Relatório de Algoritmia Avançada

Turma 3DF _ Grupo 025

1200625 – Sérgio André Oliveira Lopes
1200628 – Tiago Francisco da Silva Freitas
1201386 – Rita Ariana de Castro Ribeiro e Pereira Sobral
1202016 – Vasco Filipe Amorim de Azevedo

Data: 08/01/2023

Índice

Introdução

Neste Sprint foi-nos pedido que implementássemos um algoritmo genético de forma a obter uma solução ótima para a realização de entregas para um dado dia, visto que no Sprint anterior, concluímos que não é viável a partir de uma certa quantidade de entregas gerar todas as possibilidades e determinar qual a melhor viagem.

O algoritmo genético é utilizado para encontrar soluções em problemas complexos de otimização onde é inviável encontrar a melhor solução de forma exaustiva, como é o caso do nosso problema. É um método inspirado na teoria da evolução biológica, em que o cruzamento de gerações sucessivas gerará uma solução próxima da melhor solução.

Foi-nos então fornecido um algoritmo genético que resolvia o problema de escalonamento de tarefas e solicitado a adaptação para o contexto do nosso projeto, o planeamento de rotas. Foram também sugeridas algumas melhorias a realizar de maneira que o algoritmo fosse mais eficaz e eficiente e que gerasse mais vezes uma solução próxima da melhor.

O algoritmo genético é uma melhor abordagem quando comparada com as heurísticas desenvolvidas no Sprint anterior, pois este permite o planeamento de uma frota dado um conjunto de entregas a serem efetuadas num dado dia.

Criação da população inicial do Algoritmo Genético (AG)

A população inicial do Algoritmo Genético é gerada com o auxílio do predicado gera_populacao/3, onde iremos numa fase inicial obter todos os armazéns onde existem entregas a ser efetuadas no dia pretendido, utilizando o predicado findall/3.

```
gera_populacao(Pop,Data,Camiao):-
  populacao(TamPop),
  entregas(NumE),
  findall(ArmEntrega,entrega(_,Data,_,ArmEntrega,_,_),ListaArmazens),
  gera_populacao(TamPop,ListaArmazens,NumE,Pop,Data,Camiao).

gera_populacao(0,_,_,[],_,_):-!.

gera_populacao(TamPop,ListaArmazens,NumE,[Ind|Resto],Data,Camiao):-
  TamPop1 is TamPop-1,
  gera_populacao(TamPop1,ListaArmazens,NumE,Resto,Data,Camiao),
  ((TamPop1 == 0,!, melhorViagemArmazemMaisProximo(Data, Camiao, Ind, _));
  ((TamPop1 == 1,!, melhorViagemMelhorRelacao(Data, Camiao, Ind, _));
  (gera_individuo(ListaArmazens,NumE,Ind)))),
  not(member(Ind,Resto)).
```

Figura 1 - Predicados utilizados no processo de criação da população inicial

Depois de sabermos todos os armazéns que temos de percorrer durante a viagem podemos então proceder à criação de todos os indivíduos da população inicial.

De maneira a incluir boas soluções na população inicial, dois dos indivíduos da população inicial são criados a partir de duas heurísticas desenvolvidas no Sprint anterior, a heurística da melhor relação entre tempo e massa e a heurística do armazém mais próximo, estas heurísticas foram as escolhidas devido a serem as que obtiam os melhores resultados.

Os restantes indivíduos da população serão gerados pelo predicado gera_individuo/3 fornecido no início do Sprint.

Aleatoriedade no cruzamento entre indivíduos da população

O cruzamento entre os indivíduos da população é efetuado com recurso ao operador de cruzamento com manutenção de ordem, Order Crossover, visto que este operador é útil quando os indivíduos são representados por sequências de elementos, como é o caso do nosso problema. No algoritmo genético fornecido o cruzamento era realizado sobre os indivíduos progenitores sucessivos 2 a 2 da população, o que poderia ser uma limitação.

De maneira a gerar gerações futuras melhores, uma das melhorias implementadas no algoritmo foi a implementação da aleatoriedade no cruzamento. Para evitar que a sequência de cruzamentos fosse sempre realizada sobre os indivíduos sucessivos 2 a 2 da população, e sem efetuar alterações nos predicados fornecidos que realizam o cruzamento, optamos por antes de realizar o cruzamento efetuar uma permutação aleatória entre os indivíduos da população, e assim depois já poderemos cruzar os progenitores sucessivos 2 a 2.

