Planning Surgeries

Relatório

Instituto Superior de Engenharia do Porto

Turma 3DA

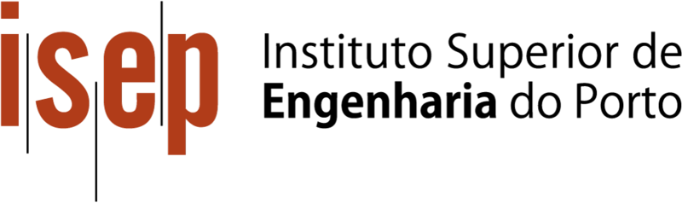
Grupo 3

1221063 Pedro Manuel Rocha Pires

1221029 Gabriel Pedro Almeida Monteiro

1220916 Rodrigo da Costa Nunes

1221031 José Silva



Índice

[Código Base 3](#_Toc183167737)

[Estudo da Complexidade 4](#_Toc183167738)

[Heurísticas 5](#_Toc183167739)

[Inclusão de Todo o Staff 6](#_Toc183167740)

[Conclusão 7](#_Toc183167741)

# Código Base

### **Descrição Geral**

Este código foi desenvolvido para organizar o agendamento de cirurgias nas salas de operação, tendo em conta a disponibilidade dos médicos. Ele utiliza uma base de conhecimento dinâmica, com factos e predicados que armazenam informações cruciais sobre horários e disponibilidade.

### **Base de Conhecimento**

### **Predicados:**

* **free\_agenda0**: Determina os períodos livres com base nos turnos previamente agendados.
* **intersect\_all\_agendas:** Calcula a interseção das disponibilidades de vários médicos para um dia específico, identificando os períodos comuns.
* **schedule\_all\_surgeries:** Organiza o agendamento de todas as cirurgias para uma sala de operação num dia determinado.
* **availability\_operation**: Determina os intervalos possíveis para realizar uma cirurgia numa sala específica e com os médicos responsáveis.

* **schedule\_first\_interval**: Seleciona o primeiro período de tempo disponível para agendar uma cirurgia, com base na sua duração.

* **insert\_agenda**: Insere um novo turno na agenda existente.
* **obtain\_better\_sol:** Tenta encontrar a alocação mais eficiente de cirurgias, levando em conta o menor tempo de término e a distribuição ideal dos médicos. Realiza permutações das cirurgias para testar diferentes ordens e substitui a solução atual pela mais eficiente, se encontrada.
* **evaluate\_final\_time**: Determina o momento em que a última cirurgia será concluída, considerando apenas as cirurgias presentes na lista.
* **list\_doctors\_agenda**: Cria uma lista com as agendas atualizadas para todo o staff num dia específico.

### **Factos:**

* **agenda\_staff**: Estabelece os turnos de trabalho de cada membro da equipa para uma data específica. Cada turno é descrito como (HoraInicio, HoraFim, CodigoTarefa).
* **timetable**: Define o período total de disponibilidade de um staff num determinado dia.

* **staff**: Cria o staff com um id, tipo, especialidade e cirurgias associadas.

* **surgery**: Cria uma cirurgia com o tipo, tempo de anestesia, tempo de cirurgia e tempo de limpeza.

* **surgery\_id:** Serve para identificar uma cirurgia.

* **assignment\_surgery:** Regista que um staffestá alocado a uma determinada cirurgia.

* **agenda\_operation\_room:** Cria e define os turnos já ocupados numa sala para um determinado dia.

# Estudo da Complexidade

O objetivo deste estudo é avaliar a capacidade do sistema de agendamento cirúrgico em lidar com cenários crescentemente complexos, medindo a escalabilidade e o desempenho ao introduzir novas cirurgias em um contexto que envolve múltiplos médicos. A análise será iniciada com três cirurgias simples e, progressivamente, novas cirurgias serão adicionadas, permitindo identificar o limite em que o sistema continua a fornecer soluções ótimas ou satisfatórias para o agendamento.

Neste estudo, é dado um foco especial às cirurgias que exigem a colaboração de dois médicos, como s100004, s100005, s100008 e s100009, pois essas operações representam um desafio adicional ao sistema.

