SPRINT 3

Algoritmia Avançada

Miguel Oliveira 1211281

Rodrigo Castro 1220636

Rodrigo Cardoso 1221083

Mário Ribeiro 1221019

Instituto Superior de Engenharia do Porto

Índice

Índice de Figuras

Introdução

O presente documento tem como objetivo apresentar o desenvolvimento e a implementação de soluções inovadoras aplicadas ao contexto hospitalar, com destaque para o uso de algoritmos genéticos (AG) no planejamento e agendamento de cirurgias, bem como para a adoção de tecnologias emergentes, como robótica e visão computacional, em procedimentos médicos. A crescente complexidade dos sistemas hospitalares exige abordagens avançadas para otimizar recursos, reduzir custos e melhorar os resultados para os pacientes. Nesse sentido, os AG representam uma ferramenta poderosa para resolver problemas de escalonamento que envolvem múltiplos critérios de decisão, como a disponibilidade de salas cirúrgicas, equipes médicas e equipamentos.

Além disso, o documento aborda o estudo do estado da arte no uso de robôs e visão computacional em procedimentos cirúrgicos. Esses avanços tecnológicos estão transformando o cenário da saúde, permitindo cirurgias mais precisas, menos invasivas e com maior segurança para os pacientes. A combinação de pesquisa científica tradicional com o suporte de inteligência artificial generativa proporcionou uma análise abrangente, explorando tanto os benefícios quanto os desafios associados a essas tecnologias.

Aleatoriedade no cruzamento entre indivíduos da população no AG

No AG fornecido o cruzamento entre indivíduos, era realizado sempre entre o primeiro e o segundo cromossoma da população, depois era o terceiro e o quarto, e assim por diante, a maneira como resolvemos este problema foi antes de ser realizado o "crossover", fizemos uma permutação aleatória da lista, assim apesar do crossover continuar a ser entre primeiro e segundo, terceiro e quarto, etc…, como alteramos a ordem da lista os elementos em cada posição são sempre aleatórios permitindo assim a variabilidade no cruzamento entre indivíduos o que nos permite evitar a formação de padrões repetitivos e limitar a diversidade genética da próxima geração.

```
random_permutation(Pop, ShuffledPop), % Shuffle the population to avoid the crossover being always between elements in the same position
crossover(ShuffledPop,NPop1),
```

Figura 1 - código que permite a aleatoriedade no cruzamento entre os indivíduos da população

Seleção da nova geração da população do AG

No AG fornecido substitui a população atual, par por par, com os seus descendentes e não há garantir que o melhor indivíduo irá para a próxima geração. Criámos então uma forma de uma percentagem dos melhores indivíduos passar para a próxima geração.

- Primeiro criámos um predicado dinâmico para guardar a percentagem de melhores indivíduos que devem permanecer para a próxima geração. *": -dynamic percentage_individuals/1".*
- Primeiro misturamos os N indivíduos da população atual com os seus descendentes obtidos através de cruzamento e mutação, depois remover indivíduos duplicados (suponhamos que acabamos com T indivíduos).
- Classificámo-los (por ordem crescente, pois o objetivo é a minimização) de acordo com a avaliação de cada um e escolhemos o top P (P maior ou igual a 1 para garantir que os indivíduos P do topo vão através).
- Removemos estes elementos da lista (vão para a próxima geração) e criamos uma nova lista com os restantes TP indivíduos, associando a cada um o produto da avaliação por um número gerado aleatoriamente entre 0 e 1.
- Em seguida, ordenamos os indivíduos desta nova lista por ordem crescente de acordo com este produto. Passamos os primeiros elementos NP desta lista para a próxima geração. Retornamos apenas o valor da avaliação inicial (e não o valor do produto do avaliação pelo número gerado aleatoriamente entre 0 e 1).

A aplicação de um método de seleção que não seja puramente elitista, como o descrito, é essencial para equilibrar a preservação de bons indivíduos com a manutenção da diversidade genética na população. Ao combinar os melhores indivíduos (top P) com outros selecionados de forma ponderada e aleatória, evitamos a estagnação em soluções locais e promovemos uma exploração mais ampla do espaço de procura. Essa abordagem garante que o algoritmo genético mantenha um progresso consistente em direção à solução ótima, sem sacrificar a capacidade de adaptação e inovação nas gerações futuras.

```
append(Pop, NPopValue, PopMergedWithDup), % Merge the old population with the new one
list_to_set(PopMergedWithDup, PopMerged), % Remove duplicates
order_population(PopMerged,PopMergedOrd), % Order the merged population
split_best_individuals(PopMergedOrd, BestIndividuals, RemainingIndividuals), % Split the best individuals from the remaining ones
length(BestIndividuals, NumBest),
population(PopSize),

NumIndividuosAirBuscarRestoLista is PopSize - NumBest,
%nl,
%write('BestIndividuals='),write(BestIndividuals),
%nl,
create_and_sort_remaining_individuals(RemainingIndividuals, NumIndividuosAirBuscarRestoLista, NextGenIndividuals),
%write('Rest of individuals for the population='),write(NextGenIndividuals),nl,nl,
append(BestIndividuals, NextGenIndividuals, NewPopulationNotOrded),
order_population(NewPopulationNotOrded, NewPopulation),
```

Figura 2 - Código dos passos que seguimos para proceder à aplicação de um método de seleção que não seja puramente elitista

```
% Gera o produto (IndList, Eval, Product) para cada indivíduo
generate_random_products([], []).  % Caso base: lista vazia.

generate_random_products([IndList * Eval | Rest], [(IndList * Product) | NewRest]) :-
    random(Rand),
    Product is Eval * Rand,
    generate_random_products(Rest, NewRest).

% Chama o predicado para gerar os produtos aleatórios
create_and_sort_remaining_individuals(RemainingIndividuals, N, NextGenIndividuals) :-
    nl, nl,
    generate_random_products(RemainingIndividuals, RandomizedProducts),
    order_population(RandomizedProducts, SortedProducts),
    %write('Resto da lista SortedProducts='), write(SortedProducts), nl,nl,
    %write('Lista original='), write(RemainingIndividuals), nl,
    extract_top_individuals(SortedProducts, N, RemainingIndividuals, NextGenIndividuals).

extract_top_individuals([], _, _, []).
extract_top_individuals(_, 0, _, []).
extract_top_individuals([Ind * _ | RestSorted], N, RemainingIndividuals, [Ind * Eval | NewRest]) :-
    N1 is N - 1,
    member(Ind * Eval, RemainingIndividuals),
    extract_top_individuals(RestSorted, N1, RemainingIndividuals, NewRest).
```

