Sprint 3 – ALGAV

Grupo 28

Tiago Marques – 1201276

Eduardo Silva – 1201371

João Vieira – 1201376

Pedro Alves – 1201381

Pedro Rocha – 1201382

janeiro, 2023

Índice de conteúdos

Índice de figuras

Índice de tabelas

1 – Introdução

Este relatório contém todas as explicações e opções tomadas relativamente às funcionalidades requisitadas no último sprint do projeto integrador, no que diz respeito à unidade curricular de Algoritmia Avançada.

As funcionalidades propostas suportam um fator comum entre todas, o Algoritmo Genético. Além da adaptação no contexto intrínseco do projeto e melhoria do mesmo, foram ainda implementadas no Algoritmo Genético, duas funcionalidades que estarão enunciadas e descritas devidamente neste relatório.

Foi solicitado, também, o estudo de métodos de Aprendizagem Automática referente ao tema em questão, a distribuição de mercadorias e/ou uso de veículos elétricos. Desta forma, procedeu-se à pesquisa de artigos científicos relacionados e ao tratamento dos mesmos.

2 – Criação da população inicial do Algoritmo Genético (AG)

A criação da população inicial do Algoritmo Genético é o primeiro procedimento a realizar aquando da execução do algoritmo. Para isso, é necessário conhecer quais os armazéns que vão receber uma entrega. A lista de armazéns que irão receber uma entrega é gerada através da utilização do predicado *findall/3*. O predicado *entrega/6* representa todas as entregas a utilizar para geração da solução, sendo que este se encontra na base de conhecimento.

Na versão do Algoritmo Genético original, durante a criação da população inicial, os indivíduos são gerados aleatoriamente. Uma das alterações que foram realizadas, foi a inserção de 2 indivíduos gerados por duas heurísticas desenvolvidas no último sprint. As heurísticas escolhidas foram:

- Heurística que avalia a massa de cada entrega;
- Heurística que avalia o tempo necessário para a realização da entrega.

Após a utilização das heurísticas, verifica-se se as duas soluções geradas são iguais. Em caso de igualdade, ocorre a troca de 2 genes de um dos indivíduos. Neste caso, troca-se os dois primeiros genes da solução gerada pela heurística do tempo através do predicado *troca_2_ele/2*, em que retorna uma nova lista com os dois primeiros genes trocados em relação à lista enviada no primeiro parâmetro.

No final, é chamado o predicado *gera_populacao/4*, que vai então gerar os indivíduos aleatoriamente. Como já foram gerados 2 indivíduos pelas heurísticas, o predicado irá gerar *TamPop-2* indivíduos, sendo o *TamPop* o tamanho desejado para a população.

Os procedimentos descritos anteriormente correspondem ao bloco de código representado na seguinte figura:

```
gera_populacao(Pop):-
    populacao(TamPop),
    entregas(NumT),
    findall(Local,entrega(_,_,_,Local,_,_),ListaArmazens),
    bfsMassa(EI2,_),
    bfs_tempo(EI1,_),
    tirar_cabeca(EI2,E2),
    tirar_cabeca(EI1, E1),
    (((E1 == E2),troca_2_ele(E1,Enovo));troca_lista(E1,Enovo)),
    write('Indivíduo gerado pela heurística do tempo= '),write(Enovo),nl,
    write('Indivíduo gerado pela heurística da massa= '),write(E2),nl,
    gera_populacao(TamPop-2,ListaArmazens,NumT,PopGerada),
    append([Enovo,E2],PopGerada,Pop),!.
```

Figura 1 - Predicado gera/1

De referir ainda que, de modo a não permitir a existências de dois indivíduos, é utilizado o predicado *member/2*, acompanhado do predicado *not/1*, em cada iteração *gera_populacao/4,* que verifica a repetição do individuo gerado na lista de indivíduos já adquiridos.

Na figura seguinte está representado um exemplo de output, em que o tamanho da população é 3.

```
?- gera(3,3,50,50, ListaFinal,Av).
IndivÃ-duo gerado pela heurÃ-stica do tempo= [V04,P02,G01,S02,P03,A01]
IndivÃ-duo gerado pela heurÃ-stica da massa= [G01,P03,V04,S02,A01,P02]
Populacao gerada = [[V04,P02,G01,S02,P03,A01],[G01,P03,V04,S02,A01,P02],[P03,A01,P02,S02,G01,V04]]
```

Figura 2 - Output da população

3 – Aleatoriedade no cruzamento entre indivíduos da população

Aquando da realização dos cruzamentos entre os indivíduos da população, estes só se verificavam para elementos consecutivos da população.

De modo a alterar a lógica inicial dos cruzamentos, utiliza-se o predicado *random_permutation/2* para permutar aleatoriamente a lista que recebe como primeiro parâmetro (neste caso corresponde à população). Assim, como o predicado *cruzamento/2*, que é responsável por realizar os cruzamentos na população, efetua cruzamentos entre 2 elementos consecutivos, como mostra a *Figura 3*, a permutação irá permitir cruzamentos que não seriam avaliados caso fosse utilizada a população sem permutações.

```
gera_geracao(N,G,Pop,TempoInicial,ListaRetorno):-
    write('Geracao '), write(N), write(':'), nl, write(Pop), nl,
    populacao(DP),
    random_permutation(Pop,RandPop),
    cruzamento(RandPop,CruzPop),
```

Figura 3 - Código referente à permutação da população e realização de cruzamentos

