 - Exemplo 3 para cirurgias que que começam em minutos diferentes

Análise Comparativa da Heurística

Número de cirurgias	Solução ótima	Tempo Final para a cirurgia em gerar a melhor solução (minutos)	Tempo Final para a cirurgia utilizando a Heurística (minutos)	Tempo de geração da melhor solução (segundos)	Tempo de geração da solução Heurística (segundos)	Solução com a Heurística
3	[(300,330,so100001), (391,441,so100002), (580,680,so100003)]	680	680	0.065	0.021	[(300,330,so100001), (391,441,so100002), (580,680,so100003)]
4	[(300,330,so100001), (331,441,so100004), (442,492,so100002), (580,680,so100003)]	680	680	0.58	0.04	(300,330,so10001), (331,441,so10004), (442,492,so10002), (580,680,so10003)]
5	[(300,330,so100001), (331,441,so100004), (442,492,so100002), (580,680,so100003), (696,726,so100005)]	726	926	4.55	0.06	[(300,330,so100001), (331,441,so100004), (442,472, so100005), (580,680,so100003), (876,926,so100002)]
6	[(300,350,so10006) (351,461,so100004),	757	926	23.86	0.068	[(300,330,so100001), (331,381,so100006),

	(462,512,so100002), (580,680,so100003), (696,726,so100005), (727,757,so100001)]					(382,492,so100004), (580,680,so100003), (696,726,so100005) (876,926, so100002)]
7	[(300,330,so100001), (331,441,so100004), (442,492,so100002), (580,680,so100003), (681,791,so100007), (801,851,so100006), (896,926,so100005)]	926	982	178.9	0.089	[(300,330,so100001), (331,381,so100006), (382,492,so100007), (580,680,so100003), (696,726,so100005), (821,931,so100004), (932, 982, so10002)]

Análise Resultados Obtidos Heurística1 vs BetterSol

Os resultados obtidos foram expectáveis, no início a heurística estava a conseguir dar uma solução igualmente boa à better_sol e à medida que fomos aumentando o número de cirurgias isso deixou de se suceder passado a haver alguma diferença entre os tempos finais obtidos para cada uma das soluções. Conseguimos também perceber o porquê de um método heurístico ser a solução mais indicada para um caso real em que normalmente temos um número mais avultado de cirurgias, pois para 7 cirurgias a solução demorou cerca de 3 minutos, o que ainda é um valor aceitável, mas para números maiores não iríamos conseguir utilizar esta solução. Além disso a discrepância de eficácia dos resultados entre esta heurística e a better_sol não foi assim tão grande, por tanto consideramos que esta heurística, apesar de não ser a melhor solução, é uma boa solução e perfeitamente aceitável.

Explicação da Heurística 2

Para o desenvolvimento desta heurística, foi definido um critério que selecciona a próxima cirurgia com base no menor número total de elementos de staff necessário para a sua realização. Este critério considera não só os médicos e anestesistas, mas também o pessoal responsável pela limpeza, garantindo uma abordagem mais completa na gestão dos recursos humanos.

O predicado que implementa esta lógica é o select_next_surgeryCriteria2. Este predicado analisa uma lista de operações disponíveis (LOpCode) e devolve:

- código da cirurgia seleccionada (BestOpCode),
- intervalo de tempo para a sua realização (Interval),
- As listas de médicos (LDoctorsSurgery), anestesistas (LStaffAnesthesy) e equipa de limpeza (LStaffCleaning).

```
% Predicado que escolhe a próxima cirurgia com base no menor número de staff envolvido
select_next_surgeryCriteria2(LOpCode, BestOpCode, Interval, LDoctorsSurgery, LStaffAnesthesy,LStaffCleaning) :-
    write('Lista de operações a ser analisada: '), write(LOpCode), nl,
    findall(
        (OpCode, Interval, LDoctorsSurgery, LStaffAnesthesy, LStaffCleaning, StaffCount),
        (
            member(OpCode, LOpCode),
            surgery_id(OpCode, OpType),
            surgery(OpType, _, _, _),
            availability_operation(OpCode, _, _, Interval, LDoctorsSurgery, LStaffAnesthesy, LStaffCleaning),
            length(LDoctorsSurgery, NumDoctors),
            length(LStaffAnesthesy, NumAnesthesy),
            length(LStaffCleaning, NumCleaning),
            StaffCount is NumDoctors + NumAnesthesy + NumCleaning
        ),
        Candidates
    ),
    maplist(
        % Extract the 5th element for sorting
        [(_, _, _, _,_, StaffCount), StaffCount]>>true,
        Candidates, StaffCounts
    ),
    % Pair candidates with their StaffCounts for sorting
    pairs_keys_values(Pairs, StaffCounts, Candidates),
    keysort(Pairs, SortedPairs),
    pairs_values(SortedPairs, SortedCandidates),

    write('Candidatos ordenados: '), write(SortedCandidates), nl,

    % Extract the best candidate
    SortedCandidates = [(BestOpCode, Interval, LDoctorsSurgery, LStaffAnesthesy, LStaffCleaning,_) | _],
    write('Melhor OpCode: '), write(BestOpCode), nl.
```

Figure 7 - select_next_surgeryCriteria2

Funcionamento do Predicado

1. **Identificação dos Candidatos**

Utiliza-se o predicado findall para construir uma lista de candidatos (Candidates), onde cada candidato inclui:

- código da operação (OpCode),
- intervalo válido para a sua realização (Interval),
- As listas de médicos, anestesistas e equipa de limpeza associados (LDoctorsSurgery, LStaffAnesthesy, LStaffCleaning),
- número total de pessoal envolvido (StaffCount), calculado como a soma do tamanho das três listas.

2. **Ordenação dos Candidatos**

A lista de candidatos é transformada em pares (Pairs), associando cada candidato ao respectivo valor de StaffCount. Estes pares são ordenados por StaffCount através do predicado keysort, priorizando as cirurgias que requerem menos pessoal.

3. **Selecção da Melhor Cirurgia**

Após a ordenação, o primeiro elemento da lista ordenada representa a cirurgia mais indicada. Este candidato é extraído e devolvido como resultado.

4. **Mensagens de Depuração**

Durante a execução, são apresentadas mensagens para facilitar a compreensão do processo:

- A lista de operações analisadas,
- A lista de candidatos ordenados pelo número de elementos de staff,
- código da cirurgia seleccionada.

Exemplo de Funcionamento

Se a lista de operações incluir diferentes quantidades de pessoal necessário, o predicado:

1. Calcula o número total de pessoal para cada cirurgia.
2. Ordena as cirurgias com base nesse número.
3. Devolve a cirurgia que requer o menor número de elementos de staff.

Este critério é especialmente útil em cenários com recursos humanos limitados, permitindo uma gestão mais eficiente do pessoal disponível, sem comprometer a qualidade do serviço prestado. O Resto do processo é semelhante à Heuristica 1, a única exceção é mesmo o criterio utilizado para a seleção dos mesmos.

Análise Comparativa da Heurística 2

Número de cirurgias	Solução ótima	Tempo Final para a cirurgia em gerar a melhor solução (minutos)	Tempo Final para a cirurgia utilizando a Heurística (minutos)	Tempo de geração da melhor solução (segundos)	Tempo de geração da solução Heurística (segundos)	Solução com a Heurística
3	[(300,330,so100001), (391,441,so100002), (580,680,so100003)]	680	680	0.065	0.034	[(300,330,so100001), (391,441,so100002), (580,680,so100003)]
4	[(300,330,so100001), (331,441,so100004), (442,492,so100002), (580,680,so100003)]	680	680	0.58	0.036	[(300, 330, so100001), (391, 441, so100002), (442, 472, so100005) (580, 680, so100003)
5	[(300,330,so100001), (331,441,so100004), (442,492,so100002), (580,680,so100003), (696,726,so100005)]	726	931	4.55	0.03	[(300, 330, so100001), (391, 441, so100002), (442, 472, so100005), (580, 680, so100003), (821, 931, so100004)
6	[(300,350,so10006) (351,461,so100004), (462,512,so100002), (580,680,so100003),	757	931	23.86	0.045	[(300, 330, so100001), (331, 381, so100006), (391, 441, so100002), (442, 472, so100005), (580, 680, so100003), (821, 931, so100004)

