****

**Relatório ALGAV**

Sprint C

**Licenciatura em Engenharia Informática**

**2022/2023**

**Turma 3DA\_Grupo03**

1171060\_André Silva

1200750\_Vasco Silva

1200774\_Cláudia Freitas

121001\_Carlos Rodrigues

**Data: 08/01/2023**

Índice

[Índice Imagens 2](#_Toc124093587)

[Índice Tabelas 2](#_Toc124093588)

[1. Introdução 3](#_Toc124093589)

[2. Criação da população inicial do Algoritmo Genético (AG) 3](#_Toc124093590)

[3. Aleatoriedade no cruzamento entre indivíduos da população 5](#_Toc124093591)

[4. Seleção da nova geração da população 6](#_Toc124093592)

[5. Análise de eficácia 7](#_Toc124093593)

[6. Parametrização da condição de término do AG 9](#_Toc124093594)

[7. Uso do Algoritmo Genético para lidar com vários camiões 10](#_Toc124093595)

[8. Estudo de métodos de Aprendizagem Automática ao problema da distribuição de mercadorias e/ou uso de veículos elétricos 12](#_Toc124093596)

[Introdução Estado de Arte 12](#_Toc124093597)

[Primeiro Artigo 12](#_Toc124093598)

[Introdução ao Estudo 12](#_Toc124093599)

[Segundo Artigo 14](#_Toc124093600)

[Resultados 18](#_Toc124093601)

[Outros Artigos 20](#_Toc124093602)

[Conclusão Estado de Arte 20](#_Toc124093603)

[Referências 20](#_Toc124093604)

## Índice Imagens

[Figura 1 Algoritmo para agrupar clientes em Clusters 12](file:///C:\Users\asilv\OneDrive\Documentos\3ano\kek_try\Projeto\lei-22-s5\Relatórios\ALGAV\SprintC\Relatório%20ALGAV.docx#_Toc123894774)

[Figura 2 Algoritmo K-Means melhorado 12](file:///C:\Users\asilv\OneDrive\Documentos\3ano\kek_try\Projeto\lei-22-s5\Relatórios\ALGAV\SprintC\Relatório%20ALGAV.docx#_Toc123894775)

[Figura 3 Visão geral da metodologia aplicada 15](#_Toc123894776)

[Figura 4 Comparação dos diferentes modelos de ML com solução ótima (DP) 17](file:///C:\Users\asilv\OneDrive\Documentos\3ano\kek_try\Projeto\lei-22-s5\Relatórios\ALGAV\SprintC\Relatório%20ALGAV.docx#_Toc123894777)

## Índice Tabelas

[Tabela 1 Aumento percentual na utilização do veículo com diminuição dos custos de armazenamento das encomendas 12](#_Toc123894840)

[Tabela 2 Aumento da percentagem na utilização de veículos com o aumento da capacidade de carga dos veículos 13](#_Toc123894841)

[Tabela 3 Distância total e máxima numa viagem pelos veículos (em KM) 14](#_Toc123894842)

[Tabela 4 Especificações do veículo usado para os cálculos 14](#_Toc123894843)

[Tabela 5 Custo e redução % dos preços para gasolina e modelos DP e AC 16](#_Toc123894844)

## Introdução

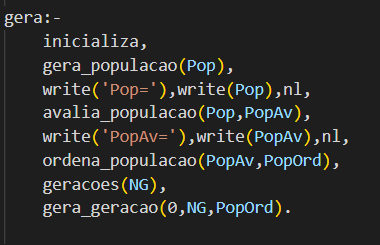
Este relatório foi realizado para o âmbito da UC de ALGAV e com vista ao Sprint C do projeto integrador.

Os objetivos para este Sprint C eram a construção de um Algoritmo Genético para o mesmo problema apresentado no Sprint B (caixeiro-viajante), era fornecido um AG base e era necessário realizar alterações com o objetivo de tornar este algoritmo melhor e mais eficaz, após isto era necessário adaptar ao caso em que é necessário lidar com vários camiões. Estes objetivos foram cumpridos estando todos descritos no relatório de forma bastante detalhada, como também a comparação do AG base com o AG alterado. Para finalizar, um último objetivo era realizar um estado de arte sobre o Machine Learning, e como podíamos aplicar ao problema descrito, objetivo também cumprido.