A permutação aleatória entre os elementos da lista foi realizada utilizando o predicado random_permutation/2. Este predicado retorna, no segundo argumento, uma versão permutada da lista presente no primeiro argumento.

```
gera_geracao(N,G,Pop,TempoInicial,TempoMaximo,0,GeracoesIguaisAnt,Estabilizacao,MelhorViagem,CamioesNecessarios,EntregasPorCamiao):-
  random_permutation(Pop,PopAleatoria),
  cruzamento(PopAleatoria,NPop1),
  mutacao(NPop1,NPop),
```

Figura 2 – Predicados utilizados para garantir a aleatoriedade no cruzamento

Seleção da nova geração da população

O algoritmo genético inicialmente fornecido substituía a população corrente pelos seus descentes, não existindo assim a garantia de que o melhor individuo passa-se para a geração seguinte, podendo então perder-se a melhor solução obtida.

Uma das melhorias então implementadas foi o método de seleção da nova geração da população, onde pretendemos que não fossem privilegiados apenas os melhores indivíduos dando alguma probabilidade de os indivíduos com pior avaliação também conseguirem passar para a próxima população, mas onde também garantimos que os dois melhores indivíduos entre a população corrente e a população gerada passam para a população seguinte.

De maneira a obter os melhores indivíduos, juntamos numa lista os elementos da população anterior com os seus descendentes, eliminando os elementos duplicados caso estes existam, e procedemos à ordenação da lista com todos elementos de acordo com os tempos das viagens correspondestes, onde depois retiramos os 2 melhores indivíduos.

```
append(Pop, NPopAv, Populacao),
sort(Populacao, Aux),
ordena_populacao(Aux,NPopOrd),
obter_melhores(NPopOrd,2,Melhores,Restantes),
```

Figura 3 – Predicados para obter os melhores indivíduos

O processo explicado anteriormente é realizado para todas as gerações e utiliza os 4 predicados representados na Figura 3.

```
obter_melhores([H|NPopOrd], 0, [],[H|NPopOrd]).
obter_melhores([Ind|NPopOrd],P,[Ind|Melhores],Restantes):-
  P1 is P-1,
  obter_melhores(NPopOrd,P1,Melhores,Restantes).
```

Figura 4 – Predicado obter_melhores/4

O predicado obter_melhores/4 retorna no terceiro argumento os dois melhores elementos e no quarto argumento os restantes elementos da população anterior e dos seus descendentes.

A seleção dos restantes indivíduos irá ser efetuada por um método que não é puramente elitista. Neste método cada indivíduo terá associado o produto do tempo da viagem por um número aleatório entre 0 e 1, sendo a nova lista ordenada por ordem crescente de acordo com o valor do produto referido anteriormente. Dessa lista os N-2 indivíduos passam para a geração seguinte, sendo N a dimensão da população.

```
probabilidade_restantes(Restantes,ProbRestantes),
ordena_populacao_probabilidade(ProbRestantes,ProbRestantesOrd),
populacao(TamPop),
ElementosEmFalta is TamPop-2,
retirar_elementos_extra(ProbRestantesOrd,ElementosEmFalta,ListaMelhores),
append(Melhores,ListaMelhores,ProxGeracao),
ordena_populacao(ProxGeracao,ProxGeracaoOrd),
```

Figura 5 – Predicados para obter os restantes elementos da população

O predicado probabilidade_restantes/2 retorna no segundo argumento uma lista que irá conter a probabilidade calculada associada a cada indivíduo.

```
probabilidade_restantes([],[]):-!.
probabilidade_restantes([Ind*Tempo|Restantes],[Ind*Tempo*Prob|ListaProb]):-
  probabilidade_restantes(Restantes,ListaProb),
  random(0.0,1.0,NumAl), Prob is NumAl * Tempo.
```

Figura 6 – Predicado probabilidade_restantes/2

O predicado ordena_populacao_probabilidade/2 é uma adaptação do predicado ordena_populacao/2 fornecido no início do Sprint.

O predicado retirar_elementos_extra/3 remove os indivíduos que não farão parte da população e retorna os elementos que fazem parte da população. O número de elementos a serem removidos é indicado pelo argumento NP.

```
retirar_elementos_extra([H|ListaProdutoRestantesOrd], 0, []).
retirar_elementos_extra([Ind*Tempo*Prob|ListaProdutoRestantesOrd],NP,[Ind*Tempo|ListaMelhores]):-
  NP1 is NP-1,
  retirar_elementos_extra(ListaProdutoRestantesOrd,NP1,ListaMelhores).
```

Figura 7 – Predicado retirar_elementos_extra/3

Análise de eficácia

De maneira a comparar a eficácia do algoritmo genético depois de efetuadas as alterações necessárias relativamente com o algoritmo base e com o predicado que gera a melhor solução irão ser fornecidas algumas tabelas que irão conter as soluções geradas por ambos os algoritmos.

