RELATÓRIO ALGAV

Sprint C

3DB Grupo 9
Vicente Cardoso (1180664)
Tiago Sousa (1191583)
Ana Beatriz Costa (1201313)
Luís Reis (1210998)
Ricardo Ribeiro (1221695)

Índice

Introdução

Este relatório foi desenvolvido no contexto da Unidade Curricular (UC) de Algoritmia Avançada (ALGAV), como parte integrante da Licenciatura em Engenharia Informática no ano letivo 2024-2025, utilizando PROLOG.

Inicialmente será apresentado o Estado da Arte relativo à aplicação de robótica e visão computacional no meio hospitalar, de forma a contextualizar e salientar a importância deste trabalho.

De seguida, é abordada a forma como se consideram diferentes salas para o agendamento de cirurgias, e como é feita essa distribuição por diversas salas.

Por fim, recorre-se a algoritmos genéticos para realizar o agendamento de cirurgias, através da criação de vários possíveis escalonamentos e selecionando-se o melhor, ao longo de várias gerações, de forma eficiente e automática.

Estado da Arte

A utilização de Inteligência Artificial (IA) em diversas áreas como forma de auxiliar o ser humano tem sido uma prática cada vez mais frequente (Davenport & Kalakota, 2019).

Dada a mudança demográfica para uma população mais envelhecida e com diversas patologias, a constante demanda por profissionais de saúde levou ao desenvolvimento de sistemas informáticos e robotizados para dar resposta a esta necessidade (Klumpp et al, 2021), sendo que se verifica que a IA consegue desempenhar funções tão bem ou melhor que o ser humano (Esteva et al, 2021).

Tipos de IA

Existem diferentes tipos de IA relevantes para a área da saúde, tais como **Aprendizagem Automática**, que evoluiu para **Redes Neuronais** artificiais e **Aprendizagem Profunda**, a **Língua Natural** e a **Visão Computacional** (Davenport & Kalakota, 2019).

A **Aprendizagem Automática** é uma técnica que treina os modelos informáticos fornecendo-lhes grande volume de informação. Na saúde, é utilizada como forma de prever que tratamentos seriam mais eficazes em determinado paciente, considerando as suas características e contexto (Davenport & Kalakota, 2019).

Uma forma mais complexa de Aprendizagem Automática é designada de **Redes Neuronais**, pela sua semelhança com o processamento de informação no sistema nervoso humano (González-Pérez et al, 2024). Recorre-se a esta técnica como forma de determinar se um paciente irá adquirir uma doença específica, pois considera o peso das variáveis para calcular um resultado (Davenport & Kalakota, 2019).

A forma mais complexa de Aprendizagem Automática é a **Aprendizagem Profunda**, que analisa grandes volumes de dados e resolve problemas complexos (Guni et al, 2024). Uma aplicação muito comum é no reconhecimento de lesões potencialmente cancerígenas em

imagens de radiologia, pois consegue detetar características não visíveis ao olho humano (Davenport & Kalakota, 2019) e padrões muito complexos (Guni et al, 2024).

A **Língua Natural** é utilizada para reconhecimento de discurso, análise e tradução de textos, convertendo discurso livre em texto estruturado (Guni et al, 2024). No contexto da saúde, é usada para gerar e interpretar documentação clínica, como registos dos pacientes e relatórios de exames médicos (Davenport & Kalakota, 2019).

Por fim, a **Visão Computacional** permite que os algoritmos analisem imagens e vídeos de modo a interpretar e extrair padrões, o que permite melhorar os registos médicos baseados em imagens (Guni et al, 2024; Lindroth et al, 2024).

Apesar de não consistir exclusivamente em IA, também se verifica o uso de **robótica** em diferentes cenários. Estes robôs são capazes de erguer e reposicionar objetos, entregar recursos hospitalares, transferir pacientes (Davenport & Kalakota, 2019), diagnosticar e tratar condições e auxiliar na reabilitação dos utentes (Silvera-Tawil, 2024).

A sua aparência pode ser semelhante à humana e podem apresentar diferentes níveis de autonomia. Para além disto, conseguem interagir com materiais prejudiciais ao ser humano, realizar atividades repetitivas com grande precisão e são imunes ao desgaste psicológico (Joseph et al, 2018; Silvera-Tawil, 2024).

Existem ainda robôs cirúrgicos, desenvolvidos especificamente para auxiliar no decorrer de uma operação, com capacidades visuais aumentadas e grande precisão para fazer incisões e suturar (Butter, 2008).

Benefícios e Utilizações

A utilização de IA no meio hospitalar pode contribuir para a segurança dos pacientes, para a gestão de riscos e cumprimento de normas. Pode também assistir na deteção precoce de condições críticas, na descoberta de novos fármacos, na precisão de diagnósticos, na

monitorização contínua de pacientes e ainda na otimização da alocação de recursos (al Kuwaiti et al, 2023; Deo & Anjankar, 2023).