Deste modo, achamos que conseguimos cumprir com todos os objetivos propostos de forma positiva.

## Criação da população inicial do Algoritmo Genético (AG)

A geração da população inicial é essencial para o algoritmo genético, pois é a partir desta população inicial que se irá desenvolver o resto do algoritmo. Em baixo vamos mostrar todo o processo para que isto aconteça. O primeiro predicado é o ***gera.*** Este é um predicado bastante importante pois é o predicado que inicializa todo o algoritmo genético. Como o que nos interessa neste tópico é a população inicial vamos olhar apenas para a parte que diz respeito a este assunto, ou seja o predicado ***gera\_populacao(Pop)*** e o ***avalia\_populacao(Pop, PopAv).***



O primeiro tem como finalidade criar a população inicial, para isto decidimos adotar a seguinte abordagem:

* Calcular 3 indivíduos através das 3 heurísticas feitas anteriormente;
* Gerar o resto da população aleatoriamente através de outro predicado.

Uma imagem com texto

Descrição gerada automaticamente

O predicado começa por ir buscar o tamanho da população ao predicado ***populacao(TamPop)*** e o numero de entregas ao predicado ***entregas(NumT)***,após isto, faz um findall de todas as entregas e chama as três heurísticas feitas no sprint passado, reduz em 3 o tamanho da população e assim temos o tamanho da população restante ( é importante salientar que o tamanho mínimo para uma população é de **3**, pois assim decidimos devido às heurísticas e porque não faria sentido estar a permutar apenas 2 indivíduos). Depois do especificado anteriormente, chamamos o predicado ***gera\_populacao*** para gerar o restante da população.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Este predicado vai gerar indivíduos de forma aleatória e garantir que não são criados indivíduos repetidos. Para finalizar o predicado ***gera\_populacao*** faz **append** dos indivíduos das heurísticas aos indivíduos criados aleatoriamente.

Contudo a lista deve estar com o custo associado logo foi desenvolvido o predicado ***avalia\_populacao.***

Uma imagem com texto

Descrição gerada automaticamente

Basicamente vai individuo a individuo e verifica o seu custo através do ***calcula\_custo* (sprint anterior)**, adicionando um \*Custo a todos.

## Aleatoriedade no cruzamento entre indivíduos da população

Para cumprir este requisito decidimos optar pelo ***random\_permutation***, predicado este fornecido pelo prolog, e resumidamente pega numa lista e faz permutações aleatórias.

Desta forma, conseguimos garantir que existe uma aleatoriedade, pois quando mandamos a lista para cruzar os pontos, ele vai cruzar os pontos 1 e 2, 3 e 4 etc., mas como houve uma permutação anterior é impossível saber quais os indivíduos que vão ser cruzados.

Uma imagem com texto

Descrição gerada automaticamente

O cruzamento acontece no predicado ***cruzamento***, que tal como explicado vai pegar nos elementos seguidos e cruzar.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

## Seleção da nova geração da população

Para este efeito foi utilizado um predicado com o nome ***roleta***.

Em suma, este predicado vai primeiramente retirar o melhor individuo da população e aos restantes vai fazer uma multiplicação do seu custo por um número random de 0 a 1 para dar oportunidade de todos passarem para a próxima geração. O predicado ***escolhe*** vai pegar nos N vencedores do torneio.

O predicado responsável por esta multiplicação é o ***roleta1*** e apenas tem este fim.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Deste modo garantimos total possibilidade de todos os individuos passarem à próxima geração, mas também garantimos que o melhor passa sempre e também que os que tem melhor custo tem mais possibilidades de passar.