Foram definidos os seguintes valores para a geração da população e foram utilizadas as seguintes entregas:

Número de Gerações: 6

Dimensão da População: 8

Probabilidade de Cruzamento: 50

Probabilidade de Mutação: 25

entrega(4439, 20230115, 200, 1, 8, 10).

entrega(4438, 20230115, 150, 9, 7, 9).

entrega(4445, 20230115, 100, 3, 5, 7).

entrega(4443, 20230115, 120, 8, 6, 8).

entrega(4449, 20230115, 300, 11, 15, 20).

entrega(4398, 20230115, 310, 17, 16, 20).

entrega(4432, 20230115, 270, 14, 14, 18).

entrega(4437, 20230115, 180, 12, 9, 11).

entrega(4451, 20230115, 220, 6, 9, 12).

entrega(4452, 20230115, 390, 13, 21, 26).

entrega(4444, 20230115, 380, 2, 20, 25).

entrega(4455, 20230115, 280, 7, 14, 19).

Número de Entregas	Melhor solução	Tempo Solução
6	[8, 3, 1, 17, 11, 9]	453.60
7	[17, 8, 3, 1, 14, 11, 9]	500.45
8	[17, 8, 3, 12, 1, 14, 11, 9]	532.44
9	[8, 3, 12, 6, 14, 1, 17, 11, 9]	562.65
10	-	618.03
11	-	-
12	-	-

Tabela 1 – Melhores Viagens

Número de Entregas	Solução AG Base	Tempo Solução	Valor Médio
6	[11,9,8,1,17,3]	513.44	567,28
7	[8,3,14,1,11,9,17]	533.68	701,39
8	[9,3,11,8,12,17,1,14]	714.90	823,32
9	[11,9,8,6,3,17,1,14,12]	750.92	907,04
10	[9,13,1,17,11,14,12,6,3,8]	898.88	1079,71
11	[3,8,14,6,17,2,1,11,9,13,12]	1022.44	1195,41
12	[17,14,8,11,7,3,13,12,2,9,6,1]	1216.79	1348,38

Tabela 2 - Análise AG Inicial

Número de Entregas	Solução AG Desenvolvido	Tempo Solução	Valor Médio
6	[17, 8, 1, 3, 11, 9]	457.98	587,83
7	[9,11,3,1,14,17,8]	529.68	665,21
8	[17,8,3,11,9,12,14,1]	585.82	845,22
9	[17,8,3,11,9,12,6,14,1]	614.85	858,19
10	[17,8,3,11,13,9,12,6,14,1]	668.82	1081,99
11	[17,8,3,2,12,6,14,1,11,13,9]	692.25	842,25
12	[17,8,3,2,12,6,14,1,7,11,13,9]	728.78	1195,61

Tabela 3 - Análise AG Modificado

Depois de efetuado uma análise sobre os valores obtidos conseguimos concluir que as modificações realizadas ao algoritmo genético inicialmente fornecido efetivamente fizeram com que o algoritmo genético gerasse soluções mais próximas da melhor solução.

Outra conclusão que obtemos foi que os valores médios do algoritmo genético modificado são bem mais elevados que o tempo da melhor solução. Isto acontece devido ao método de seleção implementado não ser puramente elitista.

Parametrização da condição de término do AG

No algoritmo genético fornecido no início do Sprint só existia uma condição de paragem, que era quando fossem determinadas todas as gerações definidas na inicialização do algoritmo.

Esta condição de paragem foi mantida, mas foram adicionadas mais duas condições de paragem, pois a geração poderá estabilizar antes de todas as gerações pretendidas terem sido determinadas ou poderá demorar mais tempo do que o pretendido a determinar todas as gerações pretendidas. Por este motivo, atualmente existem 3 condições de paragem para o algoritmo genético.

```
gera_geracao(N,G,Pop,TempoInicial,TempoMaximo,0,GeracoesIguaisAnt,Estabilizacao,MelhorViagem):-
  write('Geracao '), write(N), write(':'), nl, write(Pop), nl,
  random_permutation(Pop,PopAleatoria),
  cruzamento(PopAleatoria,NPop1),
  mutacao(NPop1,NPop),
  avalia_populacao(NPop,NPopAv,Data),
  append(Pop, NPopAv, Populacao),
  sort(Populacao, Aux),
  ordena_populacao(Aux,NPopOrd),
  obter_melhores(NPopOrd,2,Melhores,Restantes),
  probabilidade_restantes(Restantes,ProbRestantes),
  ordena_populacao_probabilidade(ProbRestantes,ProbRestantesOrd),
  populacao(TamPop),
  ElementosEmFalta is TamPop-2,
  retirar_elementos_extra(ProbRestantesOrd,ElementosEmFalta,ListaMelhores),
  append(Melhores,ListaMelhores,ProxGeracao),
  ordena_populacao(ProxGeracao,ProxGeracaoOrd),
  N1 is N+1,
  get_time(Tf),
  TempEx is Tf-TempoInicial,
  verificar_tempo_execucao(TempEx,TempoMaximo,FlagFim),
  verificar_populacao_estabilizada(Pop,ProxGeracaoOrd,GeracoesIguaisAnt,GeracoesIguais),
  gera_geracao(N1,G,ProxGeracaoOrd,TempoInicial,TempoMaximo,FlagFim,GeracoesIguais,Estabilizacao,MelhorViagem),!.
```

Figura 8 – Predicado gera_geracao/9

Foram então adicionados argumentos ao predicado gera_geracao/9 e implementados alguns predicados extra para validar se o tempo de execução tinha sido excedido ou se a população já tinha estabilizado.

