ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

P.PORTO

Relatório Algav Sprint 3

3NA Grupo 74:

- José Pedro (1211656)
- Pedro Águia (1161096)
- Rui Beloto (1121200)
- Sérgio Cardoso (1210891)

Índice

1. Introdução

Este relatório documenta o progresso alcançado pelo grupo 74 da turma 3NA no âmbito do Sprint 3 da unidade curricular de ALGAV. Ao longo deste sprint, foram abordados temas fundamentais para a adaptação e aplicação de Algoritmos Genéticos (AG) a problemas complexos, com ênfase na resolução do problema de otimização da marcação de cirurgias em blocos operatórios de hospitais.

Utilizou-se um algoritmo genético para o desenvolvimento do projeto que tinha várias particularidades, sendo exploradas em detalhe no seguinte relatório.

Por exemplo, a implementação de cruzamentos aleatórios entre indivíduos da população no AG, permite simular de forma mais realista os processos naturais de recombinação genética.

O desenvolvimento de um método de seleção para a nova geração, assegurando a preservação do melhor indivíduo da população atual sem recorrer a uma abordagem puramente elitista.

A definição de condições de término do AG para além do número fixo de gerações, permitindo maior flexibilidade e adaptabilidade.

A adaptação do AG ao problema específico do escalonamento de cirurgias, incluindo a consideração de múltiplos blocos de operação e o desenvolvimento de um método eficiente para a sua atribuição às salas de operação.

O estudo do estado da arte no uso de robots e visão por computador no contexto hospitalar, nomeadamente em cirurgias, combinando pesquisa bibliográfica com ferramentas de inteligência artificial generativa como o ChatGPT.

O relatório está estruturado de forma a apresentar, em capítulos sucessivos, a descrição das implementações realizadas, acompanhada por exemplos de código relevantes, e os resultados obtidos durante este sprint. A secção final inclui as conclusões do trabalho desenvolvido, bem como as reflexões acerca de possíveis melhorias e direções futuras.

Este documento pretende, assim, não só detalhar as soluções propostas, mas também demonstrar a forma como os desafios técnicos e teóricos foram abordados pelo grupo.

2. Aleatoriedade no cruzamento entre indivíduos da população no AG

No âmbito da implementação de um Algoritmo Genético (AG), um dos principais objetivos é evitar padrões fixos que possam limitar a diversidade da população ao longo das gerações. O código fornecido aborda precisamente um desses problemas, garantindo que o cruzamento entre indivíduos da população não seja sempre realizado entre pares predefinidos.

No caso específico do AG apresentado, o cruzamento original ocorria de forma determinística, ou seja, o 1.º cromossoma cruzava-se com o 2.º, o 3.º com o 4.º, e assim sucessivamente. Este comportamento podia levar à convergência precoce da população, reduzindo a sua capacidade de explorar o espaço de soluções de forma eficiente.

Para mitigar este problema, foi introduzida uma permutação aleatória da população antes do cruzamento. O predicado random_permutation/2 é responsável por reordenar os indivíduos de forma aleatória, garantindo que os pares formados para o cruzamento sejam diferentes a cada geração. A sequência relevante é a seguinte:

```
random_permutation(Pop, RPop), % Permutação aleatória da população
crossover(RPop, NPop1).        % Geração dos cruzamentos
```

O predicado crossover/2 é utilizado para gerar os cruzamentos entre os indivíduos. A lógica é baseada na aplicação de pontos de cruzamento aleatórios, definidos previamente pelo predicado generate_crossover_points/2. O funcionamento deste predicado é explicado a seguir.

Casos Base, se a lista de indivíduos estiver vazia, o resultado do cruzamento também será uma lista vazia:

```
crossover([], []).
```

Se existir apenas um indivíduo na lista, ele é simplesmente mantido:

```
crossover([Ind* _], [Ind]).
```

Para dois indivíduos consecutivos na população (Ind1 e Ind2), geram-se dois pontos de cruzamento aleatórios, P1 e P2, através do predicado generate_crossover_points/2.
Define-se a probabilidade de cruzamento, Pcruz, e gera-se um valor aleatório Pc.
Se o valor Pc for inferior ou igual a Pcruz, realiza-se o cruzamento utilizando o predicado cross/5.
Caso contrário, os indivíduos são mantidos inalterados.

O código implementa esta lógica da seguinte forma:

```
((Pc =< Pcruz, !,
cross(Ind1, Ind2, P1, P2, NInd1),
cross(Ind2, Ind1, P1, P2, NInd2))
;
(NInd1 = Ind1, NInd2 = Ind2)),
```

Após processar o par atual, a função é chamada recursivamente para os restantes elementos da lista.

O predicado generate_crossover_points/2 é responsável por gerar dois pontos de cruzamento aleatórios distintos. Estes pontos determinam as posições em que o material genético será trocado entre os cromossomas.

O número total de tarefas (N) define os limites dos pontos de cruzamento. Os valores possíveis para P1 e P2 variam entre 1 e N+1.

Os pontos gerados são garantidamente diferentes, conforme assegurado pelo predicado:

```
P11 \== P21, !,
```

Os pontos são ordenados para que P1 seja sempre menor do que P2.