Identificação e avaliação de riscos

A IA é capaz de analisar informação de várias fontes, detetando padrões e anomalias, bem como fazer uma análise preditiva, de forma a identificar possíveis riscos e o seu impacto. É também capaz de automatizar o escalonamento dos processos logísticos, como cirurgias e distribuição de recursos, para evitar falhas ou atrasos (Božić, n.d.).

Segurança dos pacientes

Para os hospitais a segurança dos seus utentes é uma prioridade. Assim, recorrendo a IA, é possível analisar dados em tempo real para identificar sinais precoces de deterioração da saúde dos pacientes. Pode, também, identificar fatores de risco e anomalias em imagens médicas, corrigindo terapêuticas e diagnósticos (Božić, n.d.).

Segurança e Privacidade informáticas

Os algoritmos são capazes de vigiar continuamente o tráfego na rede para identificar possíveis ameaças, como *malware* e acessos não autorizados. É também possível analisar dados sensíveis e verificar a sua conformidade com a regulamentação em vigor (Božić, n.d.).

Suporte na tomada de decisão

Uma vez que fornece informações, a IA pode apoiar na tomada de decisão. Através da análise de dados, da identificação de riscos e do seu impacto, a IA pode sugerir estratégias e opções adequadas para diferentes cenários (Božić, n.d.).

Riscos e Implicações Éticas

Contudo, existem diversos obstáculos à utilização de IA nos cuidados de saúde.

Implicações nos profissionais de saúde

Uma grande preocupação com o uso de IA é o possível desemprego dos profissionais de saúde, que seriam substituídos por ferramentas informáticas e robôs. Embora alguma automação seja possível, esta seria apenas em trabalhos mais informáticos, e não nos que exigem contacto direto com os pacientes (Davenport & Kalakota, 2019).

No entanto, mesmo nos empregos mais informáticos, a substituição por IA seria de progressão lenta, pois os algoritmos apenas são capazes de diagnosticar e categorizar imagens. A sua interpretação e a aquisição de novos dados iriam exigir avanços significativos ao nível de IA, que não deverão acontecer nos próximos 20 anos (Davenport & Kalakota, 2019).

Transparência e explicabilidade

Atualmente, os algoritmos de IA ainda não são capazes de explicar as conclusões que obtêm (Davenport & Kalakota, 2019). Uma vez que é vital que as conclusões médicas sejam justificadas e confiáveis, isto apresenta-se como um importante obstáculo na sua utilização (Elendu et al, 2023).

Qualidade da informação e *Bias*

Os algoritmos desenvolvidos podem perpetuar *bias* existentes, devido ao seu treino com informações mais centradas numa determinada população e falta de representatividade. Isto pode levar a conclusões erradas ou discriminação nos diagnósticos e tratamentos realizados por IA (Elendu et al, 2023).

Para além disto, por vezes são geradas informações sem fundamento, o que pode levar a conclusões erradas (Arshad et al, 2023).

Validade temporal da informação

Uma vez que o treino da IA é feito com dados já existentes, estes podem estar desatualizados e estudos mais recentes podem não ser ainda conhecidos pelos algoritmos. Isto é particularmente relevante na área da saúde, onde são feitos progressos regularmente (Arshad et al, 2023).

Desafios na implementação

Implementar IA nos sistemas hospitalares já existentes requer um elevado investimento na infraestrutura informática e no treinamento dos profissionais, o que pode apresentar custos financeiros e de tempo (Božić, n.d.).

Soluções e mitigação de riscos

Para reduzir os riscos associados ao uso de IA na área da saúde, podem ser implementadas as seguintes práticas.

Verificação da qualidade da informação

A implementação de regulamentação que garanta a qualidade dos dados, em termos de assertividade e representatividade, juntamente com o fornecimento de dados diversos e a verificação de *outputs* gerados pela IA, podem contribuir para reduzir a discriminação (Božić, n.d.; Elendu et al, 2023).

Fomentação da interpretação e transparência

Através do desenvolvimento de modelos capazes de explicar a sua abordagem e desenvolver documentação que apoie as suas conclusões, é possível melhorar a confiança nos resultados produzidos pela IA (Božić, n.d.).

Medidas de privacidade e segurança

A implementação de medidas de cibersegurança, como acessos restritos e encriptação, protege os dados dos pacientes de acessos indevidos (Božić, n.d.).

Supervisão humana e ética

Deve ser utilizada a IA como forma de apoio à tomada de decisão, definindo os papéis e as responsabilidades dos profissionais de saúde. É aconselhado, ainda, desenvolver mecanismos de supervisão dos resultados obtidos, de modo a garantir que estes cumprem os padrões de qualidade e os regulamentos éticos (Božić, n.d.; Elendu et al, 2023).

Competências necessárias para utilizar IA

Os profissionais de saúde, de forma a utilizarem adequadamente ferramentas de IA, teriam de possuir uma compreensão básica de como estes algoritmos funcionam (Božić, n.d.).

Para além disto, os profissionais devem possuir fortes competências de pensamento crítico, de forma a validar as conclusões oferecidas pela IA. Isto é essencial para garantir que a tomada de decisão é correta e a mais adequada para o paciente sob cuidados (Božić, n.d.).