	(696,726,so100005), (727,757,so100001)]					
7	[(300,330,so100001), (331,441,so100004), (442,492,so100002), (580,680,so100003), (681,791,so100007), (801,851,so100006), (896,926,so100005)]	926	931	178.9	0.046	[(300, 330, so100001), (331, 381, so100006), (391, 441, so100002), (471, 571, so100003), (572, 602, so100005), (681, 791, so100007), (821, 931, so100004)]

Análise Resultados Obtidos Heurística2 vs BetterSol

Os resultados obtidos foram coerentes com as expectativas, uma vez que, inicialmente, a heurística estava a conseguir gerar soluções muito próximas da solução ótima. Contudo, à medida que o número de cirurgias aumentou, começou a ser visível uma maior discrepância entre os tempos finais das soluções geradas pela heurística e a solução ótima. Para 3 e 4 cirurgias, os tempos finais das soluções foram idênticos (680 minutos), sendo que a principal diferença se verificou no tempo de geração, que foi muito mais rápido para a heurística. Nos casos de 5 e 6 cirurgias, a diferença entre as soluções geradas pela heurística e a ótima aumentou, principalmente no que diz respeito ao tempo final, mas a heurística continuou a demonstrar tempos de geração muito rápidos, com valores inferiores a 1 segundo. Para 7 cirurgias, a discrepância entre as soluções cresceu, com a heurística a gerar um tempo final de 931 minutos, em comparação com os 926 minutos da solução ótima, mas novamente com uma diferença significativa no tempo de geração, com a heurística a levar apenas 0,046 segundos.

Inclusão de toda a Equipa Necessária e Todas as Fases da Operação

Primeiramente, começámos por alterar o predicado assignment_surgery. Inicialmente, este tinha o formato assignment_surgery (surgeryId, staffId), mas revelou-se limitado, uma vez que temos staff envolvidos em diferentes fases de uma cirurgia. Por isso, alterámos o formato para assignment_surgery (surgeryId, staffId, phase), permitindo assim a atribuição do staff necessário e a distinção das suas responsabilidades em cada uma das fases da cirurgia (anestesia, cirurgia, limpeza).

```
%assignment_surgery(OpCode,Staff,Phase).
%Exemplos->
assignment_surgery(so100001,d002,surgeryPhase).
assignment_surgery(so100001,d001,anesthesyPhase).
assignment_surgery(so100001,e003,cleaningPhase).
```

Figure 8 - assignment_surgery example

De seguida, ajustámos o processo de atribuição do staff necessário a uma dada cirurgia, de forma a ter em conta as diferentes fases do procedimento. Para esta adaptação, começámos por modificar o predicado availability_operation. Anteriormente, este predicado devolvia todas as possibilidades de agendamento para a cirurgia em questão. Agora, alterámo-lo para retornar apenas o primeiro agendamento possível, evitando assim que o predicado availability_all_surgeries tenha de recorrer ao uso do schedule_first_interval, que selecionava o primeiro intervalo de uma lista de possibilidades.