O predicado que valida se o tempo de execução já foi atingido é o verificar_tempo_execucao/3. O algoritmo termina então a sua execução assim que o TempEx seja maior que o TempoMaximo caso isto aconteça antes de terem sido geradas todas as gerações pretendidas.

O predicado que valida se a população estabilizou é o verificar_populacao_estabilizada/4. O algoritmo termina então a sua execução assim que o GeracoesIguaisAnt seja igual a GeracoesIguais caso isto aconteça antes de terem sido geradas todas as gerações pretendidas.

```
verificar_tempo_execucao(TempEx,TempoMaximo,FlagFim):-
  ((TempEx < TempoMaximo,!, FlagFim is 0);(FlagFim is 0)).

verificar_populacao_estabilizada(Pop,ProxGeracaoOrd,GeracoesIguaisAnt,GeracoesIguais):-
  ((verificar_semelhanca_populacoes(Pop,ProxGeracaoOrd), !, GeracoesIguais is GeracoesIguaisAnt+1);
  (GeracoesIguais is 0)).

verificar_semelhanca_populacoes([],[]):-!.
verificar_semelhanca_populacoes([P1|Populacao],[P2|ProxGeracao]):-
  P1=P2,
  verificar_semelhanca_populacoes(Populacao,ProxGeracao).
```

Figura 9 - Predicados auxiliares para determinar a condição de paragem

```
gera_geracao(G,_,Pop,_,_,0,Estabilizacao,Estabilizacao,MelhorViagem):-!,
  write('Geracao '), write(G), write(':'), nl, write(Pop), nl,
  Pop = [MelhorViagem|_],
  write('Populacao Estabilizou'),nl.

gera_geracao(G,G,Pop,_,_,0,_,_,MelhorViagem):-!,
  write('Geracao '), write(G), write(':'), nl, write(Pop), nl,
  Pop = [MelhorViagem|_],
  write('Numero de Geracoes Pretendido Atingido'),nl.

gera_geracao(G,_,Pop,_,_,1,_,_,MelhorViagem):-!,
  write('Geracao '), write(G), write(':'), nl, write(Pop), nl,
  Pop = [MelhorViagem|_],
  write('Tempo Excedido'),nl.
```

Figura 10 - Condições de Paragem

Uso do Algoritmo Genético para lidar com vários camiões

De maneira a tornar o Algoritmo Genético o mais genérico possível foram realizadas algumas alterações para que seja possível utilizar o AG para lidar com vários camiões.

A solução desenvolvida usa o AG com a sequência de todas as entregas e admite que uma sequência de genes corresponde a um camião, podendo existir numa sequência genes correspondeste a 3 camiões. Esta solução permite que as cargas possam ser alternadas entre os diferentes camiões de maneira a obter melhor soluções do que se as cargas tivessem já pré-definidas para cada camião.

O algoritmo genético começa por determinar a quantidade de camiões necessários para realizar todas as entregas e quantas entregas cada camião transportará, sendo que o último camião poderá transportar mais que os outros caso a divisão das entregas pelos camiões não dê um número inteiro.

Os predicados implementados para obter essas informações foram os ilustrados na Figura 11.

```
determinar_quantidade_camioes(CamioesNecessarios,Data,Camiao):-
  findall(MassaEntrega,entrega(_,Data,MassaEntrega,_,_,_),Cargas),
  obter_carga_total(Cargas,0,CargaTotal),
  carateristicasCam(Camiao,_,CapacidadeCarga,_,_,_),
  QuantidadeCamioes is CargaTotal/CapacidadeCarga,
  tratar_quantidade_camioes(QuantidadeCamioes,CamioesNecessarios).

obter_carga_total([],CargaTotal,CargaTotal):-!.

obter_carga_total([H|T],CargaTotal1,CargaTotal):-
  CargaTotal2 is CargaTotal1+H,
  obter_carga_total(T,CargaTotal2,CargaTotal).

tratar_quantidade_camioes(QuantidadeCamioes,CamioesNecessarios):-
  ParteInteira is float_integer_part(QuantidadeCamioes),
  ParteDecimal is float_fractional_part(QuantidadeCamioes),
  ((ParteDecimal > 0.75,!, CamioesNecessarios is ParteInteira+2);(CamioesNecessarios is ParteInteira+1)).

determinar_entregas_por_camiao(EntregasPorCamiao,CamioesNecessarios):-
  entregas(NEntregas),
  EntregasPorCamiaoAux is NEntregas/CamioesNecessarios,
  EntregasPorCamiao is float_integer_part(EntregasPorCamiaoAux).
```