De modo a demonstrar a aleatoriedade dos cruzamentos, é apresentada a figura seguinte:

```
?- gera(3,4,50,50, ListaFinal,Av).
IndivÃ-duo gerado pela heurÃ-stica do tempo= [V04,P02,G01,S02,P03,A01]
IndivÃ-duo gerado pela heurÃ-stica da massa= [G01,P03,V04,S02,A01,P02]
Populacao gerada = [[V04,P02,G01,S02,P03,A01],[G01,P03,V04,S02,A01,P02],[V04,P02,P03,G01,S02,A01],[G01,P02,S02,P03,A01,V04]]
PopAv=[[V04,P02,G01,S02,P03,A01]*456.9438356164383,[G01,P03,V04,S02,A01,P02]*541.1061643835617,[V04,P02,P03,G01,S02,A01]*514.327397260274,[G01,P02,S02,P03,A01,V04]*506.9082191780822]
Geracao 0:
[[V04,P02,G01,S02,P03,A01]*456.9438356164383,[G01,P02,S02,P03,A01,V04]*506.9082191780822,[V04,P02,P03,G01,S02,A01]*514.327397260274,[G01,P03,V04,S02,A01,P02]*541.1061643835617]
Cruzamento entre [G01,P02,S02,P03,A01,V04] e [G01,P03,V04,S02,A01,P02]
Cruzamento entre [V04,P02,P03,G01,S02,A01] e [V04,P02,G01,S02,P03,A01]
Geracao 1:
[[V04,P02,G01,S02,P03,A01]*456.9438356164383,[G01,P02,S02,P03,A01,V04]*506.9082191780822,[V04,P02,P03,G01,S02,A01]*514.327397260274,[V04,P02,P03,S02,A01,G01]*517.508904109589]
Cruzamento entre [G01,P02,S02,P03,A01,V04] e [V04,P02,G01,S02,P03,A01]
Cruzamento entre [V04,P02,P03,S02,A01,G01] e [V04,P02,P03,G01,S02,A01]
Geracao 2:
[[V04,P02,G01,S02,P03,A01]*456.9438356164383,[V04,P02,G01,S02,A01,P03]*504.3719178082192,[G01,P02,S02,P03,A01,V04]*506.9082191780822,[V04,P02,P03,S02,A01,G01]*517.508904109589]
Cruzamento entre [V04,P02,G01,S02,P03,A01] e [V04,P02,G01,S02,A01,P03]
Cruzamento entre [G01,P02,S02,P03,A01,V04] e [V04,P02,P03,S02,A01,G01]
```

Figura 4 - Output da população e tentativas de cruzamentos efetuados

Pela análise da *Figura 4*, podemos concluir que a tentativa de cruzamentos está, de facto, a ser realizada aleatoriamente, visto que, por exemplo, da primeira para a segunda geração, existe uma tentativa de efetuar cruzamentos entre o segundo e o quarto indivíduo e entre o primeiro e o terceiro indivíduo da população, sendo estes indivíduos não consecutivos.

4 – Seleção da nova geração da população

Após obter os descendentes por cruzamento e realizar mutações, esta nova população irá ser intercalada com a já existente utilizando o predicado *union/3* que irá assegurar também a não existência de indivíduos repetidos na nova população gerada.

Como é referido nos documentos disponibilizados, é assegurada a passagem de cerca de 20% da população para a geração seguinte. Esta seleção é feita através do predicado *round/1*, em que o número de indivíduos que irão transitar de geração é dado pelo arredondamento do cálculo 0.20**DP*, sendo *DP* a dimensão da população. Numa forma de especificar os cálculos, apresentamos a tabela seguinte:

Dimensão da população	*Número de melhores indivíduos que asseguram a passagem à próxima geração*
3	0.2*3 = 0.6 ≈ 1
4	0.2*4 = 0.8 ≈ 1
5	0.2*5 = 1
6	0.2*6 = 1.2 ≈ 1
7	0.2*7 = 1.4 ≈ 1
8	0.2*8 = 1.6 ≈ 2
9	0.2*9 = 1.8 ≈ 2
10	0.2*10 = 2

Tabela 1 - Número de melhores indivíduos que asseguram a passagem à próxima geração

A lista que contém os melhores elementos é denominada de *Select* e a lista com os restantes elementos é denominada de *Rest*. De seguida, esta última passa por um processo de seleção não elitista, dando oportunidade a todos os indivíduos da lista de transitarem para a geração seguinte e não apenas aos melhores da mesma.

Por base nos documentos de suporte ao projeto, é implementado o predicado *valores_avaliacao/2* em que irá multiplicar por um número aleatório entre 0 e 1, a avaliação de cada indivíduo. Assim, aquando da chamada do predicado *nao_elitista/4*, este último irá utilizar a lista retornada pelo predicado anteriormente descrito e irá ordená-la através do predicado *orderna_populacao/2*. Após a ordenação e selecionando os primeiros *IndividuosSG* (calculado através da diferença entre a dimensão da população e o número de melhores indivíduos que asseguram a passagem à próxima geração) indivíduos, o predicado retorna então a lista dos selecionados.

```
nao_elitista(NIndividuos,RestNovo,P,MPopOrd):-
    IndividuosSG is NIndividuos - P,
    ordena_populacao(RestNovo,RestNovoOrd),
    select_first_n(RestNovoOrd,IndividuosSG,MPopOrd).
```

Figura 5 - Predicado nao_elitista/4

Assim, garante-se que a seleção dos indivíduos não é elitista. No final, utiliza-se o predicado *append/*3 para então juntar os melhores indivíduos selecionados aos que foram escolhidos por torneio. Toda a explicação apresentada é referente ao seguinte bloco de código:

```
    P1 is round(0.20*DP),
    ((P1 < 1, P is P1+1); P is P1),
    select_first_n(PopTotalOrd,P,Select),
    remove_elements(PopTotalOrd,P,Rest),


    valores_avaliacao(Rest, RestNovo),
    nao_elitista(DP,RestNovo,P,MPopTotal),

    lista_com_produto_avaliacao(MPopTotal, MPopTotalOriginal),

    avalia_populacao(MPopTotalOriginal, MPopTotalAv),

    append(Select,MPopTotalAv,MPopTotalAp),
```