Uma vez que a IA está em constante evolução, os profissionais devem ser capazes de acompanhar esse desenvolvimento, adaptando-se às novas tecnologias e atualizando os seus conhecimentos (Božić, n.d.).

Atribuição de operações a salas de cirurgia

De forma a ser possível alocar cirurgias a várias salas, foi desenvolvido o predicado **allocate_surgeries.** Este recorre a vários outros predicados, que serão explicados de seguida.

```
allocate_surgeries(Date) :-
  free_agendas_for_all_rooms(Date, _),
  initialize_room_summary(Date),
  retractall(operation_assigned_to_room(_, Date, _)),
  sort_surgeries_by_time(SortedSurgeries),
  forall(member((SurgeryId, SurgeryTime), SortedSurgeries), (
    allocate_surgery_to_best_room(SurgeryId, SurgeryTime, Date)
  )).
```

Agenda das salas

O predicado **free_agendas_for_all_rooms** permite encontrar todos os intervalos de tempo disponível para cada uma das salas, a partir do horário ocupado, recorrendo ao predicado **free_agenda0,** abordado no relatório anterior.

```
free_agendas_for_all_rooms(Date, FreeAgendas) :-
        findall((Room, FreeSlots),
        (agenda_operation_room(Room, Date, Agenda),
         free_agenda0(Agenda, FreeSlots),
         (retractall(room_free_agenda(Room, Date, _)),
         assertz(room_free_agenda(Room, Date, FreeSlots)))
         ), FreeAgendas).
```

Ocupação das salas antes das operações serem distribuídas

O predicado **initialize_room_summary** é usado para criar factos dinâmicos para cada sala, com o tempo total de ocupação de cada uma, usando o predicado **calculate_sum_from_agenda**, apresentado de seguida.

Isto é necessário para que ao selecionar a melhor sala, seja possível calcular se a percentagem da sua ocupação é superior ao limite máximo de 80%.

```
initialize_room_summary(Date) :-
  retractall(room_summary(_, _, _)),
  findall(Room, room_free_agenda(Room, Date, _), Rooms),
  forall(member(Room, Rooms),
  (calculate_sum_from_agenda(Room, Date, SumOfOperations),
  assertz(room_summary(Room, 1440, SumOfOperations)))).
```

Por sua vez, o predicado **calculate_sum_from_agenda** permite calcular o tempo total de ocupação de uma sala, com base na agenda da sala com as operações já marcadas. O predicado **sum_agenda_slots** é responsável por realizar os cálculos necessários.

```
calculate_sum_from_agenda(Room, Date, SumOfOperations) :-
  agenda_operation_room(Room, Date, Agenda),
  sum_agenda_slots(Agenda, SumOfOperations).


sum_agenda_slots([], 0).
sum_agenda_slots([(Start, End, _) | Rest], TotalTime) :-
  SlotTime is End - Start,
  sum_agenda_slots(Rest, RestTotal),
  TotalTime is SlotTime + RestTotal.
```

Operações a realizar

O predicado **sort_surgeries_by_time** permite ordenar as operações por ordem decrescente de duração. Isto é necessário para que o tempo total ocupado de cada sala seja o mais equilibrado possível, pois atribui-se a cirurgia mais longa à sala menos ocupada.

```
sort_surgeries_by_time(SortedSurgeries) :-
        findall((Id, TotalTime), surgery_with_time(Id, TotalTime), SurgeryList),
        sort(2, @>=, SurgeryList, SortedSurgeries).
```

Por sua vez, o predicado **surgery_with_time** permite saber o tempo que cada operação demora, com base no seu tipo, usando o predicado **surgery_total_time,** que calcula o tempo total de cada tipo de operação.

```
surgery_with_time(Id, TotalTime) :-
       surgery_id(Id, Type),
       surgery_total_time(Type, TotalTime).


surgery_total_time(Type, TotalTime) :-
          surgery_type(Type, Prep, Duration, Recovery),
           TotalTime is Prep + Duration + Recovery.
```

Atribuição das operações

O predicado **allocate_surgery_to_best_room** é utilizado para encontrar a melhor sala para atribuir a cirurgia, tendo em conta o limite máximo de ocupação da sala.

Depois de possuir todos os **room_summaries,** é calculada a nova percentagem de ocupação da sala com a adição do tempo da operação, para verificar se o tempo de ocupação não ultrapassou o limite de 80%.