## Análise de eficácia

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **6 entregas** | **7 entregas** | **8 entregas** | **9 entregas** | **10 entregas** | **11 entregas** |
| **Valor da melhor solução total do problema** | 507.06 | 626.61 | 705.43 | 744.93 | 845.34 | 913.32 |
| **Valor da melhor solução/indivíduo da população final do AG desenvolvido** | 575.13 | 703.78 | 758.02 | 886.76 | 970.36 | 1127.21 |
| **Valor da melhor solução/indivíduo da população final do AG base** | 586.63 | 775.13 | 833.20 | 999.80 | 1134.17 | 1149.22 |
| **Valor médio de todas as soluções/indivíduos da população final do AG desenvolvido** | 653.67 | 789.17 | 881.54 | 1018.10 | 1207.50 | 1305.17 |
| **Valor médio de todas as soluções/indivíduos da população final do AG base** | 695.46 | 827.98 | 950.75 | 1133.67 | 1232.76 | 1456.78 |

**Entregas utilizadas:**

6 entregas - 6449,6438,6398,6445,6443,6439

7 entregas - 6449,6438,6398,6445,6443,6439,6432

8 entregas - 6437, 6449,6438,6398,6445,6443,6439,6432

9 entregas – 6451, 6437, 6449,6438,6398,6445,6443,6439,6432

10 entregas- 6452, 6451, 6437, 6449,6438,6398,6445,6443,6439,6432

11 entregas – 6444, 6452, 6451, 6437, 6449,6438,6398,6445,6443,6439,6432

**Forma como foi utilizado o AG, para a obtenção dos valores da tabela:**

Número de novas Gerações: 10.

Dimensão da População: 6.

Probabilidade de Cruzamento (%): 75.

Probabilidade de Mutacao (%): 15.

Podemos concluir coma a análise da tabela que o AG desenvolvido chega a valores bastante interessantes tendo em conta o valor da melhor solução, e sabendo que foram usados somente 6 indivíduos por geração e 10 gerações. Obviamente, os valores obtidos pelo AG mesmo sendo positivos no nosso ponto de vista raramente chegam à solução ótima e fica cada vez mais difícil quando sobe o número de entregas, para que o valor se aproxime devemos aumentar o número de gerações e indivíduos por geração.

O AG de base como podemos ver tem sempre valores piores, isto porque está bastante simplificado, sendo que a geração inicial não conta com as heurísticas ao contrário do alterado, os melhores indivíduos melhores não passam automaticamente, entre outras alterações feitas por nós, que no final fizeram toda a diferença.

Concluindo, o AG alterado tem resultados muito melhores comparado com o AG base fornecido, isto deve-se ao facto das alterações e melhorias feitas, quando comparado com a melhor solução tem algumas diferenças, diferenças estas consideradas normais e achamos que no geral os resultados foram bastante positivos.

## Parametrização da condição de término do AG

Contamos com 2 condições possíveis de término e que trabalham de forma simultânea.

A primeira é se atingirmos um determinado valor ou um número inferior no custo. Para isto no predicado ***gera\_geracao*** temos uma verificação que verifica se o melhor valor da geração anterior não passa neste requisito. Mas é possível nunca passar este requisito e por isso temos outra condição de término.

Uma imagem com texto

Descrição gerada automaticamente

Esta condição é cumprida através de um predicado de término e basicamente este predicado é cumprido quando um número de gerações máximo é atingido. A cada geração é adicionado 1 a um variável n e esta condição é cumprida quando o n chega ao valor de g (número máximo de gerações, introduzido pelo utilizador).



Assim, garantimos duas condições diferentes de término tal como pedido e sem ficar preso num *Loop* infinito a procurar um valor impossível de ser alcançado.