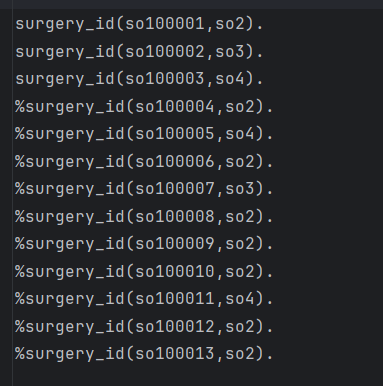
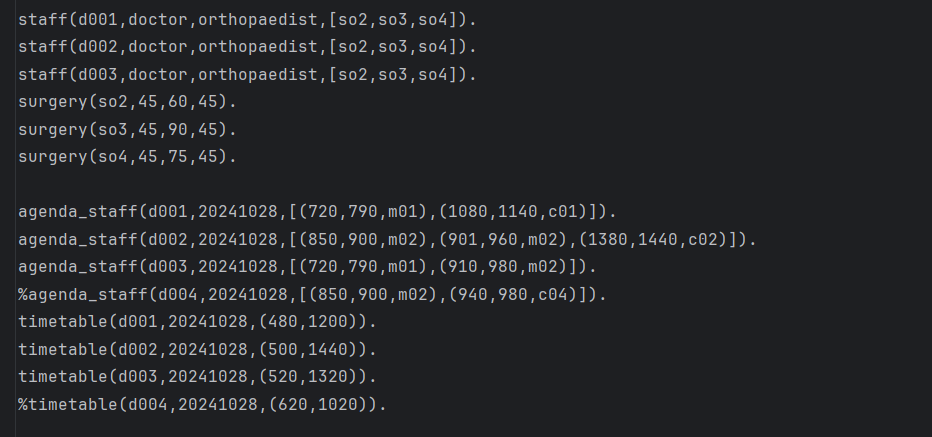
A partir deste processo incremental, busca-se entender até que ponto o sistema é capaz de lidar com a crescente complexidade, identificando potenciais limitações e oportunidades de melhoria no planejamento e na alocação de recursos médicos.

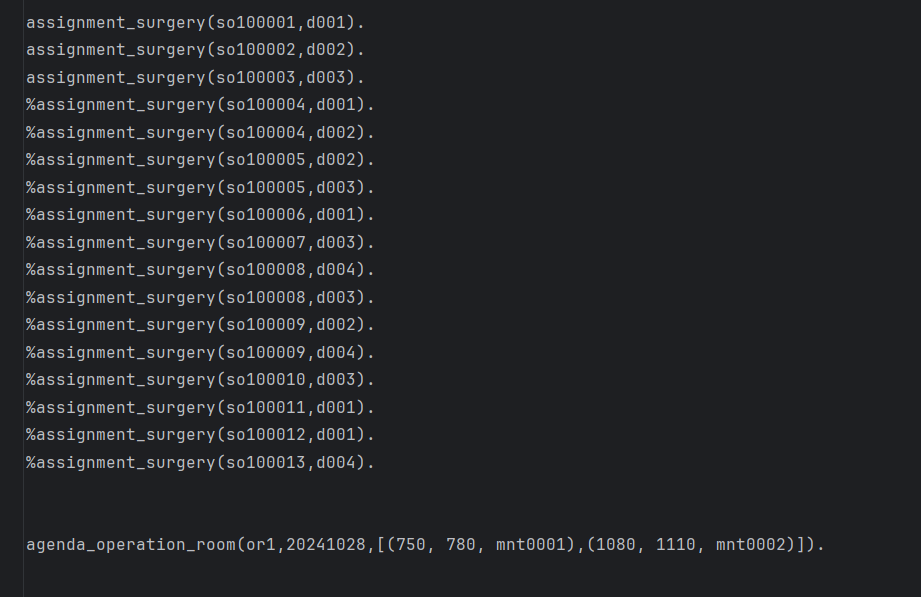
O algoritmo tem complexidade **O(n!)** devido ao uso de **permutation/2**, que gera todas as ordens possíveis das n cirurgias, pois existem **n! combinações**. Cada permutação realiza verificações e atualizações, mas o crescimento fatorial domina o tempo de execução, tornando o algoritmo lento para muitos elementos. Por isso, começaremos este estudo com poucas cirurgias.