A lógica recursiva do predicado permite que novos pontos sejam gerados caso os inicialmente

```
330 generate_generation(Room,Day,StartTime,N,G,Pop,StableCount):-
331     population(PopSize),
332     order_population(Pop,PopOrdShow),
            ...
357     random_permutation(Pop,RPop),
358     crossover(RPop,NPop1),
            ...
380     generate_generation(Room,Day,StartTime,N1,G,Result,NewStableCount).
            ...
469 %Gera os crossovers
470 crossover([ ],[ ]).
471 crossover([Ind*_],[Ind]).
472 crossover([Ind1*_,Ind2*_|Rest],[NInd1,NInd2|Rest1]):-
473 generate_crossover_points(P1,P2),
474 prob_crossover(Pcruz),random(0.0,1.0,Pc),
475 ((Pc =< Pcruz,!,
476         cross(Ind1,Ind2,P1,P2,NInd1),
477  cross(Ind2,Ind1,P1,P2,NInd2))
478 ;
479 (NInd1=Ind1,NInd2=Ind2)),
480 crossover(Rest,Rest1).
481
482
483 generate_crossover_points(P1,P2):- generate_crossover_points1(P1,P2).
484 generate_crossover_points1(P1,P2):-
485 tasks(N),
486 NTemp is N+1,
487 random(1,NTemp,P11),
488 random(1,NTemp,P21),
489 P11\==P21,!,
490 ((P11<P21,!,P1=P11,P2=P21);P1=P21,P2=P11).
491 generate_crossover_points1(P1,P2):-
492 generate_crossover_points1(P1,P2).
```

calculados sejam inválidos ou repetidos. Na figura é possível ver o excerto de codigo completo referente a este ponto.

Para além de crossover, também pode acontecer mutation/2. Este predicado aplica o processo de mutação a uma população de indivíduos. Ele percorre recursivamente a lista de indivíduos e decide, com base numa probabilidade, se cada indivíduo será mutado ou não. Quando a lista de indivíduos está vazia, a lista resultante também estará vazia:

mutation([],[]).

Para cada indivíduo Ind da população:

A probabilidade de mutação (Pmut) é avaliada.
Um número aleatório entre 0.0 e 1.0 (Pm) é gerado.
Se Pm for menor que Pmut, a mutação ocorre; caso contrário, o indivíduo permanece inalterado:

((Pm < Pmut,!, mutacao1(Ind, NInd)) ; NInd = Ind),

A mutação é aplicada ao resto da população de forma recursiva:

mutation(Rest, Rest1).

Se a mutação for aplicada a um indivíduo, o predicado mutacao1/2 é chamado para gerar um novo indivíduo (NInd). Este predicado seleciona dois pontos (P1 e P2) onde ocorrerá a troca de genes no indivíduo, utilizando o predicado generate_crossover_points/2:

```
generate_crossover_points(P1, P2),
```

A mutação entre os pontos P1 e P2 é realizada pelo predicado mutacao22/4:

```
mutacao22(Ind, P1, P2, NInd).
```

Este predicado implementa a lógica para identificar os genes entre os pontos de mutação P1 e P2 e trocá-los. Quando o ponto inicial de mutação (P1) é alcançado, chama-se mutacao23/5 para realizar a troca entre os genes P1 e P2:

```
mutacao22([G1|Ind], 1, P2, [G2|NInd]) :-
    !, P21 is P2-1,
    mutacao23(G1, P21, Ind, G2, NInd).
```

Para outros casos, reduz-se os pontos P1 e P2 até alcançar os índices desejados:

```
mutacao22([G|Ind], P1, P2, [G|NInd]) :-
    P11 is P1-1, P21 is P2-1,
    mutacao22(Ind, P11, P21, NInd).
```

Este predicado realiza a troca de dois genes (G1 e G2) quando o ponto final de mutação (P2) é atingido. Quando P2 é igual a 1, o gene no índice P2 (G2) é trocado com o gene inicial (G1):

```
mutacao23(G1, 1, [G2|Ind], G2, [G1|Ind]) :- !.
```

Caso contrário, reduz-se o índice P2 até alcançar o ponto final:

```
mutacao23(G1, P, [G|Ind], G2, [G|NInd]) :-
    P1 is P-1,
    mutacao23(G1, P1, Ind, G2, NInd).
```

3. Seleção da nova geração da população do AG de forma não elitista.

A seleção da nova geração é muito importante, sendo responsável por determinar quais os indivíduos da população atual e quais dos seus descendentes irão formar a população da próxima geração. Este trecho de código implemento com uma seleção não elitista, garante que o melhor indivíduo entre a população atual e os descendentes é preservado, mas evita a estagnação causada por uma abordagem puramente elitista.

Avalia-se a pontuação dos indivíduos da nova população (NPop) utilizando o predicado evaluate_population/4. Os indivíduos são ordenados pela sua pontuação após avaliação, em ordem decrescente, através do predicado order_population/2.

O predicado non_elitist_selection/4 é responsável por combinar as populações, filtrar duplicados e selecionar os melhores indivíduos (top 20%) da nova geração.

O melhor indivíduo global (Best) é inserido na população final, substituindo o pior indivíduo caso ainda não esteja presente. Este processo é realizado pelo predicado replace_worst/3.

A seleção não elitista é implementada no predicado non_elitist_selection/4. Esta abordagem combina a população atual (CurrentPop) com os novos descendentes (NewPop) e remove duplicados, promovendo diversidade.

O predicado merge_populations/3 une as populações atuais e descendente. De seguida, remove_duplicates/2 filtra os indivíduos repetidos:

```
merge_populations(CurrentPop, NewPop, CombinedPop),
remove_duplicates(CombinedPop, FilteredPop).
```

Depois de ordenar os indivíduos pela sua qualidade (order_population/2), seleccionam-se os 20% melhores utilizando o predicado select_top_20_percent/3. Os restantes indivíduos são armazenados em Remaining para aplicação de variabilidade aleatória:

```
select_top_20_percent(OrderedFPop, TopP, Remaining).
```

A variabilidade na seleção é introduzida através do predicado apply_random_factor/2, que multiplica os valores de pontuação dos indivíduos restantes por um fator aleatório entre 0 e 1:

```
apply_random_factor(Remaining, Result).
```

A nova população é ordenada pela pontuação ajustada, e o melhor indivíduo (BestZ) é identificado e extraído para ser preservado:

```
order_population3(Result, Sorted),
find_best_individual(Sorted, BestZ),
remove_Z(BestZ, Best).
```