É possível alocar a cirurgia às salas que não ultrapassarem este limite, sendo estas ordenadas por ordem crescente de percentagem de tempo ocupado. Por fim, a sala que ficar no início desta ordenação vai ser a sala a que vai ser atribuída a operação.

```
allocate_surgery_to_best_room(SurgeryId, SurgeryTime, Date) :-
  findall((Room, TotalTime, SumOfOp), room_summary(Room, TotalTime, SumOfOp), RoomSummaries),
  findall((Room, NewRatio), (
    member((Room, TotalTime, SumOfOp), RoomSummaries),
    NewSum is SumOfOp + SurgeryTime,
    NewRatio is NewSum / TotalTime,
    NewRatio =< 0.8
  ), ValidRooms),
  sort(2, @=<, ValidRooms, [ (BestRoom, _) | _ ]),
  update_room_summary(BestRoom, SurgeryTime, SurgeryId, Date).
```

O predicado anterior recorre ao predicado **update_room_summary,** usado para eliminar e criar novamente o facto **room_summary** que contém o ID da sala e o tempo total de ocupação atualizado. Isto é necessário para manter o tempo total atualizado sempre que uma operação é atribuída a uma determinada sala.

```
update_room_summary(Room, SurgeryTime, SurgeryId, Date) :-
  retract(room_summary(Room, TotalTime, SumOfOp)),
  NewSum is SumOfOp + SurgeryTime,
  assertz(room_summary(Room, TotalTime, NewSum)),
  assertz(operation_assigned_to_room(Room, Date, SurgeryId)).
```

Adaptação do Algoritmo Genético para o escalonamento de cirurgias

O algoritmo genético fornecido é utilizado para gerar o escalonamento de cirurgias, existindo, no entanto, algumas alterações que foram efetuadas, relativamente à consideração da sala e do dia para os quais se quer agendar cirurgias.

Inicialmente, o predicado **start** agenda as cirurgias para cada sala, de forma recursiva, recorrendo ao predicado **comeca_aqui**. De seguida, procede à atualização das agendas das salas e do staff que participa nas cirurgias agendadas.

```
start([],_,[]).
start([Room|Rest], Day, [Result|Res]):-
  comeca_aqui(Room, Day),
  findall(agenda_staff1(D, Day, AG), agenda_staff1(D, Day, AG), LAG),
  agenda_operation_room1(Room, Day, A),
  assert(resultPerRoom(Room, A, LAG)),
  retract(agenda_operation_room2(Room,_,_)),
  retract(agenda_operation_room1(Room,AR2,AR3)),
  assertz(agenda_operation_room2(Room,AR2,AR3)),
  retractall(agenda_staff2(_,_,_)),
  findall(_, (agenda_staff1(AS1,AS2,AS3), assertz(agenda_staff2(AS1,AS2,AS3))),_),
  retractall(tarefas(_)),
  start(Rest, Day, Res).
```

Por sua vez, o predicado **começa_aqui** inicia com a eliminação de dados já existentes e inicializa os parâmetros com os valores definidos. De seguida, é criada uma população inicial de indivíduos (**gera_populacao_schedule**), que são avaliados (**avalia_populacao_schedule**) e ordenados conforme o seu fitness (**ordena_populacao**).

Este processo repete-se ao longo do número de gerações definidas (**gera_geracao_schedule**), sendo que no fim obtém-se a melhor geração e desta o melhor indivíduo, portanto, o melhor escalonamento de cirurgias para um determinado dia e sala.

Por fim, verifica se todas as cirurgias podem ser agendadas, e atualiza as agendas da sala e staff, recorrendo a predicados explicados no relatório anterior.

```
comeca_aqui(Room,Day):-
  retractall(day(_)),
  retractall(room(_)),
  retractall(geracoes(_)),
  retractall(populacao(_)),
  retractall(prob_cruzamento(_)),
  retractall(prob_mutacao(_)),!,
  asserta(day(Day)),
  asserta(room(Room)),
  asserta(geracoes(5)),
  asserta(populacao(4)),
  asserta(prob_cruzamento(0.70)),
  asserta(prob_mutacao(0.01)),!,
  gera_populacao_schedule(Pop),
  length(Pop, NPop),
  write('Day='),write(Day),nl,
  write('Pop='),write(NPop),write(':'),write(Pop),nl,    !,
  avalia_populacao_schedule(Room,Day,Pop,PopAv),
  write('PopAv='),write(PopAv),nl,
  ordena_populacao(PopAv,PopOrd),
  geracoes(NG),    !,
  gera_geracao_schedule(Room,Day,0,NG,PopOrd),!,
  latest_gen_indivuduals(PPP),
```

```
  reverse(PPP, [P*V|R]),
  (availability_all_surgeries1(Room, Day, P); true),
  agenda_operation_room1(Room, Day, A),
  write('agenda_operation_room1='), write(A), nl,
  findall(agenda_staff1(D, Day, AG), agenda_staff1(D, Day, AG), LAG),
  write('L_agenda_staff1='), write(LAG), nl.
```

Gerar população inicial

Relativamente aos novos predicados utilizados, o predicado **gera_populacao_schedule** cria a população inicial para uma determinada sala, com base no tamanho definido para essa população e dependendo de quantas operações precisam de marcação.