Figura 6 - Código referente à seleção dos indivíduos para a nova geração

Para demonstrar que a seleção não é elitista, analisemos o output seguinte:

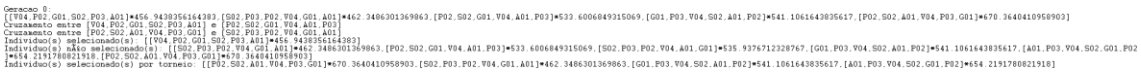

Figura 7 - Output da população e dos indivíduos selecionados para integrarem a próxima geração

Neste caso, foi selecionado o melhor indivíduo da população, os restantes indivíduos não selecionados participaram no torneio em que, por exemplo, o pior indivíduo foi selecionado para integrar a seleção seguinte, enquanto o terceiro melhor indivíduo da população atual não foi selecionado. Concluímos que é feita uma seleção não elitista dos indivíduos para ingressarem na geração seguinte.

5 – Análise de eficácia

Primeiramente, iremos comparar entre o melhor indivíduo da última geração do Algoritmo Genético desenvolvido com o de base e o valor médio das avaliações de todos os indivíduos da população final de cada algoritmo.

Sendo assim, apresentamos as seguintes tabelas:

Dimensão da população	5
Número de gerações	5
Probabilidade de cruzamento	0,5
Probabilidade de mutação	0,25

Tabela 2 - Parametrização de ambos os Algoritmos Genéticos

	AG base		AG desenvolvido	
Tentativas	**Melhor indivíduo** (Minutos)	**Média dos indivíduos** (Minutos)	**Melhor indivíduo** (Minutos)	**Média dos indivíduos** (Minutos)
1	462,36	499,20	456,94	530,46
2	511,14	558,17	386,55	506,78
3	492,87	586,49	456,78	501,23

Tabela 3 - Valores retornados pelos Algoritmos Genéticos

Pela análise da *Tabela 3*, verificamos que os melhores indivíduos gerados pelo Algoritmo Genético desenvolvido são melhores em relação aos que são gerados pelo Algoritmo Genético base. Optou-se por apresentar 3 tentativas de modo a despistar casos “felizes” ou “infelizes” para ambos os algoritmos. Verificamos também que a média dos indivíduos da última geração é, de maneira geral, menor no Algoritmo Genético desenvolvido que no de base.

Assim sendo, iremos agora analisar valores para casos de 6 a 12 entregas. A parametrização utilizada nos algoritmos é descrita na tabela seguinte:

Dimensão da população	Número de entregas a realizar
Número de gerações	10
Probabilidade de cruzamento	0,5
Probabilidade de mutação	0,25

Tabela 4 - Parametrização de ambos os Algoritmos Genéticos

A tabela abaixo retratada contém, além dos dois algoritmos, um terceiro algoritmo que gera todas as soluções e escolhe a melhor. São apresentados os valores para cada número de entregas a realizar.

		AG base		AG desenvolvido	
Número de entregas	**Melhor solução total do problema** (Minutos)	**Melhor indivíduo** (Minutos)	**Média dos indivíduos** (Minutos)	**Melhor indivíduo** (Minutos)	**Média dos indivíduos** (Minutos)
6	380,05	506,10	544,24	456,94	513,89
		484,20	546,18	441,82	491,45
		495,11	524,38	454,69	512,88
7	533,87	676,37	789,59	633,62	717,80
		682,95	856,44	550,03	655,71
		694,55	828,01	565,33	734,13
8	613,12	840,03	975,16	740,40	908,48
		815,73	1044,19	650,57	756,21
		793,34	997,06	730,95	858,68
9	669,60	838,56	1121,43	766,93	899,04
		968,67	1147,69	808,39	979,11
		861,91	1090,74	724,27	952,55
10	-	1150,09	1406,27	814,49	1137,21
		1216,46	1446,08	814,49	1177,00
		1111,71	1370,26	814.49	1111,52
11	-	1348,61	1549,33	1057,36	1395,26
		1303,96	1519,62	1162,26	1395,12
		1267,59	1529,99	1229,25	1374,98
12	-	1437,41	1746,15	1102,99	1273,71
		1412,41	1631,29	1142,98	1523,42
		1306,87	1627,23	1104,72	1347,24

Tabela 5 - Valores retornados pelos Algoritmos Genéticos e gerador de todas as soluções

Para complementar a análise da *Tabela 5*, é apresentado o seguinte gráfico:

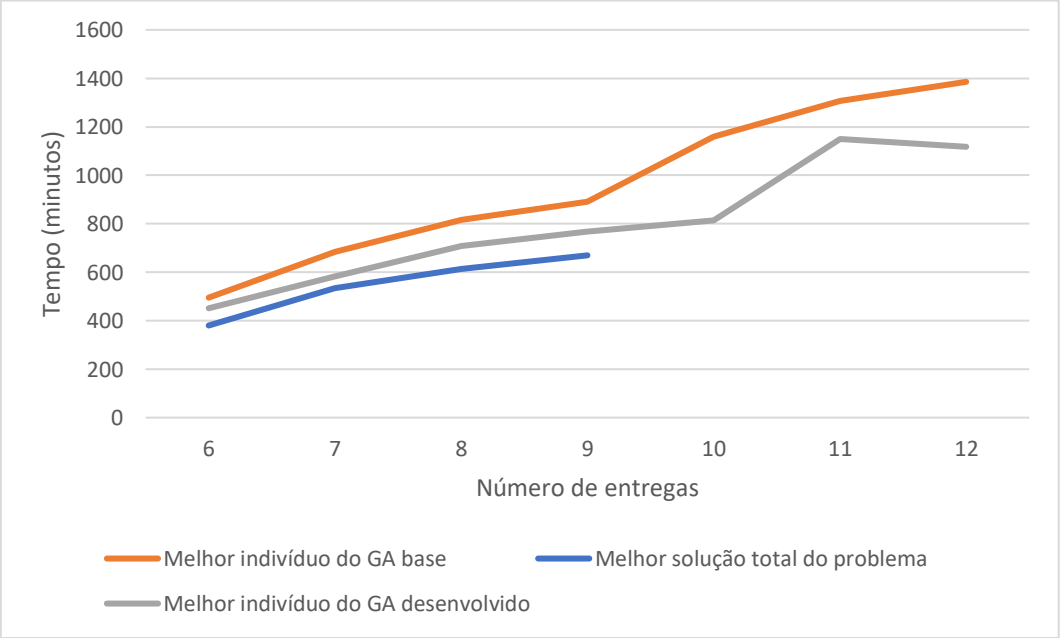

Figura 8 - Gráfico de análise à Tabela 5

Pela análise do gráfico e da tabela anteriores, podemos concluir que o Algoritmo Genético desenvolvido gera melhores soluções que o Algoritmo Genético base. De referir ainda que é apresentada melhor solução total do problema são apresentados apenas valores até às 9 entregas, visto que a partir desse número de entregas não é possível gerar valores devido à *stack* ser insuficiente.

Podemos referir também que a diferença entre os valores gerados pelos dois algoritmos é exponencial, logo, quanto maior o número de entregas a realizar, mais díspares serão os valores gerados por ambos os algoritmos.

O gráfico apresentado foi realizado através da média das tentativas para os valores gerados para cada número de entregas.

Concluindo, o algoritmo desenvolvido, com todas as modificações e melhorias aconselhadas nos documentos disponibilizados pela unidade curricular, gera melhores soluções que o algoritmo base. Naturalmente, o algoritmo desenvolvido no sprint anterior, em que analisa todas as soluções e retorna a ótima, irá sempre gerar uma melhor solução. No entanto, consideramos que o Algoritmo Genético é mais eficiente e mais dinâmico na forma de gerar soluções.

6 – Parametrização da condição de término do AG

Além da condição de término natural do Algoritmo Genético, em que termina a sua execução quando atingir o número de gerações especificadas, foi implementada uma nova condição de término. De dentre as condições de término referidas, a que se optou foi pela definição de um tempo máximo de execução do algoritmo.

Assim sendo, no início da execução do algoritmo, é utilizado o predicado *get_time/1*, que indica o tempo atual, ficando assim com uma referência do tempo de quando o algoritmo começou a sua execução.

Para verificar se o tempo limite de execução foi atingido, é utilizado novamente o predicado *get_time/1* em cada iteração do predicado *gera_geracao/5*, verificando através da diferença do tempo atual e o tempo do início de execução do algoritmo se esta é maior que o limite estipulado.

```
gera_geracao(_,_,Pop,TempoInicial,Pop):-
    get_time(TempoAtual),
    TempoConsumido is TempoAtual-TempoInicial,
    TempoConsumido > 0.1,
    write('Tempo esgotado: '), write(TempoConsumido),nl,!.
```

Figura 9 - Verificação da ultrapassagem do tempo limite de execução

Caso ultrapasse o limite, a execução é terminada e é apresentada uma mensagem de aviso contendo o tempo de execução, em segundos, como mostra a figura seguinte.

```
Tempo esgotado: 0.11302495002746582
```

Figura 10 – Output de quando o tempo limite é excedido

7 – Uso do Algoritmo Genético para lidar com vários camiões

Quando a carga das entregas definidas para um dado dia é superior à capacidade do camião (que se pressupõe a mesma para todos, isto é, 4300 kg), as entregas são divididas entre mais do que um camião, com ajuda do Algoritmo Genético. Para que a definição do número de camiões a realizar entregas para esse dia possa ser feita, é necessário dividir a carga total das entregas pela capacidade do camião, arredondando um valor para cima. Se após a divisão, a parte decimal for superior a 0.8, acrescenta-se 1 ao valor arredondando, de modo a evitar uma grande frequência de violações de capacidade da carga.

```
num_camioes(N):-soma_carga(Soma), N1 is Soma/4300, floor(N1, Inteira), Decimal is N1-Inteira,
 ((Decimal > 0.8, N is Inteira+2); (Decimal > 0, N is Inteira+1)).

soma_carga(Soma):-findall(Carga, entrega(_,_,Carga,_,_,_), Lista), soma_lista(Lista, Soma).

soma_lista([],0).
soma_lista([H|T],Soma):-soma_lista(T,Soma1), Soma is Soma1 + H.
```

Figura 11 - Definição do número de camiões para um dado dia de entregas

Após isto, é chamado o predicado *avalia_populacao_mquc/2,* que determina quantas entregas serão atribuídas a cada camião através da divisão entre o número de entregas para o dado dia e o número de camiões que irão realizar as mesmas. O valor resultante será arredondado para a unidade abaixo, e o último camião ficará com o resto. De seguida, é definido o tempo total que cada um necessita para levar ao destino as mesmas.

```
avalia_populacao_mquc([],[]).
avalia_populacao_mquc([Lista|Resto],[Lista*Soma|Resto1]):-
    camioes(Num_camioes), entregas(Num_entregas), EntregasCamiao is Num_entregas/Num_camioes,
    floor(EntregasCamiao, EntregasCamiaoFloor),
    separar_lista_mais_que_um_camiao_2(Lista, _, Avaliacao,EntregasCamiaoFloor, Num_camioes),
    format_avaliacao(Avaliacao, AvaliacaoFinal),
    somarAv(AvaliacaoFinal, Soma),
    avalia_populacao_mquc(Resto,Resto1).

somarAv([],0).
somarAv([H|T],Soma):-
    somarAv(T,Soma1),
    Soma is Soma1 + H.
```