## Uso do Algoritmo Genético para lidar com vários camiões

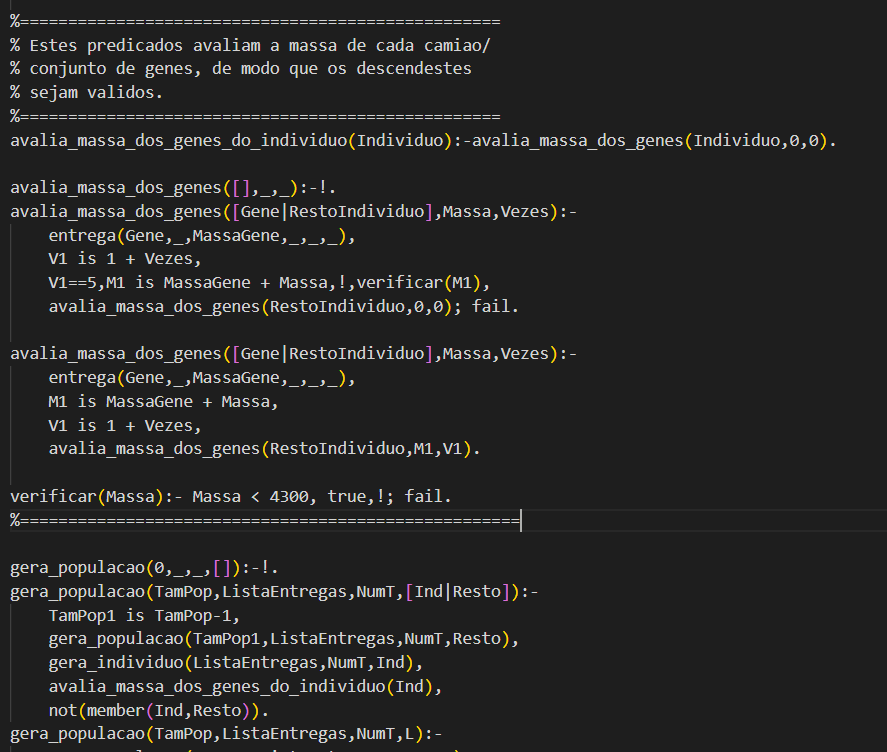
Para esta tarefa, o algoritmo genético teve de sofrer algumas modificações, sendo elas:

1. **Validar os indivíduos:**

Assumindo que em cada individuo, do gene 1 ao 5, do 6 ao 10 e do 11 ao 16 corresponde, respetivamente, ao camião 1,2 e 3. Então em cada conjunto de genes o peso não pode ser superior à capacidade de carga de cada camião.

Como todos os camiões apresentam a mesma capacidade de carga, então verificou-se se o descendente era valido, ou seja, se em cada conjunto de genes do individuo a massa não era superior a 4300.

Caso fosse superior, o descendente era eliminado, não fazendo parte da população.



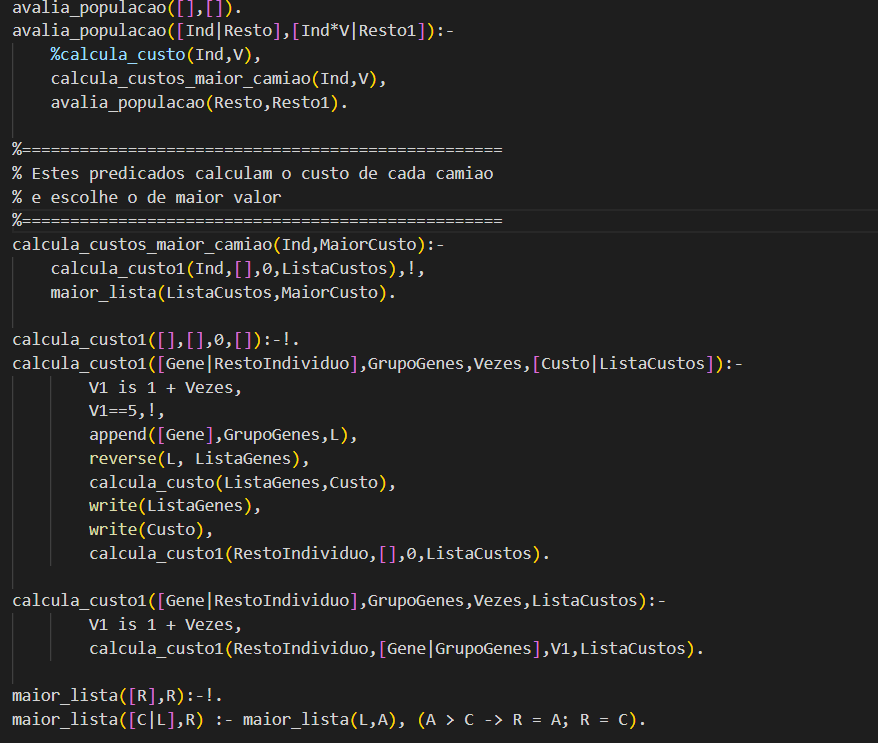
O predicado avalia\_massa\_dos\_genes\_do\_individuo, contem o individuo gerado e vai validar.

Caso esta validação falhe, o individuo não será agregado à população.