Figura 11

Um pormenor que tivemos de ter em consideração foi que nem todos os indivíduos gerados pelo algoritmo eram possíveis de acontecer devido a excederem a capacidade de carga do camião associado e por esse motivo foram implementados alguns predicados para garantir que se fosse gerado um indivíduo inválido este seria substituído por um válido.

```
avalia_individuo(_,_,_,_,1,ViagemValida,Data):-
  ViagemValida is 1,!.

avalia_individuo(_,_,_,_,2,ViagemValida,Data):-
  ViagemValida is 0,!.

avalia_individuo(Ind,AvaliacoesRealizadas,CamioesNecessarios,EntregasPorCamiao,0,ViagemValida,Data):-
  ((AvaliacoesRealizadas < (CamioesNecessarios),!, obter_x_elementos(EntregasPorCamiao,Ind,EntregasCamiao),
  obterCargaCamiao(Data,EntregasCamiao,CargaViagem,CargaTotal), Flag is 0);
  (obterCargaCamiao(Data,Ind,CargaViagem,CargaTotal),Flag is 2)),
  ((CargaTotal > 4300,!,avalia_individuo(_,CamioesNecessarios,CamioesNecessarios,_,1,ViagemValida,Data));
  (AvaliacaoAtualizada is AvaliacoesRealizadas+1,remover_x_elementos(EntregasPorCamiao,Ind,IndAtualizada),
  avalia_individuo(IndAtualizada,AvaliacaoAtualizada,CamioesNecessarios,EntregasPorCamiao,Flag,ViagemValida,Data))).

valida_populacao([],_,_,PopResultante,PopAtualizada,_):-
  PopAtualizada = PopResultante,!.

valida_populacao([P1|Resto],CamioesNecessarios,EntregasPorCamiao,PopResultante,Solucao,Data):-
  avalia_individuo(P1,1,CamioesNecessarios,EntregasPorCamiao,0,ViagemValida,Data),
  ((ViagemValida == 1,!, random_permutation(P1,NovoMembro),append(Resto,[NovoMembro],PopNova),
   valida_populacao(PopNova,CamioesNecessarios,EntregasPorCamiao,PopResultante,Solucao,Data));
   (append(PopResultante,[P1],PopAtualizada),valida_populacao(Resto,CamioesNecessarios,EntregasPorCamiao,PopAtualizada,Solucao,Data))).
```

Figura 12

O predicado é chamado antes de a população ser avaliada, tanto no predicado gera_geracao/11 como no predicado gera/6.

O predicado que avalia os indivíduos teve de ser modificado ligeiramente para fazer tantas avaliações como o número de camiões existentes e guardar apenas o maior valor de todas as avaliações de um indivíduo, repetindo o processo para todos os indivíduos da população, como é visível na Figura 13.

```
avalia_populacao([],[],_,_,_):-!.
avalia_populacao([Ind|Resto],[Ind*V|Resto1],CamioesNecessarios,EntregasPorCamiao,Data):-
  Camioes is truncate(CamioesNecessarios),
  obter_tempo_viagem(Ind,0,0,Camioes,EntregasPorCamiao,V,Data),
  avalia_populacao(Resto,Resto1,CamioesNecessarios,EntregasPorCamiao,Data).

obter_tempo_viagem(_,Tempo,Camioes,Camioes,_,TempoMaior,_):-
  (TempoMaior is Tempo),
  !.

obter_tempo_viagem(Ind,Tempo,TemposCalculados,CamioesNecessarios,EntregasPorCamiao,TempoMaior,Data):-
  ((TemposCalculados < (CamioesNecessarios - 1),!, obter_x_elementos(EntregasPorCamiao,Ind,EntregasCamiao),
  determinarTempo(Data,eTruck01, EntregasCamiao, TempoViagem));
  (determinarTempo(Data,eTruck01, Ind, TempoViagem))),
  ((TempoViagem > Tempo,!, NovoTempo is TempoViagem);(NovoTempo is Tempo)),
  VezesCalculadas is TemposCalculados+1,
  ((VezesCalculadas == (CamioesNecessarios),!,
  obter_tempo_viagem(IndAtualizada,NovoTempo,VezesCalculadas,CamioesNecessarios,EntregasPorCamiao,TempoMaior,Data),!);
  (remover_x_elementos(EntregasPorCamiao,Ind,IndAtualizada),
  obter_tempo_viagem(IndAtualizada,NovoTempo,VezesCalculadas,CamioesNecessarios,EntregasPorCamiao,TempoMaior,Data))).
```

Figura 13

O algoritmo funciona na mesma caso só seja preciso um camião para realizar todas as entregas, sendo então este o nosso algoritmo genético final que se encontra no ficheiro AlgoritmoGeneticoAtualizadoVariosCamioes.pl.