Figura 12 - Determinação do número de entregas a realizar por cada camião e o tempo total para realizá-las

Depois de atribuídas as entregas a cada camião, é necessário certificar que a soma da carga da mesma não ultrapassa a capacidade do camião. Se em algum momento, em algum camião, tal se verificar, a lista de entregas é permutada com as listas de entregas dos outros, através do predicado *random_premutation/2* e, isto acontece, até não ocorrer em nenhum dos camiões a infração de capacidade de carga.

```
verificar_carga(Lista, ListaApta):-camioes(Num_camioes), entregas(Num_entregas), EntregasCamiao is Num_entregas/Num_camioes,
floor(EntregasCamiao, Inteira), carga_por_camiao(Lista, Inteira,_,Num_camioes,_,ListaApta),!.

carga_por_camiao([],_,_,_,_,[]):-!.
carga_por_camiao([H|T],Inteira,Soma,Num_camioes,Baralhar,ListaApta):-
carga_por_camiao2(H,Inteira,_, _,Num_camioes,Baralhar, Flag),
(Flag is 1,!, removerAv(H,ListaSemAv), random_permutation(ListaSemAv,ListaPermutada),
avalia_individuo(ListaPermutada, ListaPermutadaAv),

carga_por_camiao([ListaPermutadaAv|T], Inteira, Soma,Num_camioes,_,ListaApta)).

carga_por_camiao([H|T],Inteira,Soma,Num_camioes,Baralhar,[H|ListaApta]):-(carga_por_camiao(T,Inteira,Soma,Num_camioes,Baralhar,ListaApta)).

carga_por_camiao2([]*_,_,0,[],_,_,_):-!.

carga_por_camiao2([H|T]*_,Inteira,Soma,Lista,Num_camioes, Baralhar, Flag):- Baralhar \== 1,!,
carga_por_camiao3([H|T],Inteira,Soma, Lista,Num_camioes),
((Soma > 4300,!, Baralhar1 is 1); Baralhar1 is 0),
Numero_camioes is Num_camioes - 1, subtract([H|T], Lista, Lista2),
carga_por_camiao2(Lista2*_, Inteira, _,_,Numero_camioes,Baralhar1,Flag1),
((Flag1 \== 1,!, Flag is Baralhar1);(Flag is Flag1)).

carga_por_camiao2(_,_,_,_,_,1,_):-!.

carga_por_camiao3([],_,0,[],1):-!.
carga_por_camiao3(_, 0, 0, [],_):-!.
carga_por_camiao3([H|T],Inteira,Soma,[H|Lista],Num_camioes):-
((Num_camioes is 1,!, ParteInteira is Inteira + 1); (ParteInteira is Inteira)), Inteira1 is ParteInteira - 1,
carga_por_camiao3(T,Inteira1,Soma1,Lista,Num_camioes), entrega(_,_,Carga,H,_,_), Soma is Carga+Soma1.

removerAv(Lista*_,Lista):-!.
```

Figura 13 - Verificação da violação da capacidade de carga do camião

Finalmente, a população é ordenada e as seguintes gerações são geradas, com recurso a cruzamentos e mutações e com a chamada ao método não elitista, que permite a indivíduos com pior avaliação poderem passar à geração seguinte. No final, a população é novamente ordenada por ordem crescente de avaliação, ou seja, tempo total para a realização das entregas.

Apesar de esta solução permitir atribuir as entregas a vários camiões, o facto de o último camião ficar com mais entregas pode aumentar o número de permutações e, consequentemente, o tempo de execução do método e, no pior dos casos, nunca atingir uma solução. No entanto, a probabilidade de tal acontecer é reduzida, a não ser que a totalidade da carga das entregas seja muito elevada e haja poucos camiões.

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

8.1. Inteligência Artificial na Logística

A Inteligência Artificial tem sido usada para aperfeiçoar os procedimentos tradicionais da logística, aumentando a confiabilidade e o processamento rápido na escolha de rotas de entrega, reduzindo assim os custos de transporte [1].

Giuffrida, refere que o crescimento do *e-commerce* está a resultar no aumento da expectativa dos consumidores perante um serviço de entrega customizado e no aumento da pressão nos fornecedores nas cadeias de abastecimento [1]. Refere também que para além se reduzir a segurança rodoviária graças à circulação de veículos pesados, a circulação dos mesmos aumenta o congestionamento do trânsito e aumenta a emissão de gases na atmosfera (25% de CO_2 e 30 a 50% de PM e NO_x) [1].

Face à otimização das técnicas tradicionais (p.e. uso de sensores e *IoT*) têm sido geradas diversas informações que precisam de ser tratadas, sendo assim cultivado o uso de Big Data e técnicas analíticas [1].

A Logística de Última Milha (Last Mile Logistics) refere-se à parte do ciclo de vida das cadeias de abastecimento, onde as mercadorias são entregues desde o último ponto de trânsito até ao local final de entrega. [2]

8.2. Soluções existentes

8.2.1. Roteamento de veículos

8.2.1.1. Otimização do roteamento de veículos (VRO)

A otimização do roteamento de veículos, é um conceito que procura calcular a rota de entrega mais ideal através da interpretação e gestão de dados e aplicação da inteligência preditiva [2].

