Relatório ALGAV

Sprint 3



Instituto Superior de Engenharia do Porto – Janeiro 2025

Turma 3DI – Grupo 51

Gonçalo Costa – 1220897

Gonçalo Ribeiro- 1220702

Guilherme Pinto – 1221074

José Sá – 1220612

Índice

[Introdução 4](#_Toc187004046)

[Aleatoriedade no cruzamento entre indivíduos da população no AG 4](#_Toc187004047)

[Explicação Detalhada do Código 4](#_Toc187004048)

[Predicados Base: crossover([], []) e crossover([Ind], [Ind]) 4](#_Toc187004049)

[Cruzamento Principal: crossover([Ind1, Ind2 | Rest], [NewInd1, NewInd2 | NewRest]) 5](#_Toc187004050)

[Operador de Mutação (Contexto de Aleatoriedade) 6](#_Toc187004051)

[Seleção da nova geração da população do AG 7](#_Toc187004052)

[Criação da Próxima Geração (Parcialmente Elitista) 7](#_Toc187004053)

[Geração de Descendentes (Crossover + Mutação) 8](#_Toc187004054)

[Combinação e Remoção de Duplos 8](#_Toc187004055)

[Ordenação por Fitness (Descendente) 8](#_Toc187004056)

[Retenção do Top P% (Componente Elitista) 8](#_Toc187004057)

[Seleção Aleatória para Preencher o Resto (Evita Elitismo “Puro”) 9](#_Toc187004058)

[Parametrização da condição de término do AG 10](#_Toc187004059)

[Leitura de Parâmetros e Cálculo da Geração Atual 10](#_Toc187004060)

[Verificação se o Fitness-Alvo é atingido 11](#_Toc187004061)

[Resumo do Fluxo 11](#_Toc187004062)

[Adaptação do Algoritmo Genético 11](#_Toc187004063)

[Inicialização 12](#_Toc187004064)

[Avaliação (Fitness) 13](#_Toc187004065)

[No código de Tarefas: 13](#_Toc187004066)

[No código de Cirurgias: 13](#_Toc187004067)

[Resumo 15](#_Toc187004068)

[Estudo do Estado da Arte de aplicação de Robots e Visão por Computador no contexto hospitalar 19](#_Toc187004069)

[Robots 19](#_Toc187004070)

[Introdução 19](#_Toc187004071)

[Evolução da tecnologia de robots de cirurgia 19](#_Toc187004072)

[Aplicações de robots cirúrgicos 20](#_Toc187004073)

[Benefícios e Limitações da tecnologia 20](#_Toc187004074)

[Tendências futuras 21](#_Toc187004075)

[Uso de robots fora de cirurgias 21](#_Toc187004076)

[Visão por computador 21](#_Toc187004077)

[Introdução 21](#_Toc187004078)

[Aplicações de Visão por Computador em cirurgias 21](#_Toc187004079)

[Desafios e tendências futuras da tecnologia 22](#_Toc187004080)

[Conclusões 23](#_Toc187004081)

[Bibliografia 24](#_Toc187004082)

# Introdução

Este relatório é destinado à UC de ALGAV no âmbito do Sprint 3 do Projeto Integrador.

# Aleatoriedade no cruzamento entre indivíduos da população no AG

Em um **Algoritmo Genético (AG)**, o cruzamento (ou *crossover*) é o operador que **combina** partes de dois indivíduos (pais) para gerar novos indivíduos (filhos). A **aleatoriedade** entra em cena em vários pontos:

1. **Probabilidade de Cruzamento**: Controla se dois indivíduos selecionados para reprodução realmente se recombinam ou são apenas copiados integralmente.
2. **Posições de Corte ou Segmento**: Determinam quais *partes* dos pais vão ser trocadas. A escolha desses pontos é aleatória.

Essa aleatoriedade é importante para **manter diversidade genética** na população e evitar que o algoritmo convirja rapidamente para uma solução local.

## Explicação Detalhada do Código

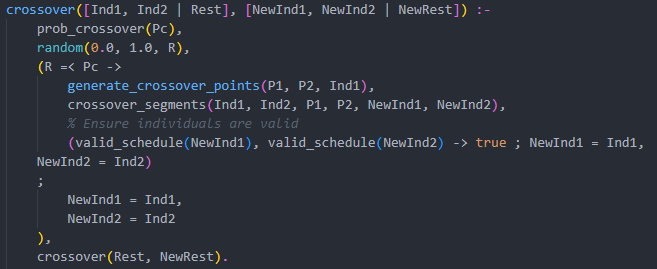
### Predicados Base: crossover([], []) e crossover([Ind], [Ind])



* **Primeira cláusula** (crossover([], [])): Se a lista de indivíduos estiver **vazia**, não há nada para cruzar; o resultado é outra lista vazia.
* **Segunda cláusula** (crossover([Ind], [Ind])): Se existir **apenas um indivíduo**, não é possível cruzá-lo com mais ninguém; então, ele é simplesmente copiado adiante sem alteração.

Esses **casos base** evitam erros quando tentamos cruzar menos de dois indivíduos.

### Cruzamento Principal: crossover([Ind1, Ind2 | Rest], [NewInd1, NewInd2 | NewRest])

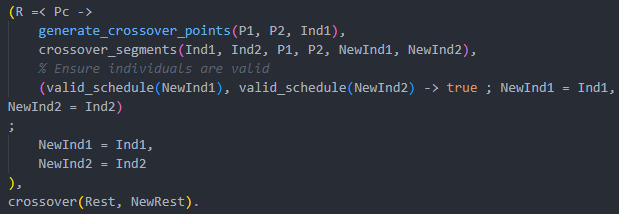


#### Leitura da probabilidade de cruzamento:



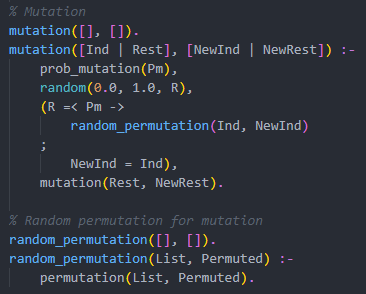
* Pc é a **probabilidade de cruzar** (ex.: 0.8 para 80%).
* random(0.0, 1.0, R) gera um número entre **0 e 1** que decide **aleatoriamente** se o crossover ocorre.