Nas Figuras 14 e 15, temos dois exemplos de possíveis resultados quando invocado o predicado gera/6.

```
0]2542372874,[008,009,003,001,011]*470.15826271186444,[003,011,008,001,009]*475.21694915254244,[003,011,008,009,001]*586.0665254237289,[003,011,001,008,0
09]*599.0088983050848]
Geracao 4:
[[011,009,008,003,001]*430.33898305084745,[008,001,003,009,011]*431.99300847457636,[011,003,001,008,009]*463.02902542372874,[008,009,003,001,011]*470.15
826271186444,[008,009,003,011,001]*486.32139830508476,[003,011,008,009,001]*501.56864406779664,[003,011,008,009,001]*586.0665254237289,[003,011,001,008,
009]*599.0088983050848]
Geracao 5:
[[008,001,003,011,009]*412.569279661017,[011,009,008,003,001]*430.33898305084745,[008,001,003,009,011]*431.99300847457636,[011,009,003,001,008]*459.9499
9999999993,[011,003,001,008,009]*463.02902542372874,[001,011,008,003,009]*466.0533898305085,[008,009,003,001,011]*470.15826271186444,[003,011,008,009,00
1]*586.0665254237289]
Geracao 6:
[[008,001,003,011,009]*412.569279661017,[011,009,008,003,001]*430.33898305084745,[011,009,003,001,008]*459.94999999999993,[011,003,001,008,009]*463.0290
2542372874,[001,011,008,003,009]*466.0533898305085,[011,009,008,003,001]*500.38559322033905,[008,001,003,011,009]*525.1582627118644,[003,011,008,009,001
]*586.0665254237289]
Numero de Geracoes Pretendido Atingido


MelhorViagem = ["008", "001", "003", "011", "009"],
TempoViagem = 412.569279661017,
CamioesNecessarios = 1.0,
EntregasPorCamiao = 5.0 ,
```

Figura 14

```
1,010,013,008,007,002,012,003]*685.4832627118644,[011,012,009,007,010,008,013,001,016,003,015,017,014,002,004,006]*819.6974576271186,[011,014,009,007,01
0,008,013,001,016,003,015,017,012,002,004,006]*819.6974576271186]
Geracao 4:
[[006,012,008,003,002,017,014,001,010,007,016,009,015,011,004,013]*500.33644067796611,[006,012,003,002,017,014,001,010,016,009,015,008,007,011,004,013]*5
57.4353813559323,[012,010,001,015,013,003,009,004,011,016,002,008,017,014,006,007]*606.6555084745762,[012,002,017,001,014,003,010,016,009,015,008,007,01
1,004,013,006]*620.6197033898305,[016,014,006,007,002,015,003,010,004,012,009,017,001,008,011,013]*652.1084745762713,[011,001,003,002,017,014,006,015,01
6,004,009,012,010,008,007,013]*659.2800847457627,[006,001,009,004,013,014,016,017,011,010,015,008,007,002,012,003]*685.4832627118644,[011,014,009,007,01
0,008,013,001,016,003,015,017,012,002,004,006]*819.6974576271186]
Geracao 5:
[[003,002,006,012,008,017,014,001,010,007,016,009,015,011,004,013]*500.33644067796611,[006,012,008,003,002,017,014,001,010,007,016,009,015,011,004,013]*5
00.33644067796611,[006,012,003,002,017,014,001,010,016,009,015,008,007,011,004,013]*557.4353813559323,[012,002,014,001,017,003,010,016,009,015,008,007,01
1,004,013,006]*572.4783898305085,[012,002,017,001,014,003,010,016,009,015,008,007,011,004,013,006]*620.6197033898305,[016,014,006,007,002,015,003,010,00
4,012,009,017,001,008,011,013]*652.1084745762713,[011,001,003,002,017,014,006,015,016,004,009,012,010,008,007,013]*659.2800847457627,[003,002,006,001,01
1,014,017,015,016,004,009,012,010,008,007,013]*659.2800847457627]
Geracao 6:
[[003,002,006,012,008,017,014,001,010,007,016,009,015,011,004,013]*500.33644067796611,[006,012,008,003,002,017,014,001,010,007,016,009,015,011,004,013]*5
00.33644067796611,[006,012,003,002,017,014,001,010,016,009,015,008,007,011,004,013]*557.4353813559323,[012,002,014,001,017,003,010,016,009,015,008,007,01
1,004,013,006]*572.4783898305085,[012,002,017,001,014,003,010,006,009,015,008,007,011,004,013,016]*620.6197033898305,[009,016,007,003,002,006,014,012,00
4,017,010,001,013,008,015,011]*644.4334745762712,[016,014,006,007,002,015,003,010,004,012,009,017,001,008,011,013]*652.1084745762713,[003,002,006,001,01
1,014,017,015,016,004,009,012,010,008,007,013]*659.2800847457627]
Numero de Geracoes Pretendido Atingido


MelhorViagem = ["003", "002", "006", "012", "008", "017", "014", "001", "010"|...],
TempoViagem = 500.33644067796611,
CamioesNecessarios = 3.0,
EntregasPorCamiao = 5.0.
```