Nos dias de hoje, o VRO é também uma ferramenta utilizada para reduzir a poluição ambiental. Com o contínuo crescimento do número de encomendas, é importante encontrar rotas que, gastando o mínimo de recursos, consigam suprir o máximo de encomendas realizadas. O VRO calcula a rota de acordo com condições específicas, desde a distância percorrida, o número de veículos utilizados e o tempo total de transporte [3].

De modo a ilustrar um exemplo que dá uso à otimização de rotas, apresentamos o exemplo da UBER[4]. A UBER – prestadora de serviços eletrónicos na área do transporte privado – trabalha constantemente com o tratamento de informação para entregar ao cliente um serviço de qualidade e preciso, entregando a estimativa entre duas zonas numa determinada hora, num determinado dia[4]. Os condutores da Uber utilizam o smartphone para trabalhar, o mesmo está sempre conectado à internet e com o GPS ligado, a cada viagem a informação é anonimizada, contudo o tempo e localização entre zonas é recolhido e guardado a cada quatro segundos o que resulta na sincronização de condutores com clientes, direção do ícone do automóvel na aplicação e sobretudo, estimativas precisas[4].

8.2.1.2. Problema do roteamento de veículos (VRP)

A Janela de Tempo corresponde ao número de clientes a serem visitados, num determinado intervalo. O artigo revê os algoritmos exatos propostos nas últimas três décadas[5]. O *Problema de Encaminhamento de veículos com Janela de Tempo* (VRPTW) tem sido amplamente estudado na literatura *Operations Research* (OR) dadas as suas aplicações cada vez mais difundidas, que vão desde o agendamento de autocarros escolares até à entrega de pacotes[5].

Foram desenvolvidos vários métodos de *Machine Learning* (ML) para combater problemas combinados e para alavancar uma estrutura complexa de dados, mas pouco tem sido feito para aplicar estas técnicas ao VRPTW[5].

O artigo trata possibilidades de resolução da VRPTW em larga escala sem o roteamento clássico ou de encaminhamento, propondo um algoritmo de duas fases[5]. Na primeira, um algoritmo de *clustering* (agrupamento de dados) alavancando as *Árvores de Classificação Ideal* (OCT) que visa dividir os clientes em subconjuntos menores. Já na segunda fase, é apresentada uma **aprendizagem de reforço** de *ator-crítico* (RL) para resolver o VRPTW nestes agrupamentos de clientes mais pequenos[5].

Os resultados mostram que a abordagem de *clustering* é competitiva no que diz respeito a um *clustering K-means-based*, produzindo melhorias até 5% em termos de número de veículos, e que uma abordagem RL pode resolver com sucesso instâncias VRPTW de tamanho médio, proporcionando resultados de otimização semelhantes aos solucionadores industriais de última geração[5].

8.2.2. Previsão do tempo de entrega

As empresas organizam e planeiam as rotas de entrega com base no tempo estimado entre paragens, se estas estimativas não forem precisas a qualidade do serviço entregue não é favorável. O artigo avalia a viabilidade das técnicas de ML para avaliar os tempos de previsão das entregas – previsão do tempo final da entrega e verificação se o mesmo cumpre a janela predefinida[6]. Para a avaliação, os modelos treinados usaram informação gerada de dados GPS coletados em Medellín, Colômbia e foram comparados com modelos de duração de perigo[6].

Relativamente aos resultados da avaliação, concluiu-se que embora os *Modelos de Regressão* (RM) e modelos de classificação serem usados para prever o tempo de entregas, os *Modelos de duração de risco* (HDM) são recomendados face aos modelos de regressão para previsões de tempo de entrega. O *K-nearest-neighbor* (KNN) fornece melhores resultados ao prever se o tempo de previsão de entrega excede o limite que a combinação entre dois níveis (KNN e RM/HDM) melhora a previsão do tempo de entrega[6].

A Amazon tem trabalhado em conjunto com a Inawsisdom, parceiro de consultoria *premier* da AWS, para ajudar várias empresas no setor do transporte e logística com metodologias de ML para prever prazos de entrega precisos[7]. A Aramex - empresa de logística no Médio Oriente - está a conduzir um programa de revolução para o comércio digital, e está a processar milhares de remessas para todo o mundo dando uso a ferramentas de *crowdsourcing*, investindo em novas tecnologias e veículos autónomos[7]. O modelo de ML da Amazon ajudou a aumentar a precisão dos tempos de entrega em 74% e os volumes de *call center* em 40%[7].

8.2.3. Previsão da energia – veículos elétricos

8.2.3.1. Autonomia limitada

A autonomia limitada de um veículo elétrico é um aspeto muito importante a se ter em conta, pois o mesmo pode ser afetado por fatores incertos como as condições de circulação.

O artigo trata o Problema de Roteamento de Veículos Elétricos dependentes do tempo com restrições de chance (EVRP-CC) e recarga parcial[8]. O método de roteamento é dividido em duas etapas, a primeira procura os melhores percursos, a segunda otimiza as rotas. Propôs-se uma abordagem probabilística de aprendizagem de uma máquina de Bayesian – paradigma de construção de modelos estatísticos baseados no Teorema de Bayes - para se prever o consumo de energia esperado e a variação para as ligações rodoviárias, caminhos e rotas[8]. O estudo foi validado com dados de autocarros elétricos que fazem a rota Gotemburgo-Suécia bem como com simulações realistas para o tráfego de 24 horas na cidade de Luxemburgo conectado a um modelo de veículo de alta-fidelidade[8]. As soluções de roteamento são comparadas com uma formulação determinística do problema semelhante às encontradas na literatura[8].

Os resultados indicam alta precisão para a previsão de energia, bem como economia de energia e maior confiabilidade para as rotas[8].