#### Decisão de fazer ou não o crossover:



* Se R for **menor ou igual** a Pc, então **ocorre** o cruzamento.
* Caso contrário, **não ocorre** crossover; os filhos (NewInd1 e NewInd2) são **cópias** exatas dos pais.

### Operador de Mutação (Contexto de Aleatoriedade)



* **random(0.0, 1.0, R)** novamente decide se ocorre mutação baseada em Pm.
* Se sim, random\_permutation(Ind, NewInd) gera uma **permutação aleatória** do indivíduo para alterar sua ordem interna.
* Com isso, **pequenas ou grandes mudanças** podem ocorrer, preservando a

A **introdução de aleatoriedade no processo de cruzamento** é um componente essencial para garantir que o Algoritmo Genético (AG) explore amplamente o espaço de soluções. Ao definir uma probabilidade de cruzamento e pontos de corte selecionados aleatoriamente, o AG consegue:

1. **Combinar traços de maneira imprevisível**: Criar descendentes que mesclam partes dos pais de formas pouco convencionais, ampliando o espectro de soluções possíveis.
2. **Evitar convergência prematura**: Ao não se limitar a uma troca fixa ou determinística de segmentos, o algoritmo diminui as chances de toda a população se fixar rapidamente em poucas regiões do espaço de busca.
3. **Aproveitar características dos dois pais**: Permitir que os filhos herdem não apenas as partes mais “fortes” (já que nem sempre é evidente quais segmentos são efetivamente os melhores), mas também outras combinações de genes que poderiam levar a soluções inovadoras.
4. **Manter a variabilidade**: Mesmo que determinados indivíduos apresentem alto valor de fitness, o uso de probabilidades e cortes aleatórios faz com que surjam soluções novas e inesperadas, mantendo a população rica em diversidade genética.

Dessa forma, o **crossover aleatório** assegura tanto uma exploração contínua quanto a possibilidade de incorporar avanços gerados em diferentes indivíduos, promovendo melhorias sucessivas ao longo das gerações.

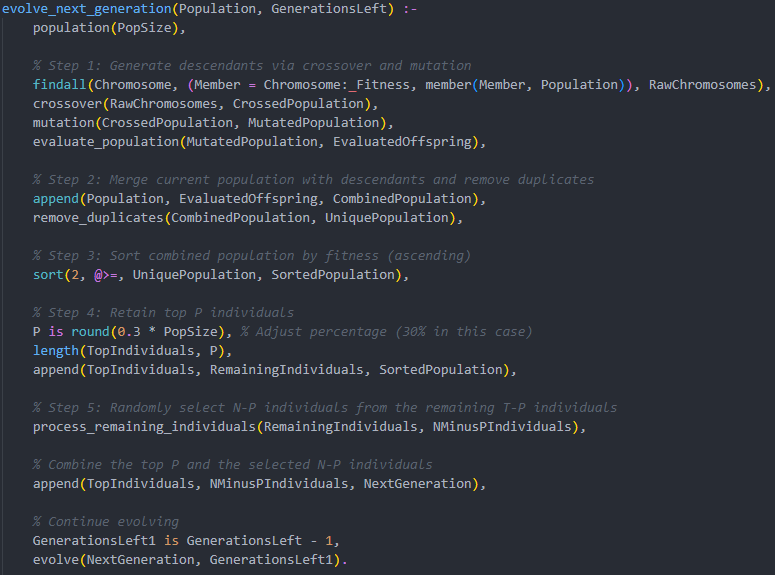
# Seleção da nova geração da população do AG

A **seleção da nova geração** descrita nesse trecho de código procura **equilibrar** duas características fundamentais de um Algoritmo Genético (AG):

1. **Conservar os melhores indivíduos** já descobertos (componente elitista).
2. **Manter diversidade** na população, dando oportunidade a indivíduos não ótimos (evitar um elitismo “puro” e a consequente perda de variedade genética).

A seguir, descrevemos como cada passo do código implementa essa ideia.

## Criação da Próxima Geração (Parcialmente Elitista)

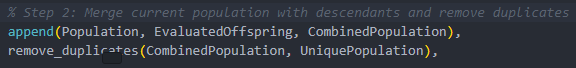


### Geração de Descendentes (Crossover + Mutação)



* **RawChromosomes**: Extrai apenas as partes “genéticas” (ex.: a lista de cirurgias) da população (Indivíduo: Fitness).
* **crossover/2** e **mutation/2**: Geram novas variações (descendentes).
* **evaluate\_population/2**: Avalia cada novo indivíduo, atribuindo-lhe um valor de fitness.

### Combinação e Remoção de Duplos



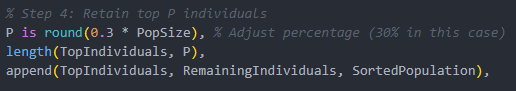
* **append(Population, EvaluatedOffspring, CombinedPopulation)**: Junta a população atual com os novos descendentes.
* **remove\_duplicates/2**: Elimina indivíduos idênticos.

### Ordenação por Fitness (Descendente)



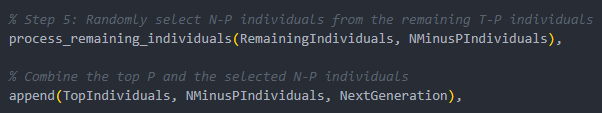
* Ordena a lista resultante (**SortedPopulation**) de **maior para menor** valor de fitness.
* Assim, o melhor indivíduo fica em **primeiro** lugar (SortedPopulation[0]).

### Retenção do Top P% (Componente Elitista)



* Seleciona automaticamente os **top P** indivíduos (30% do tamanho total da população, nesse exemplo).
* **TopIndividuals** é a lista contendo os melhores (por fitness). Isso garante que **o melhor indivíduo** não se perca na próxima geração.