Figura 15

Estudo de métodos de Aprendizagem Automática ao problema da distribuição de mercadorias e/ou uso de veículos elétricos

A Aprendizagem Automática, conceito inventado por Arthur Samuel, é uma área importante da Inteligência Artificial que se concentra no desenvolvimento de sistemas de computadores que consigam aprender com os dados e adaptarem-se a novas informações, sem que seja necessário a existência de uma programação explícita, aprendendo com a experiência e aperfeiçoando assim o seu desempenho. [1]

Este conceito pode ser aplicado praticamente em todas as áreas, mas neste estudo vamos focar-nos no problema da distribuição de mercadorias e veículos elétricos, onde existem já várias aplicações possíveis.

O Machine Learning permite que as empresas que necessitam de serviços de logística analisem grandes quantidades de dados e melhorem a sua gestão. É viável usá-lo para otimizar operações e minimizar erros que os humanos podem cometer, reduzindo assim custos para a empresa devido ao processamento rápido na escolha de melhores rotas e uma melhor gestão da frota. [2]

São vários os temas onde existe até ao momento uma grande variedade de soluções que utilizam algoritmos de Machine Learning sendo alguns deles os seguintes: Otimização da Rota, Controlo dos Veículos, Consumo de Energia de Veículos Elétricos e Diagnóstico de problemas em veículos elétricos.

Otimização da Rota e Controlo dos Veículos

Uma parte substancial dos custos de empresas que têm de efetuar o transporte de mercadorias entre diferentes armazéns poderá estar relacionada com uma má definição das rotas adotadas, um mau controlo dos veículos, visto serem processos com uma grande complexidade associada.

Em problemas de logística como o do caixeiro-viajante e o controlo da rota de veículos, os modelos de otimização já são utilizados há muito tempo, e têm vindo a tornar o procedimento mais simplificado à medida que a inteligência artificial evolui. [3] Alguns dos mais estudados e úteis são o Problema de Controlo de Rota de Veículos (VRP) e o Problema de Controlo de Rota de Veículos com Janelas de Tempo (VRPTW). [4]

O objetivo do VRP é o de encontrar um conjunto de rotas de entrega que satisfaça alguns requisitos ou restrições e forneça uma solução ótima para o problema, e tem vindo a atrair o interesse de vários investigadores devido ao seu papel importante no planeamento de sistemas de logística e noutras áreas como é o caso do sequenciamento de tarefas. [5]

O VRPTW é uma generalização do VRP e pode ser analisado como um problema que tenta otimizar o uso de uma frota de veículos que deve fazer um número de entregas, especificando que entregas devem ser realizadas por cada veículo de maneira a minimizar o tempo, tendo em atenção a capacidade do veículo e restrições de tempo, como os necessários para efetuar o carregamento das baterias caso seja necessário. [5]

Desde que o primeiro VRP foi apresentado por Dantzig e Ramser em 1959, muitos outros algoritmos foram propostos para resolver o VRP clássico ou suas diversas variantes e foi relatado em 2002 por Toth e Vigo que o uso de Aprendizagem Automática nos processos de logística normalmente traduz-se em economias que variam entre 5% e 20% nos custos de transportes. [5]

Consumo de energia de veículos elétricos

No mundo atual, onde a sustentabilidade ambiental é um tema bastante presente e a energia um bem de primeira necessidade, os modelos de consumo de energia que podem prever o consumo da mesma de forma eficiente e confiável são bastante importantes, para aperfeiçoar o planeamento e gestão das infraestruturas de carregamento e para melhorar a eficiência do consumo de energia dos veículos elétricos. [6]

Recentemente algumas abordagens baseadas em Aprendizagem Automática foram aplicadas para prever este tema. O LightGBM e o XGBOOST são alguns dos algoritmos de Machine Learning inovadores que provaram ter um melhor desempenho na previsão do que os modelos tradicionais, estes algoritmos pertencem à família de algoritmos de árvore de decisão. [6]

Os modelos enunciados anteriormente para realizar com precisão a previsão do consumo de energia de veículos elétricos têm em atenção a influência de fatores internos e externos, como a velocidade da

viagem, a distância percorrida, o uso do ar condicionado, a temperatura ambiente e inclinação da estrada e as interdependências entre os diferentes fatores. [6]

Podemos então concluir que com a utilização da Aprendizagem Automática, existirá uma previsão mais precisa da energia consumida o que fará com que seja realizada uma melhor utilização da energia disponível conseguindo assim uma melhor gestão no que diz respeito às infraestruturas de carregamento e ao planeamento da rota dos veículos elétricos, pois um estudo revelou que maior parte dos condutores só utiliza 70% da energia disponível da bateria devido à incerteza. [6]