Este predicado recorre ao **gera_individuo_schedule** que cria um indivíduo, representando um possível agendamento de operações. Este é composto pelas diferentes cirurgias selecionadas de forma aleatória, garantindo diversidade entre indivíduos.

```
gera_populacao_schedule(Pop):-
  populacao(TamPop),
  day(Dr),
  room(Rr),
  findall(OpCode, operation_assigned_to_room(Rr, Dr, OpCode), ListaTarefas),
  length(ListaTarefas, NumT),
  assert(tarefas(NumT)),
  gera_populacao_schedule(TamPop,ListaTarefas,NumT,Pop).


gera_populacao_schedule(0,_,_,[]):-!.
gera_populacao_schedule(TamPop,ListaTarefas,NumT,[Ind|Resto]):-
  TamPop1 is TamPop-1,!,
  gera_populacao_schedule(TamPop1,ListaTarefas,NumT,Resto),!,
```

```
  gera_individuo_schedule(ListaTarefas,NumT,Ind),
  not(member(Ind,Resto)).


gera_individuo_schedule([],_,[]):-!.
gera_individuo_schedule([G],1,[G]):-!.
gera_individuo_schedule(ListaTarefas,NumT,[G|Resto]):-
  NumTemp is NumT + 1,
  randseq(1, NumT, [N|R]),
  retira(N,ListaTarefas,G,NovaLista),!,  %write('     gera_individuo_schedule: retira~_next
'),     nl,
  NumT1 is NumT-1,
  gera_individuo_schedule(NovaLista,NumT1,Resto).
```

Avaliar população

O predicado **avalia_populacao_schedule** avalia toda a população de forma a atribuir um valor de fitness a cada individuo, para uma determinada sala e dia. Este, por sua vez, recorre ao **avalia_schedule** para avaliar o fitness de um único individuo.

Este último consulta as agendas do staff de modo a ver as suas disponibilidades e caso todas as cirurgias possam ser agendadas, calcula o tempo final da última cirurgia e retira-o ao tempo total de um dia. Assim, o fitness é definido como o tempo desde que a última cirurgia acaba até ao fim do dia, sendo que quanto mais cedo acabar, maior o fitness.

```
avalia_populacao_schedule(_,_,[],[]).
avalia_populacao_schedule(Room,Day,[Ind|Resto],[Ind*V|Resto1]):-
 avalia_schedule(Ind,Room,Day,V),
 avalia_populacao_schedule(Room,Day, Resto,Resto1).


avalia_schedule(Seq,Room,Day,V):-
```

```
    retractall(agenda_staff1(_,_,_)),
    retractall(agenda_operation_room1(Room,_,_)),
    retractall(availability(_,_,_)),
    findall(_,(agenda_staff2(D,Day,Agenda),assertz(agenda_staff1(D,Day,Agenda))),_),
     agenda_operation_room2(Room,Day,Agenda),assert(agenda_operation_room1(Roo
m,Day,Agenda)),
     findall(_,(agenda_staff1(D,Day,L),free_agenda0(L,LFA),adapt_timetable(D,Day,LFA,L
FA2),assertz(availability(D,Day,LFA2))),_),
  (
    (
      availability_all_surgeries1(Seq, Room, Day),
      agenda_operation_room1(Room, Day, AgendaNova),!,
      reverse(AgendaNova, AgendaR),
      evaluate_final_time(AgendaR,Seq,Tfinal),
      V is 1441 - Tfinal
    ) ;
    (
      V is 0
    )
  ).
```

Ordenar população

O predicado **ordena_população** não sofreu qualquer alteração, pelo que manteve as funcionalidades fornecidas. Assim, recorre ao algoritmo de ordenação *Bubble Sort* para ordenar os indivíduos da população de forma crescente, com base no seu *fitness*, ou seja, ordena os possíveis escalonamentos com base no tempo final da última cirurgia, ficando em primeiro lugar o escalonamento cuja última cirurgia acaba mais tarde.

O algoritmo principal **comeca_aqui** irá depois reverter esta lista, de forma a obter em primeiro lugar o escalonamento em que a última cirurgia acaba mais cedo, que é o cenário procurado.

```
ordena_populacao(PopAv,PopAvOrd):-
   bsort(PopAv,PopAvOrd).

bsort([X],[X]):-!.
bsort([X|Xs],Ys):-
   bsort(Xs,Zs),
   btroca([X|Zs],Ys).

btroca([X],[X]):-!.

btroca([X*VX,Y*VY|L1],[Y*VY|L2]):-
   VX>VY,!,
   btroca([X*VX|L1],L2).

btroca([X|L1],[X|L2]):-btroca(L1,L2).
```

Criar novas gerações

Por sua vez, o predicado **gera_geracao_schedule** é responsável por criar várias gerações, evoluindo a população inicial. Através de cruzamento (*crossover*), são gerados novos indivíduos, aos quais são aplicadas mutações para aumentar a diversidade. Estes depois são avaliados, selecionados os que apresentam melhor fitness e ordenados, repetindo-se o processo até ao número de gerações desejado.

```
gera_geracao_schedule(_,_,G,G,Pop):-!,
   ordena_populacao(Pop,NPopOrd),
```

```
  retractall(latest_gen_indivuduals(_)),
  asserta(latest_gen_indivuduals(NPopOrd)).

gera_geracao_schedule(Room,Day,N,G,Pop):-
  write('Geracao '), N2 is N+1 ,write(N2), write(':'), nl,
  cruzamento(Pop,NPop1),
  mutacao(NPop1,NPop),
  avalia_populacao_schedule(Room,Day,NPop,NPopAv),
  append(Pop, NPopAv, NovaPop),
  pop_nova_geracao(NovaPop, PopNova),
  N1 is N+1,
  retractall(latest_gen_indivuduals(_)),
  ordena_populacao(PopNova,NPopOrd),
  asserta(latest_gen_indivuduals(NPopOrd)),
  gera_geracao_schedule(Room,Day,N1,G,PopNova).
```