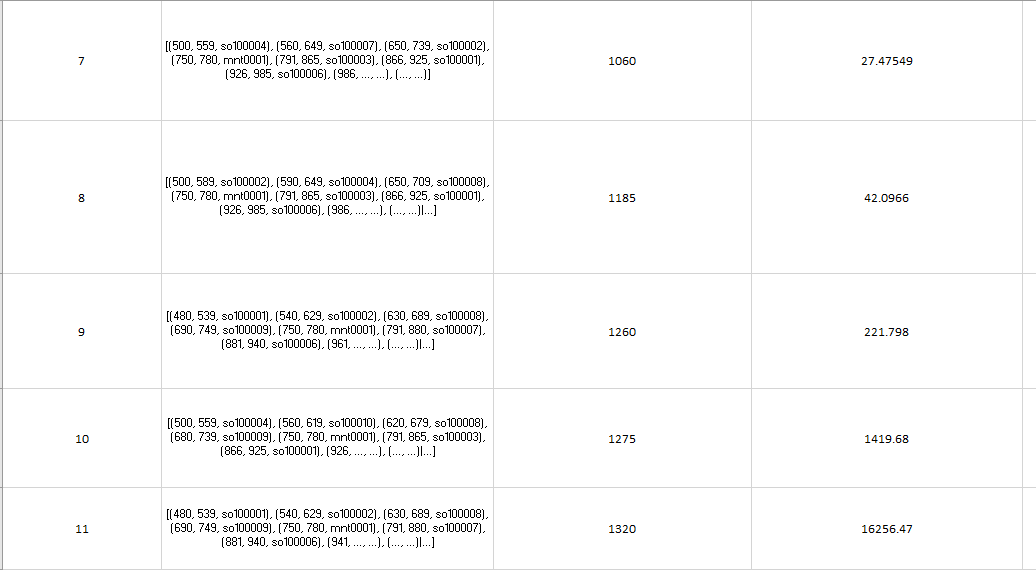
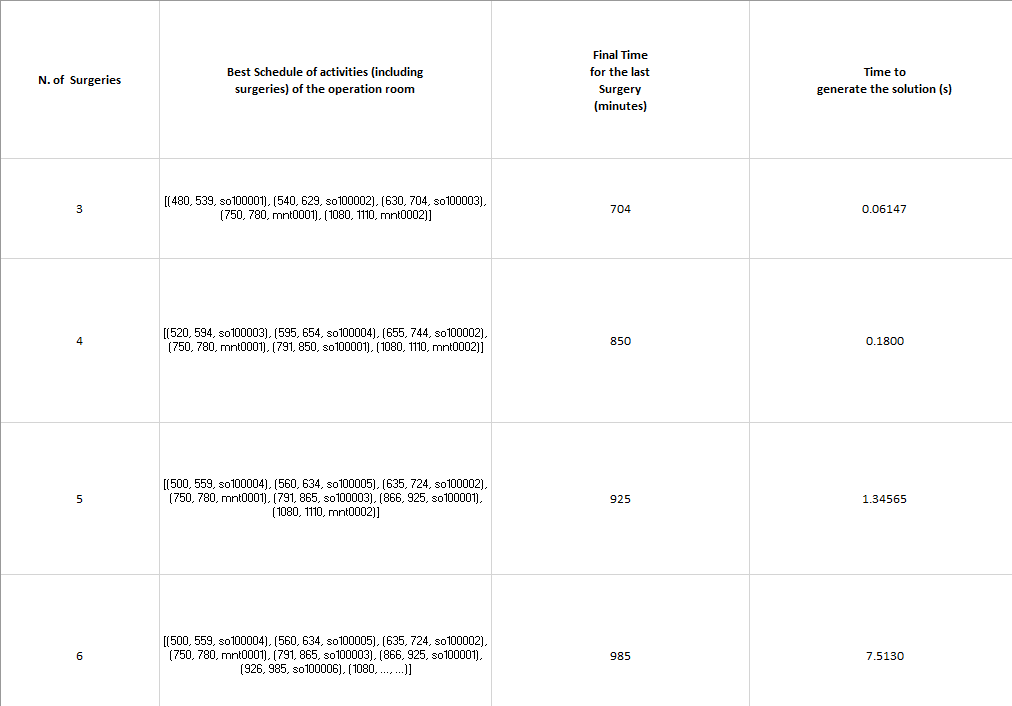


**Dados utilizados na análise da complexidade:**





**Resultados da análise da complexidade:**



Como é possível observar, os resultados obtidos coincidem com o que era espectável, sendo que para ser gerado um resultado com apenas três cirurgias o resultado consegue ser apresentado em apenas 0.06147 segundos em contraste ao resultado com 11 cirurgias que demora um tempo de 16256.47 segundos, que é representado por 4.516 horas.

No caso, de serem adicionadas ainda mais cirurgias, o melhor caso possível iria demorar ainda mais tempo a ser obtido, o que se poderá tornar um problema no caso de precisamos de gerar a solução num tempo determinado, por exemplo, se começarmos hoje às 10 da manhã a gerar uma solução e precisarmos da mesma até ao dia depois de amanhã às 6 da tarde teremos 56 horas para gerar a solução. Isto poderá ser feito para a melhor solução possível, no caso de haver, por exemplo, 11 cirurgias que demorará mais ou menos 4 horas a gerar resultados, mas conforme cresça o número de cirurgias a serem geradas, isto poderá já não ser possível, por exemplo, se tivermos mais de 20 cirurgias a serem geradas poderá não existir solução a tempo da necessidade desses horários, por isso foi-nos pedido também a realização de heurísticas que não irão gerar obrigatoriamente o melhor resultado, mas gerarão resultados mais rápidos, o que permitirá corrigir o problema acima mencionado.