### Seleção Aleatória para Preencher o Resto (Evita Elitismo “Puro”)



* Entre os demais indivíduos (“RemainingIndividuals”), seleciona-se **aleatoriamente** alguns para compor o resto da população.
* **process\_remaining\_individuals/2** embaralha e escolhe o número necessário de indivíduos para chegar a PopSize.
* Ao final, **NextGeneration** = {melhores P} + {selecionados aleatoriamente}.

Esse procedimento de seleção garante:

* **Pelo menos o melhor indivíduo passa adiante** (pois ele estará no top P).
* **Não é um método elitista extremo**, visto que não se descarta totalmente os indivíduos médios ou até os piores, que podem guardar combinações de “genes” úteis.
* O algoritmo retém a capacidade de **melhorar**, geração após geração, sem comprometer a diversidade genética da população.

# Parametrização da condição de término do AG

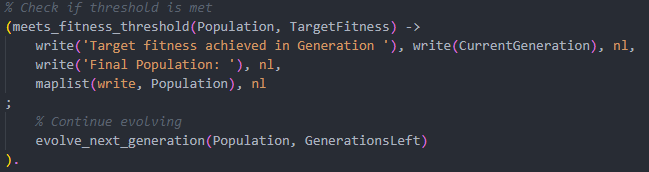


## Leitura de Parâmetros e Cálculo da Geração Atual



* **generations(TotalGenerations)**: Lê o número **máximo** de gerações definido pelo utilizador (ou configurado dinamicamente).
* **CurrentGeneration is TotalGenerations - GenerationsLeft + 1**: Calcula o **índice da geração atual**.
* Exemplo: se TotalGenerations = 10 e GenerationsLeft = 8, então CurrentGeneration = 10 - 8 + 1 = 3.
* Isso serve para imprimir corretamente o número da geração (p. ex., "Generation 3").
* **target\_fitness(TargetFitness)**: Recupera a **fitness-alvo** (condição adicional de término). Caso algum indivíduo atinja ou ultrapasse esse valor, podemos encerrar antes de consumir todas as gerações.

## Verificação se o Fitness-Alvo é atingido



* **meets\_fitness\_threshold(Population, TargetFitness)**: Verifica se **algum** indivíduo da população atinge ou supera TargetFitness.
* Se **sim**, significa que obtivemos um resultado satisfatório.
* O algoritmo imprime uma mensagem de sucesso, mostra a população final e **encerra** (não chama mais passos evolutivos).
* Caso contrário (**não** atinge a meta), o código chama:

evolve\_next\_generation(Population, GenerationsLeft)

Ou seja, manda gerar **descendentes** e continuar o processo.

## Resumo do Fluxo

1. **Identificar** qual geração atual estamos processando.
2. **Evitar impressões duplicadas** da mesma geração.
3. **Verificar** se a população já **atendeu** ao critério de qualidade (TargetFitness).
   1. Se sim, **encerra**.
   2. Se não, **prossegue** com o próximo passo (cruzamento, mutação, seleção…).

Desse modo, o predicado **evolve/2** atua como um **“guarda”** em cada geração:

* Decide se o AG deve **parar** por já ter atingido o objetivo ou por ter alcançado o limite de gerações (essa última verificação normalmente ocorre quando GenerationsLeft se torna zero).
* Se nenhum critério de parada é verdadeiro, chama-se o fluxo de geração seguinte, onde ocorrem crossover, mutação e seleção dos indivíduos para a **próxima** população.

# Adaptação do Algoritmo Genético

O **problema de escalonamento de cirurgias** em salas de operação (blocos operatórios) é, conceitualmente, **similar** ao **problema de escalonamento de tarefas em máquinas** (ou sequenciamento de tarefas). Em ambos os casos, o objetivo é **ordenar** ou **atribuir** uma lista de “jobs” (cirurgias ou tarefas) de forma a **otimizar** um critério (por exemplo, minimizar atrasos ou maximizar eficiência).

Na **versão original** (código de escalonamento de tarefas), cada “tarefa” tem:

* Tempo de processamento (ProcessTime)
* Data de entrega ou prazo (*DueTimeDueTime*DueTime)
* Uma penalidade (PenaltyWeight) associada a atrasos

Já na **versão de escalonamento de cirurgias**, cada cirurgia tem:

* Duração (Duration)
* Médicos e suas janelas de disponibilidade (DoctorStart/End)
* Sala de operação com limite de horário (RoomStart/End)

O **Algoritmo Genético (AG)** em ambas versões contém elementos semelhantes:

1. **Parâmetros Globais** (tamanho da população, número de gerações, probabilidades de crossover e mutação).
2. **Função de avaliação** (*fitness*) adaptada ao objetivo específico.
3. **Seleção de crossover**, **mutação** e **formação da nova geração** (ou seja, “evolve\_next\_generation” ou “generate\_generation”).

A **principal diferença** está em **como** cada indivíduo é **avaliado** (função de fitness) e, em segundo lugar, em **como** se validam as restrições de agendamento (disponibilidade de médicos/salas versus prazos e penalidades).

## Inicialização

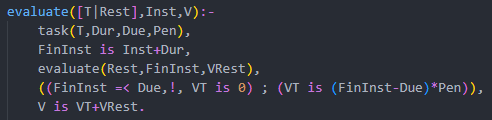
* **Versão Tarefas**:
  + Lê do utilizador: número de gerações, tamanho da população, probabilidade de crossover, probabilidade de mutação.
  + Chama generate\_population/1 para construir a população inicial de sequências de tarefas.
* **Versão Cirurgias**:
  + Lê informações similares (nº de gerações, pop size, Pc, Pm), mas **também** pergunta qual sala será utilizada (initialize\_surgery\_room/1).
  + Em seguida, gera a população inicial de **conjuntos de cirurgias** válidos e parte para a evolução.

Em ambas, a lógica é: (1) pergunte ao usuário, (2) atribua ao banco dinâmico, (3) gere a população.