Aleatoriedade no cruzamento entre indivíduos da população do Algoritmo Genético

Ao criar indivíduos, estes são gerados de forma aleatória, utilizando o **randseq**. Este garante que a escolha de cirurgias a agendar seja aleatória, o que permite que cada indivíduo tenha uma combinação diferente de agendamento.

Para além disto, ao garantir not(member), é possível evitar que a população possua indivíduos iguais.

```
gera_populacao_schedule(0,_,_,[]):-!.
gera_populacao_schedule(TamPop,ListaTarefas,NumT,[Ind|Resto]):-
   TamPop1 is TamPop-1,!,
   gera_populacao_schedule(TamPop1,ListaTarefas,NumT,Resto),!,
   gera_individuo_schedule(ListaTarefas,NumT,Ind),
   not(member(Ind,Resto)).

gera_individuo_schedule([],_,[]):-!.
gera_individuo_schedule([G],1,[G]):-!.
gera_individuo_schedule(ListaTarefas,NumT,[G|Resto]):-
   NumTemp is NumT + 1, % To use with random
   randseq(1, NumT, [N|R]),
   retira(N,ListaTarefas,G,NovaLista),!,
   NumT1 is NumT-1,
   gera_individuo_schedule(NovaLista,NumT1,Resto).
```

Para além disso, quando se cria uma geração são selecionados indivíduos de forma aleatória, para depois serem comparados e selecionados os que apresentam melhor fitness, como mostra o predicado **torneio**.

```
torneio(Populacao, TorneioSelecionado):-
   torneio_seleciona(Populacao, TorneioSelecionado).


torneio_seleciona([], _).
torneio_seleciona([Last], []).
torneio_seleciona(Populacao, [Selecionado|RestoSelecionados]) :-
   select_aleatorio(Populacao, Ind1*V1, PopResto1),
   select_aleatorio(PopResto1, Ind2*V2, PopResto2),
   (
      V1 >= V2, Selecionado = Ind1*V1;
      V2 > V1, Selecionado = Ind2*V2
   ),
   torneio_seleciona(PopResto2, RestoSelecionados).


select_aleatorio(Populacao, Ind*V, PopResto) :-
   random_permutation(Populacao, [Ind*V|PopResto]).
```

Seleção da nova geração da população do Algoritmo Genético

A nova geração é gerada pelo predicado **pop_nova_geracao**, que começa por selecionar os melhores indivíduos gerados a partir da população antiga, de modo a preservar a qualidade da população.

De seguida, os restantes indivíduos da população antiga passam pelo **torneio**, que compara pares aleatórios de indivíduos e escolhe o melhor de cada par.

Assim, a nova geração é constituída pelos melhores indivíduos e alguns indivíduos aleatórios, que apresentam bom fitness, de modo a garantir qualidade mas também diversidade.

```
pop_nova_geracao(PopAntiga, Resultado):-
   elite(E),
   populacao(NPop),
   nl,NrElem1 is E/100,
 NrElem is floor(NrElem1 * NPop),%write('pop_nova_geracao:NrElem: '),write(NrElem),nl,
   ordena_populacao(PopAntiga, PopAntiga1),
   reverse(PopAntiga1, PopAntiga2),
   sublist(PopAntiga2, NrElem, Elite, RestoAntiga),
   torneio(RestoAntiga, Resultado_Torneio),
   NrTorneiro is NPop - NrElem,
   sublist(Resultado_Torneio, NrTorneiro, B, Fim),
   append(Elite, B, Resultado).
```

Parametrização da condição de término do Algoritmo Genético

Número máximo de gerações

De forma ao algoritmo genético parar de gerar novas soluções, é considerado o número máximo de gerações desejadas, definido no predicado inicial **comeca_aqui**.

```
comeca_aqui(Room,Day):-
   ...
   asserta(geracoes(5)),
   ...
   geracoes(NG),!,
   gera_geracao_schedule(Room,Day,0,NG,PopOrd),!,
   ...
```

No predicado **gera_geracao_schedule**, responsável por criar gerações, é atribuído um valor a cada geração criada, que vai sendo incrementado. Quando o valor da geração for equivalente ao número de gerações desejadas, este predicado acaba a sua execução e obtém-se a geração final.

```
gera_geracao_schedule(_,_,G,G,Pop):-!,
   ordena_populacao(Pop,NPopOrd),
   retractall(latest_gen_indivuduals(_)),
   asserta(latest_gen_indivuduals(NPopOrd)).


gera_geracao_schedule(Room,Day,N,G,Pop):-
   write('Geracao '), N2 is N+1 ,write(N2), write(':'), nl,
   cruzamento(Pop,NPop1),
   mutacao(NPop1,NPop),
```

```
  avalia_populacao_schedule(Room,Day,NPop,NPopAv),
  append(Pop, NPopAv, NovaPop),
  pop_nova_geracao(NovaPop, PopNova),
  N1 is N+1,
  retractall(latest_gen_indivuduals(_)),
  ordena_populacao(PopNova,NPopOrd),
  asserta(latest_gen_indivuduals(NPopOrd)),
  gera_geracao_schedule(Room,Day,N1,G,PopNova).
```