Figura 3- Código do predicado que faz a seleção dos elementos não elite da população

```
?- generate.
Number of new generations: 2.
Population size: |: 4.
Probability of crossover (%):|: 50.
Probability of mutation (%):|: 20.
Percentage of best individuals to be kept for the next generation (%): |: 20.
Lower cost wanted: |: 1.
Time limit (in seconds): |: 10000000000000.
Date to schedule (yyyymmdd): |: 20251130.
Generation 0:
[[so100004,so100005,so100003,so100001,so100002]*441,[so100003,so100005,so100004,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

PopMerged=[[so100004,so100005,so100003,so100001,so100002]*441,[so100003,so100005,so100004,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

BestIndividuals=[[so100004,so100005,so100003,so100001,so100002]*441]
RemainingIndividuals=[[so100003,so100005,so100004,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

Resto da lista SortedProducts=[[so100003,so100005,so100004,so100002,so100001]*22.242346554983747,[so100004,so100001,so100003,so100002,so100005]*140.7807728680408,[so100001,so100004,so100003,so100005,so100002]*311.5857892596985]

Rest of individuals for the next gen=[[so100003,so100005,so100004,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

NewPopulation=[[so100004,so100005,so100003,so100001,so100002]*441,[so100003,so100005,so100004,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

Generation 1:
[[so100004,so100005,so100003,so100001,so100002]*441,[so100003,so100005,so100004,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

PopMerged=[[so100004,so100005,so100003,so100001,so100002]*441,[so100003,so100005,so100004,so100002,so100001]*451,[so100003,so100004,so100005,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

BestIndividuals=[[so100004,so100005,so100003,so100001,so100002]*441]
RemainingIndividuals=[[so100003,so100005,so100004,so100002,so100001]*451,[so100003,so100004,so100005,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

Resto da lista SortedProducts=[[so100004,so100001,so100003,so100002,so100005]*13.783498968000314,[so100003,so100004,so100005,so100002,so100001]*131.50979563334897,[so100001,so100004,so100003,so100005,so100002]*196.86818052852735,[so100003,so100005,so100004,so100002,so100001]*254.30237879630275]

Rest of individuals for the next gen=[[so100004,so100001,so100003,so100002,so100005]*461,[so100003,so100004,so100005,so100002,so100001]*451,[so100001,so100004,so100003,so100005,so100002]*461]

NewPopulation=[[so100004,so100005,so100003,so100001,so100002]*441,[so100003,so100004,so100005,so100002,so100001]*451,[so100004,so100001,so100003,so100002,so100005]*461,[so100001,so100004,so100003,so100005,so100002]*461]

true .
```

Figura 4 - Exemplo funcionamento adaptação método seleção realizado

```
split_best_individuals(Population, BestIndividuals, RemainingIndividuals) :-
    percentage_individuals(Percentage),
    length(Population, TotalSize),
    N is max(1, round(TotalSize * Percentage)), % Calculate N accordingly with the percentage (min 1)
    length(BestIndividuals, N),
    append(BestIndividuals, RemainingIndividuals, Population). % Divide a população
      You, 2 weeks ago • improving genetic algorithm provided
```

Figura 5 - Código do predicado que separa os n melhores indivíduos do resto da população

Parametrização da condição de término do AG

No AG fornecido a condição de término era feita através dum número de gerações, o algoritmo corria até chegarmos a um determinado número de geração.
Nós desenvolvemos para além da condição de término existente, outras duas condições, uma baseada no tempo de geração da solução, e outra baseada na avaliação mínima pretendida para um determinado elemento da população.

Para implementar a condição de tempo de término o que fizemos foi, primeiramente criar um predicado para guardar o valor desejado pelo utilizador para limite de tempo.

Quando iniciamos a execução do algoritmo obtemos o tempo inicial através do predicado get_time/1 depois criamos a primeira população, se no final da criação da primeira população o limite de tempo tiver sido excedido o algoritmo para, caso contrário parte para a criação de outra geração, e antes de iniciar uma nova geração verifica se o limite de tempo estabelecido não foi ultrapassado, caso tenha sido, interrompe o algoritmo.

Esta estratégia que adotámos não interrompe o algoritmo no instante exato em que o tempo limite é ultrapassado, apenas pode interromper no final da criação de uma determinada geração, a vantagem desta abordagem é termos sempre dados consistentes, para evitar problemas do género, o algoritmo criava a população e de seguida ia fazer a avaliação da mesma mas o tempo tinha sido ultrapassado logo não iria fazer esse passo, como nós implementámos o tempo final pode ser ou não ligeiramente superior ao pretendido mas garante sempre um término consistente do algoritmo, garantindo que mesmo que o tempo limite tenha sido ultrapassado durante passo intermediários da criação de uma nova população, todos eles são executados e só no fim da geração ser criado o algoritmo termina.

Para implementar a condição de término baseada na avaliação mínima pretendida para um determinado elemento da população,primeiramente criámos um predicado para guardar o valor pretendido pelo utilizador como avaliação mínima pretendida, de seguida o que fizemos foi depois de avaliar e ordenar segundo essa avaliação uma população, sabemos que o indivíduo considerado melhor, segundo o nosso método de avaliação é o primeiro elemento da lista, logo o que fazemos é comparar o valor desse elemento com o valor pretendido, se o valor for inferior ou igual ao valor pretendido quer dizer que arranjámos uma solução que satisfaz a condição, caso seja superior continuamos para a próxima geração.