Além disso, enquanto antes era devolvida uma lista genérica de médicos, agora é necessário distinguir entre o staff de anestesia, cirurgia e limpeza. Dentro do predicado, começámos por obter todos os membros do staff atribuídos à cirurgia para as fases de anestesia, cirurgia e limpeza, bem como as interseções das respetivas agendas livres. Em seguida, consultámos a agenda da sala de operações para obter os seus horários disponíveis. Por fim, invocámos o predicado find_first_interval, responsável por determinar o horário compatível entre as agendas da sala de operações e do staff das diferentes fases.

```
availability_operation(OpCode, Room, Day, Interval, LStaffSurgeryPhase,LStaffAnesthesyPhase,LStaffCleaningPhase) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, TAnesthesia, TSurgery, TCleaning),
    findall(Staff, assignment_surgery(OpCode, Staff, surgeryPhase), LStaffSurgeryPhase),
    findall(Staff, assignment_surgery(OpCode, Staff, anesthesyPhase), LStaffAnesthesyPhase),
    findall(Staff, assignment_surgery(OpCode, Staff, cleaningPhase), LStaffCleaningPhase),

    intersect_all_agendas(LStaffSurgeryPhase, Day, LASurgery),
    intersect_all_agendas(LStaffAnesthesyPhase, Day, LAAnesthesy),
    intersect_all_agendas(LStaffCleaningPhase, Day, LACleaning),

    agenda_operation_room1(Room,Day,LAgenda),
    free_agenda0(LAgenda,LFAgRoom),
    find_valid_interval(LAAnesthesy, LASurgery, LACleaning,LFAgRoom, TAnesthesia, TSurgery, TCleaning, Interval),!,
    write('Cirurgia'),write(OpCode),write(' tem Intervalo disponivel : '), write(Interval), nl.
```

Figure 9 - availability_operation

O predicado find_first_interval recebe como parâmetros a agenda livre do staff de anestesia, do staff de cirurgia, do staff de limpeza, a agenda livre da sala de operações e os tempos necessários para as fases de anestesia, cirurgia e limpeza. Este predicado retorna o primeiro intervalo disponível. O que ele faz é iterar pelos intervalos de disponibilidade da sala de operações minuto a minuto, verificando a compatibilidade entre os horários do staff e da sala. Entrando mais ao pormenor neste predicado, ele primeiro verifica-se se todo o tempo de duração da cirurgia cabe na disponibilidade da sala. Caso contrário, avança-se para o próximo intervalo livre da sala de operações. Se a condição for cumprida, verifica-se que o staff de anestesia deve estar presente durante o tempo de anestesia e o tempo de cirurgia, que o staff de cirurgia deve estar presente apenas durante o tempo de cirurgia e que o staff de limpeza deve estar presente desde o final da cirurgia até ao término do tempo necessário para a limpeza. Quando encontrarmos o primeiro minuto inicial que cumpra todas estas condições o predicado retorna o intervalo correspondente, na forma (tempo início do período de anestesia, tempo final da limpeza), representando o período total de ocupação da sala para essa cirurgia. Se porventura não for encontrado nenhum intervalo retornará no argumento do intervalo o valor false.

```
find_valid_interval(StaffAnesthesiaAvailable, StaffSurgeryAvailable, StaffCleaningAvailable,RoomAvailable, TAnesthesia, TSurgery, TCleaning, Interval) :-
    % Para cada intervalo de disponibilidade da sala
    member((RoomStart, RoomEnd), RoomAvailable),
    TotalTime is TAnesthesia+TSurgery+TCleaning,  % Soma do tempo de anestesia, cirurgia e limpeza
    MaxStart is RoomEnd - TotalTime,  % Calcula o último minuto em que o processo pode começar
    between(RoomStart, MaxStart, StartAnesthesia), % Para cada minuto entre o início e o último minuto de cada bloco de tempo disponível da sala
    % Assegurar que o intervalo total (anestesia + cirurgia + limpeza) caiba dentro do dia
    StartAnesthesia + TAnesthesia + TSurgery =< 1440,
    StartSurgery is StartAnesthesia + TAnesthesia,
    StartCleaning is StartSurgery + TSurgery,
    % Verific a disponibilidade da sala de operação
    RoomStart =< StartAnesthesia,
    RoomEnd >= (StartAnesthesia + TAnesthesia + TSurgery + TCleaning),
    % Verifica a disponibilidade do staff de anestesia
    member((AnesthesiaStart, AnesthesiaEnd), StaffAnesthesiaAvailable),
    AnesthesiaStart =< StartAnesthesia,
    AnesthesiaEnd >= (StartAnesthesia + TAnesthesia + TSurgery),
    % Verifica a disponibilidade do staff de cirurgia
    member((SurgeryStart, SurgeryEnd), StaffSurgeryAvailable),
    SurgeryStart =< StartSurgery,
    SurgeryEnd >= (StartSurgery + TSurgery),
    % Verifica a disponibilidade do staff de limpeza
    member((CleaningStart, CleaningEnd), StaffCleaningAvailable),
    CleaningStart =< StartCleaning,
    CleaningEnd >= (StartSurgery + TSurgery + TCleaning),

    % Calcular o fim do processo
    EndProcess is StartAnesthesia + TAnesthesia + TSurgery + TCleaning,
    % Retornar o intervalo
    Interval = (StartAnesthesia, EndProcess),!.
% Caso não encontre nenhum intervalo válido, retorna false
find_valid_interval(_, _, _, _, _, _, _, false) :-…
```