2. **Avaliação dos indivíduos:**

Após a validação do individuo, caso este seja descendente ocorre a face de avaliação.

Uma vez mais, do gene 1 ao 5 é calculado o custo, tal como do gene 6 ao 10 e do gene 11 ao 16. Depois dos três custos, é assumido como custo do individuo o maior custo.



3. **Encontro do melhor individuo**

Por fim, para obtermos a melhor solução de planeamento, encontramos o melhor descendente. Este encontrasse na última geração e é o que apresenta menor custo.

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

### Introdução Estado de Arte

Com base na nossa pesquisa, o conceito de Machine Leaning (aprendizagem automática) define um ramo da inteligência artificial que estuda a forma como os sistemas computacionais aprendem e desenvolvem capacidades, de acordo com a experiência e reconhecimento de padrões.

“Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience.” [1]

De forma a compreender melhor a aplicação de *Machine Learning* no mundo real e relacionando com o trabalho que desenvolvemos, pesquisamos artigos científicos onde este ramo é usado ao nível do planeamento da distribuição de mercadorias e gestão de veículos elétricos. 2 dos artigos são analisados em alguma profundidade e referenciamos 5 artigos relacionados com o tema.

### Primeiro Artigo

Artigo: [Clustering model and algorithm for production inventory and distribution problem](https://doi.org/10.1016/j.apm.2013.05.029) [2]

### Introdução ao Estudo

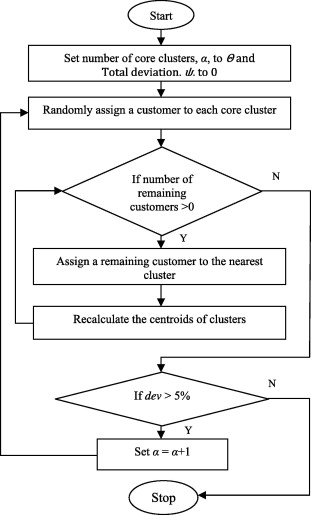
De acordo com a pesquisa efetuada o primeiro artigo que achamos importante realçar foca-se no uso de métodos de *clustering* para reduzir os custos na distribuição de entregas de uma frota.

*Clustering* é uma técnica de *machine learning* em que um conjunto de dados é dividido em grupos (*clusters*) baseado nas suas semelhanças e identificação de padrões. Existem diversos métodos de *clustering* dentro os quais se realçam: *k-means*, *hierarchical clustering* e *expectation maximisation*. [3] [4]

Neste artigo é usado o método *k-means* para agrupar uma lista de clientes em *clusters* tendo em consideração a distância entre os clientes, capacidade de carga e tempo máximo de entrega com o objetivo de minimizar os custos de transporte e manutenção das encomendas.

#### Descrição do Estudo

Primeiramente os clientes foram atribuídos a *clusters* com base na sua localização geográfica. O algoritmo para agrupar os clientes está resumido na Figura 1.



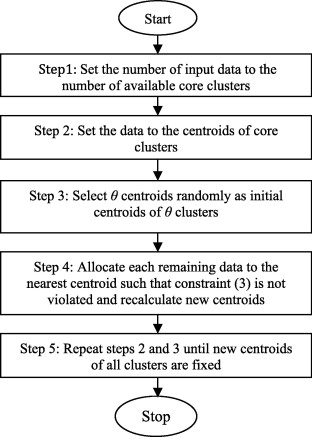


Figura Algoritmo para agrupar clientes em Clusters

No passo seguinte, os *clusters* finais dos clientes são criados usando um algoritmo *K-Means* melhorado (Figura 2).

Figura Algoritmo K-Means melhorado

O propósito do algoritmo melhorado é pegar nos *clusters* iniciais e refiná-los tendo em consideração os custos. Centroides iniciais para os *clusters* são escolhidos aleatoriamente e os objetos restantes são alocados para o centroide mais próximo que não viola as limitações impostas. O algoritmo é repetido até os centroides de todos os *clusters* estarem arranjados. A solução final é depois usada no *tabu search algoritm*.

Em *tabu search* é procurado melhorar a solução atual.