O predicado replace_worst/3 assegura que o melhor indivíduo global (BestInd) seja incluído na nova geração. Caso o melhor indivíduo já esteja presente, a população não é alterada. Caso contrário, o pior indivíduo da população é substituído:

```
replace_worst(Pop, BestInd, Result) :-
(   member(BestInd, Pop)
    ->  Result = Pop
;   append(Front, [_WorstInd], Pop),
    append(Front, [BestInd], Result)
).
```

A população final é ajustada ao tamanho máximo permitido (PopSize) no predicado replace_worst_in_pop/4. Indivíduos duplicados ou redundantes são excluídos, garantindo que apenas os melhores sejam mantidos:

```
exclude(already_in_pop(Pop), FilteredPop, UniqueFilteredPop),
append(UniqueFilteredPop, Pop, CombinedPop),
sort(CombinedPop, SortedPop),
length(NewPop, PopSize),
append(NewPop, _, SortedPop).
```

4. Parametrização das condições de término

Para além do tradicional número de gerações, outras condições de término são introduzidas, como limite de tempo, pontuação da solução ou estabilidade da população. A seguir, detalha-se cada abordagem e a sua implementação.

4.1. Número Máximo de Gerações (NG)

Esta é a condição padrão, o AG é executado por um número predefinido de gerações (NG). Após atingir este número, o algoritmo termina e apresenta a melhor solução encontrada.

4.2. Limite de Tempo

A execução pode ser limitada por um período de tempo (em segundos). Esta abordagem é útil em cenários onde o tempo de execução é crítico ou quando o algoritmo tem um número elevado de gerações ou quando acontece um erro e entra em loop por algum motivo.
A condição de tempo é verificada em cada geração através da diferença entre o tempo atual e o tempo de início (TotalTime):

```
get_time(CurrentTime),
TotalTime is CurrentTime - StartTime,
(   TotalTime > 1000
->  write('Time limit reached! Terminating...'), nl,
PopOrdShow = [BestSolution|_],
throw(best_solution_reached(BestSolution))
;   true
).
```

Se o limite de tempo for ultrapassado, o algoritmo termina imediatamente e retorna a melhor solução encontrada até então.

4.3. Solução com Avaliação Específica

Outra condição de término ocorre quando o algoritmo encontra uma solução com uma avaliação igual ou inferior a um valor especificado.
Por exemplo, o algoritmo pode ser configurado para terminar ao atingir uma solução com uma pontuação inferior a 940:

```
PopOrdShow = [_*FirstIndValue | _],
FirstIndValue < 940
->  write('Specific evaluation reached! Terminating...'), nl,
PopOrdShow = [BestSolution|_],
throw(best_solution_reached(BestSolution))
;   true
).
```

Esta abordagem é útil para problemas onde se conhece um limite aceitável para a solução, mas é acompanhada de uma salvaguarda contra loops infinitos caso o valor não seja atingido que é a paragem por tempo.

4.4. Estabilidade da População

A estabilidade da população é definida como o caso em que a população permanece inalterada por G gerações consecutivas.
Para verificar a estabilidade, é necessário comparar a população atual (Pop) com a geração anterior (Result):

```
(   Result = Pop
->  NewStableCount is StableCount + 1
;   NewStableCount is 0
).
```

Se a estabilidade for mantida por 10 gerações consecutivas, o algoritmo termina:

```
(   NewStableCount >= 10
->  write('Population stable! Terminating...'), nl,
PopOrdShow = [BestSolution|_],
throw(best_solution_reached(BestSolution))
;   true
).
```

Este critério é frequentemente utilizado em algoritmos genéticos elitistas, onde a seleção favorece sempre os melhores indivíduos, mas também pode ser implementado com métodos não elitistas.

4.5. Combinação de Critérios

No algoritmo proposto, várias condições podem ser verificadas em simultâneo. Por exemplo:

- Se o tempo for ultrapassado.
- Se a solução com a avaliação desejada for atingida.
- Se a população estabilizar.

Esta flexibilidade permite adaptar o algoritmo a diferentes problemas e contextos.

O predicado principal do AG combina estas condições para determinar se o algoritmo deve continuar ou terminar. A estrutura de controlo assegura que a melhor solução encontrada é capturada e exibida:

```
catch(
generate_generation(Room, Day, StartTime, 0, NG, PopOrd, _),
best_solution_reached(BestSolution),
(
write('Best solution reached: '), write(BestSolution), nl,
retract(solution_found(false)),
assertz(solution_found(true))
)
).
```

Se nenhuma das condições de término for satisfeita, o AG continua até completar todas as gerações configuradas:

```
solution_found(SolutionFound),
(
SolutionFound = true
->
true
;
write('Completed all generations, final solution: '), nl,
order_population(PopOrd, PopOrdShow),
PopOrdShow = [BestSolution|_],
write(BestSolution), nl
).
```

5. Adaptação do Algoritmo Genético para o problema do Escalonamento de Cirurgias a Blocos de Operação de Hospitais

Para adaptar o algoritmo base fornecido foram precisas algumas alterações, nomeadamente, inserir os factos das cirurgias e calendário do staff. Utilizei o algoritmo desenvolvido no sprint 2 para trazer toda a lógica de marcação de cirurgias numa sala, num dia especifico. Esse algoritmo vai ser chamado durante a execução do algoritmo genético em duas situações, na avaliação das populações e no momento final de marcação das cirurgias.

```
209 generate(Room,Day,BestSolution):-
210     initialize,
211     %Gera populaçao inicial de individuos
212     findall(OpCode,surgery_id(OpCode,_,Room),LOpCode),
213     length(LOpCode, NumT),
214     assertz(tasks(NumT)),
215     generate_population(Room,Pop),
216     evaluate_population(Room,Day,Pop,PopValue),
217     order_population(PopValue,PopOrd),
218     generations(NG),
```

