|  |  |
| --- | --- |
| Inteligência Artifícial - 3º ano do Mestrado Integrado em Engenharia Informática e Computação | |
| Otimização da Distribuição de Produtos |
| Relatório | |



|  |
| --- |
| Trabalho elaborado por:  Luís Gonçalves – ei12080 – [ei12080@fe.up.pt](mailto:ei12080@fe.up.pt)  Ricardo Loureiro – ei12034 – ei12034@fe.up.pt  Tiago Ferreira – ei12051 – ei12051@fe.up.pt  5/6/2015 |

Objectivo

No âmbito da Unidade Curricular de Inteligência Artificial foi-nos proposta a criação de um algoritmo de otimização para um problema de distribuição de produtos. Com o objetivo de resolver de uma forma eficaz será utilizado um algoritmo genético de modo a minimizar o custo de necessário para satisfazer os diversos pontos de venda.

Especificação

# Análise detalhada do tema

Um fabricante possuí várias Fábricas de produção que têm de abastecer diversos Pontos de Venda a nível nacional. Estas Fábricas têm um limite de produção e encontram-se a uma distância de cada Pontos de Venda. Os pontos de venda requerem mensalmente uma quantidade necessária de produtos tendo de ser satisfeitos pelas fábricas de produção.

O custo de venda é diretamente proporcional à distância entre a Fábrica de produção e o Ponto de Venda, com o objetivo de minimizar este custo pretende-se encontrar para cada ponto de venda as fábricas que os vão fornecer. Será aplicado um algoritmo genético tendo em conta diversos pontos de venda, várias Fábricas de produção, várias distâncias (basiadas na localização das Fábricas e Pontos de Venda) e diferentes capacidades requeridas pelos Pontos de Venda.

# Ilustração de cenários

# Abordagem

## Algoritmo Adoptado

O Algoritmo utilizado neste projeto foi **Algoritmo Genético** que é ideal para encontrar soluções para problemas complexos. Frequentemente usado em áreas como a engenharia por forma a criar produtos próximos dos ideais devido à capacidade de percorrer/pesquisar dentro de uma grande gama de combinações e parametros e encontrar o melhor.

Como sabemos estes algoritmos são baseados no processo de seleção natural, isso significa que estes tomam as propriedades fundamentais da seleção natural e aplicam-nas a qualquer problema que estejamos a tentar resolver.

O processo do Algoritmo Genético consiste basicamente num conjunto de passos:

1. Inicialização – Criação de uma população inicial. Esta população é geralmente gerada aleatoriamente e pode ser de qualquer tamanho desejado, desde um pequeno número de indivíduos até milhares.
2. Avaliação - Cada membro da população é então avaliado e calcula-se uma “aptidão” para cada indivíduo. O valor de “fitness” é calculado consoante os nossos requisitos desejados. Estes requisitos podem ser simples, 'algoritmos mais rápidos são melhores', ou mais complexas, 'materiais mais fortes são melhores, mas eles não deve ser muito pesado ". No nosso caso foi tida em conta a distância e a quantidade de produtos a ser distribuidos.
3. Seleção - Queremos constantemente melhorar o condicionamento físico geral das nossas populações.A seleção ajuda-nos a fazer isso, descartando os projetos menos aptos e só mantendo os melhores indivíduos na população. Existem alguns métodos de seleção diferentes, mas a idéia básica é o mesmo, torná-lo mais provável que os indivíduos mais aptos sejão selecionados para a nossa próxima geração.
4. Crossover - Durante o “Crossover” 0criaos novos indivíduos, combinando aspectos de indivíduos selecionados. A esperança é que através da combinação de certos traços de dois ou mais indivíduos, vamos criar uma prole mais “fit”, que irá idealmente herdar as melhores características de cada um de seus pais.
5. Mutação – É necessário adicionar um pouco de aleatoriedade à genética das nossas populações de outra forma cada combinação de soluções estaria demasiado dependente da geração inicial. A Mutação faz mudanças muito pequenas de forma aleatória para um genoma de um conjunto de indivíduos.
6. Repetir - Agora nós temos a nossa próxima geração podemos começar de novo a partir do passo dois até chegar a uma condição que nos pareça ideal ou até termos atingido um número de repetições “suficientes”.