Tabela Aumento percentual na utilização do veículo com diminuição dos custos de armazenamento das encomendas

Uma imagem com mesa

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Tabela Aumento da percentagem na utilização de veículos com o aumento da capacidade de carga dos veículos

Como podemos ver, a utilização média de 1 veículo aumenta com a diminuição dos custos de manutenção das encomendas (Tabela 1) e com o aumento da capacidade de carga (Tabela 2), pois, ao diminuir o custo ou aumentar a capacidade de carga, podem ser agrupadas mais encomendas por cada utilização, tornando os veículos mais eficientes e reduzindo assim o custo das operações.

#### Conclusão do Estudo

Os resultados mostram que o modelo proposto pode ser usado para agrupar os clientes por *cluster* mais efetivamente do que usando apenas as localizações geográficas, dando uma melhor qualidade na solução.

### Segundo Artigo

Artigo: [Demand-Side Management Using Deep Learning for Smart Charging of Electric Vehicles](https://ieeexplore.ieee.org/document/8299470) [5]

#### Introdução ao Estudo

O segundo artigo escolhido utiliza várias técnicas de *machine learning* (ML) para calcular a maneira mais eficiente de carregar a bateria de veículos elétricos sendo uma dessas técnicas *deep leaning*.

Métodos *deep learning* inspiram-se no funcionamento e estrutura do cérebro, mais especificamente nas redes neuronais. Envolve treinar redes neuronais artificiais com grandes quantidades de dados, permitindo que a rede aprenda e faça decisões inteligentes por si própria. [6]

No contexto duma rede elétrica inteligente, *Demand-Side Management* (DSM) é uma estratégia para controlar o consumo de energia dos consumidores encorajando-os a mudar os seus hábitos energéticos. Desta maneira, tenta-se que a energia seja usada de maneira mais eficiente para puder responder ao aumento da procura. [7]

Dado o rápido crescimento na adoção de veículos elétricos (VE), o risco de sobrecarregar a rede energética nos picos de uso de energia aumenta. Existe, por isso, um interesse para integrar DSM no carregamento de VE usando, por exemplo, preços dinâmicos (reduzir os preços da energia nas horas de menor consumo).

Este estudo propõe uma resposta baseada em ML para controlar o carregamento de VE em resposta aos preços da energia em tempo-real para que os custos sejam minimizados.

#### Descrição do Estudo

Para testar este modelo foram usados os dados GPS de 17 veículos diferentes (número de viagens, data, tempo, posição, velocidade e velocidade máxima permitida na localização do veículo) no verão de 2008 e inverno de 2009. As distâncias percorridas por cada veículo, assim como a distância máxima de uma viagem podem ser vistas na seguinte tabela:

Uma imagem com mesa

Descrição gerada automaticamente

Tabela Distância total e máxima numa viagem pelos veículos (em KM)

Destes dados foram simuladas viagens e sessões de carregamento atuando como se todos os veículos tivessem as mesmas especificações, particularmente a bateria.

Uma imagem com mesa

Descrição gerada automaticamente

Tabela Especificações do veículo usado para os cálculos

Neste estudo são comparados os dados de 2 períodos de tempo: verão 2008 e inverno 2009. Os dados dos meses de junho e julho 08 são usados como dados de treino para o verão e agosto 08 o mês que se pretende testar. Da mesma forma, dezembro 08 e janeiro 09 são usados como treino para o inverno e fevereiro 09 como teste.

Primeiro foi criado um modelo para calcular as ações ótimas com total conhecimento da informação atual e futura usando *Dynamic Programming* (DP).

DP é um método para resolver problemas de otimização repartindo-os em problemas mais pequenos e guardando as soluções. As soluções dos *subproblems* são depois combinadas para resolver o problema principal. [8]

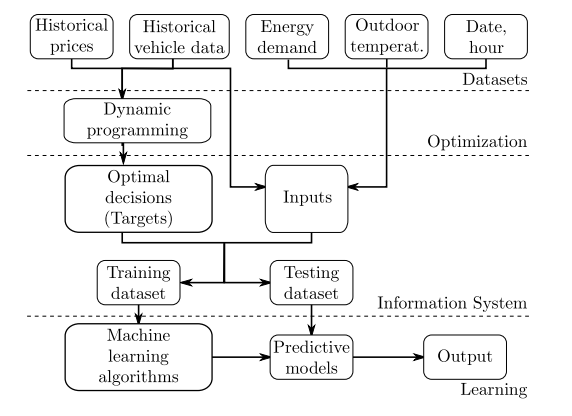


Figura Visão geral da metodologia aplicada

Os valores obtidos com DP não conseguem ser atingidos em situações reais, pois a sua computação necessita de conhecimento perfeito do futuro.