Pela figura é possivel observar que agora o número de tasks vai ser definido pelo número de cirurgias a marcar.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
generate_population(Room,Pop):-
    %Tamanho da populaçao
    population(PopSize),
    findall(OpCode,surgery_id(OpCode,_,Room),LOpCode),
    length(LOpCode, NumT),
    generate_population(PopSize,LOpCode,NumT,Pop).

generate_population(0,_,_,[]):-!.
generate_population(PopSize,LOpCode,NumT,[Ind|Rest]):-
    PopSize1 is PopSize-1,
    generate_population(PopSize1,LOpCode,NumT,Rest),
    %Gera individuo
    generate_individual(LOpCode,NumT,Ind),
    %Verifica se não é duplicado
    not(member(Ind,Rest)).
generate_population(PopSize,LOpCode,NumT,L):-
    generate_population(PopSize,LOpCode,NumT,L).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Para o generate population mais uma vez utilizo o total de cirurgias a marcar para enviar a lista de cirurgias. Assim, as populações vão ser geradas tendo em conta as cirurgias e não tasks.

```
evaluate_population(_,_,[],[]).
evaluate_population(Room,Day,[Ind|Rest],[Ind*TFin|EvaluatedRest]) :-
    schedule_all_surgeries(Room,Day,Ind,TFin),
    evaluate_population(Room,Day,Rest,EvaluatedRest).

```

Quando o algoritmo passar para a fase de avaliar as populações entra no predicado schedule all surgeries. Esse predicado é onde a lógica de lidar com a marcação das cirurgias do sprint 2 está. A partir dai, vai simular as marcações das cirurgias e é obtido a pontuação consoante a hora de marcação da ultima cirurgia, tal como era o críterio do algoritmo no sprint 2. Pontuações mais baixas são mais desejadas.

```
147 definitive_schedule([],_,_):-!.
148 definitive_schedule([LOpCode|Rest],Day,[Room|RestRooms]):-
149     retractall(agenda_operation_room1(_,_,_)),
150     schedule_all_surgeries(Room,Day,LOpCode,TFin),
151     format("Sala de Operações ~w~n", [Room]),
152     (   agenda_operation_room1(Room, Day, AgendaRoom),
153         AgendaRoom \= [] ->
154         format("  Agenda para sala ~w no dia ~w: ~w~n", [Room, Day, AgendaRoom])
155     ;   write("  Nenhuma cirurgia agendada.\n")
156     ),
157     findall(Staff, agenda_staff1(Staff, Day, _), Staffs),
158     forall(
159         member(Staff, Staffs),
160         (
161             (retract(agenda_staff1(Staff, Day, Agenda)) -> true; Agenda = []),
162             (retract(agenda_staff2(Staff, Day, Agenda1)) -> true; Agenda1 = []),
163             append(Agenda, Agenda1, TotalAgenda),
164             assertz(agenda_staff2(Staff, Day, TotalAgenda))
165         )
166     ),
167     definitive_schedule(Rest,Day,RestRooms).
168
```

Após o algoritmo terminar temos a melhor solução possivel. O que vai acontecer é que essa solução é usada para marcar definitivamente as cirurgias no calendário das salas e staff.

6. Consideração de vários blocos de operação.

O algoritmo é iniciado pelo predicado principal startGA/1, que identifica as salas disponíveis e remove quaisquer variaveis dinâmicas antigas relacionadas com as cirurgias, soluções armazenadas, ou agendas temporárias, garantindo que o ambiente esteja limpo para uma nova execução.

O predicado assign_surgery_room/0 é chamado para realizar uma atribuição inicial das cirurgias às salas disponíveis. E é aqui que consideramos os vários blocos e fazemos uma atribuição automática, tal como pedido, das cirurgias pelas salas disponiveis.

```
120 %Começo do algoritmo:
121 startGA(Day):-
122     findall(Room,agenda_operation_room(Room,_,_),Rooms),
123     retractall(surgery_id(_,_,_)),
124     retractall(best_solution_storage(_)),
125     assign_surgery_room,
126     run(Rooms,Day),
```

O que acontece no assign_surgery_room é que são recolhidas todas as cirurgias que neste momento ainda não têm uma sala atribuida. É feita uma listagem e assign_rooms_to_surgeries permite fazer a atribuição da sala para cada cirurgia.