Tempo máximo de execução

Para além disso, é considerado o tempo máximo de execução, definido no predicado **full**.

```
full(Day):-
  assert(time_limit(3)),
  get_time(StartTime),
  ...
```

No predicado **gera_geracao_schedule**, se a cada criação de uma nova geração o tempo decorrido ultrapassar o tempo máximo definido, o algoritmo interrompe essa criação e continua a sua lógica com as gerações já obtidas.

```
gera_geracao_schedule(_,_,G,G,Pop):-!,
  ordena_populacao(Pop,NPopOrd),
  retractall(latest_gen_indivuduals(_)),
  asserta(latest_gen_indivuduals(NPopOrd)).


gera_geracao_schedule(Room,Day,N,G,Pop):-
  (
    get_time(CurrentTime),
```

```
    time_limit(A),
    start_time(StartTime),
    Dif is CurrentTime - StartTime,
    Dif > A, !
  )
  ;
  (
    cruzamento(Pop,NPop1),
    mutacao(NPop1,NPop),
    avalia_populacao_schedule(Room,Day,NPop,NPopAv),
    append(Pop, NPopAv, NovaPop),
    pop_nova_geracao(NovaPop, PopNova),
    N1 is N+1,
    retractall(latest_gen_indivuduals(_)),
    ordena_populacao(PopNova,NPopOrd),
    asserta(latest_gen_indivuduals(NPopOrd)),
    gera_geracao_schedule(Room,Day,N1,G,PopNova)
  ).
```

Escolha do Algoritmo a usar

De forma a realizar o escalonamento de cirurgias para uma determinada sala e dia, foram desenvolvidos diferentes algoritmos, nomeadamente:

- ***brute force***, que gera todas as combinações possíveis e escolhe a melhor
- **heurísticas**, que vão construindo um caminho satisfatório medida que o percorrem
- **algoritmo genético**, que cruza e seleciona indivíduos para obter soluções de alta qualidade

A escolha do algoritmo a utilizar, para diferentes problemas, depende da sua dimensão.

Se for apenas considerada uma sala e existirem até seis operações a agendar, inclusive, será utilizado ***brute force,*** uma vez que o número de soluções não é muito extenso.

Por sua vez, for necessário alocar mais de seis cirurgias na mesma sala é utilizada **heurística**, que embora não garanta uma solução ótima, é uma alternativa mais eficiente.

Caso o problema englobe várias salas de operação seria utilizado o **algoritmo genético**, capaz de incluir diversas salas e explorar diferentes combinações de forma eficiente.

O predicado **full**, após obter as salas e staff disponíveis e saber as cirurgias que necessitam de agendamento, decide que algoritmo usar e exibe os resultados do agendamento no fim.

```
full(Day):-
  retractall(resultPerRoom(_,_,_)),
  retractall(operation_assigned_to_room(_,_,_)),
  retractall(agenda_operation_room2(_,_,_)),
  retractall(agenda_staff2(_,_,_)),
  allocate_surgeries(Day),
  findall(Room, agenda_operation_room(Room,B,C), LRoom),
  findall(_, (agenda_staff(AS1,AS2,AS3), assertz(agenda_staff2(AS1,AS2,AS3))),_),
```

```
 findall(_,(agenda_operation_room(AR1,AR2,AR3),
assertz(agenda_operation_room2(AR1,AR2,AR3))),_),
  length(LRoom, LLRoom),
  (
    (
      LLRoom = 1,!,
      (
        LRoom = [Room1],
        findall(sID, surgery_id(sID, _), LsID),
        length(LsID, NLsID),
        (
          NLsID =< 6,
          brute_force(Room1, Day),!
        );
        (
          obtain_heuristic_solution(Room1, Day)
        )
      )
    )
    ;
    start(LRoom, Day, Result)
  ),
  findall(resultPerRoom(Room, A, LAG), resultPerRoom(Room, A, LAG), L),
  nl,nl,nl,write('Final: '), write(L), nl.
```

Conclusão

Em suma, a integração de IA no meio hospitalar pode melhorar a segurança dos pacientes e a eficiência e qualidade cirúrgica. Apesar de existirem desafios e riscos, a utilização efetiva e segura de IA permite aos hospitais prestarem melhores cuidados aos seus pacientes e uma melhor gestão de riscos (Božić, n.d.).

Uma das formas de implementar IA no meio hospitalar é através de algoritmos genéticos, como aquele que foi apresentado para realizar o escalonamento de cirurgias, de forma automática e eficiente.