# Heurísticas

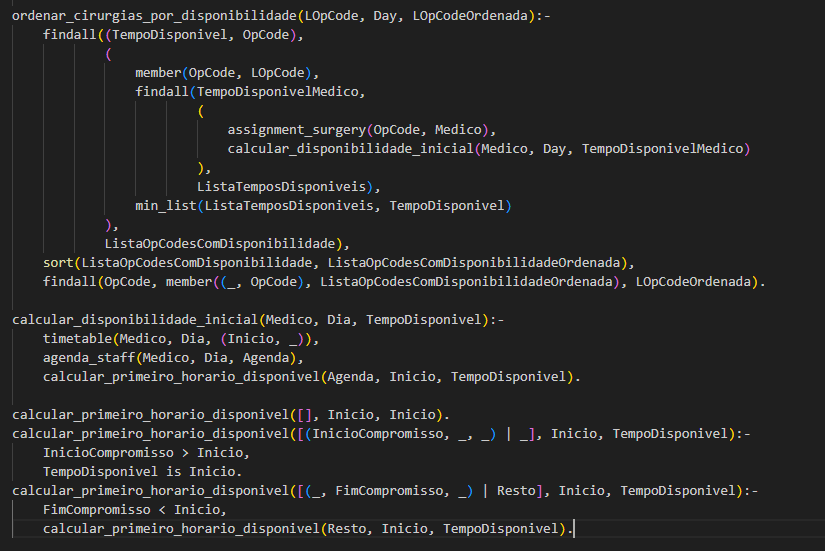
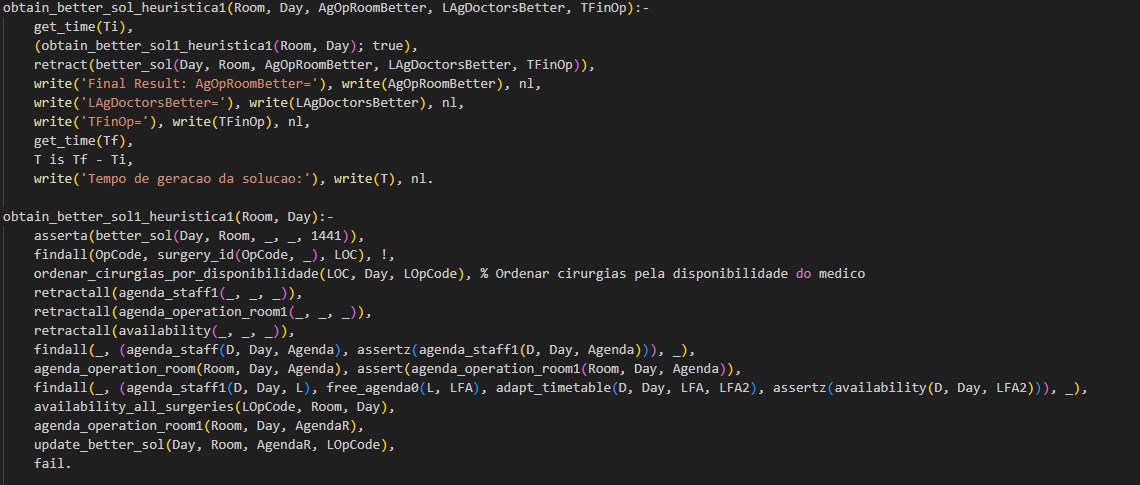
Para contornar os problemas de escalabilidade associados à complexidade fatorial do algoritmo exato de permutação, foram implementadas duas heurísticas distintas. O objetivo desta abordagem é gerar soluções de agendamento mais rapidamente, mesmo que não sejam necessariamente as ótimas. As heurísticas permitem encontrar boas soluções em um intervalo de tempo aceitável, o que é essencial em cenários onde a geração de uma solução exata seria inviável devido a restrições de tempo ou à grande quantidade de cirurgias a serem agendadas.

# Heurística baseada no tempo mais cedo em que os médicos responsáveis estão disponíveis.

### **Funcionamento**

1. **Inicialização e Preparação de Dados**
   1. O programa começa por registrar o tempo de execução, para medir o desempenho da heurística utilizada.
   2. Inicializa a melhor solução provisória com um valor alto para tempo total (1441 minutos), como base para comparação.
2. **Ordenação de Cirurgias**
   1. As cirurgias são ordenadas de acordo com a **disponibilidade inicial dos médicos** atribuídos a cada uma.
   2. Disponibilidade inicial refere-se ao primeiro momento em que o médico está livre para começar a cirurgia, considerando:
      1. O horário de início do seu turno.
      2. A duração de compromissos anteriores no dia.
      3. A duração necessária para realizar a cirurgia.
3. **Alocação de Cirurgias**
   1. Após ordenar as cirurgias, o programa ajusta os horários das agendas e verifica a viabilidade da alocação em relação aos médicos e à sala de operações.
   2. A heurística tenta minimizar o tempo total necessário para realizar todas as cirurgias agendadas.
4. **Atualização da Melhor Solução**
   1. Caso seja encontrada uma solução melhor (ou seja, com menor tempo total), ela substitui a solução provisória.
   2. No final do processo, a melhor solução é exibida.
5. **Mensuração de Tempo**
   1. O tempo total necessário para gerar a solução é calculado e exibido, permitindo avaliar o desempenho do algoritmo.

**Heurística utilizada:**



### **Principais Funções**

#### **Função ordenar\_cirurgias\_por\_disponibilidade**

* Ordena as cirurgias com base no **tempo mais cedo** em que os médicos responsáveis por elas estão disponíveis.
* Verifica todos os compromissos existentes na agenda de cada médico e calcula o primeiro horário viável.

#### **Função calcular\_disponibilidade\_inicial**

* Avalia a disponibilidade inicial de cada médico no dia, considerando o início do turno e os compromissos agendados.
* Garante que apenas horários viáveis para a duração da cirurgia sejam considerados.

