A picture containing shape

Description automatically generated

**Relatório Sprint C**

**Turma 3DG-Grupo 37**

1161274- Artur Muiria

1190827- Luís Araújo

1191240- Danilton Lopes

1201183- João Oliveira

**Professor:**

Ana Vieira, AIV

**Unidade Curricular:**

Algoritmia Avançada

**Data: 08/01/2023**

Índice

[**Introdução** 1](#_Toc124105431)

[**Uma solução para o problema de planeamento usando Algoritmos Genéticos** 1](#_Toc124105432)

[Alterações nos Algoritmos Genéticos 1](#_Toc124105433)

[**O predicado gera** 2](#_Toc124105434)

[**O predicado gera\_populacao** 3](#_Toc124105435)

[**O predicado gera\_individuo** 4](#_Toc124105436)

[**Os predicados avalia\_populacao e avalia** 5](#_Toc124105437)

[**O predicado gera\_geracao** 5](#_Toc124105438)

[**O predicado melhor\_cruzamento e escolher\_das\_geracoes** 6](#_Toc124105439)

[**Análise de qualidade** 6](#_Toc124105440)

[Predicado atribuir\_lotes 6](#_Toc124105441)

[**Estudo bibliográfico sobre Aprendizagem Automática (“*Machine Learning”*) 7**](#_Toc124105442)

[**Bibliografia 10**](#_Toc124105443)

**Introdução**

O presente relatório tem como objetivo apresentar como foi feita a adaptação dos algoritmos genéticos para gerar um planeamento de entregas mais eficiente e também como foi utilizado a solução adaptada para que haja um planeamento para uma frota de camiões. O trabalho apresentou algumas dificuldades mas, no fim, foi implementado com sucesso.

**Uma solução para o problema de planeamento usando Algoritmos Genéticos**

Neste tópico será explicado como foi o processo de adaptação dos algoritmos genéticos para o planeamento de entregas. Neste foi também desenvolvido um predicado capaz de tratar de planeamento de entregas de um camião ou de frota.

**Alterações nos Algoritmos Genéticos**

De facto, foram feitas algumas alterações aos predicados que foram dados como base pelos professores para que se conseguisse fazer o planeamento das entregas. As alterações que foram realizadas estão nos predicados, e foram também adicionados novos predicados com o mesmo propósito, mas com número diferente de argumentos:

* **gera**;
* **gera\_populacao** (os dois predicados, com número diferente de argumentos);
* **gera\_individuo**;
* **avalia\_populacao**;
* **avalia**;
* **gera\_geracao**;
* entre outros que foram alterados o mínimo para que se possa manter a coerência, que seriam alteração de “tarefas” para “entregas”.

Também foram adicionados novos predicados para se poder obter o melhor caminho desenvolvido com a solução: o predicado **melhor\_cruzamento**, que utiliza o predicado para comparar a melhor solução atual com as gerações obtidas e ver se precisa substituir ou não e o predicado **escolher\_das\_geracoes**.

**O predicado gera**

Uma imagem com texto

Descrição gerada automaticamente

*Figura 1 - Predicado gera*

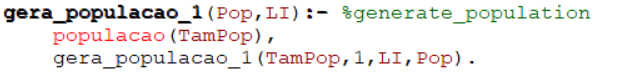
Diferente do método que nos havia sido oferecido, este passou a ser menos interativo para que possa ser realizado posteriormente para suportar pedidos do tipo HTTP. Este recebe os dados para que seja realizado o planeamento das entregas num determinado dia, e outras irá buscar à base de dados.

Os argumentos utilizados serão:

* **ListaGenes**, que representa as entregas do dia determinado no pedido;
* **NumGeracoes**, que representa a quantidade de gerações pretendidas pelo gestor de logística que deseja o planeamento;
* **TamGenes**, será calculado no momento do pedido ou depois quando o predicado de planeamento de frotas chama o método gera;
* **TamPop**, representa o tamanho da população a ser desenvolvido. Uma das coisas que durante os testes foram observadas foram: o tamanho de população não funcionava bem, ou simplesmente não funcionava quando o tamanho da população era maior que o tamanho de genes. Por isso o número de genes e o tamanho da população terão o mesmo valor;
* **ProbC**, que representa a probabilidade de cruzamento;
* **ProbM**, que representa a possibilidade de mutação;
* **Geracoes**, é uma variável não declarada que depois receberá todos as gerações geradas pelo predicado de **gera\_geracoes**;
* **MelhorCaminho**, que representará a melhor solução encontrado pelo método.