## Avaliação (Fitness)

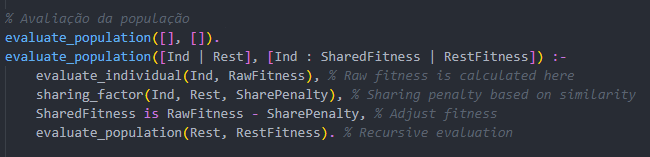
Aqui está a **maior distinção**:

### No código de Tarefas:



* O valor de fitness é baseado em **atraso** (*FinInst−DueFinInst - Due*FinInst−Due) multiplicado pela penalidade da tarefa (PenaltyWeight).
* Quanto **maior** o atraso, maior a penalidade, portanto **pior** a fitness.

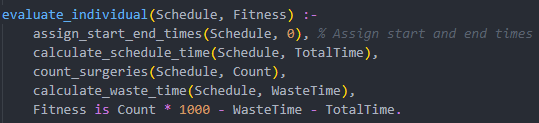
### No código de Cirurgias:



1. Se a lista de indivíduos estiver vazia, a lista resultante de avaliações também é vazia.
2. Para cada indivíduo Ind na população:
   1. **(1)** Calcula o fitness “bruto” através de evaluate\_individual/2.
   2. **(2)** Chama sharing\_factor/3 para estimar uma penalização baseada na **semelhança** com outros indivíduos.
   3. **(3)** A fitness final (SharedFitness) é RawFitness - SharePenalty.
   4. **(4)** Regista o resultado como Ind : SharedFitness e chama a recursão para avaliar o resto dos indivíduos (Rest).

**Resultado**: um conjunto de pares (Ind : FitnessAjustada) representando cada indivíduo e seu valor de fitness final.

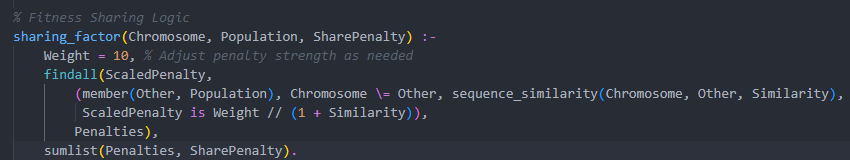
#### Fitness Bruto:



1. **assign\_start\_end\_times(Schedule, 0)**
   1. Percorre a lista de cirurgias (ou procedimentos) em Schedule, atribuindo tempos de início e fim.
   2. Baseia-se na disponibilidade do médico, na sala, e no final da cirurgia anterior.
   3. Ao final, cada cirurgia em Schedule estará com StartTime e EndTime definidos.
2. **calculate\_schedule\_time(Schedule, TotalTime)**
   1. Mede o **tempo total** desde o começo do agendamento até a última cirurgia.
   2. Por exemplo, se a primeira cirurgia começou em t=480 e a última terminou em t=720, TotalTime seria 720 - 480 = 240 minutos (exemplo simplificado).
3. **count\_surgeries(Schedule, Count)**
   1. Conta quantas cirurgias (Count) estão efetivamente no Schedule.
4. **calculate\_waste\_time(Schedule, WasteTime)**
   1. Mede quanto tempo **ocioso** existe entre as cirurgias (intervalos em que nem sala nem médico estão sendo usados).
   2. Também pode considerar o tempo ocioso inicial, se a cirurgia começar depois do início possível da sala/médico.
5. **Fitness is Count \* 1000 - WasteTime - TotalTime**
   1. **Objetivo**: quanto **mais cirurgias** (multiplicadas por 1000), melhor.
   2. Mas **tempo ocioso** (WasteTime) e **tempo total** (TotalTime) reduzem o valor de fitness.
   3. Assim, o algoritmo procura “encaixar” o maior número de cirurgias possível, em **menos tempo total**, **minimizando** intervalos mortos.

O valor resultante (Fitness) é a pontuação **bruta** do indivíduo, sem considerar a diversidade/população.

#### Penalização de Similaridade: sharing\_factor (Ind, Rest, SharePenalty)



* **Compara** Ind com cada outro indivíduo em Rest.
* Mede a **semelhança** (por exemplo, quantas cirurgias iguais na mesma posição, ou quantos genes idênticos).
* Gera uma penalização maior se o indivíduo for muito parecido com vários outros, para **estimular** a diversidade.

É comum em Algoritmos Genéticos usar “fitness sharing” para **evitar convergência prematura** — se muitos indivíduos forem praticamente iguais, eles se penalizam mutuamente, mantendo a população diversa.

O resultado é armazenado em SharePenalty: SharedFitness is RawFitness - SharePenalty

## Resumo

1. **evaluate\_individual/2**: Encontra o valor “bruto” de fitness baseado em:
   1. **Número** de cirurgias realizadas (Count).
   2. **Tempo total** de ocupação (TotalTime).
   3. **Tempo ocioso** (WasteTime).
2. **sharing\_factor/3**: Calcula quanto essa solução se parece com as demais, gerando uma penalização que reduz o fitness para favorecer a **diversidade**.
3. **evaluate\_population/2**: Aplica esses passos para **toda** a população, armazenando cada indivíduo com seu respetivo fitness final.

Assim, cada indivíduo obtém um valor de fitness que **recompensa** encaixar muitas cirurgias em pouco tempo, com pouca inatividade, mas **penaliza** soluções muito semelhantes a outras já existentes. Esse **equilíbrio** ajuda o AG a evoluir buscando **bons agendamentos** e, simultaneamente, **manter variedade** entre as soluções.