Figure 10 - find_first_interval

No predicado availability_all_surgeries, adicionámos o método calculate_intervals, que, a partir de um intervalo e dos tempos de atribuídos a cada fase da cirurgia, calcula o minuto inicial e final em que o staff de anestesia, cirurgia e limpeza irão presente na sala. Este cálculo é necessário para que possamos posteriormente inserir os respetivos tempos nas agendas.

Outra alteração significativa foi a substituição do antigo insert_agenda_staff por três insert_agenda_staff, um para cada staff pertencente a uma fase da cirurgia. Anteriormente, os blocos de tempo eram iguais para todos os membros do staff, mas, agora, como o staff está presente em momentos diferentes na sala de operações, é necessário inserir os tempos de forma diferenciada. Assim, ajustámos a agenda do staff de anestesia com os respetivos tempos inicial e final, aplicámos a mesma lógica para ajustar a agenda do staff de cirurgia e, finalmente, procedemos à inserção dos tempos ajustados para o staff de limpeza, com o tempo inicial e final adaptado à ao intervalo atribuída à limpeza.

```
availability_all_surgeries([],_,_).
availability_all_surgeries([OpCode|LOpCode],Room,Day):-
    surgery_id(OpCode,OpType),surgery(OpType,TAnesthesy,TSurgery,TCleaning),
    availability_operation(OpCode,Room,Day,Interval,LDoctorsSurgery,LStaffAnesthesy,LStaffCleaning),
    write('Cirurgia a ser agendada: '), write(OpCode), write(' - Intervalo: '), write(Interval), nl,
    calculate_intervals(Interval,TAnesthesy,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,LStaffAnesthesy),
    insert_agenda_staff((MinuteStartCleaning,MinuteEndProcess,OpCode),Day,LStaffCleaning),
    retractall(availability(_,_,_)),
    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).
```

Figure 11 - availability_all_surgeries

```
% O predicado calcula os intervalos de tempo para anestesia, cirurgia e limpeza
calculate_intervals((Start, End), TAnesthesia, TSurgery, TCleaning, MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess) :-
    % O início da anestesia é o início do intervalo (Start)
    MinuteStartAnesthesia = Start,
    % O início da cirurgia é o final da anestesia, ou seja, após a duração de anestesia
    MinuteStartSurgery is MinuteStartAnesthesia + TAnesthesia,
    % O início da limpeza é o final da cirurgia, ou seja, após a duração da cirurgia
    MinuteStartCleaning is MinuteStartSurgery + TSurgery,
    % O fim do processo é o final da limpeza, ou seja, após a duração da limpeza
    MinuteEndProcess is MinuteStartCleaning + TCleaning.
```

Figure 12 - calculate_intervals

Conclusões

Em conclusão, a implementação de heurísticas para o agendamento de cirurgias demonstrou ser uma abordagem eficaz para otimizar a utilização dos recursos hospitalares, considerando as limitações de tempo e a complexidade das operações. Através das duas heurísticas propostas, foi possível reduzir o tempo de espera dos pacientes e melhorar a alocação dos recursos humanos, especialmente no que diz respeito à disponibilidade de médicos, anestesistas e outros profissionais envolvidos no processo cirúrgico. Embora os métodos tradicionais de agendamento, que exploram todas as possibilidades, possam fornecer soluções ótimas, a utilização de heurísticas apresentou uma solução prática e eficiente para cenários de maior escala, onde o número de cirurgias e variáveis envolvidas torna o problema computacionalmente desafiador.