O método inicia-se com a eliminação dos valores de gerações, entregas, população, probabilidade de mutação e cruzamento que já existem na base de conhecimento para os valores que recebe nos parâmetros. De seguida faz chamada de conjunto de predicado na ordem especifica que já era predefinida pelos algoritmos genéticos. Estes processos são: a obtenção da **população**, a **avaliação** da população, a **ordenação** aleatória e assim garantimos que o cruzamento seja feita de aleatória sempre, faz as **gerações** da população, avalia-os e vai **buscar** a **melhor solução** gerada.

**O predicado gera\_populacao**



*Figura 2 – 1º Predicado gera\_populacao*

Este predicado recebe como parâmetro uma variável não declarada, **Pop,** que representa a população gerada, e a variável com a lista das entregas que serão processadas para a criação de população.

Os passos seguintes é a busca do tamanho da população e por fim chama o predicado com mesmo nome, mas com número de argumentos diferente.

Uma imagem com texto

Descrição gerada automaticamente

*Figura 3 - 2º Predicado gera\_populacao*

Os argumentos que este predicado recebe são:

* **TamPop**, que representa o tamanho da população a ser criada;
* **OpIndex**, representa o índice da opção de como gerar um individuo, sendo as 3 primeiras das heurísticas, e o restante será gerado aleatoriamente;
* **LG**, a lista de genes para a criação de indivíduos;
* **Resto**, que representará uma variável não instanciada para receber a população que será gerada.

A seguir, o predicado segue as seguintes instruções com o objetivo de gerar indivíduos. Estas instruções fazem, sequencialmente, o seguinte:

Diminui o número do tamanho de população para a recursividade, aumenta o número de índice da opção para gerar o individuo. E a seguir faz a recursividade com os dados alterados e o que deveria se manter, depois chama o predicado de para gerar o individuo e caso já exista ele tenta novamente.

**O predicado gera\_individuo**

Uma imagem com texto

Descrição gerada automaticamente

*Figura 4 - Predicado gera\_individuo*

Este predicado tem como objetivo gerar um individuo, no qual podemos ver que nos 3 primeiros predicados utilizam os predicados desenvolvidos para tratar das heurísticas. Já o seguinte irá gerar um individuo com os genes organizados de forma aleatória.

Uma imagem com texto

Descrição gerada automaticamente

*Figura 5 - Predicado gera\_individuo\_aleatorio*

Este predicado, como dito anteriormente, coloca os genes de forma aleatória. Pelas instruções a serem seguidas, ele gera um número aleatório, que esteja entre 1 e o tamanho dos genes + 1, e depois busca-se o elemento que esteja naquela posição da lista. Quando encontra, devolve e retira-se da lista, diminui-se o tamanho das entregas e faz-se a recursividade.

**Os predicados avalia\_populacao e avalia**

Este predicado irá fazer a avaliação da população gera e posteriormente as gerações que foram criadas.

Uma imagem com texto

Descrição gerada automaticamente

*Figura 6 - Predicado avalia\_populacao e avalia*

Para o método avalia utiliza-se o predicado do sprint passado para avaliar o planeamento. Há uma necessidade de transformar a lista de genes da população para uma lista de armazém, uma vez que o predicado do custo irá fazer o tempo total que seria para percorrer os armazéns que criaram a entrega.

**O predicado gera\_geracao**

Uma imagem com texto

Descrição gerada automaticamente

*Figura 7 - Predicado gera\_geracao*

Este predicado sofreu mínimas alterações, tendo apenas de ser colocado uma variável não instanciada para receber as gerações geradas para no fim serem comparadas e escolher a melhor.

**O predicado melhor\_cruzamento e escolher\_das\_geracoes**

Uma imagem com texto

Descrição gerada automaticamente

*Figura 8 - Predicado melhor\_cruzamento e escolher\_das\_geracoes*

Para escolher o melhor caminho gerado foram desenvolvidos 2 predicado: um para comparar o atual melhor resultado com os que estão na geração total e outro para fazer com todos as gerações que foram geradas.