Procedemos à implementação destas duas condições de paragem pois a implementação por exemplo só da condição de paragem a partir da avaliação não faria sentido pois o utilizador pode escolher um valor que nunca vai ser obtido ou que não é possível, logo a existência em simultâneo de uma condição de paragem baseada no tempo é super importante para o algoritmo ter forma de terminar, visto que esta é sempre possível ser obtida.

```
generate_generation(N,G,Pop,StartTime):-
    %write('Generation '), write(N), write(':'), nl, write(Pop), nl,
    get_time(CurrentTime),
    time_limit(TimeLimit),
    ElapsedTime is CurrentTime - StartTime,

    ( ElapsedTime > TimeLimit ->
        %write('Tempo limite atingido após '), write(ElapsedTime), write(' segundos.'), nl,
        !
    ;

    random_permutation(Pop, ShuffledPop), % Shuffle the population to avoid the crossover being always between elements in the same position
    crossover(ShuffledPop,NPop1),

    mutation(NPop1,NPop),

    evaluate_population(NPop,NPopValue),

    append(Pop, NPopValue, PopMergedWithDup), % Merge the old population with the new one
    list_to_set(PopMergedWithDup, PopMerged), % Remove duplicates
    order_population(PopMerged,PopMergedOrd), % Order the merged population
    split_best_individuals(PopMergedOrd, BestIndividuals, RemainingIndividuals), % Split the best individuals from the remaining ones
    length(BestIndividuals, NumBest),
    population(PopSize),

    NumIndividuosAirBuscarRestoLista is PopSize - NumBest,
    %nl,      You, 2 days ago • endpoint in planning for genetic algorithm
    %write('BestIndividuals='),write(BestIndividuals),
    %nl,
    create_and_sort_remaining_individuals(RemainingIndividuals, NumIndividuosAirBuscarRestoLista, NextGenIndividuals),
    %write('Rest of individuals for the population='),write(NextGenIndividuals),nl,nl,
    append(BestIndividuals, NextGenIndividuals, NewPopulationNotOrded),
    order_population(NewPopulationNotOrded, NewPopulation),
    %write('NewPopulation='),write(NewPopulation),nl,nl,
    lowerCostWanted(LowerCost),
    ( NewPopulation = [_*V|_] ->
        ( V =< LowerCost ->
            %write('Lower cost found!'), nl,
            !
        ;
            N1 is N + 1,
            generate_generation(N1, G, NewPopulation, StartTime)
        )
    ;
        N1 is N + 1,
        generate_generation(N1, G, NewPopulation, StartTime)
    )
    ).
```

Figura 6 - Código com as 2 condições de paragem implementadas

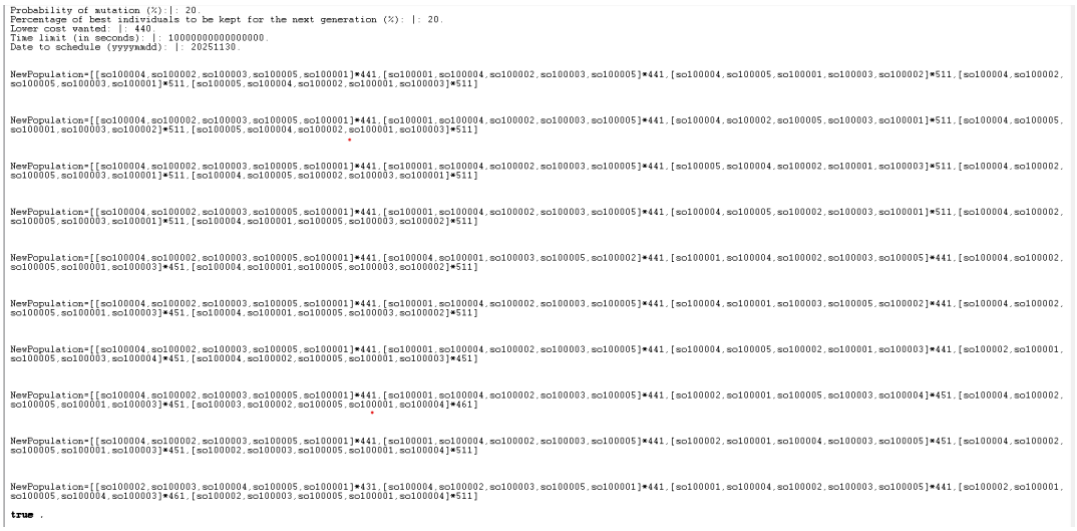

Figura 7 - Amostra resultados critério encontrar a partir de avaliação mínima desejada

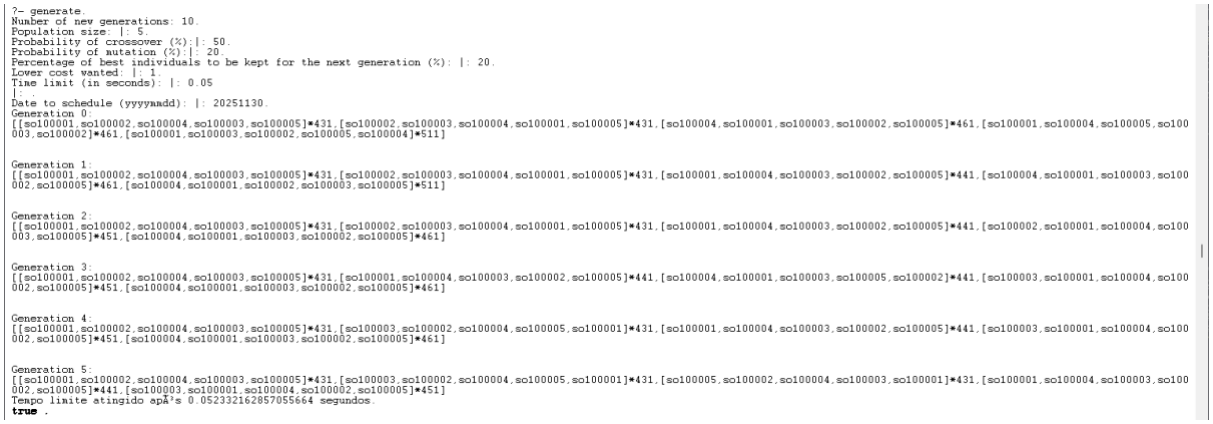

Figura 8 - Amostra resultados critério por tempo de execução

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

O AG fornecido foi desenvolvido para resolver um problema de distribuição de tarefas, no nosso caso, a distribuição é de cirurgias por salas de operação.

Nós analisando o código fornecido percebemos que o único sítio em que o nosso problema era diferente do problema fornecido, era no método de avaliação e no método de geração de primeira população pois a nossa população logicamente tem indivíduos diferentes, ou seja, foi por aí que alterámos.