#### **Função obtain\_better\_sol\_heuristica1**

* Coordena todo o processo de busca pela melhor solução, integrando a ordenação de cirurgias, a atualização de tabelas de agenda e a comparação de soluções.

### **Exemplo de Aplicação**

Se um médico inicia o turno às 8:00 e tem um compromisso às 8:30, ele **não estará disponível para realizar uma cirurgia de 60 minutos** (tipo so2) às 8:00, pois não há tempo suficiente antes do próximo compromisso. Neste caso, a heurística ajustará a ordem para priorizar outro médico ou outro horário.

**Comparação da complexidade com a melhor solução:**



### **Conclusão**

Este programa oferece uma solução eficiente para o agendamento de cirurgias, minimizando conflitos e otimizando o uso de recursos médicos e das salas de operação. O uso de uma heurística baseada em disponibilidade inicial dos médicos permite uma gestão mais prática e eficaz dos horários.

# Heurística baseada no tempo de ocupação do staff

A heurística implementada segue uma abordagem iterativa que prioriza o agendamento das cirurgias dos médicos com maior percentagem de ocupação relativa ao tempo disponível restante. A cada iteração, calcula-se a percentagem de ocupação de todos os médicos com base no tempo necessário para as cirurgias restantes e no tempo livre total de cada um. A cirurgia atribuída primeiro é aquela que envolve o médico com a maior percentagem de ocupação, garantindo que os recursos mais críticos sejam alocados de forma eficiente. Após cada agendamento, os tempos disponíveis e as percentagens de ocupação são recalculados, considerando as cirurgias já atribuídas, para ajustar as prioridades dinamicamente. Esse processo é repetido até que todas as cirurgias sejam agendadas, garantindo um planejamento que reduz o risco de subutilização ou conflitos no uso dos recursos médicos.

**Heurística utilizada:**

### **1. obtain\_better\_sol\_heuristica2/5**

Este predicado principal gera uma solução de agendamento utilizando a heurística e calcula o tempo total gasto na geração da solução.

* **Parâmetros**:
  + Room: Sala onde as operações ocorrem.
  + Day: Dia para o qual o agendamento é feito.
  + AgOpRoomBetter: Agenda final das operações na sala (melhor solução encontrada).
  + LAgDoctorsBetter: Lista das agendas dos médicos envolvidos (melhor solução encontrada).
  + TFinOp: Tempo final da última cirurgia realizada.
* **Passos**:
  + Regista o tempo inicial (get\_time(Ti)).
  + Chama obtain\_better\_sol2/2, que tenta gerar uma solução para o agendamento.
  + Recupera a melhor solução armazenada no fato dinâmico better\_sol/5.
  + Calcula o tempo da última cirurgia realizada usando calcular\_tempo\_final\_realizado/3.
  + Calcula o tempo total gasto na geração da solução (T = Tf - Ti) e imprime-o.

### **2. calcular\_tempo\_final\_realizado/3**

Calcula o tempo final da última cirurgia realizada, considerando as agendas dos médicos e as operações agendadas.

* **Parâmetros**:
  + AgOpRoom: Agenda das operações na sala.
  + LAgDoctors: Lista das agendas dos médicos.
  + TFinOp: Resultado, o maior tempo final das operações realizadas.
* **Funcionamento**:
  + Coleta todos os tempos finais (Tfin) das operações realizadas.
  + Para cada operação na sala (AgOpRoom), verifica se está na agenda de algum médico (LAgDoctors).
  + Se não houver operações realizadas (ListaFins = []), o tempo final é 0.
  + Caso contrário, retorna o maior tempo final da lista.

### **3. obtain\_better\_sol2/2**

Este predicado busca uma solução de agendamento para as operações e atualiza a melhor solução encontrada.

* **Parâmetros**:
  + Room: Sala onde as operações ocorrem.
  + Day: Dia para o qual o agendamento é feito.
* **Funcionamento**:
  + Inicializa um fato dinâmico better\_sol/5 com uma solução inicial de pior qualidade.
  + Obtém todos os códigos de cirurgias (LOC) usando surgery\_id/2.
  + Gera todas as permutações possíveis de cirurgias (LOpCode).
  + Limpa fatos dinâmicos (agenda\_staff1, agenda\_operation\_room1, availability) e os redefine com os dados atuais.
  + Ajusta a disponibilidade dos médicos.
  + Avalia a solução gerada e atualiza a melhor solução encontrada usando update\_better\_sol2/4.
* **Nota**: O uso de fail no final força o Prolog a buscar todas as permutações possíveis.

### **4. update\_better\_sol2/4**

Atualiza a melhor solução de agendamento, se a solução atual for superior.

* **Parâmetros**:
  + Day: Dia do agendamento.
  + Room: Sala das operações.
  + Agenda: Agenda atual das operações.
  + LOpCode: Lista de códigos das cirurgias.
* **Funcionamento**:
  + Obtém a maior percentagem de ocupação dos médicos para a solução atual (calcular\_percentagem\_maxima/3).
  + Compara a percentagem com a da melhor solução (MelhorPercentagem).
  + Se a solução atual for melhor:
    - Atualiza o fato better\_sol/5.
    - Recalcula as agendas dos médicos (list\_doctors\_agenda/3).
    - Imprime a nova melhor solução.