As ações ótimas calculadas com DP tornam-se os alvos desejados para aprender sendo algoritmos de aprendizagem usados para inferir modelos que conseguem tomar decisões em tempo real para este problema. Os métodos de aprendizagem usados foram: *Threshold-Based Rule* (TBR)*, k-Nearest Neighbors* (KNN)*, Shallow Neural Network* (SNN)*, and Deep Neural Networks* (DNN).

### Resultados

Primeiro foi testado o modelo *Always Charge* (AC) em o veículo carrega mal esteja estacionado desde que a bateria não esteja cheia. A seguinte tabela mostra cada veículo para 3 situações: 1) depende de gasolina, 2) estratégia ótima (DP) e 3) modelo AC.

Uma imagem com texto, recibo

Descrição gerada automaticamente

Tabela Custo e redução % dos preços para gasolina e modelos DP e AC

Como podemos verificar a simples estratégia AC apresenta uma redução média do custo de 68% e 56% no verão e inverno respetivamente. No entanto, existe uma diferença significativa para a estratégia ótima que consegue resultados de 95% e 88% no verão e inverno.

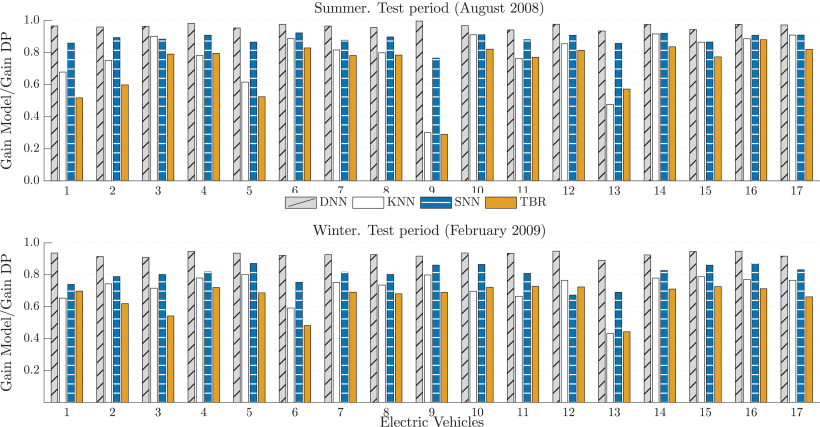
Foram depois testados os modelos de *Machine Learning*.

Figura Comparação dos diferentes modelos de ML com solução ótima (DP)

A figura 4 mostra o rácio entre modelos ML e DP. Um valor perto de 1 indica um modelo com uma eficácia perto da solução ótima gerada com DP.

A abordagem DNN mostra claramente ser a melhor com um rácio médio perto de 0.95, tendo um resultado tão bom como a solução ótima para o veículo 9 durante o verão. No entanto, ao contrário do DP que necessita de conhecimento prévio do futuro, o modelo DNN pode ser aplicado no mundo real.

Outros métodos como SNN também mostraram boas prestações com ratios médio de 0.88 e 0.80, sendo esta média apenas 8% mais baixa que o DNN.

Finalmente, o método KNN apresenta rácios médios de 0.77 e 0.72 e o método TBR 0.71 e 0.66. Apesar de estes métodos não atingirem a performance do DNN ou do SNN continuam a ser uma melhoria substancial do que a simples estratégia de AC.

#### Conclusão do Estudo

Os resultados mostram que o modelo baseado em *Deep Neural Network* apresentou os melhores resultados.

Pode-se também verificar que quanto maior a complexidade da técnica utilizada, melhor foram os resultados.

### Outros Artigos

Outros artigos que achamos importante destacar:

[Optimal delivery routing with wider drone-delivery areas along a shorter truck-route](https://doi.org/10.1016/j.eswa.2018.03.032) [9] – estuda o planeamento de rotas na distribuição de entregas usanto drones e camiões.