No método de avaliação o que fizemos foi, para cada elemento da população chamamos o método schedule_surgeries_fromListForMultipleRooms que recebe a data e a sequência de cirurgias que estamos a avaliar, este método cria um facto do tipo lastSurgeryTime, que nos dá o valor a utilizar como avaliador de cada indivíduo, sendo este o tempo final da cirurgia marcada para mais tarde.

O schedule_surgeries_fromListForMultipleRooms não existia previamente, tivemos que o criar tanto por causa de termos que dividir as cirurgias por mais do que uma sala(tópico seguinte), mas também porque o predicado que tínhamos anteriormente, o Schedule_all_surgeries não criava nenhum facto que nos permitisse saber qual a cirurgia marcada para mais tarde, por tanto as nossas adaptações foram nesse sentido, adaptámos o predicado para criar um facto com esse valor para podermos utilizar com avaliação de cada indivíduo.

Na geração da primeira população a nossa alteração foi simples em vez de fazermos um findAll para as tasks fizemos para as cirurgias de forma a retornar os OpCode de todas as cirurgias a marcar.

```
generate_population(Pop):-
    population(PopSize),
    surgeries_number(NumT),
    findall(OpCode,surgery_id(OpCode,_),LOpCode),
    generate_population(PopSize,LOpCode,NumT,Pop).
```

Figura 9 - Código do generate_population

```
evaluate(Seq,V):-
    dateToSchedule(Data),
    schedule_surgeries_fromListForMultipleRooms(Data,Seq), lastSurgeryTime(V).
```

Figura 10 - Código do evaluate

```
schedule_surgeries_fromListForMultipleRooms(Day,LOpCode):-
    retractall(lastSurgeryTime(_)),
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(_,_,_)),
    retractall(availability(_,_,_)),
    findall(_,(agenda_staff(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda))),_),
    findall((Or), (agenda_operation_room(Or, Date, Agenda), assert(agenda_operation_room1(Or, Date, Agenda))), Lrooms),
    %write('Lista de rooms = '), write(Lrooms), nl,
    findall(_,(agenda_staff1(D,Date,L),free_agenda0(L,LFA),adapt_timetable(D,Date,LFA,LFA2),assertz(availability(D,Date,LFA2))),_),
    asserta(lastSurgeryTime(0)),
    availability_all_surgeries4(LOpCode,Lrooms,Day),!.

availability_all_surgeries4([], _, _) :-!.
availability_all_surgeries4([OpCode|LOpCode], [], Day):-
    % Reinicia a lista de salas caso esteja vazia
    findall(Or, agenda_operation_room(Or, Date, Agenda), Lrooms),
    availability_all_surgeries4([OpCode|LOpCode], Lrooms, Day),!.

availability_all_surgeries4([OpCode|LOpCode], [Room|LRooms], Day):-
    surgery_id(OpCode, OpType),
    %write('A atribuir a cirurgia'),write(OpCode),write(' à sala '),write(Room),nl,
    surgery(OpType, TAnesthesy, TSurgery, TCleaning),
    availability_operation(OpCode, Room, Day, Interval, LDoctorsSurgery, LStaffAnesthesy, LStaffCleaning),
    calculate_intervals(Interval, TAnesthesy, TSurgery, TCleaning, MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess),
    %write('Intervalo calculado: '), write((MinuteStartAnesthesia, MinuteEndProcess)), nl,
    retract(agenda_operation_room1(Room, Day, Agenda)),
    insert_agenda((MinuteStartAnesthesia, MinuteEndProcess, OpCode), Agenda, Agenda1),
    assertz(agenda_operation_room1(Room, Day, Agenda1)),
    %write('Agenda atualizada da sala'),write(Room),write(' atualizada para -->') ,write(Agenda1), nl,
    insert_agenda_staff((MinuteStartSurgery, MinuteStartCleaning, OpCode), Day, LDoctorsSurgery),
    insert_agenda_staff((MinuteStartAnesthesia, MinuteStartCleaning, OpCode), Day, LStaffAnesthesy),
    insert_agenda_staff((MinuteStartCleaning, MinuteEndProcess, OpCode), Day, LStaffCleaning),
    lastSurgeryTime(LastSurgeryTime),
    %write('Último tempo de cirurgia: '), write(LastSurgeryTime), nl,
    % Atualizando LastSurgeryTime apenas se for maior que o valor atual
    %write('Minuto de fim do processo: '), write(MinuteEndProcess), nl,


    (LastSurgeryTime < MinuteEndProcess ->      You, 2 days ago • creating method to distribute surgeries acroos …
        %write('Atualizando lastSurgeryTime'), nl,
        retract(lastSurgeryTime(_)),
        assertz(lastSurgeryTime(MinuteEndProcess)),
        !
    ;
        true
    ),
```

Figura 11 - código do schedule_surgeries_fromListForMultipleRooms

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

Tal como já foi dito anteriormente criamos o método **schedule_surgeries_fromListForMultipleRooms** que recebe a data e a sequência de cirurgias que estamos a avaliar. No entanto para adaptar o agendamento das cirurgias aos requisitos pedidos foi feita uma pequena adaptação do método **schedule_surgeries_fromListForMultipleRooms**, chamamos-lhe de ***schedule_surgeries_fromListForMultipleRoomsOptimizedByMario.***

```
% Predicado principal otimizado
schedule_surgeries_fromListForMultipleRoomsOptimizedByMario(Day, LOpCode) :-
    retractall(lastSurgeryTime(_)),
    retractall(agenda_staff1(_, _, _)),
    retractall(agenda_operation_room1(_, _, _)),
    retractall(availability(_, _, _)),
    findall(_, (agenda_staff(D, Day, Agenda), assertz(agenda_staff1(D, Day, Agenda))), _),
    findall((Or), (agenda_operation_room(Or, Day, Agenda), assert(agenda_operation_room1(Or, Day, Agenda))), LRooms),
    findall(_, (agenda_staff1(D, Day, L), free_agenda0(L, LFA), adapt_timetable(D, Day, LFA, LFA2), assertz(availability(D, Day, LFA2))), _),
    asserta(lastSurgeryTime(0)),
    LRooms = [FirstRoom | _], % Inicializa com a primeira sala
    availability_all_surgeries_optimized(LOpCode, FirstRoom, LRooms, Day), !.
```

Figura 12 - Adaptação do scheduleMultipleRooms

O que diferencia esta adaptação do método é que o método chama o método **availability_all_surgeires_optimized/4** no qual também é necessária a inicialização do parâmetro FirstRoom (explicação do porquê mais abaixo).