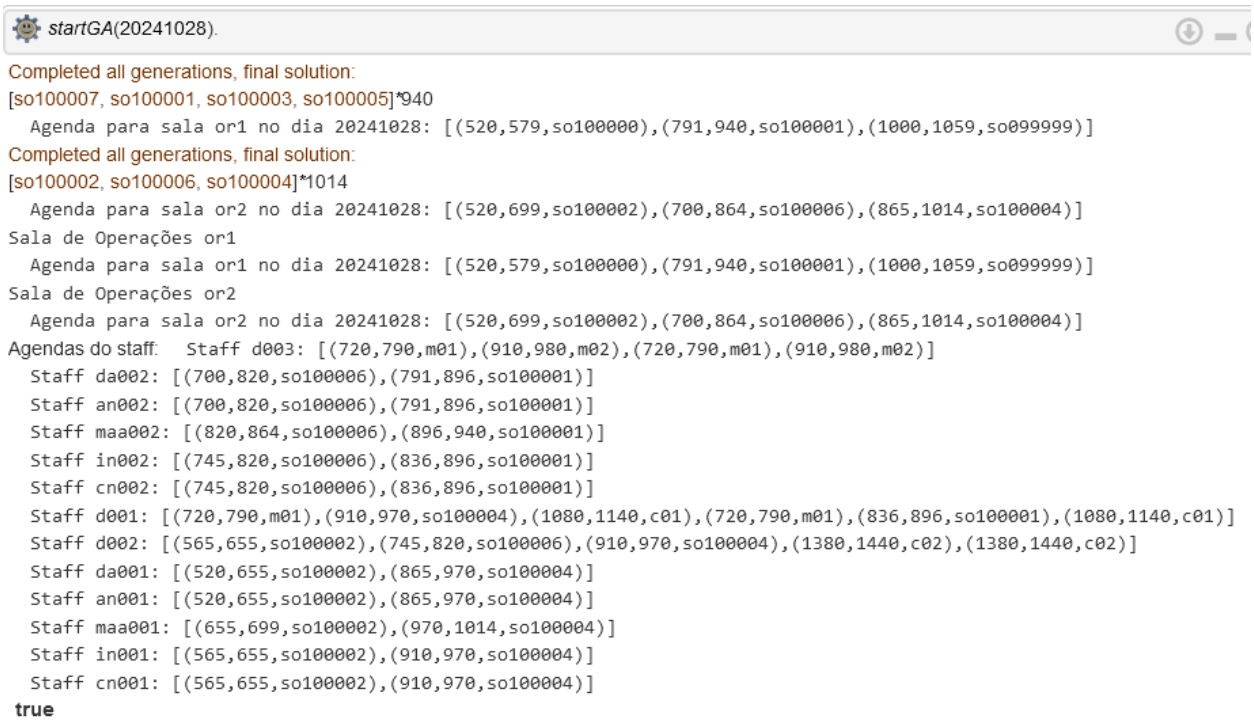
```
90 % Predicado principal para atribuir as salas às cirurgias
91 assign_surgery_room:-
92     % Lista todas as salas disponíveis
93     findall(Room, agenda_operation_room(Room, _, _), Rooms),
94     % Recupera todas as cirurgias a serem agendadas
95     findall(Surgery, surgery_to_schedule(Surgery,_),Surgeries),
96     findall(Type, surgery_to_schedule(_,Type),Types),
97     % Atribui as salas às cirurgias de forma sequencial
98     assign_rooms_to_surgeries(Surgeries,Types,Rooms,0).
99
100 % Atribui as salas às cirurgias usando um índice
101 assign_rooms_to_surgeries([],_,_,_). % Caso base: sem cirurgias para agendar
102 assign_rooms_to_surgeries([Surgery|Rest],[Type|RestTypes],Rooms,Index) :-
103     % Calcula o índice da sala atual (cíclico)
104     length(Rooms, NumRooms),
105     RoomIndex is Index mod NumRooms,
106     nth0(RoomIndex, Rooms, AssignedRoom),
107     assertz(surgery_id(Surgery,Type,AssignedRoom)),
108     % Passa para a próxima cirurgia e incrementa o índice
109     NextIndex is Index + 1,
110     assign_rooms_to_surgeries(Rest,RestTypes,Rooms,NextIndex).
```

Essa atribuição é feita sequencialmente, ou seja, a primeira cirurgia é atribuida à primeira sala, a segunda à segunda e assim sucessivamente até voltar à primeira sala.

```
113 %Começo do algoritmo:
114 startGA(Day):-
115     findall(Room,agenda_operation_room(Room,_,_),Rooms),
116     retractall(surgery_id(_,_,_)),
117     retractall(best_solution_storage(_)),
118     assign_surgery_room,
119     run(Rooms,Day),
120     findall(Solution, best_solution_storage(Solution), BestSolutions),
121     retractall(agenda_operation_room1(_,_,_)),
122     retractall(availability(_,_,_)),
123     retractall(agenda_staff2(_,_,_)),
124     definitive_schedule(BestSolutions,Day,Rooms),
125     write("\nAgendas do staff:\n"),
126     findall(Staff, agenda_staff2(Staff, Day, _), Staffs),
127     sort(Staffs,SortedStaff),
128         forall(member(Staff, Staffs),
129            (agenda_staff2(Staff, Day, AgendaStaffs),
130             format("  Staff ~w: ~w~n", [Staff, AgendaStaffs]))
131     )
132     .
```

```
158 run([],_):-!.
159 run([Room|Rest],Day):-
160     generate(Room,Day,BestSolution),
161     % Extrair apenas a parte relevante da solução.
162     extract_solution(BestSolution, CleanSolution),
163     % Adicionar a solução à variável dinâmica.
164     assertz(best_solution_storage(CleanSolution)),
165     run(Rest,Day).
```

Após essa atribuição o algoritmo encontra a melhor solução para cada sala, no entanto, tem em conta as agendas do staff a todo o momento porque um médico pode fazer uma cirurgia numa sala mas, ter uma marcação seguinte na outra sala, por exemplo. Então durante a marcação das cirurgias todo o staff é tido em conta e persiste ao longo do algoritmo.



O resultado final é a marcação das melhores opções obtidas como se pode ver no exemplo da figura.

7. Estudo do estado da arte

A evolução tecnológica tem transformado profundamente o setor da saúde. Com o aumento no uso de dispositivos avançados, tornou-se possível capturar dados intraoperatórios detalhados, abrindo o caminho para a aplicação de algoritmos de visão computacional (CV) na análise e interpretação desses dados visuais. Esta aplicação traz diversos benefícios para o setor, dando apoio à tomada de decisão dos cirurgiões, melhora a segurança da cirurgia e amplia o acesso aos cuidados cirúrgicos.

No entanto, o desenvolvimento e validação destes algoritmos, especialmente baseados em Machine Learning dependem principalmente do Dataset que é fornecido. De facto, um dos maiores problemas é o processo de fabricação do conjunto de dados ou Dataset que pode implicar alguns problemas relativos ao custo elevado dos equipamentos, exigências éticas rigorosas, necessidade de consentimento dos pacientes e acesso restrito a ambientes hospitalares.