## Técnicas e Representação do conhecimento

A representação do conhecimento apresentada no Relatório Intermédio foi desde então corrigida tendo em conta as sugestões da Docente da turma e depois de alguns aperfeiçoamentos chegámos à representação final que se encontra, então, no projeto final.

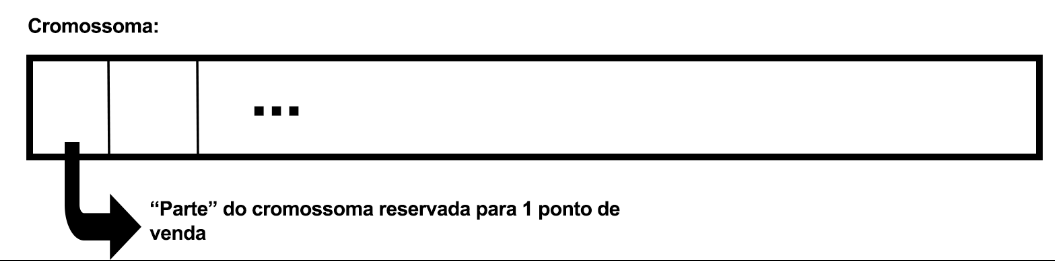
Cada Chromossoma é divido pelo número de pontos de venda:

Figure 1: Representação do Chromossoma

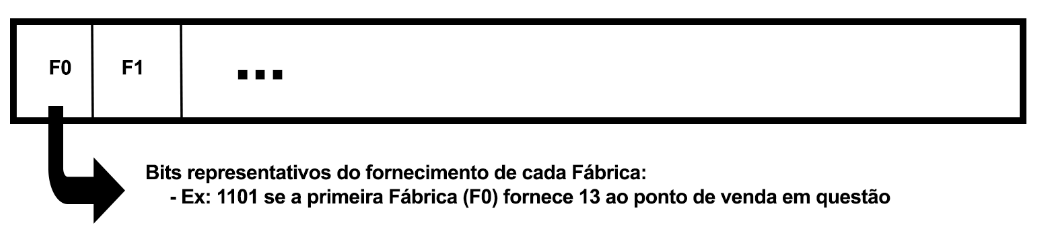
Depois, cada ponto de venda tem um espaço reservado a cada fábrica indicando o que a mesma o fornece:

Figure 2: Representação da parte do Chromossoma reservada a um Ponto de Venda

Assim sendo, de forma sucinta, representamos em bits o que cada fábrica fornecea cada ponto de serviço por forma a ter toda a informação necessária nos Chromossomas mutável. Optámos pela representação binária por forma a facilitar os métodos específicos do Algoritmo Genético, mais especificamente o *crossover* e a mutação.

## Métricas

### Função de Penalização (*Penalty*):

A Função de Penalização tem em conta sealguma das fábricas envia mais do que produz bem como se algum ponto de venda recebe mais ou menos do que o pedido, penalizando mais quanto maior for cada um dos valores.

Penalização(Xᵢ) =

E[Fᵢ] - Excesso de produção de cada Fábrica.

If( Produção < Envio) {

E[Fi] += (Envio - Produção);

}

R[Pᵢ] – O que sobra ou o excesso do recebido por cada Ponto de Venda

R[i] += (Envio - Produção);

ABS(R[i]);

### Função de Adaptação (*Fitness*):

A função de adaptação para quando existe penalização é diretamente proporcional ao inverso do somatório da distância com a penalição.