Para dividir as cirurgias por múltiplas salas tivemos em atenção o ratio de cada sala. O ratio de uma sala corresponde à divisão da soma dos tempos das operações agendadas para a sala e a soma dos tempos livres da sala. Utilizamos o ratio para garantir que a sala em questão nunca ficaria sobrelotada e, para tal, usamos um valor de referência de o,8 de ratio (que as salas não poderiam exceder). Foi também solicitada uma análise, na qual deveríamos ter em atenção o uso excessivo de salas, ou seja, usar muitas salas quando possivelmente conseguimos usar menos salas e mantê-las com um ratio dentro dos limites.
Para cumprir estes requisitos realizamos um método (**availability_all_surgeires_optimized/4)** que analisará uma lista de cirurgias a ser agendada **(LOpCode)** e as dividirá por uma lista de

salas **(LRooms)** até não haver mais cirurgias para agendar ou não haver mais espaço nas salas (mantendo-as sempre dentro do ratio limite) para agendar as cirurgias.

```
% Predicado principal otimizado
availability_all_surgeries_optimized([], _, _, _) :-
    write("Todas as cirurgias foram alocadas com sucesso."), nl.

availability_all_surgeries_optimized([OpCode | LOpCode], CurrentRoom, LRooms, Day) :-
    format("Tentando alocar cirurgia ~w na sala ~w para o dia ~w.~n", [OpCode, CurrentRoom, Day]),
    (   try_allocate_surgery(OpCode, LRooms, Day)
    ->  availability_all_surgeries_optimized(LOpCode, CurrentRoom, LRooms, Day)
    ;   write("Não há mais espaço nas salas. Todas as cirurgias possíveis foram devidamente alocadas."), nl,
        fail % Este fail interrompe o processo para indicar que houve rejeições.
    ).

% Predicado auxiliar para tentar alocar a cirurgia em alguma sala
try_allocate_surgery(_, [], _) :-
    !, fail.  % Se não houver mais salas, falha.

try_allocate_surgery(OpCode, [Room | RestRooms], Day) :-
    (   can_fit_in_room(OpCode, Room, Day)
    ->  format("Cirurgia ~w cabe na sala ~w. Alocando...~n", [OpCode, Room]),
        allocate_surgery(OpCode, Room, Day)
    ;   format("Cirurgia ~w não cabe na sala ~w. Tentando próxima sala...~n", [OpCode, Room]),
        try_allocate_surgery(OpCode, RestRooms, Day)
    ).
```

Figura 13 - availability_all_surgeires_optimized/4

O processo de agendamento é começado na **FirstRoom** que será a primeira sala na lista de salas. É utilizado o método **can_fit_in_room** que vai verificar se uma cirurgia pode ser agendada na sala selecionada (**FirstRoom**) para o dia especificado (**Day**). Este método calcula o novo ratio da sala caso a cirurgia seja alocada e verifica se o ratio permanece abaixo ou igual ao limite de 0.8.

Caso a cirurgia não caiba na sala atual, o método tenta alocá-la na próxima sala disponível na lista, utilizando o predicado **try_allocate_surgery**. Este predicado percorre recursivamente a lista de salas (**LRooms**) até encontrar uma sala que possua espaço suficiente para a cirurgia ou esgotar todas as salas, indicando que a cirurgia não pode ser agendada. Desta forma garantimos que uma sala só será usada caso não seja estritamente necessário, o que fará com que haja sempre salas disponíveis para emergências.

Para calcular se uma cirurgia pode ser alocada em uma sala, utilizamos o predicado **calculate_room_ratio**, que avalia o impacto da cirurgia no ratio da sala considerando os tempos de anestesia, cirurgia e limpeza. Caso a sala possua espaço suficiente, a cirurgia é alocada utilizando o predicado **allocate_surgery**, que atualiza a agenda da sala, os horários do staff e outros dados relevantes.

O predicado principal, **availability_all_surgeries_optimized**, itera sobre a lista de cirurgias (**LOpCode**) e tenta alocar cada cirurgia na lista de salas (**LRooms**). Se todas as cirurgias forem alocadas com sucesso, o programa retorna uma mensagem indicando sucesso. Caso contrário, informa que não há mais espaço disponível para alocar as cirurgias restantes.

```
% Verificar se a cirurgia pode ser alocada
can_fit_in_room(OpCode, Room, Day) :-
    calculate_room_ratio(OpCode, Room, Day, NewRatio),
    format("Calculando ratio para cirurgia ~w na sala ~w: ~2f~n", [OpCode, Room, NewRatio]),
    NewRatio =< 0.8.

% Calcular o novo ratio da sala se a cirurgia for adicionada
calculate_room_ratio(OpCode, Room, Day, NewRatio) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, TAnesthesy, TSurgery, TCleaning),
    TotalTime is TAnesthesy + TSurgery + TCleaning,
    room_free_time(Room, Day, FreeTime),
    room_current_usage(Room, Day, UsedTime),
    NewRatio is (UsedTime + TotalTime) / FreeTime.

% Alocar a cirurgia à sala atual
allocate_surgery(OpCode, Room, Day) :-
    surgery_id(OpCode, OpType),
    surgery(OpType, TAnesthesy, TSurgery, TCleaning),
    availability_operation(OpCode, Room, Day, Interval, LDoctorsSurgery, LStaffAnesthesy, LStaffCleaning),
    calculate_intervals(Interval, TAnesthesy, TSurgery, TCleaning, MinuteStartAnesthesia, MinuteStartSurgery, MinuteStartCleaning, MinuteEndProcess),
    retract(agenda_operation_room1(Room, Day, Agenda)),
    insert_agenda((MinuteStartAnesthesia, MinuteEndProcess, OpCode), Agenda, Agenda1),
    assertz(agenda_operation_room1(Room, Day, Agenda1)),
    insert_agenda_staff((MinuteStartSurgery, MinuteStartCleaning, OpCode), Day, LDoctorsSurgery),
    insert_agenda_staff((MinuteStartAnesthesia, MinuteStartCleaning, OpCode), Day, LStaffAnesthesy),
    insert_agenda_staff((MinuteStartCleaning, MinuteEndProcess, OpCode), Day, LStaffCleaning),
    lastSurgeryTime(LastSurgeryTime),
    (LastSurgeryTime < MinuteEndProcess ->
        retract(lastSurgeryTime(_)),
        assert(lastSurgeryTime(MinuteEndProcess))
    ; true).
```

Figura 14 - Fit RoomRatio e Allocate

Os tempos disponíveis e utilizados em cada sala são calculados utilizando métodos auxiliares que analisam a agenda da sala para o dia específico. Esses cálculos são essenciais para determinar se uma cirurgia pode ser alocada sem ultrapassar o limite do ratio.