**Análise de qualidade**

Com análise de nós obtemos os seguintes resultados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Quant. Entregas | Melhor Heuritica | Custo (Min) | Custo GA (Min) | Tempo GA |
| 6 | Massa e Distancia | 491,1898 | 413,4138 | 0,0018 |
| 7 | Distancia | 537,7494 | 445,4983 | 0,0032 |
| 8 | Distancia | 584,2695 | 499,9972 | 0,0044 |
| 9 | Distancia | 621,5595 | 492,7460 | 0,0044 |
| 10 | Distancia | 674,6593 | 535,2424 | 0,005 |
| 11 | Distancia | 672,6589 | 544,6530 | 0,0071 |
| 12 | Distancia | 723,7246 | 518,2585 | 0,0075 |

O que se pode concluir é que, com desde os algoritmos genéticos, como esperado, apresentam a melhor solução do planeamento das entregas, tendo um desempenho com até uma diferença de mais de 3 horas.

**Predicado atribuir\_lotes**

Uma imagem com texto

Descrição gerada automaticamente

Este predicado busca distribuir uma lista de entregas à uma lista de camiões, no qual como parâmetro:

* **LC**, representa a lista de camiões;
* **NG**, que representa a quantidade gerações desejadas pelo utilizador;
* **ProbC**, representa a probabilidade de cruzamento;
* **ProbM**, representa a probabilidade de mutação;
* **Geracoes**, representa as gerações geradas pelos algoritmos genéticos;
* **MCS**, representa o melhor caminho gerado.

Depois de realizado distribuir as entregas entre os camiões ela chama o predicado que vai dividir as entregas e fazer o planeamento destas entregas.

Uma imagem com texto

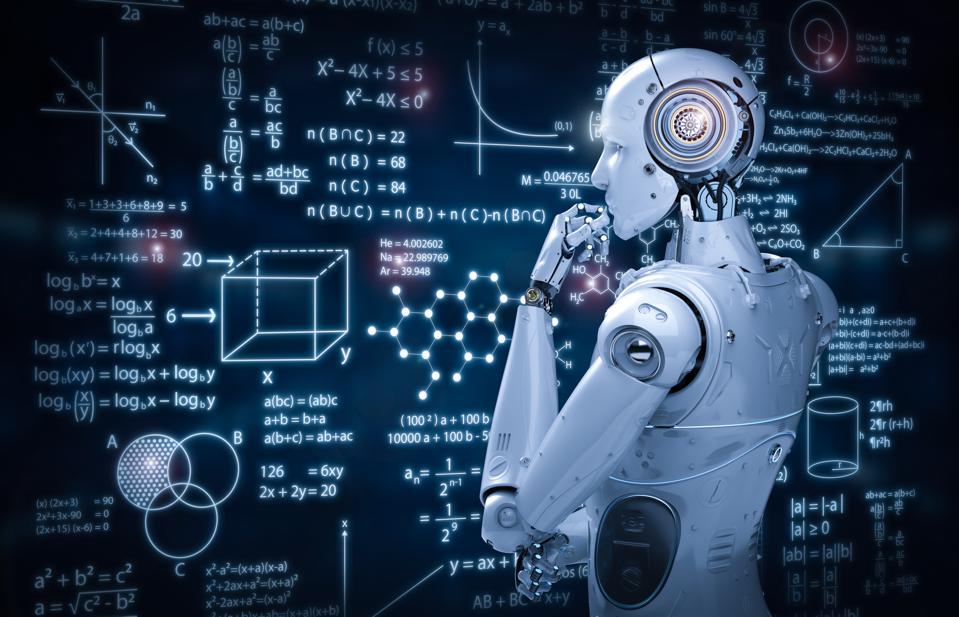
Descrição gerada automaticamente

# Estudo bibliográfico sobre Aprendizagem Automática (“*Machine Learning”*)

A aprendizagem automática é um tipo de inteligência artificial (*AI*) e foi criado com a evolução da tecnologia, e que, até ao dia de hoje, está a ser aperfeiçoada. Permite que as aplicações de software fiquem mais precisas na previsão de resultados sem serem programados para tal. Os seus algoritmos utilizam dados históricos como entrada para conseguir prever novos valores de saída.

O termo *Machine Learning* veio de Arthur Samuel, cientista da computação da IBM e pioneiro em IA (Inteligência Artificial) e jogos de computador. Samuel criou um programa de computador para jogar damas. Quanto mais o programa jogava, mais ele aprendia com a experiência, com o uso de algoritmos que fazia previsões.