Adaptação(Xᵢ) =

dᵢ = Distãncia da fábrica a um ponto de venda, caso esta o forneça.

Se a penalização for nula então a adaptação dependerá só da distância e vice-versa.

Desenvolvimento

# Linguagem e Ambiente

A aplicação foi, na sua totalidade, desenvolvida em java sendo que os IDEs utilizados foram o IntelliJ e o Eclipse Juno tanto em Windows como em MacOS.

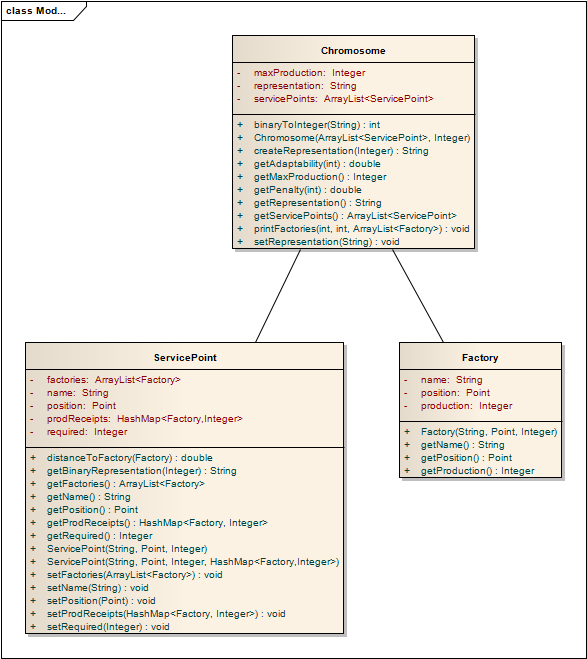
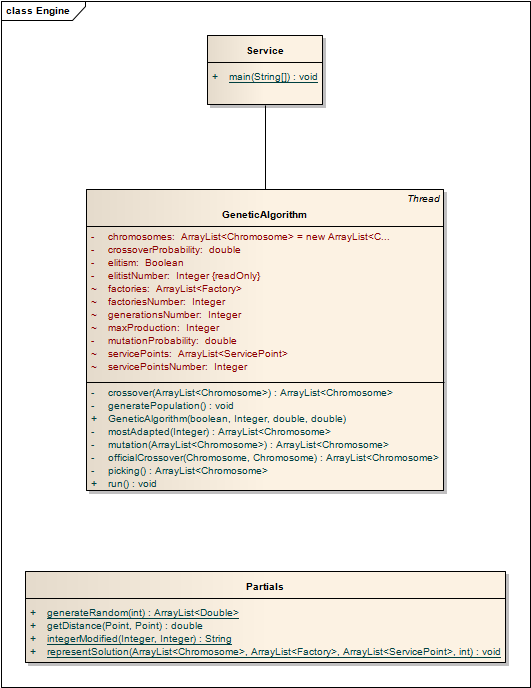
# Estrutura da Aplicação

Por forma a manter o código organizado, dividimo-lo em dois *packages*: o *Engine Package*, que é o responsável pelo funcionamento da aplicação, e o *Models Packages*, em que se encontram as classes representativas consoante o modelo adoptado de representação.

No *Engine Package* temos a classe *GeneticAlgorithm* responsável pela execução do algoritmo na sua totalidade, a classe *Service* onde é chamado e inicializado o *GeneticAlgorithm* e inseridos os valores relativos ao mesmo (se é elitista, a probabilidade de crossover e a probabilidade de mutação), por fim temos a class *Partials* usada para calculos auxiliares e representação da solução final.

No *Models Package* contem a class *Factory* onde são guardadas as informações relativas à fabrica, a class *Chromossomes* onde além da informação de cada Chromossoma também contem a função de Adaptação e Penalização, e, por fim, a class *Service Point* que contém toda a informação dos *Service Point*.

# Diagramas de Classes



Experiências

Conclusões