O predicado **calculate_free_time** calcula o tempo livre total da sala baseado na sua agenda. Caso não haja nenhum agendamento, considera-se que o dia inteiro (1440 minutos) está disponível. Se existirem agendamentos, o predicado percorre a lista de intervalos ocupados e subtrai a soma dos intervalos ocupados do tempo total disponível no dia.

Já o predicado **calculate_used_time** computa o tempo total utilizado numa sala. Ele percorre a lista de agendamentos da sala, somando a duração de cada intervalo ocupado (definido como a diferença entre o horário de início e término de cada operação).

Os predicados **room_free_time** e **room_current_usage** utilizam os cálculos acima para obter, respetivamente, o tempo livre e o tempo utilizado numa sala no dia especificado. Eles consultam a agenda de operações para o dia (**agenda_operation_room1/3**), garantindo que o cálculo é baseado nos agendamentos atualizados. Adicionalmente, mensagens de depuração (**format/2**) são utilizadas para fornecer informações detalhadas sobre o estado da agenda e os tempos calculados, facilitando a análise e validação do funcionamento do algoritmo.

Exemplo de teste com os seguintes dados:

```
% Agenda staff
agenda_staff(d001,20251130,[]).
agenda_staff(d002,20251130,[]).
agenda_staff(d003,20251130,[]).
agenda_staff(d004,20251130,[]).
agenda_staff(d005,20251130,[]).
agenda_staff(e001,20251130,[]).
agenda_staff(e002,20251130,[]).
agenda_staff(e003,20251130,[]).

% Timetable
timetable(d001,20251130,(300,1400)).
timetable(d002,20251130,(300,1400)).
timetable(d003,20251130,(300,1400)).
timetable(d004,20251130,(300,1400)).
timetable(d005,20251130,(300,1400)).
timetable(e001,20251130,(300,1400)).
timetable(e002,20251130,(300,1400)).
timetable(e003,20251130,(300,1400)).

% Surgery
surgery(so2,5,15,10).
surgery(so3,10,15,25).
surgery(so4,30,50,20).
surgery(so5,20,60,30).
surgery(so6,500,500,30).
surgery(so7,20,60,30).
surgery(so8,600,60,30).


% Surgery IDs
surgery_id(so100001,so2).
surgery_id(so100002,so3).
surgery_id(so100003,so4).
surgery_id(so100004,so5).
surgery_id(so100005,so6).
surgery_id(so100006,so7).
surgery_id(so100007,so8).
```

Figura 15 - Dados para teste

Resultado:

```
% c:/Users/mario/OneDrive - Instituto Superior de Engenharia do Porto/Prolog/algorithmByMarioForUS731.pl compiled 0.00 sec, -7 clauses
?- schedule_surgeries_fromListForMultipleRoomsOptimizedByMario(20251130, [so100001, so100002, so100003, so100004, so100005, so100006,  so100007]).
Tentando alocar cirurgia so100001 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: []
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando ratio para cirurgia so100001 na sala or1: 0.02
Cirurgia so100001 cabe na sala or1. Alocando...
Tentando alocar cirurgia so100002 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: [(300,330,so100001)]
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 330.
Calculando ratio para cirurgia so100002 na sala or1: 0.06
Cirurgia so100002 cabe na sala or1. Alocando...
Tentando alocar cirurgia so100003 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: [(300,330,so100001),(331,381,so100002)]
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 330.
Calculando tempo utilizado: Inicio 331, Fim 381.
Calculando ratio para cirurgia so100003 na sala or1: 0.13
Cirurgia so100003 cabe na sala or1. Alocando...
Tentando alocar cirurgia so100004 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: [(300,330,so100001),(331,381,so100002),(382,482,so100003)]
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 330.
Calculando tempo utilizado: Inicio 331, Fim 381.
Calculando tempo utilizado: Inicio 382, Fim 482.
Calculando ratio para cirurgia so100004 na sala or1: 0.23
Cirurgia so100004 cabe na sala or1. Alocando...
Tentando alocar cirurgia so100005 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: [(300,330,so100001),(331,381,so100002),(382,482,so100003),(483,593,so100004)]
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 330.
Calculando tempo utilizado: Inicio 331, Fim 381.
Calculando tempo utilizado: Inicio 382, Fim 482.
Calculando tempo utilizado: Inicio 483, Fim 593.
Calculando ratio para cirurgia so100005 na sala or1: 1.15
Cirurgia so100005 não cabe na sala or1. Tentando próxima sala...
Obtendo tempo livre para sala or2 no dia 20251130.
Agenda da sala or2 no dia 20251130: []
Obtendo tempo utilizado para sala or2 no dia 20251130.
Calculando ratio para cirurgia so100005 na sala or2: 0.72
Cirurgia so100005 cabe na sala or2. Alocando...
Tentando alocar cirurgia so100006 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: [(300,330,so100001),(331,381,so100002),(382,482,so100003),(483,593,so100004)]
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 330.
Calculando tempo utilizado: Inicio 331, Fim 381.
Calculando tempo utilizado: Inicio 382, Fim 482.
Calculando tempo utilizado: Inicio 483, Fim 593.
Calculando ratio para cirurgia so100006 na sala or1: 0.35
Cirurgia so100006 cabe na sala or1. Alocando...
Tentando alocar cirurgia so100007 na sala or1 para o dia 20251130.
```