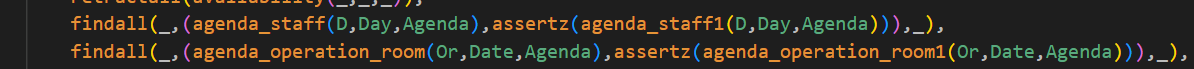
Consideração de vários blocos de operação, com um método de atribuição das operações às salas

Análise

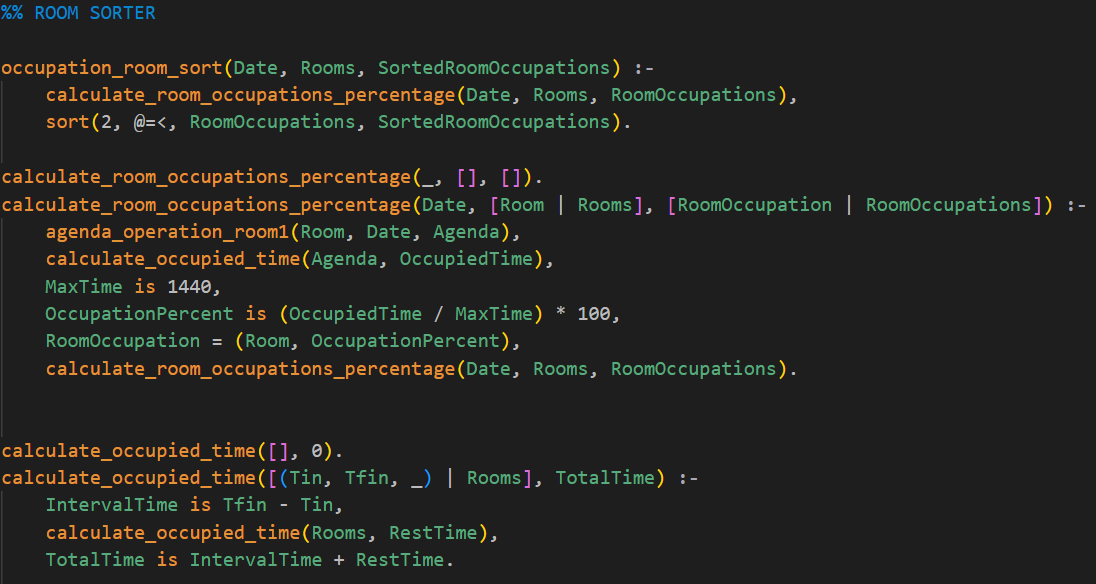
O algoritmo de atribuição das operações às salas implementa um sistema para organizar cirurgias em várias salas de operação, garantindo que os recursos disponíveis sejam alocados de forma eficiente. A lógica principal é gerida pela função ‘schedule\_all\_surgeries/2’, que coordena a alocação de salas, horários e equipas médicas para cada cirurgia, considerando restrições como disponibilidade de recursos e tempos de cirurgia.

Implementação e descrição do seu funcionamento

1. **Inicialização das Agendas**: As agendas das salas e equipas são copiadas para tabelas temporárias, utilizando os predicados ‘agenda\_staff1’ e ‘agenda\_operation\_room1’para trabalhar de forma isolada e evitar alterações no estado original.

Neste código, ele está a buscar todas as agendas existentes (agenda\_staff e agenda\_operation\_room). De seguida, usa o ’assertz/1’ para inserir essas agendas nas tabelas temporárias.

1. **Ordenação de salas:** As salas são organizadas com base nas suas taxas de ocupação, utilizando o predicado ‘occupation\_room\_sort/3’.



Neste exemplo, as percentagens de ocupação são calculadas para cada sala e ordenadas em ‘SortedRoomOccupations’. Isto é feito, primeiramente, através do cálculo do tempo ocupado em cada sala ‘calculate\_occupied\_time/2’. Esse tempo depois é convertido em percentagem de ocupação através da fórmula que se encontra na imagem. No final, através da função sort as salas são ordenadas por ordem crescente de ocupação.

1. **Verificação de Disponibilidade**: As agendas de disponibilidade das salas e equipas são calculadas através de interseções, assegurando que os recursos necessários (médicos, anestesistas, enfermeiros) estejam disponíveis

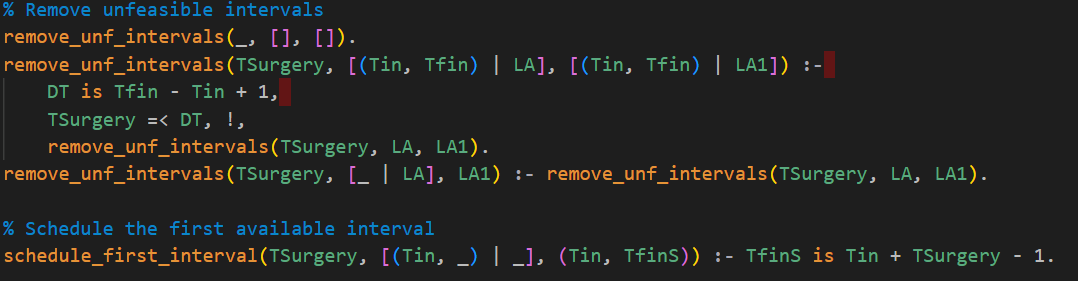


Através deste código, obtemos a interseção entre as agendas de todos os recursos envolvidos. Por exemplo, se uma sala e uma equipa têm os seguintes intervalos:  
Sala : [(7h-10h), (13h-17)]

Equipa: [(9h-12h)]

Isto resultará numa interseção de [(9h-10h)].

1. **Alocação de Cirurgia**: A cirurgia é atribuída ao primeiro intervalo disponível numa sala, com o auxílio de funções como ‘remove\_unf\_intervals/3’ para eliminar intervalos inviáveis e ‘schedule\_first\_interval/3’ para selecionar o intervalo apropriado.

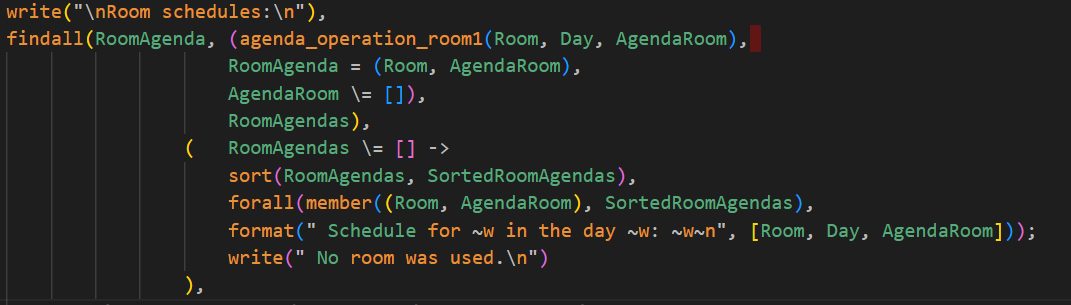


Um exemplo para melhor entender o código é o seguinte:  
- Se tivermos uma sala com os intervalos disponíveis de [(7h-9h), (10h-13h)] se quisermos agendar uma cirurgia de 3 horas teria de ser no segundo intervalo (10h-13h).