[Electric vehicle routing problem with machine learning for energy prediction](https://doi.org/10.1016/j.trb.2020.12.007) [10] – estuda o cálculo de rotas para veículos elétricos com possíveis carregamentos parciais.

[Large-scale constrained clustering for rationalizing pickup and delivery operations](https://doi.org/10.1016/j.trb.2008.10.003) [11] – estuda o uso de *clusters* para reduzir os custos duma empresa de distribuição em operações de recolha e envio de encomendas.

[Evaluation of machine learning methodologies to predict stop delivery times from GPS data](https://doi.org/10.1016/j.trc.2019.10.018) [12] – estuda a previsão do tempo de entrega usando métodos de *machine learning*.

[Delivery Management System Based on Vehicles Monitoring and a Machine-Learning Mechanism](https://doi.org/10.1109/vtcfall.2018.8690619) [13] – utiliza *machine learning* para aumentar a eficiência e diminuir os tempos de entrega em cenários de *food delivery*.

### Conclusão Estado de Arte

Existe uma forte investigação nesta área com novos estudos e aplicações de *machine learning* no tema de distribuição de mercadorias e gestão de veículos elétricos a aparecer com bastante frequência. Algo que deve continuar/aumentar no futuro.

### Referências

[1] Machine Learning Definition | Tom M. Mitchell| McGraw-Hill Science/Engineering/Math; (March 1, 1997), Page 1 | <http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html>

[2] Nananukul, N. (2013). Clustering model and algorithm for production inventory and distribution problem. Applied Mathematical Modelling, 37(24), 9846–9857. <https://doi.org/10.1016/j.apm.2013.05.029>

[3] Ramos, C. (n.d.). 6 - Domínios da IA 6.4 - Aprendizagem Automática [12]

[4] Google. (n.d.). What is clustering? &nbsp;|&nbsp; machine learning &nbsp;|&nbsp; google developers. Google. Retrieved January 5, 2023, from <https://developers.google.com/machine-learning/clustering/overview>

[5] Lopez, K. L., Gagne, C., &amp; Gardner, M.-A. (2019). Demand-side management using Deep Learning for smart charging of electric vehicles. IEEE Transactions on Smart Grid, 10(3), 2683–2691. <https://doi.org/10.1109/tsg.2018.2808247>

[6] Brownlee, J. (2020, August 14). A tour of machine learning algorithms. MachineLearningMastery.com. Retrieved January 6, 2023, from <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms>

[7] What is demand side management? Enel X. (n.d.). Retrieved January 6, 2023, from <https://corporate.enelx.com/en/question-and-answers/what-is-demand-side-management>

[8] Kafkes, A. (2017, June 1). Demystifying dynamic programming. freeCodeCamp.org. Retrieved January 5, 2023, from <https://www.freecodecamp.org/news/demystifying-dynamic-programming-3efafb8d4296/>

[9] Chang, Y. S., &amp; Lee, H. J. (2018). Optimal Delivery Routing with wider drone-delivery areas along a shorter truck-route. Expert Systems with Applications, 104, 307–317. <https://doi.org/10.1016/j.eswa.2018.03.032>

[10] Basso, R., Kulcsár, B., &amp; Sanchez-Diaz, I. (2021). Electric vehicle routing problem with machine learning for energy prediction. Transportation Research Part B: Methodological, 145, 24–55. <https://doi.org/10.1016/j.trb.2020.12.007>

[11] Bard, J. F., &amp; Jarrah, A. I. (2009). Large-scale constrained clustering for rationalizing pickup and delivery operations. Transportation Research Part B: Methodological, 43(5), 542–561. <https://doi.org/10.1016/j.trb.2008.10.003>

[12] Hughes, S., Moreno, S., Yushimito, W. F., &amp; Huerta-Cánepa, G. (2019). Evaluation of machine learning methodologies to predict stop delivery times from GPS Data. Transportation Research Part C: Emerging Technologies, 109, 289–304. <https://doi.org/10.1016/j.trc.2019.10.018>

[13] Habault, G., Taniguchi, Y., &amp; Yamanaka, N. (2018). Delivery management system based on vehicles monitoring and a machine-learning mechanism. 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall). <https://doi.org/10.1109/vtcfall.2018.8690619>