O estado da arte procura examinar o panorama atual destas tecnologias, destacando as suas aplicações, desafios e potenciais futuros. São discutidos casos práticos de sucesso e analisados os principais entraves que ainda restringem uma adoção mais ampla destas inovações. Além disso, são exploradas as perspetivas de desenvolvimento futuro, bem como os desafios éticos e regulatórios que surgem com a expansão do uso de robôs e sistemas de visão computacional no ambiente hospitalar.

7.1. Perspetiva Geral e Relevância no uso da Tecnologia

A robótica experimentou um crescimento significativo ao longo do século passado, impulsionado pelo avanço da eletrónica e da computação. Esse desenvolvimento levou a robôs, sendo amplamente utilizados na indústria, automatizarem operações repetitivas, melhorando a eficiência e a precisão. A partir da década de 1980, a cirurgia minimamente invasiva (MIS) ganhou popularidade, e atualmente uma variedade crescente de procedimentos é realizada por meio de técnicas laparoscópicas, que oferecem vários benefícios em relação à cirurgia aberta, como redução do tempo de hospitalização, menor taxa de complicações parietais, menor dor pós-operatória, melhores resultados estéticos e recuperação mais rápida. (1)

Computer Vision (CV) é a área de estudo que aborda como os computadores podem compreender imagens ou vídeos digitais e procura automatizar tarefas que podem ser realizadas pelo sistema visual humano. Como este campo lida com todos os processos de aquisição de informações do mundo real por computadores, o termo “CV” abrange desde hardware para captura de imagens até o reconhecimento de imagens baseado em Inteligência Artificial. O reconhecimento de imagens baseado em AI para tarefas simples, como identificar imagens estáticas em um dado instante de tempo, progrediu a tal ponto que, em anos recentes, já é comparável ao desempenho humano. (2)

De facto, centenas de milhões de cirurgias são efetuadas em todo o mundo e, como tal, a rápida ascensão do uso destas tecnologias deve–se ao uso das câmaras de fibra ótica e sensores para captar informação detalhada na cirurgia. Apesar de muitos avanços e estudos relacionados ao uso de técnicas de CV, estas tecnologias ainda não são amplamente utilizadas na prática clínica para fins diagnósticos ou terapêuticos. Utilizando a colecistectomia laparoscópica como exemplo, é possível analisar as técnicas atuais de CV aplicadas à cirurgia minimamente invasiva e as suas aplicações clínicas. (3)

7.2. Tecnologias e Métodos Atuais

- Cirurgia Laparoscópica: Rastreamento de Instrumentos (4)

A colecistectomia laparoscópica, um procedimento endoscópico rotineiro, envolve a remoção da vesícula biliar através de pequenas incisões no abdômen, com a ajuda de um endoscópio de vídeo e instrumentos adicionais. O abdômen é inflado com dióxido de carbono para criar espaço para a visualização e execução da cirurgia. O processo geralmente dura entre 30 a 60 minutos. Durante a cirurgia, grandes quantidades de dados de sequência de imagens são geradas, embora nem todos sejam registados. Essas imagens, geralmente monoscópicas, podem ser usadas para compreender e analisar os vídeos de cirurgia utilizando visão computacional. (4)

Uma aplicação relevante de visão computacional é o visual servoing em robôs endoscópicos. Normalmente, um assistente humano é responsável por segurar e mover o endoscópio durante a cirurgia, mas essa prática tem limitações, como a escassez de assistentes qualificados e a fadiga humana. A telemanipulação, onde um robô movimenta o endoscópio com base na combinação de entendimento automático de vídeo e controlo do cirurgião, oferece uma alternativa. (4)

Além disso, aplicações offline para compreensão de vídeo endoscópico incluem indexação, recuperação e análise de cirurgias. Embora haja grande interesse em sistemas de recuperação de informações visuais baseados em conteúdo, ferramentas especializadas para o domínio médico ainda não existem. Este trabalho propõe um método para rastrear instrumentos cirúrgicos em vídeos endoscópicos, com o objetivo de ajudar na inferência do campo de visão para controle do robô ou na modelagem e reconhecimento de ações cirúrgicas para anotação de vídeo. (4)

- Visualização Craniofacial e Simulação Cirúrgica Baseada em Raios-X (5)

A tecnologia apresentada neste artigo traz uma nova abordagem para a visualização craniofacial e a simulação cirúrgica, substituindo o uso de tomografia computadorizada (CT), que envolve altas doses de radiação, por uma técnica que utiliza apenas três raios-X convencionais. A inovação principal é a reconstrução 3D do crânio com base em imagens de raios-X e um ultrassom, o que

torna o processo mais acessível e de menor custo, além de reduzir a exposição à radiação. A tecnologia também permite a criação de um modelo 3D da face usando marcadores de chumbo colocados no rosto do paciente, proporcionando uma visualização detalhada da estrutura craniana interna, útil para planeamento cirúrgico e diagnóstico, especialmente em pacientes pediátricos, para quem a exposição à radiação precisa ser minimizada. (5)

- Uma forma autônoma de detetar e quantificar cataratas usando Visão Computacional (6)

O método baseia-se na deteção de cataratas. Uma catarata é uma opacificação da lente normalmente transparente do olho, e a visão turva causada por cataratas pode aumentar as dificuldades em atividades diárias, como leitura e condução. As cataratas são a principal causa de cegueira em pessoas idosas, e dados do National Eye Institute (NEI) mostram que mais de 65% das pessoas com 80 anos ou mais são diagnosticadas com cataratas. Contudo, existem limitações nos métodos atuais de deteção de cataratas, que incluem: Falta de habilidade clínica, dado que as cataratas são diagnosticadas por oftalmologistas utilizando biomicroscopia de lâmpada de fenda e escalas clínicas estabelecidas. Isto representa uma barreira, especialmente em áreas rurais onde há escassez de oftalmologistas especializados. Outra limitação reside no método utilizado para calcular a potência da lente intraocular (IOL). Atualmente, o algoritmo usado para o cálculo não é padronizado e, por vezes, depende do julgamento subjetivo do cirurgião. (6)