O *ML* mostrou ser valioso porque pode resolver problemas numa velocidade e escala que não pode ser duplicada pela mente humana. Com grandes quantidades de capacidade computacional por trás de uma única tarefa ou várias tarefas específicas, as máquinas podem ser treinadas para identificar padrões e relacionamentos entre dados de entrada e automatizar processos de rotina.



De facto, existem quatro abordagens básicas: aprendizagem supervisionada, aprendizagem sem supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço. O tipo de algoritmo que os cientistas de dados escolhem utilizar depende do tipo de dados que desejam prever. Os 4 tipos de aprendizagens são:

* **Aprendizagem supervisionada**: Neste tipo de aprendizagem automática, os cientistas de dados fornecem algoritmos com dados de formação rotulados e definem as variáveis que querem que o algoritmo avalie para as correlações. Especificada tanto a entrada como a saída do algoritmo;
* **Aprendizagem sem supervisão**: Este tipo de aprendizagem automática envolve algoritmos que treinam em dados não rotulados. O algoritmo verifica através de conjuntos de dados à procura de qualquer ligação significativa. Os dados que os algoritmos treinam, bem como as previsões ou recomendações que eles output são pré-determinados;
* **Aprendizagem semi-supervisionada**: Esta abordagem ao *Machine Learning* envolve uma mistura dos dois tipos anteriores. Os cientistas de dados podem alimentar um algoritmo maioritariamente rotulado de dados de treino, mas o modelo é livre para explorar os dados por si só e desenvolver a sua própria compreensão do conjunto de dados;
* **Aprendizagem de reforço**: Os cientistas de dados normalmente usam a aprendizagem de reforço para ensinar uma máquina a completar um processo em várias etapas para o qual existem regras claramente definidas. Os cientistas de dados programam um algoritmo para completar uma tarefa e dar-lhe pistas positivas ou negativas à medida que funciona como completar uma tarefa. Mas, na maior parte das vezes, o algoritmo decide por si só que passos tomar ao longo do caminho;

Chart, diagram

Description automatically generated

No que diz respeito à cadeia de fornecimento, a AA em logística podem ser bastante benéficas. É viável usá-lo para otimizar operações, minimizar erros que as pessoas podem cometer ou perder, e prever possibilidades e dificuldades futuras. Uma cadeia de abastecimento eficiente e flexível é uma vantagem significativa numa indústria altamente competitiva. Como resultado, as empresas estão a procurar ferramentas para os ajudar a otimizar as suas operações e a tomar decisões para aumentar a eficiência operacional e a felicidade dos clientes, bem como para reduzir os custos económicos e ambientais. O desenvolvimento mais significativo nesta indústria é a transformação digital da cadeia de abastecimento, que continua a ser um grande problema para muitos operadores de transportes.

A picture containing indoor, automaton

Description automatically generated

Agora em casos mais práticos onde se usa o *ML* é no exemplo da UPS que recorre a uma ferramenta que serve para gerir o seu sistema de frotas que é a ORION (Otimização Integrada em Estrada e Navegação). Os seus algoritmos avançados criam rotas ideais para os condutores de entregas a partir dos dados fornecidos pelos clientes, condutores e veículos e podem alterar as rotas em movimento com base na alteração das condições meteorológicas ou acidentes. Em última análise, irá olhar para as entregas que ainda precisam de ser concluídas e continuar a otimizar as rotas. A poupança de custos e tempo e a redução de emissões com base apenas nesta otimização é extraordinária — a UPS espera reduzir as milhas de entrega em 100 milhões.

**Conclusão**

Concluindo, *Machine Learning* traz bastantes benefícios especialmente na indústria de logística precisamente por causa dos seus custos reduzidos, otimizações nas operações com eficiência e baixo uso de recursos. Vai continuar a fazer parte do nosso dia a dia como também evoluir cada vez mais.

# Bibliografia

<https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML>

<https://www.expert.ai/blog/machine-learning-definition/>

<https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML>

<https://serengetitech.com/business/9-use-cases-of-machine-learning-in-logistics/>

<https://www.forbes.com/sites/bernardmarr/2018/06/15/the-brilliant-ways-ups-uses-artificial-intelligence-machine-learning-and-big-data/>