### **5. calcular\_percentagem\_maxima/3**

Calcula a maior percentagem de ocupação dos médicos para um conjunto de cirurgias.

* **Parâmetros**:
  + Day: Dia do agendamento.
  + LOpCode: Lista de códigos das cirurgias.
  + PercentagemMaxima: Resultado, a maior percentagem de ocupação entre os médicos.
* **Funcionamento**:
  + Para cada cirurgia (OpCode), obtém o médico responsável e calcula sua percentagem de ocupação (calcular\_ocupacao/3).
  + Retorna o maior valor da lista de percentagens.

### **6. calcular\_ocupacao/3**

Calcula a percentagem de ocupação de um médico em relação ao tempo disponível.

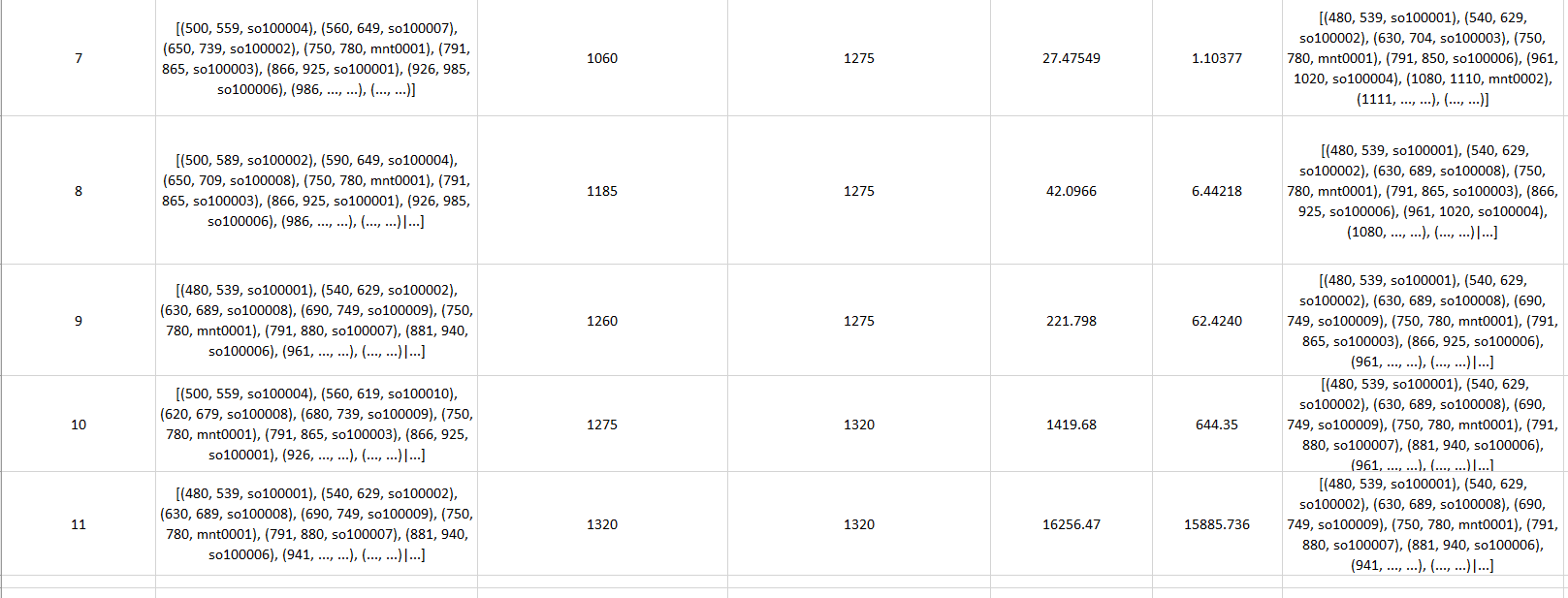
* **Parâmetros**:
  + Medico: Médico analisado.
  + Dia: Dia do agendamento.
  + Percentagem: Percentagem de ocupação do médico.
* **Funcionamento**:
  + Obtém a agenda do médico (agenda\_staff1/3).
  + Calcula o tempo total ocupado pelas cirurgias na agenda (calcular\_tempo\_ocupado/2).
  + Obtém o tempo total disponível do médico (timetable/3).
  + Calcula a percentagem como (TempoTotalOcupado / TempoDisponivel) \* 100.

### **7. calcular\_tempo\_ocupado/2**

Soma o tempo total ocupado pelas cirurgias na agenda de um médico.

* **Parâmetros**:
  + Lista de intervalos na agenda ([(Tin, Tfin, OpCode), ...]).
  + TempoTotalOcupado: Resultado, tempo total ocupado.
* **Funcionamento**:
  + Para cada cirurgia, calcula a duração (Tfin - Tin).
  + Soma todas as durações recursivamente.
  + Retorna o tempo total.