O algoritmo foi implementado utilizando as bibliotecas Python OpenCV, Numpy e FPDF. Ele é estruturado em três etapas principais. Primeiramente, o sistema recebe entradas do usuário, como nome, idade e a imagem a ser processada. Em seguida, a região do olho é manualmente selecionada com o uso do rato, gerando uma imagem recortada para análise posterior. Por fim, a imagem é processada no formato HSV, permitindo a segmentação de cores. Máscaras de branco, preto e castanho são aplicadas para distinguir a brancura do globo ocular e considerar variações na cor dos olhos. Contornos e comparações de áreas ajudam a identificar e isolar as regiões afetadas pela catarata. (6)

7.3. Casos de Sucesso

- Sistema da Vinci da Intuitive Surgical

O sistema da Vinci foi o primeiro equipamento de cirurgia robótica a obter aprovação da FDA em 2000 para procedimentos laparoscópicos gerais. Ele permite que os cirurgiões executem uma variedade de intervenções minimamente invasivas com precisão e eficácia comprovadas clinicamente. Este sistema é amplamente utilizado em diferentes tipos de cirurgias, incluindo cardíacas, colorretais, ginecológicas, de cabeça e pescoço, torácicas, urológicas e gerais minimamente invasivas. Até o momento, mais de 8,5 milhões de procedimentos foram realizados com a ajuda dessa tecnologia. (7)

- Ion by Intuitive Surgical

O Ion, outro sistema cirúrgico robótico da Intuitive, recebeu autorização 510(k) da FDA em fevereiro de 2019. Este dispositivo é um sistema endoluminal robótico projetado para realizar biópsias minimamente invasivas em áreas profundas do pulmão. Ele utiliza um cateter robótico que navega pelas vias aéreas estreitas do pulmão até o local de interesse, guiando ferramentas de biópsia, como pinças e agulhas, para o tecido pulmonar alvo. O Ion emprega tecnologia de detecção de forma por fibra óptica para controlar a posição do cateter durante a navegação, que é travado ao alcançar a área desejada, garantindo estabilidade e precisão na biópsia. O cateter permite movimentos de 180° em todas as direções para ajustes finos na coleta de amostras.(7)

- Mako by Stryker

O Sistema Mako é uma tecnologia de cirurgia robótica que realiza procedimentos de joelho parcial, quadril total e joelho total. Ele usa informações de tomografia computadorizada para criar um modelo 3D da estrutura óssea do paciente, possibilitando um planejamento cirúrgico mais preciso. Esse sistema é amplamente utilizado em cirurgias de substituição de quadril e joelho, proporcionando maior exatidão e melhores resultados para os pacientes. (7

7.4. Desafios Éticos e Regulamentares

A adoção desta tecnologia no ambiente hospitalar traz avanços consideráveis, mas também levanta questões éticas e regulamentares que precisam ser avaliadas com atenção.

- Privacidade e Segurança dos Dados

O uso de tecnologias avançadas envolve a recolha e o processamento de grandes volumes de dados dos pacientes. Garantir a confidencialidade e a segurança dessas informações é essencial para preservar a confiança dos pacientes e cumprir as normas de proteção de dados, como o RGPD da União Europeia. A aderência a essas regulamentações garante que os dados sejam tratados de maneira ética e segura. (8)

- Responsabilidade em Caso de Erros

A determinação de responsabilidade em casos de erros durante procedimentos assistidos por robôs e sistemas de inteligência artificial (IA) é uma questão jurídica complexa. Identificar quem é o responsável em tais situações, seja o desenvolvedor da tecnologia, o cirurgião ou outra parte envolvida, exige uma abordagem clara e precisa. A literatura especializada sublinha a urgência de criar marcos legais que tratem de forma específica e eficaz as questões relacionadas à responsabilidade em casos de falhas desses sistemas, a fim de garantir uma responsabilização justa e adequada, além de preservar a segurança dos pacientes.

- Impacto no Emprego e na Formação Profissional

A automação de procedimentos médicos pode resultar na substituição de mão de obra qualificada em algumas funções, o que gera preocupações sobre o desemprego e a necessidade de requalificação dos profissionais da área. Além disso, a formação de novos profissionais deve ser adaptada para incorporar uma gama mais ampla de competências, focando-se nas tecnologias emergentes. Isso exige um redesenho do currículo educacional para garantir que os futuros profissionais estejam preparados para lidar com as inovações tecnológicas e possam aproveitar ao máximo os avanços da medicina digital.

7.5. Futuro

Level of Autonomy	0: No Autonomy	1: Driver/Surgeon Assistance:	2: Partial Automation:	3: Conditional Automation:	4: High Automation:	5: Full Autonomy:
Automobile	The driver performs all driving tasks.	Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design.	Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all time	Driver is a necessity, but is not required to monitor the environment. The driver must be ready to tale control of the vehicle at all times with notice.	The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.	The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.
Surgery	The surgeon performs all surgical tasks.	Robot-assisted surgery **AI-based image navigation**	Operator is a necessity, but the function of assistant and scopist can be omitted? **Operator:** Task and sub-task execution, e.g., autonomous blunt dissection, suturing, etc. **Assistant:** autonomous surgical view exposure by robot arm. **Scopist:** autonomous endoscopic view control by robot scope holder.		The surgical robot is capable of performing all surgical functions under certain conditions? The surgeon may have the option to control the surgical robot?	The surgical robot is capable of performing all surgical functions under all conditions? The surgeon may have the option to control the surgical robot?