Diagnóstico de problemas em veículos elétricos

A direção assistida elétrica (EPS) tem vindo a tornar-se mais sofisticada e amplamente adotada na indústria automobilística. Mas como todos os sistemas tem algumas falhas associadas que podem incluir anomalias ou falhas de componentes, como sensores, bem como falhas iminentes, como rolamentos danificados ou defeituosos que afetam o atrito. [7]

A deteção de anomalias é uma técnica para identificar alguns desvios inesperados de pontos de dados de padrões típicos em um conjunto de dados, e a deteção antecipada de possíveis falhas no veículo podem minimizar atrasos de entregas, custos elevados de reparação, acidentes e proporcionar uma maior segurança aos trabalhadores da empresa.

As três principais abordagens para detetar as anomalias são model-based, knowledge-based e orientadas por dados, sendo que nos últimos anos, a abordagem orientada a dados demonstrou alto potencial com o desenvolvimento de modelos de deep learning. No entanto, obter dados com um evento atípico é difícil e tem custos elevados associados, o que limita o uso do método tradicional baseado em dados. O uso de um algoritmo de machine learning para deteção de anomalias fornece uma solução para esse desafio. [7]

Conclusões

Após a realização deste estudo bibliográfico, conseguimos concluir a importância que a Aprendizagem Automática tem para as empresas de logística, visto que permite otimizar a gestão de frota, uma vez que realiza melhores previsões e tem em conta aspetos que não eram tidos em consideração nas metodologias mais tradicionais no momento de efetuar o planeamento de uma rota, com uma percentagem de erro muito baixa, traduzindo-se na redução de custos para a empresa.

Cada vez mais a frota das empresas é composta maioritariamente por veículos elétricos, mas associado a este fator, temos a necessidade de uma boa gestão da energia das baterias, por parte da empresa. Para isso, uma boa adoção de metodologias de Aprendizagem Automática é manifestamente necessária.

Conclusões

Após a implementação das funcionalidades pedidas no âmbito da unidade curricular de Algoritmia Avançada, do terceiro ano da Licenciatura em Engenharia Informática, do Instituto Superior de Engenharia do Porto, conseguimos concluir que:

- O algoritmo genético que contem as alterações implementadas gera melhores soluções do que o AG fornecido no início do Sprint, e soluções, muitas vezes, próximas da melhor solução.
- O Machine Learning permite uma redução de custos às empresas e melhora a gestão das mesmas no que diz respeito aos processos de logística e aos veículos elétricos.

Assim, podemos concluir, as evidências apresentadas neste relatório demonstram claramente que o algoritmo genético e a aprendizagem automática ajudam a processar rapidamente grandes quantidades de informações disponíveis e tomar decisões sem ter de considerar todas as opções possíveis, reduzindo assim os custos para as empresas e um melhor planeamento da sua frota de camiões e gerenciamento de entregas.

Bibliografia

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, First Edit. O'Reilly Media, 2017.

[2] "9 Use Cases of Machine Learning in Logistics | Serengeti." https://serengetitech.com/business/9-use-cases-of-machine-learning-in-logistics/ (accessed Jan. 05, 2023).

[3] N. Giuffrida, J. Fajardo-Calderin, A. D. Masegosa, F. Werner, M. Steudter, and F. Pilla, "Optimization and Machine Learning Applied to Last-Mile Logistics: A Review," *Sustain.*, vol. 14, no. 9, 2022, doi: 10.3390/su14095329.

[4] Z. Tarapata, W. Kulas, and R. Antkiewicz, "Machine learning algorithms for the problem of optimizing the distribution of parcels in time-dependent networks: the case study," *Arch. Transp.*, vol. 61, no. 1, pp. 133–147, 2022, doi: 10.5604/01.3001.0015.8269.

[5] L. C. Yeun, W. a N. R. Ismail, K. Omar, and M. Zirour, "Vehicle Routing Problem : Models and Solutions," *J. Qual. Meas. Anal.*, vol. 4, no. 1, pp. 205–218, 2008.

[6] I. Ullah, K. Liu, T. Yamamoto, R. E. Al Mamlook, and A. Jamal, "A comparative performance of machine learning algorithm to predict electric vehicles energy consumption: A path towards sustainability," *Energy Environ.*, vol. 33, no. 8, pp. 1583–1612, 2022, doi: 10.1177/0958305X211044998.

[7] L. W. Alabe, K. Kea, Y. Han, Y. J. Min, and T. Kim, "A Deep Learning Approach to Detect Anomalies in an Electric Power Steering System," *Sensors*, vol. 22, no. 22, pp. 1–14, 2022, doi: 10.3390/s22228981.