**Comparação da complexidade com a melhor solução:**



Esta heurística apresenta uma abordagem eficiente para o problema de agendamento de cirurgias, especialmente em cenários onde a otimização do uso dos recursos médicos é crítica. Apesar de não garantir a solução mais otimizada, a implementação iterativa e dinâmica permite ajustar as prioridades com base no contexto de ocupação da staff, permitindo um resultado mais rápido.

# Inclusão de Todo o Staff

A implementação de um predicado que conseguisse fazer o agendamento das cirurgias marcadas foi feita a partir da adaptação de predicados já existentes no código base com a adição de novos.

O predicado responsável por toda a ação de tal agendamento é o ***schedule\_all\_surgeries/2***, fornecido no código base. Foi feita uma ligeira alteração na sua estrutura de forma a identificar cada uma das suas porções com um predicado de nome apropriado. Apesar de tudo, a sua funcionalidade permanece **inalterada**:

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

1-Predicado schedule\_all\_surgeries/3

A partir daqui, entremos no predicado ***availability\_all\_surgeries/3***, que possui a mesma assinatura do predicado base, mas será aqui que a **inclusão do staff começará**.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

2-Predicado availability\_all\_surgeries/3

Sucintamente, o predicado irá iterar sobre todas as cirurgias existentes, irá guardar os valores das várias durações do tipo de cirurgia que é e de seguida temos a chamada de dois predicados auxiliares, que serão apresentados mais à frente em maior detalhe: ***availability\_operation/7*** e ***calculate\_intervals/8***. De uma forma simples, estes predicados irão: **definir a disponibilidade dos vários membros do staff, encontrando o primeiro intervalo válido para que a operação seja executada,** e **calcular os instantes iniciais de cada fase da operação** (anestesia, cirurgia, limpeza), respetivamente.

Depois temos uma simples redefinição da disponibilidade da sala escolhida para o agendamento, no facto dinâmico ***agenda\_operation\_room1/3***, seguida da atualização das agendas de todos os membros do staff baseada nas suas especialidades, i.e., anestesia/cirurgia/limpeza. Finalmente, temos uma chamada recursiva ao próprio predicado para que o processo de agendamento seja feito para as restantes cirurgias.

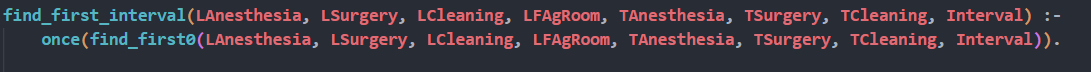
Passemos à compreensão do ***availabiltiy\_operation/7***.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

3-Predicado availability\_operation/7

Começando com uma simples definição das listas de staff requerido para uma determinada operação baseadas nas especialidades de cada um. Essas listas levam também em consideração a necessidade de os horários serem coincidentes entre si, fazendo uso do predicado ***intersect\_all\_agendas/3*** para cada lista de staff. De seguida é definida a disponibilidade da sala passada como argumento e finalmente é feita a chamada do predicado ***find\_first\_interval/8***, predicado auxiliar que chamará o predicado ***find\_first0/8***.



4-Predicado find\_first\_interval/8

O predicado ***find\_first0/8*** vai ser responsável por encontrar um possível intervalo (de acordo com critérios que serão explicados mais à frente quando for abordado) em que a operação pode decorrer. Isto pode resultar em vários possíveis intervalos, por isso o ***find\_first\_interval/8*** chama-o embutido no predicado nativo ***once/1*** que retorna imediatamente o intervalo encontrado na primeira instância em que o ***find\_first0*** for bem-sucedido.

Antes de passar à explicação de ***find\_first0/8***, segue-se uma explicação do que é pretendido com este predicado. Queremos uma forma de encontrar um intervalo em que possamos agendar uma cirurgia, considerando que[1]:

* **Anestesistas** devem participar na operação desde o **início da fase de anestesia** **até** **ao fim da fase de cirurgia**;
* **Cirurgiões** devem participar na operação desde o **fim da fase de anestesia até ao fim da fase de cirurgia**;
* **Técnicos de limpeza** devem participar na operação desde o **fim da fase de cirurgia até ao fim da fase de limpeza**.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

5-Predicado find\_first0/8

Começamos por definir como intervalo de referência cada um dos intervalos de disponibilidade da sala em que a operação poderá possivelmente decorrer (***member((RoomStart, RoomEnd), RoomsAvailable***).

De seguida temos a definição do tempo total da operação, e definimos o instante de tempo **máximo** a partir do qual a operação poderá começar, uma vez que durando um **tempo total** caso escolhêssemos um instante inicial que após a operação definisse um instante final fora do intervalo de disponibilidade da sala o agendamento não seria válido.

Agora, temos a chamada do predicado nativo ***between/3***, que recebe dois argumentos e retorna todos os valores que se encontram entre eles, inclusivé. Por exemplo, ***between(1,5,R).*** iria retornar como valores possíveis de R = {1, 2, 3, 4, 5}.