Figura 16 - Resultado Teste

```
Tentando alocar cirurgia so100007 na sala or1 para o dia 20251130.
Obtendo tempo livre para sala or1 no dia 20251130.
Agenda da sala or1 no dia 20251130: [(300,330,so100001),(331,381,so100002),(382,482,so100003),(483,593,so100004),(594,704,so100006)]
Obtendo tempo utilizado para sala or1 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 330.
Calculando tempo utilizado: Inicio 331, Fim 381.
Calculando tempo utilizado: Inicio 382, Fim 482.
Calculando tempo utilizado: Inicio 483, Fim 593.
Calculando tempo utilizado: Inicio 594, Fim 704.
Calculando ratio para cirurgia so100007 na sala or1: 1.05
Cirurgia so100007 não cabe na sala or1. Tentando próxima sala...
Obtendo tempo livre para sala or2 no dia 20251130.
Agenda da sala or2 no dia 20251130: [(300,1330,so100005)]
Obtendo tempo utilizado para sala or2 no dia 20251130.
Calculando tempo utilizado: Inicio 300, Fim 1330.
Calculando ratio para cirurgia so100007 na sala or2: 4.20
Cirurgia so100007 não cabe na sala or2. Tentando próxima sala...
Obtendo tempo livre para sala or3 no dia 20251130.
Agenda da sala or3 no dia 20251130: []
Obtendo tempo utilizado para sala or3 no dia 20251130.
Calculando ratio para cirurgia so100007 na sala or3: 0.48
Cirurgia so100007 cabe na sala or3. Alocando...
Todas as cirurgias foram alocadas com sucesso.
true.

?-
```

Figura 17 - Resultado Teste

Análise do Estado da Arte da Aplicação de Robôs e Visão Computacional em Cirurgias

Este estudo tem como objetivo apresentar um panorama detalhado da aplicação de robôs e visão computacional em cirurgias. Ele combina pesquisa baseada em humanos com as capacidades da IA generativa, analisando as tecnologias e seus mecanismos, além de destacar as contribuições de cada abordagem.

Contextualização Histórica

A **cirurgia robótica** (CR) [1] é uma evolução da cirurgia minimamente invasiva que combina a medicina cirúrgica, a robótica e a engenharia. O conceito de CR surgiu de pesquisas em robótica financiadas pela NASA e pela Agência de Projetos de Pesquisa Avançada de Defesa dos Estados Unidos durante os anos 1970 [1].

O objetivo inicial era criar um sistema que permitisse que os procedimentos cirúrgicos fossem controlados remotamente, substituindo assim os cirurgiões em ambientes perigosos ou difíceis de alcançar, como em zonas de guerra e em missões espaciais.

Foi só nos anos 80, mais propiamente em 1985, que o primeiro robô cirúrgico foi implementado na sociedade. Chamado de ***Programmable Universal Machine for Assembly 560*** (**PUMA 560**) [1], foi empregado em uma biópsia neurocirúrgica na *Westinghouse Electric* em Pittsburgh, Pensilvânia [1], nos Estados Unidos da América. O sistema utilizava a linguagem de programação VAL II para traduzir comandos simples em sinais elétricos que acionavam o robô. O seu braço possuía 6 juntas que possibilitavam uma movimentação articulada e precisa no momento da cirurgia [2].

Em 1988, o ***Imperial College of London*** desenvolveu um sistema robótico chamado **ProBot** [1]. Este robô foi projetado para auxiliar na remoção de excesso de tecidos na próstata. O robô possuía quatro eixos de movimento, uma lâmina rotativa de alta velocidade para a remoção e um tamanho compacto adequado para procedimentos analíticos da próstata [1].

Nos anos 1990, a empresa ***Computer Motion*** desenvolveu o braço robótico ***Automated Endoscope System for Optimal Positioning*** **(AESOP)** [1], que recebeu aprovação da FDA (Food and Drug Administration), tornando-se no primeiro robô cirúrgico de controlo remoto [1].

Como evoluiu nos últimos 5 anos

Nos últimos 5 anos, a cirurgia robótica evoluiu significativamente com o desenvolvimento de novas tecnologias e a expansão de aplicações clínicas.

O sistema robótico ***Senhance* (*Telelap ALF-X*)** [3] foi introduzido na área da medicina **ginecológica**. Uma das principais características do *Senhance* é o **feedback tátil** [3], que proporciona ao cirurgião informações táteis sobre a força e a direção aplicadas no local cirúrgico, através de contramovimentos na consola do cirurgião. Também inclui um **sistema de rastreamento ocular** [3], que controla os movimentos da câmara e amplia a imagem quando a cabeça do cirurgião se aproxima do ecrã. O *Senhance* foi utilizado em **remoções do útero** para doenças benignas e malignas, bem como em **cortes de zonas cancerígenas renais**.

Desenvolvido na China, o ***Micro Hand S*** [3] é um sistema robótico cirúrgico minimamente invasivo doméstico [3]. O robô é composto por uma consola de cirurgião e um carrinho funcional por controlo remoto. Foram usados em **perfurações gástricas** e **remoções de apendicite aguda**.

O sistema robótico ***Flex*** [3] oferece **acesso de porta única** e permite a visualização de locais anatómicos de difícil acesso. Implementado pela empresa ***Medrobotic***, uma das mais conceituadas no mercado dos robôs cirurgiões, o sistema possui um escopo flexível, semelhante a uma cobra, com segmentos articulados e canais para o uso de instrumentos flexíveis [3]. O *Flex* foi usado no **tratamento de nódulos benignos e malignos que se desenvolvem na glândula da tiroide**.

O robô ***Aquablation*** [3], usado em terapias para a **remoção dos excessos de tecidos da próstata** (já citado anteriormente) combina um cistoscópio integrado com ultrassom intraoperatório. O sistema integrado ***AquaBeam*** [3], integrado no sistema robótico, utiliza software para calcular as taxas de fluxo com base no comprimento, profundidade e largura da remoção parcial necessária.

Desenvolvido pela **Mazor Robotics**, a máquina ***MAZOR X*** [3] é um sistema de orientação robótica em cirurgias da **coluna**. O software ***MAZOR X Align*** [3] auxilia no planeamento da correção de deformidades da coluna, combinando automaticamente as vértebras vistas em imagens Raio-X com as imagens tomográficas (vistas de cima).

Vantagens e Desvantagens da Cirurgia Robótica

Os sistemas robóticos cirúrgicos apresentam uma série de vantagens e desvantagens, que têm sido objeto de estudo e desenvolvimento contínuo.

Os sistemas robóticos oferecem **maior precisão** e **destreza** [3] em comparação com as cirurgias tradicionais, devido aos seus braços robóticos **articulados** e à capacidade de realizar movimentos complexos com maior facilidade.