Assim, este considera uma amostra inicial de possíveis agendamentos de cirurgias e através de cruzamentos e mutações vão sendo geradas novas possibilidades, até se encontrar o melhor horário para cada sala, respeitando a disponibilidade da sala e dos profissionais de saúde.

Apesar do relatório anterior considerar o uso de algoritmos de *brute force* e heurísticas para realizar este escalonamento, o presente relatório mostrou que o algoritmo genético é mais adequado quando se possui um grande volume de resultados.

Para além disto, este trabalho permitiu também considerar diversas salas de operação para sofrerem agendamento, sem afetar a eficiência da geração de horários.

Deste modo, é possível concluir que a IA é uma ferramenta essencial no agendamento de cirurgias, devendo continuar a ser utilizada e a evoluir.

Referências

al Kuwaiti, A., Nazer, K., Al-Reedy, A., Al-Shehri, S., Al-Muhanna, A., Subbarayalu, A. V., al Muhanna, D., & Al-Muhanna, F. A. (2023). A Review of the Role of Artificial Intelligence in Healthcare. In *Journal of Personalized Medicine* (Vol. 13, Issue 6). MDPI. https://doi.org/10.3390/jpm13060951

Arshad, H. B., Butt, S. A., Khan, S. U., Javed, Z., & Nasir, K. (2023). ChatGPT and Artificial Intelligence in Hospital Level Research: Potential, Precautions, and Prospects. *Methodist DeBakey Cardiovascular Journal*, *19*(5), 77–84. https://doi.org/10.14797/mdcvj.1290

Božić, V. (n.d.). Integrated Risk Management and Artificial Intelligence in Hospital. In *Journal of AI* (Vol. 63, Issue 7). APA.

Butter, M., Rensma, A., van Boxsel, J., Kalisingh, S., Schoone, M., Leis, M., Gelderblom, G., J., Cremers, G., de Wilt, M., Kortekaas, W., Thielmann, A. & Cuhls, K., Sachinopoulou, A. & Korhonen, I. (2008). *Robotics for Healthcare Final Report*.

Davenport, T., & Kalakota, R. (2019). DIGITAL TECHNOLOGY The potential for artificial intelligence in healthcare. In *Future Healthcare Journal* (Vol. 6, Issue 2).

Deo, N., & Anjankar, A. (2023). Artificial Intelligence With Robotics in Healthcare: A Narrative Review of Its Viability in India. *Cureus*. https://doi.org/10.7759/cureus.39416

Elendu, C., Amaechi, D. C., Elendu, T. C., Jingwa, K. A., Okoye, O. K., John Okah, M., Ladele, J. A., Farah, A. H., & Alimi, H. A. (2023). Ethical implications of AI and robotics in healthcare: A review. In *Medicine (United States)* (Vol. 102, Issue 50, p. E36671). Lippincott Williams and Wilkins. https://doi.org/10.1097/MD.0000000000036671

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J., & Socher, R. (2021). Deep learning-enabled medical computer vision. In *npj Digital Medicine* (Vol. 4, Issue 1). Nature Research. https://doi.org/10.1038/s41746-020-00376-2

González-Pérez, Y., Montero Delgado, A., & Martinez Sesmero, J. M. (2024). Acercando la inteligencia artificial a los servicios de farmacia hospitalaria. *Farmacia Hospitalaria*, *48*, S35–S44. https://doi.org/10.1016/j.farma.2024.02.007

Guni, A., Varma, P., Zhang, J., Fehervari, M., & Ashrafian, H. (2024). Artificial Intelligence in Surgery: The Future is Now. *European Surgical Research*. https://doi.org/10.1159/000536393

Joseph, A., Christian, B., Abiodun, A. A., & Oyawale, F. (2018). A review on humanoid robotics in healthcare. *MATEC Web of Conferences*, *153*. https://doi.org/10.1051/matecconf/201815302004

Klumpp, M., Hintze, M., Immonen, M., Ródenas-Rigla, F., Pilati, F., Aparicio-Martínez, F., Çelebi, D., Liebig, T., Jirstrand, M., Urbann, O., Hedman, M., Lipponen, J. A., Bicciato, S., Radan, A. P., Valdivieso, B., Thronicke, W., Gunopulos, D., & Delgado-Gonzalo, R. (2021). Artificial intelligence for hospital health care: Application cases and answers to challenges in european hospitals. *Healthcare (Switzerland)*, *9*(8). https://doi.org/10.3390/healthcare9080961

Lindroth, H., Nalaie, K., Raghu, R., Ayala, I. N., Busch, C., Bhattacharyya, A., Moreno Franco, P., Diedrich, D. A., Pickering, B. W., & Herasevich, V. (2024). Applied Artificial Intelligence in Healthcare: A Review of Computer Vision Technology Application in Hospital Settings. In *Journal of Imaging* (Vol. 10, Issue 4). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/jimaging10040081

Silvera-Tawil, D. (2024). Robotics in Healthcare: A Survey. *SN Computer Science*, *5*(1). https://doi.org/10.1007/s42979-023-02551-0