8.2.3.2. Demanda de energia

Sendo a energia um bem de primeira necessidade, devido à brutalidade da sua utilização, existem grandes chances de interrupção do fornecimento de energia. É de extrema importância a segurança do fornecimento de energia. O artigo propõe um modelo de **aprendizagem por reforço** de como lidar com as incertezas na oferta e demanda de energia considerando que os veículos elétricos possuem vários recursos energéticos – painéis fotovoltaicos, baterias – que partilham unidades de geração de energia e armazenamento reduzindo assim custos de energia[9].

O desempenho do modelo de **aprendizagem por reforço** é avaliado sob diferentes configurações de consumidores e veículos elétricos, e comparado com os resultados do CPLEX e três algoritmos heurísticos[9].

Os resultados da simulação demonstram que o algoritmo de aprendizado por reforço pode reduzir os custos de energia em até 22,05%, 22,57% e 19,33% em comparação com os resultados do algoritmo genético, otimização de enxame de partículas e algoritmo de enxame de peixes artificiais, respetivamente[7].

8.2.4. Prevenção de Erros

O minimizar a ocorrência do envio de encomendas com a morada destino incorreta, que resulta em custos financeiros e ecológicos inúteis, propõe uma abordagem de correspondência de entidades e um sistema para as validações de entidades de Transporte de Logística através de **técnicas de *Word Embedding* e Aprendizagem Supervisionada**[10] .

8.2.5. Inovação

Como referido, o aumento do número de encomendas tem um impacto negativo no ambiente e na segurança. Desta forma, novos conceitos de última milha incluem entregas usando *drones* ou pequenos robôs autónomos terrestres, distribuição de mercadorias por veículos elétricos, entregas no porta-malas de carros estacionados, distribuição combinada de transporte de carga com veículos públicos e privados, logística reversa, *crowdsourcing* de entregas ou armários móveis com fechadura[2, p.2][11].

8.3 Conclusões do estudo bibliográfico

A otimização das técnicas tradicionais, que leva à utilização da *Big Data* e técnicas analíticas é uma mais-valia para as organizações, uma vez que através da análise das informações recolhidas, é possível fazer balanços e tomar decisões que potenciam lucros e dizimam custos ambientais[1].

A otimização do Roteamento De Veículos é mais uma prova de que a otimização dos recursos eleva a produtividade da organização, já que são constantemente tidos em conta diversos aspetos que nas metodologias mais tradicionais não eram (p.e. volatilidade do trânsito)[3].

Relativamente ao problema do Roteamento De Veículos Numa Janela De Tempo, através da implementação da aprendizagem por reforço foram notadas melhorias relativamente ao número de veículos necessários[5].

De forma às empresas cumprirem os seus prazos de entrega, o planeamento e organização das rotas e recursos têm se ser bem alocados. A adoção de metodologias ML é uma mais-valia, como ilustrado no exemplo da Amazon, em que aumentou a precisão dos tempos de entrega em 74% e os volumes de *Call Center* em 40%[7].

A utilização e adoção de veículos elétricos nesta indústria é cada vez mais percetível, e para isso existe uma necessidade de uma boa gestão da energia das baterias[5][8].

Como em todas as áreas de interesse, existe sempre espaço para a inovação, já se fala em utilizar *drones* e robôs para efetuar o transporte de mercadorias para se reduzir a poluição e o trânsito de veículos de mercadorias nas grandes cidades[2][11].

9 – Avaliação da questão das alterações dinâmicas das entregas

Para a realização das alterações dinâmicas foi necessário a criação dinâmica da base de conhecimento para o algoritmo genético. Primeiramente, foi criado um *handler* que recebe os *requests* vindos de http://localhost:4201/api/Planning e redireciona para o predicado *plan/1*.

```
:- http_handler('/api/Planning', plan, []).

:-dynamic (paths/7).
:-dynamic (trucks/6).
:-dynamic (warehouses/2).
:-dynamic (deliveries/6).
:- json_object dto(id:string, email:string).
:- json_object joyDistressDto(joy:float, distress:float).

:- initialization
    http_server(http_dispatch,[port(4201)]).
```

Figura 14 - Handler e inicialização do servidor prolog

No predicado *plan/1*, definimos os parâmetros que vão ser recebidos do *request*, para depois enviar para o predicado *planeamento_plate_data/9*, quando esse método acabar, é formatado uma resposta para o *request* com as informações necessárias em formato *JSON*.

No método *planeamento_plate_data/9* é onde acontece a parte principal da funcionalidade, visto que é aqui onde iremos chamar os predicados que fazem os *requests* às respetivas bases de dados, interpretação da informação vinda da base de dados e por fim, escrever num ficheiro chamado de *KnowledgeBase.pl*.

Como podemos ver no caso dos caminhos, na *Figura 15*, com o predicado *http_open/3* será feito o *request* para o modulo de logística, que retornará em formato *JSON* o respetivo pedido. Utilizamos o predicado *json_read_dict/3* e o *getPathsInfo/3* para tratar a informação do *JSON* recebido do *request*.

```
getPaths(Plate):-
    (destroy_paths();true),
    atom_concat('http://localhost:3000', '/api/Paths', Url),
    http_open(Url, ResultJSON, []),
    json_read_dict(ResultJSON, ResultObj),
    getPathsInfo(ResultObj, Plate, ResultValue),
    createDynamicPaths(ResultValue),
    length(ResultValue,Tam),
    writePathsToFile(ResultValue,'KnowledgeBase.pl',0,Tam),
    close(ResultJSON).
```

Figura 15 – Exemplo de predicado responsável por fazer request aos módulos e preparar para escrita no ficheiro

Para terminar, no predicado *writePathsToFile/4*, são construídas as várias strings do ficheiro com a informação anteriormente tratada. Este conjunto de *strings* vai ser então adicionado ao ficheiro *KnowledgeBase.pl*.