A tecnologia de **visualização 3D** [3] de alta-definição melhora a perceção da profundidade e os detalhes anatômicos, proporcionando uma visão **mais clara e detalhada** da zona cirúrgica. Permitem procedimentos cirúrgicos **minimamente invasivos** [3], onde reduz-se o trauma cirúrgico, a dor pós-operatória, o tempo de recuperação e as cicatrizes causadas pela cirurgia.

Sendo uma tecnologia em expansão, abriu-se espaço para mais inovação e concorrência, resultando em mais opções e sistemas avançados.

Apesar das grandes vantagens da cirurgia robótica, esta também acarreta desvantagens, tal como o **preço muito alto** dos sistemas robóticos [3], o que pode limitar o acesso a essa tecnologia em alguns hospitais e países. Além do preço, será necessária uma **curva de aprendizagem** para a familiarização do uso de robôs cirúrgicos [3].

O Futuro da Cirurgia Robótica

O futuro da cirurgia robótica apresenta um cenário repleto de inovações tecnológicas e transformações nos cuidados de saúde. Entre os avanços mais promissores estão a maior precisão e personalização dos procedimentos, o desenvolvimento de instrumentos "inteligentes" que ampliam as capacidades cirúrgicas, a automação de tarefas que simplifica operações complexas e a exploração de novas aplicações para atender a uma variedade de necessidades médicas.

Essa evolução tecnológica está diretamente ligada à transformação dos cuidados de saúde. As cirurgias robóticas estão cada vez mais seguras e menos invasivas, com a incorporação de **inteligência artificial** e ***machine learning***, que aprimoram desde o diagnóstico até a execução dos procedimentos. Além disso, a **telecirurgia** [4] surge como uma possibilidade concreta, permitindo que cirurgiões realizem operações à distância, enquanto os instrumentos são personalizados para atender às especificidades de cada paciente.

No que diz respeito ao planeamento e ao *feedback* durante os procedimentos, a **impressão 3D** poderá desempenhar um papel fundamental, com a possibilidade de criação de próteses totalmente personalizadas [5].

A formação em cirurgia robótica também está a ser reinventada, com o foco na **educação contínua** [5]. Simulações avançadas e plataformas de treinamento específicas estão a ser desenvolvidas para capacitar profissionais, assim garantindo que eles acompanhem as inovações tecnológicas e estejam preparados para enfrentar os desafios do futuro da medicina robótica.

Contribuição da IA Generativa e Humanos

A inteligência artificial contribui significativamente em diversas áreas, como na **análise de dados** [6], ao extrair informações relevantes de grandes volumes de conteúdo. Além disso, auxilia na organização ao criar uma estrutura coerente para os dados analisados e na geração de conteúdo, produzindo textos descritivos e até mesmo ilustrações para complementar informações.

Apesar das capacidades da IA, as tarefas humanas continuam indispensáveis. Entre elas, destaca-se a **validação**, que consiste em conferir a precisão das informações geradas. Também são importantes a **interpretação** e o **alinhamento** dessas informações [6] com as necessidades específicas do ambiente hospitalar.

O refinamento e a edição garantem maior **clareza** e **fluidez** [6] ao texto produzido, enquanto as recomendações humanas transformam os resultados em **conclusões práticas** para aplicação no quotidiano.

É importante ressaltar que a IA generativa funciona como uma ferramenta auxiliar, mas o seu uso requer supervisão humana para garantir a qualidade e a adequação das informações geradas

Conclusão

A combinação de pesquisa humana e IA generativa é uma abordagem promissora para o estudo da aplicação de robôs e visão computacional em cirurgias. Enquanto os humanos fornecem expertise, a IA auxilia na análise de dados e geração de conteúdo.

A colaboração entre ambas permite criar estudos abrangentes e informativos, auxiliando na tomada de decisão sobre a implementação dessas tecnologias. A validação e interpretação humanas são cruciais, especialmente nos aspetos éticos e de segurança. A adoção responsável dessas tecnologias pode transformar os cuidados de saúde, com o objetivo de democratizar o acesso e melhorar os resultados clínicos.

Conclusões

O uso de algoritmos genéticos no contexto hospitalar demonstrou ser uma abordagem eficaz para lidar com os desafios do escalonamento de cirurgias. A capacidade desses algoritmos de explorar grandes espaços de solução e otimizar múltiplos critérios simultaneamente oferece uma vantagem significativa sobre métodos tradicionais. As adaptações realizadas no modelo base, como a incorporação de parâmetros específicos para o ambiente hospitalar, resultaram em soluções mais adequadas à realidade operacional, permitindo não apenas o planejamento eficiente das cirurgias, mas também uma melhor alocação de recursos críticos, como salas, equipes médicas e equipamentos.

Adicionalmente, a análise do uso de robôs e visão computacional em cirurgias mostrou o potencial transformador dessas tecnologias na medicina. Robôs cirúrgicos, por exemplo, aumentam a precisão dos procedimentos e reduzem os riscos associados a cirurgias invasivas. Já a visão computacional possibilita diagnósticos mais rápidos e precisos, além de facilitar o monitoramento em tempo real de procedimentos médicos. No entanto, a adoção dessas tecnologias ainda enfrenta desafios, como altos custos, barreiras de treinamento e questões regulatórias, que precisam ser superados para que possam ser amplamente utilizadas.

Referências

[1] Y. River-Moreno, “Robotic Surgery: A Comprehensive Review of the Literature and Current Trends,” Cureus, Barcelona, Espanha, 2023.

[2] J. Rutherford, “USING THE PUMA 560 ROBOT”.

[3] C. Gosrisirikul, “New era of robotic surgical systems,” ASES, Seul, Coreia do Sul, 2018.

[4] C. Vaz, “Cirurgia Robótica: Uma Revolução Silenciosa na Medicina Moderna,” Gazeta, 2024.

[5] H. d. M. Rodrigues, “A Cirurgia Robótica e a Navegação por Computador na Artoplasia do Joelho - O Presente e o Futuro,” Coimbra, Março 2018.

[6] J. M. F. Fernandes, “Otimização das Práticas de Saúde: Integração de Big Data e Inteligência Artificial no Diagnóstico Médico,” Lisboa, Junho 2024.

[7] Cirúrgia Robótica Da Vinci, Brasil: Strattner.