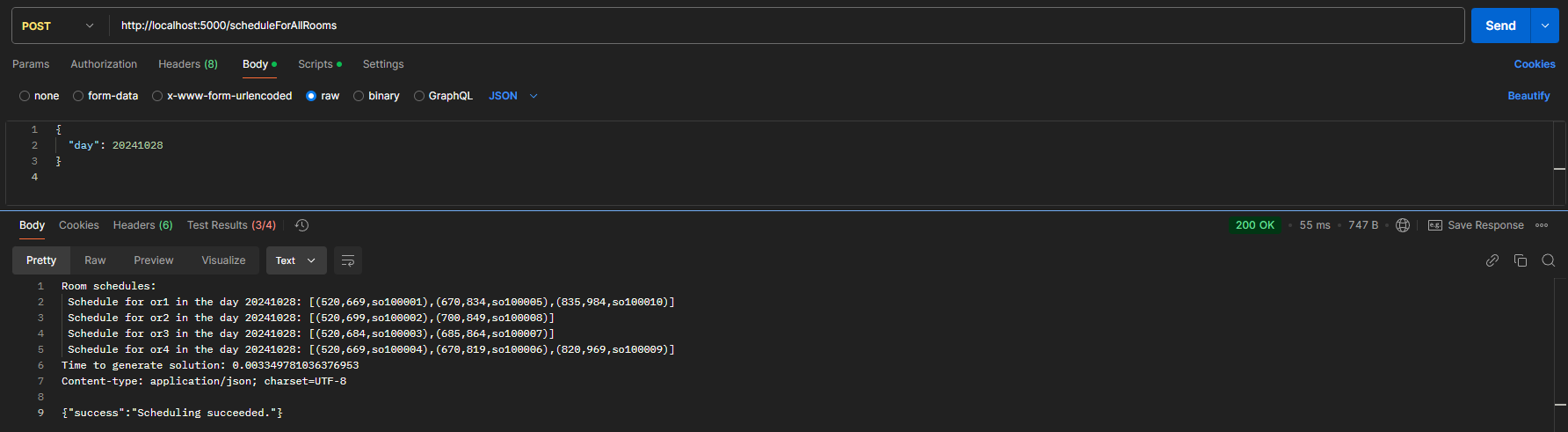
1. **Atualização das Agendas**: Após a alocação, as agendas das salas e equipas são atualizadas para incluir a cirurgia recém-alocada, garantindo a consistência com o novo planeamento. No código, isso está inserido no ‘insert\_agenda\_doctors/3’ onde este adiciona o intervalo na agenda da sala para os doctors enquanto no ‘insert\_agenda\_staff/3’ faz a mesma coisa para as restantes equipas envolvidas



Com este conjunto de passos conseguimos cobrir os principais aspetos do algoritmo de atribuição, demonstrando assim o que era pretendido e o seu funcionamento para maximizar o uso eficiente de recursos disponíveis.

A escrita dos resultados é feita também no ‘schedule\_all\_surgeries/2’ através do seguinte ‘write’:

Um exemplo do programa a correr é o seguinte (para o exemplo, usamos 4 salas e 10 cirurgias para um dia):



# Estudo do Estado da Arte de aplicação de Robots e Visão por Computador no contexto hospitalar

## Robots

### Introdução

A integração de robots nos hospitais trouxe uma revolução à área médica, permitindo realizar, por exemplo, cirurgias minimamente invasivas com maior precisão, menor tempo de recuperação para os pacientes, e a melhoria de resultados em geral. Robots são agora uma ferramenta indispensável em hospitais. Este estudo então explora o atual estado da arte de robots em contexto hospitalar, focando em cirurgias com auxílio de robots.

### Evolução da tecnologia de robots de cirurgia

Esta tecnologia começou a surgir no final do século XX. O primeiro robot de renome usado em cirurgias foi o Puma 560, que em 1985 auxiliou uma biopsia neurocirúrgica. Outros robots foram surgindo nas décadas de 80 e 90, como o AESOP (usado para endoscopia) e o ZEUS, mas em 2000 a FDA (Food and Drug Administration) aprovou o sistema cirúrgico da Vinci, que é agora o robot cirúrgico mais usado no mundo. (Lanfranco et al., 2004; Rivero-Moreno et al., 2023)

Em relação a avanços tecnológicos recentes, pedimos ao ChatGPT para dar alguns exemplos, e foi isto que foi dado (o texto gerado está em itálico, e foi traduzido e em certas secções parafraseado do output original, todas as referências nesta secção foram dadas pelo ChatGPT):

* ***Uso de inteligência artificial:*** *Vários sistemas robóticos agora integram Inteligência Artificial para melhorar a precisão e personalizar os procedimentos. Por exemplo, o robot cirúrgico Smith+Nephew's Cori usa AI para um posicionamento de implantes personalizado e para a medida da laxidez das articulações durante cirurgia de joelho. (Top Advancements in Robotic-Assisted Surgery: 2024 Edition, n.d.)*
* ***Desenvolvimentos telecirurgicos:*** *Avanços em telecomunicações estão a tornar cirurgia remota viável. Por exemplo, o sistema robótico spaceMIRA foi usado para um procedimento simulado na Estação Espacial Internacional (o sistema foi controlado por cirurgiões na Terra). (7 Robotic Assisted Surgery Trends | Sagentia Innovation, n.d.)*
* ***Mais aplicações:*** *A FDA tem vindo a aprovar mais plataformas robóticas para procedimentos mais especializados. Por exemplo os sistemas da Vinci X e Xi da Intuitive Surgical mostram taxas de sobrevivência a longo prazo altas para cirurgias de cancro de próstata.* (*Top Advancements in Robotic-Assisted Surgery: 2024 Edition*, n.d.)