Ora, chamando este predicado para definir os valores entre o instante de disponibilidade da sala e o instante máximo de início, iremos averiguar se definindo como instante de início da anestesia para cada um desses temos uma solução satisfatória.

Por exemplo, começando a anestesia, ao adicionar o tempo total da operação, temos de garantir que não excede o tempo total de um dia de trabalho. Depois, consoante o instante inicial de anestesia definimos valores de referência para os instantes de início de cirurgia e limpeza.

Depois segue-se um procedimento de verificação das disponibilidades do staff, feito para cada especialidade de uma forma muito semelhante entre si. Consideremos por exemplo o caso dos anestesistas:

Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente

6-Verificação da validade do instante inicial da anestesia

Agora, iremos verificar cada intervalo de disponibilidade de um anestesista. Um intervalo válido de anestesia é aquele que tenha um início de disponibilidade inferior ou igual ao instante inicial da anestesia, e que tenha um fim de disponibilidade superior ou igual ao instante final da cirurgia (***StartAnesthesia + TAnesthesia + TSurgery***).

No caso dos restantes membros do staff, as mesmas condições são verificadas de acordo com os critérios [1] apropriados a cada tipo de especialidade.

No final de todas estas verificações, verificamos se a disponibilidade do quarto de operação é válida consoante o instante de início da operação e retornamos o intervalo da operação de acordo com tais valores.

Finalmente, retornando ao predicado ***availability\_all\_surgeries/3***, o predicado ***calculate\_intervals/8*** serve apenas para esconder, atrás de um nome sugestivo da sua função, a definição dos instantes de início de cada procedimento da operação:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

7-Predicado calculate\_intervals/8

Estes valores são depois utilizados em ***availability\_all\_surgeries/3*** para que a inserção das novas agendas do quarto e do staff seja apropriada.

# Conclusão

### **Análise de Dados e Resultados do Relatório**

O relatório fornece uma análise aprofundada sobre o desempenho do sistema de agendamento de cirurgias, considerando cenários crescentes de complexidade. A análise concentra se nos seguintes pontos:

#### **Estudo da Complexidade:**

O desempenho do algoritmo principal foi analisado ao agendar cirurgias em diferentes cenários. Os resultados mostram que o sistema lida bem com poucos casos, mas enfrenta limitações significativas à medida que o número de cirurgias aumenta:

* **Cenários Testados:**
  + Para 3 cirurgias, o algoritmo executa em 0,06147 segundos.
  + Para 11 cirurgias, o tempo necessário aumenta drasticamente para 16256,47 segundos (~4,52 horas).
* **Crescimento Fatorial:**

A análise destaca que a complexidade do algoritmo, devido ao uso de permutações (O(n!)), torna inviável a sua aplicação direta em casos com mais de 11 cirurgias, especialmente em situações com prazos apertados.

#### **Resultados de Heurísticas:**

Para superar as limitações do algoritmo exato, foram implementadas duas heurísticas, cujos desempenhos e abordagens estão descritos abaixo:

1. **Heurística 1: Disponibilidade Inicial**
   1. **Abordagem:** Prioriza cirurgias com base no primeiro horário disponível dos médicos.
   2. **Resultados:** Reduz o tempo total necessário para encontrar uma solução. É mais indicada em cenários com recursos limitados e onde decisões rápidas são críticas.
   3. **Benefício:** Agiliza o planejamento sem comprometer significativamente a qualidade do agendamento.
2. **Heurística 2: Tempo de Ocupação do Staff**
   1. **Abordagem:** Prioriza médicos com maior ocupação relativa, ajustando dinamicamente as agendas para evitar subutilização de recursos e conflitos.
   2. **Resultados:** Permite uma melhor alocação dos recursos críticos. Embora não garanta a solução ideal, oferece resultados mais rápidos, especialmente em contextos complexos.

**Comparação entre Heurísticas:**

Ambas são eficazes em reduzir o tempo de execução em relação ao algoritmo exato, com a Heurística 2 sendo mais vantajosa para cenários onde a utilização otimizada de todos os médicos é essencial.

#### **Impactos Operacionais:**

Os dados obtidos mostram que:

* **Eficiência:** O sistema, mesmo com heurísticas, consegue lidar com a maior parte dos cenários hospitalares, otimizando o uso de salas e recursos médicos.
* **Limitações:** O algoritmo exato é inadequado para situações com mais de 20 cirurgias devido ao tempo de processamento inviável. Nesse contexto, as heurísticas são indispensáveis.

### **Conclusão dos Resultados:**

Os dados e análises indicam que:

* A transição do algoritmo exato para heurísticas é essencial para lidar com a complexidade crescente do problema.
* As heurísticas oferecem soluções rápidas e práticas para cenários reais, embora possam sacrificar a solução ótima.
* O sistema demonstra eficácia geral no planejamento de cirurgias, mas pode ser aprimorado para suportar uma carga ainda maior e explorar tecnologias avançadas, como aprendizado de máquina, para melhorar a eficiência.