Como mostrado na figura, quando o futuro da cirurgia é comparado à inovação tecnológica nos automóveis, acreditamos que a cirurgia autónoma nos aguarda além da cirurgia de navegação por imagem. Não há dúvida de que o reconhecimento de imagem baseado em IA se tornará uma tecnologia fundamental essencial para a realização da cirurgia autónoma.

No campo dos automóveis, o nível de autonomia 1 é definido como "o veículo é controlado pelo motorista, mas algumas funções assistidas de direção podem estar incluídas no design do veículo". Da mesma forma, a navegação por imagem baseada em IA e a cirurgia assistida por robô ainda se enquadram no nível de autonomia 1. Para que a cirurgia atinja o próximo nível de autonomia, a IA deve, no mínimo, analisar e fornecer etapas cirúrgicas, anatomia, instrumentos, etc., com precisão em tempo real e robusta em todas as situações. Um nível de autonomia muito maior será necessário

para substituir a função do operador, e espera-se que o desafio mais recente no campo da IA na cirurgia seja o de substituir parcialmente a função do assistente e do operador do escopista. (2)

8. Conclusão

O presente trabalho abordou a implementação e adaptação de algoritmos genéticos para o problema de escalonamento de cirurgias a blocos operatórios em hospitais, explorando várias facetas da aplicação desta técnica. Com uma abordagem estruturada e incremental, foram tratados tópicos fundamentais desde os princípios do AG até a sua contextualização no domínio hospitalar.

A introdução de aleatoriedade no cruzamento garante a diversidade genética na população, essencial para evitar convergências prematuras para soluções subótimas. Essa abordagem reforça a robustez do AG, permitindo uma exploração mais ampla do espaço de soluções.

A seleção da nova geração foi projetada para equilibrar diversidade e pontuação, garantindo que o melhor indivíduo entre a população atual e os seus descendentes sempre seja preservado. Contudo, o método evita um comportamento puramente elitista, promovendo variações e assegurando que a população continue a evoluir ao longo das gerações.

Para além do número de gerações, outras condições de término foram consideradas, como a estabilização da população e a obtenção de uma solução com avaliação específica. Estas condições tornam o algoritmo mais adaptável a diferentes cenários, mitigando riscos como ciclos infinitos ou execuções excessivamente longas.

A adaptação do AG ao problema hospitalar focou-se em atribuir eficientemente cirurgias a blocos operatórios. A modelagem do problema considerou restrições práticas, como a disponibilidade de salas, a duração das cirurgias e as necessidades específicas de recursos. O AG mostrou-se uma ferramenta promissora para este tipo de problema combinatório, com capacidade de gerar soluções otimizadas em tempo útil.

A consideração de vários blocos operatórios e a atribuição sequencial inicial das cirurgias às salas garantiram uma base sólida para a otimização posterior pelo AG. Este método assegura que todas as cirurgias têm uma sala atribuída antes da execução do AG, promovendo equilíbrio e eficiência.

A análise do estado da arte evidenciou a crescente aplicação de robots e sistemas de visão por computador no contexto hospitalar, especialmente em cirurgias assistidas. Estas tecnologias têm demonstrado potencial para melhorar a precisão e a eficiência dos procedimentos cirúrgicos, complementando sistemas de planeamento como os desenvolvidos neste trabalho. A combinação de ferramentas de IA generativa, como o ChatGPT, com pesquisa bibliográfica tradicional revelou-se uma abordagem útil para ampliar e diversificar as fontes de informação.

9. Referências

(1) - F. Pugin, P. Bucher, P. Morel
“History of robotic surgery: From AESOP® and ZEUS® to da Vinci”
Disponível em: https://www.sciencedirect.com/science/article/pii/S1878788611000324

(2) - Daichi Kitaguchi, Nobuyoshi Takeshita, Hiro Hasegawa, Masaaki Ito
“Artificial intelligence-based computer vision in surgery: Recent advances and future perspectives”
Disponível em: https://onlinelibrary.wiley.com/doi/pdfdirect/10.1002/ags3.12513

(3) - Pietro Mascagni, Deepak Alapatt, Luca Sestini, Maria S. Altieri, Amin Madani, Yusuke Watanabe, Adnan Alseidi, Jay A. Redan, Sergio Alfieri, Guido Costamagna, Ivo Boškoski, Nicolas Padoy & Daniel A. Hashimoto
“Computer vision in surgery: from potential to clinical value”
Disponível em: https://www.nature.com/articles/s41746-022-00707-5

(4) - S. J. McKenna , H. Nait Charif , and T. Frank
“Towards Video Understanding of Laparoscopic Surgery: Instrument Tracking”
Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=674ffeb0a37dd0bd298953917e44f9ba3dad8025

(5) - Junjun Pan, Jian J Zhang, Yanning Zhang, Hong Zhou
“X-ray Based Craniofacial Visualization and Surgery Simulation”
Disponível em: https://nccastaff.bournemouth.ac.uk/pjunjun/research/X-ray/X-ray%20Based%20Craniofacial%20Visualization%20and%20Surgery%20Simulation.pdf

(6) - Raghav Sharma, Reetu Jain
“An Autonomous way to detect and quantify Cataracts using Computer Vision”
Disponível em: https://d1wqtxts1xzle7.cloudfront.net/

(7) - Shawn Tsuda, Dmitry Oleynikov, Jon Gould, Dan Azagury, Bryan Sandler, Matthew Hutter, Sharona Ross, Eric Haas, Fred Brody & Richard Satava
“Top 5 Robotic Surgery Systems”
Disponível em: https://docwirenews.com/post/top-5-robotic-surgery-systems

(8) - Murdoch, B. (2021).
Privacy and artificial intelligence: challenges for protecting health information in a new era. BMC Medical Ethics.
Disponível em: BMC Medical Ethics