### Aplicações de robots cirúrgicos

Robots cirúrgicos são utilizados em diversas áreas da medicina, como por exemplo:

* **Urologia:** O uso de robots para cirurgia prostatectomia radical laparoscópica é extremamente prevalente, com 88% das cirurgias deste tipo terem sido assistidas por um robot entre 2017 e 2019 no Reino Unido. (Hughes et al., 2023)
* **Cardiologia:** Devido à precisão das cirurgias relacionadas a esta área (plastia valvar mitral, troca valvar mitral, etc.), o uso de robots é um grande apoio para aumentar a taxa de sucesso destas operações. Por exemplo, em um estudo relacionado entre março de 2010 a março de 2013, no Hospital Israelita Albert Einstein, São Paulo, Brazil, foram realizadas cirurgias a 21 pacientes, sem nenhum óbito ter ocorrido, com um periodo medio de acompanhamento de 684±346 dias. (Poffo et al., 2013)

### Benefícios e Limitações da tecnologia

Os benefícios desta tecnologia incluem o facto que é minimamente invasiva, levando a menor perda de sangue, dor e tempo de recuperação; maior precisão, minimizando dano a zonas do corpo fora do que é necessário, e menor fatiga dos cirurgiões para procedimentos mais demorados.

Apesar destes benefícios o uso de robots em cirurgias também vem com alguns desafios, como o custo alto (por exemplo, de acordo com o Annual Report de 2021 da Intuitive Surgical, os custos de máquinas da Vinci rondam entre 0.5 e 2.5 milhões de dólares (Intuitive Surgical Annual Report 2021, 2021)), uma alta curva de aprendizagem para utilizar estas máquinas, e feedback háptico limitado.

### Tendências futuras

No futuro, esta tecnologia pode vir a avançar de diversas maneiras, como no uso de Inteligência Artificial, no desenvolvimento de robots mais autómatos, o desenvolvimento de sistemas mais pequenos e portáteis, e o avanço em tecnologias de telecomunicação que permitem cirurgias remotas.

### Uso de robots fora de cirurgias

Num contexto hospitalar, cirurgias não são o único caso onde se verifica o uso de robots. Para reabilitação, por exemplo, existe exoesqueletos como o EksoGT que ajudam pacientes a reaprender a andar ou a usar os braços (*EksoGTTM Robotic Exoskeleton | Northwestern Medicine*, n.d.). Também existe o robot Xenex, para desinfeção de superfícies com o uso de luz ultravioleta (*Xenex Microbial Reduction*, n.d.), e o robot TUG que auxilia no transporte de itens por um hospital.

## Visão por computador

### Introdução

A visão por computador é uma subsecção de Inteligência Artificial que tem vindo a transformar diversas indústrias, incluindo a medicina. Em hospitais o uso desta tecnologia tem sido critica, especialmente em cirurgias, melhorando a precisão, eficiência, e a segurança dos pacientes. Este estudo explora, então, o estado da arte desta tecnologia no contexto de cirurgias, com enfâse nos avanços, desafios, e áreas a desenvolver no futuro.

### Aplicações de Visão por Computador em cirurgias

* **Sistemas de navegação cirúrgicas e analise de imagens durante operações:** O uso de visão por computador é essencial para sistemas de navegação e para analise de imagens durante operações, que ajudam os cirurgiões durante intervenções em diversas áreas. Por exemplo, a analise de imagens endoscópicas de alta-definição melhora as intervenções em diversas áreas como broncoscopia, rinoscopia e cirurgia laparoscópica. (Yang & Etsuko, 2020)
* **Treino de cirurgiões:** A visão por Computadores pode ser usada para avaliar a performance de cirurgiões em treino, para de seguida lhe serem dados feedback. Por exemplo, através de modelos de visão de computador pode ser analisado as poses da mão de um cirurgião a realizar uma operação, com uma precisão de 88.35%, superando métodos baseados em sensores. Depois da análise, é fornecido ao cirurgião feedback, sendo que o sistema identifica áreas de melhoria na técnica dele. (Bkheet et al., 2023)
* **Cirurgias robô assistidas:** Robots de cirurgia utilizam visão por computador para auxiliar na operação. Algoritmos de CV analisam feeds de vídeo em tempo real para guiar os braços robóticos, ajudando a diminuir tremores e melhorando os resultados de cirurgias minimamente invasivas. (Cho et al., 2024) Avanços mais recentes também melhoraram a segmentação das ferramentas em uso pela máquina do resto da imagem. (Shvets et al., 2018)

### Desafios e tendências futuras da tecnologia

Apesar de esta ser uma tecnologia com grande potencial nesta área, ainda está numa fase muito inicial, por isso verifica-se alguns desafios importantes que é preciso avançar, como:

* Esta tecnologia depende de muitos dados de boa qualidade, algo que pode ser raro na medicina.
* A integração destes sistemas em workflows e ferramentas já existentes pode vir a ser um desafio técnico.
* Os dados que esta tecnologia necessita precisam de ser obtidos de forma ética de forma a cumprir com os direitos de privacidade dos pacientes.

Apesar disto, nesta tecnologia está a ver-se tendências que podem vir a influenciar como ela é desenvolvida no futuro, como o melhoramento de deep learning e inteligência artificial, a junção desta tecnologia com Realidade Aumentada através de dispositivos portáteis para ajudar nos procedimentos cirúrgicos, e a redução de latência nos sistemas para analise mais rápida.