Posteriormente à escrita da base de conhecimentos, é chamado o algoritmo de planeamento selecionado e no fim da sua execução o resultado é persistido na Base de dados.

```
writePathsToFile([],_,_,_).
writePathsToFile([P,IS,IE,D,T,E,EX|L],Ficheiro,Modo,Tam):-
    T1 is T.hours*60 + T.minutes + T.seconds/60,
    EX1 is EX.hours*60 + EX.minutes + EX.seconds/60,
    open(Ficheiro,append,Writer),
    string_concat('',"dadosCam_t_e_ta('",Dados),
    string_concat(Dados,P,Dados1),
    string_concat(Dados1,"','",Dados2),
    string_concat(Dados2,IS,Dados3),
    string_concat(Dados3,"','",Dados4),
    string_concat(Dados4,IE,Dados5),
    string_concat(Dados5,"',",Dados6),
    string_concat(Dados6,D,Dados7),
    string_concat(Dados7,',',Dados8),
    string_concat(Dados8,T1,Dados9),
    string_concat(Dados9,',',Dados10),
    string_concat(Dados10,E,Dados11),
    string_concat(Dados11,',',Dados12),
    string_concat(Dados12,EX1,Dados13),
    string_concat(Dados13,').',Dados14),
    write(Writer,Dados14),
    nl(Writer),
    ModoNovo is Modo + 7,
    writePathsToFile(L,Ficheiro,ModoNovo,Tam),
    close(Writer).
```

Figura 16 - Exemplo de predicado para escrita no ficheiro

Em forma de conclusão, foi abordada desta maneira a realização da funcionalidade para que sempre que fosse executado um *request* para o planeamento, a informação a ser utilizada nos algoritmos fosse gerada dinamicamente, de acordo com o que estava nas respetivas bases de dados naquele momento. Assim, quando for removida uma entrega, apenas é necessário voltar a utilizar o método de planeamento.

10 – Conclusões

Este relatório abordou quer o Algoritmo Genético desenvolvido neste sprint, as alterações feitas e uma completa análise à eficácia do mesmo, como também relata a importância da Inteligência Artificial no ramo da logística, enumerando algumas aplicações.

Com isto, além de todas as conclusões alcançadas em cada tópico do relatório, podemos destacar também que a aplicabilidade do Algoritmo Genético é vasta, podendo ser utilizado em vários setores relevantes no quotidiano da comunidade.

Na nossa opinião, a Inteligência Artificial assumirá um papel cada vez mais presente no dia a dia de todos, uma vez que melhora processos tradicionais, otimizando assim recursos e metodologias.

Em suma, a realização deste trabalho não só fomentou, em todos os elementos do grupo, competências técnicas referentes ao tema em questão, como despertou o interesse do grupo nesta área.

Referências

[1] 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. 2022, Vol. 14, Page 5329*, vol. 14, no. 9, p. 5329, Apr. 2022, doi: 10.3390/SU14095329.

[2] P. Jucha, "Use of artificial intelligence in last mile delivery," *SHS Web Conf.*, vol. 92, p. 04011, 2021, doi: 10.1051/SHSCONF/20219204011.

[3] "AI as the Ultimate Disrupter in Logistics: How to Manage Last-Mile Costs? | by ODSC - Open Data Science | Medium." https://odsc.medium.com/ai-as-the-ultimate-disrupter-in-logistics-how-to-manage-last-mile-costs-c4874e8f2ea0 (accessed Dec. 29, 2022).

[4] "Uber Movement: Travel Times Calculation Methodology."

[5] J. Poullet, "Leveraging Machine Learning to Solve the Vehicle Routing Problem with Time Windows," 2020.

[6] S. Hughes, S. Moreno, W. F. Yushimito, and G. Huerta-Cánepa, "Evaluation of machine learning methodologies to predict stop delivery times from GPS data," *Transp. Res. Part C Emerg. Technol.*, vol. 109, pp. 289–304, Dec. 2019, doi: 10.1016/J.TRC.2019.10.018.

[7] "How to Predict Shipments' Time of Delivery with Cloud-based Machine Learning Models | AWS for Industries." https://aws.amazon.com/pt/blogs/industries/how-to-predict-shipments-time-of-delivery-with-cloud-based-machine-learning-models/ (accessed Jan. 01, 2023).

[8] R. Basso, B. Kulcsár, and I. Sanchez-Diaz, "Electric vehicle routing problem with machine learning for energy prediction," *Transp. Res. Part B Methodol.*, vol. 145, pp. 24–55, Mar. 2021, doi: 10.1016/J.TRB.2020.12.007.

[9] M. Alqahtani and M. Hu, "Dynamic energy scheduling and routing of multiple electric vehicles using deep reinforcement learning," *Energy*, vol. 244, p. 122626, Apr. 2022, doi: 10.1016/J.ENERGY.2021.122626.

[10] Y. Guermazi, S. Sellami, and O. Boucelma, "Address Validation in Transportation and Logistics: A Machine Learning Based Entity Matching Approach," *Commun. Comput. Inf. Sci.*, vol. 1323, pp. 320–334, 2020, doi: 10.1007/978-3-030-65965-3_21/COVER.

[11] A. Arishi, K. Krishnan, and M. Arishi, "Machine learning approach for truck-drones based last-mile delivery in the era of industry 4.0," *Eng. Appl. Artif. Intell.*, vol. 116, p. 105439, Nov. 2022, doi: 10.1016/J.ENGAPPAI.2022.105439.