# Conclusões

O desenvolvimento do Projeto Integrador na UC de ALGAV, tendo como foco o Sprint 3, evidenciou como a aplicação de Algoritmos Genéticos (AG) pode ser adaptada a diferentes problemas de escalonamento — desde o agendamento de tarefas até a marcação de cirurgias em múltiplas salas de operação. Ao longo deste relatório, foram detalhados os elementos estruturantes do AG, incluindo:

* Operador de Crossover: determinante para recombinar traços de indivíduos e manter a variabilidade na população;
* Operador de Mutação: promotor de alterações pontuais que potencializam a exploração de novas regiões no espaço de busca;
* Seleção: combinando elitismo (retenção dos melhores indivíduos) e escolhas aleatórias, possibilita equilibrar qualidade e diversidade;
* Condições de Término: monitorizando a geração atual e/ou metas de fitness, o AG evita execuções desnecessárias;
* Adaptações ao Contexto: diferenças na função de avaliação e nas restrições de alocação (salas, disponibilidade de médicos, entre outros) ilustram a versatilidade do algoritmo.

No caso das cirurgias, a lógica de agendamento baseou-se na verificação rigorosa de recursos (salas, equipas médicas) e intervalos de disponibilidade, procurando alocar procedimentos de forma eficiente e contínua. Já a versão voltada para tarefas seguiu um raciocínio semelhante, embora com ênfase em prazos e penalizações por atrasos.

As soluções implementadas demonstram a robustez e adaptabilidade do Algoritmo Genético, desde a construção de populações iniciais até a seleção da próxima geração, culminando em processos de escalonamento mais eficientes. A exploração equilibrada entre manutenção de bons resultados e incentivo à diversidade destaca-se como fator crucial para a obtenção de melhorias sustentáveis a cada iteração, reforçando a relevância desse paradigma para resolver problemas de natureza combinatória e de alocação de recursos.

Sobre o estudo do estado da arte, concluímos que o uso de robots em hospitais é indispensável. Esta tecnologia permite realizar cirurgias minimamente invasivas de uma forma segura, eficaz, e com tempo curto de recuperação do paciente. Robots também são usados fora de cirurgias para diversas tarefas, como limpeza e logística.

Visão por computador, por outro lado, ainda está numa fase muito inicial, com avanços críticos ainda a ser feitos nos dias de hoje. A nossa expectativa é que em breve, com o desenvolvimento de tecnologias de Inteligência Artificial e Deep Learning, esta tecnologia consiga ser bastante útil em cirurgias.

# Bibliografia

*7 Robotic Assisted Surgery trends | Sagentia Innovation*. (n.d.). Retrieved December 30, 2024, from https://www.sagentiainnovation.com/insights/robotic-assisted-surgery-emerging-segments/

Bkheet, E., D’Angelo, A. L., Goldbraikh, A., & Laufer, S. (2023). Using hand pose estimation to automate open surgery training feedback. *International Journal of Computer Assisted Radiology and Surgery*, *18*(7), 1279–1285. https://doi.org/10.1007/S11548-023-02947-6

Cho, S. M., Joo, H. H., Golla, P., Sahu, M., Shankar, A., Trakimas, D. R., Creighton, F., Akst, L., Taylor, R. H., & Galaiya, D. (2024). Tremor Assessment in Robot-Assisted Microlaryngeal Surgery Using Computer Vision-Based Tool Tracking. *Otolaryngology–Head and Neck Surgery*, *171*(1), 188–196. https://doi.org/10.1002/OHN.714

*EksoGTTM Robotic Exoskeleton | Northwestern Medicine*. (n.d.). Retrieved December 30, 2024, from https://www.nm.org/conditions-and-care-areas/treatments/eksoGT-robotic-exoskeleton

Hughes, T., Rai, B., Madaan, S., Chedgy, E., & Somani, B. (2023). The Availability, Cost, Limitations, Learning Curve and Future of Robotic Systems in Urology and Prostate Cancer Surgery. *JOURNAL OF CLINICAL MEDICINE*, *12*(6), 2268. https://doi.org/10.3390/JCM12062268

*Intuitive Surgical Annual Report 2021*. (2021). https://isrg.intuitive.com/static-files/704322bf-cb0d-4ed1-954c-8eb46a070f70

Lanfranco, A. R., Castellanos, A. E., Desai, J. P., & Meyers, W. C. (2004). Robotic Surgery: A Current Perspective. *Annals of Surgery*, *239*(1), 14. https://doi.org/10.1097/01.SLA.0000103020.19595.7D

Poffo, R., Parrilha Toschi, A., Bastos Pope, R., Celullare, A. L., Benício, A., Henrique Fischer, C., Campos Vieira, M. L., Teruya, A., Hatanaka, D. M., Rusca, G. F., & Makdisse, M. (2013). Robotic surgery in Cardiology: a safe and effective procedure. *Einstein (16794508)*, *11*(3), 296–302. https://openurl.ebsco.com/contentitem/a9h:90605246?sid=ebsco:plink:crawler&id=ebsco:a9h:90605246&crl=c

Rivero-Moreno, Y., Echevarria, S., Vidal-Valderrama, C., Stefano-Pianetti, L., Cordova-Guilarte, J., Navarro-Gonzalez, J., Acevedo-Rodríguez, J., Dorado-Avila, G., Osorio-Romero, L., Chavez-Campos, C., & Acero-Alvarracín, K. (2023). Robotic Surgery: A Comprehensive Review of the Literature and Current Trends. *Cureus*, *15*(7), e42370. https://doi.org/10.7759/CUREUS.42370

Shvets, A. A., Rakhlin, A., Kalinin, A. A., & Iglovikov, V. I. (2018). Automatic Instrument Segmentation in Robot-Assisted Surgery using Deep Learning. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, 624–628. https://doi.org/10.1109/ICMLA.2018.00100

*Top Advancements in Robotic-Assisted Surgery: 2024 Edition*. (n.d.). Retrieved December 30, 2024, from https://idataresearch.com/advancements-in-robotic-assisted-surgery/

*Xenex Microbial Reduction*. (n.d.). Retrieved December 30, 2024, from https://xenex.com/

Yang, L., & Etsuko, K. (2020). Review on vision-based tracking in surgical navigation. In *IET Cyber-systems and Robotics* (Vol. 2, Issue 3, pp. 107–121). Institution of Engineering and Technology. https://doi.org/10.1049/iet-csr.2020.0013