Aplicação de Algoritmos Genéticos na otimização da Calendarização de Exames

Inteligência Artificial

3º Ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do grupo:

Hugo Cunha - 201404587
Ilona Generalova- 201400035
João Castro - 201404896

Porto, 21 de Maio de 2017

Índice

1. Introdução / Objetivo

Com o nosso projeto temos como objetivo principal perceber e aplicar os algoritmos genéticos a uma situação do mundo real, neste caso, a otimização da calendarização de exames para uma determinada entidade.

O nosso tema, tal como referido acima, é resolver a problemática da calendarização de exames de uma universidade, escola ou outra entidade que tenha o binómio exame-alunos bem definido, assim como datas limites.

Como conjunto de dados *input* teremos um conjunto de unidades curriculares (utilizámos os exames a ser realizados na FEUP, no MIEIC, na época de 2016/2017, segundo semestre) e os alunos inscritos a exame nessas mesmas unidades curriculares. Além disso, teremos ainda duas datas limites que marcam o início e o final da época normal de exames. Ao aplicar o algoritmo genético teremos, ainda, atenção a que alguns requisitos estejam a ser cumpridos, como de evitar um aluno ter exames a duas unidades curriculares em dias seguidos.

Ao nosso programa estão associadas algumas restrições: um aluno não pode ter mais que um exame por dia, nem em dias consecutivos. No final da execução do nosso programa esperamos obter uma boa solução de calendarização de exames para todos os estudantes, cumprindo todas as restrições estabelecidas.

2. Especificação

2.1 População Inicial

A população inicial constitui, à partida, uma solução para o problema. Daí o grupo ter tido em conta tanto o tamanho como a diversidade de indivíduos utilizados, pois populações pequenas podem não ser suficientes para cruzamentos, assim como muito grandes podem fazer com que o algoritmos se torne demasiado lento. De entre algumas formas de gerar população inicial, o grupo decidiu optar pela geração aleatória. No entanto é também acrescentada diversidade manual, como por exemplo a não existência de alunos do primeiro ano com exames do quarto.

2.2 Função de otimização

A função de otimização tem um impacto muito grande em como o algoritmo evolui, pois tem como objetivo avaliar a qualidade de uma determinada solução, atribuindo um valor específico. É possível afirmar que, em teoria, uma solução é tanto melhor quanto maior for o valor retornado por esta função.

Na verdade, a função de otimização é utilizada a cada iteração/geração do algoritmo, portanto o objetivo do grupo foi, desde o princípio, tentar obter a melhor solução possível consumindo o mínimo de recursos e tempo possível.

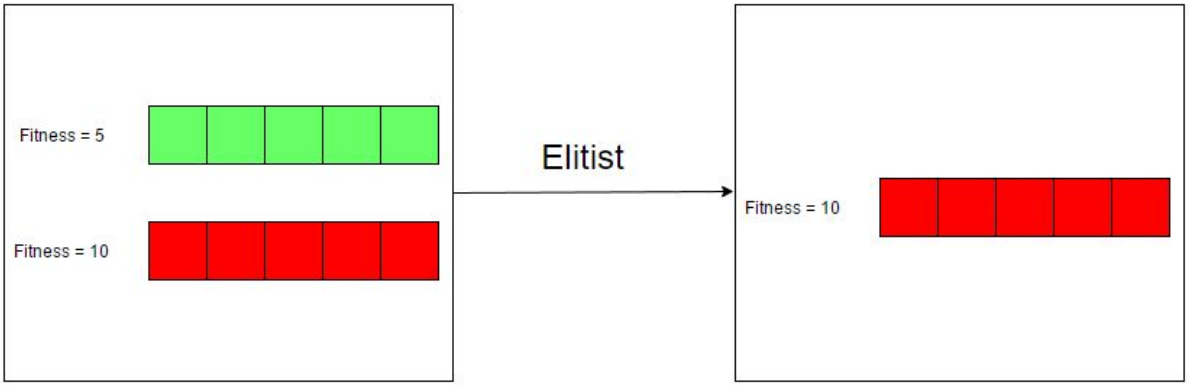
2.3 Seleção de indivíduos para reprodução

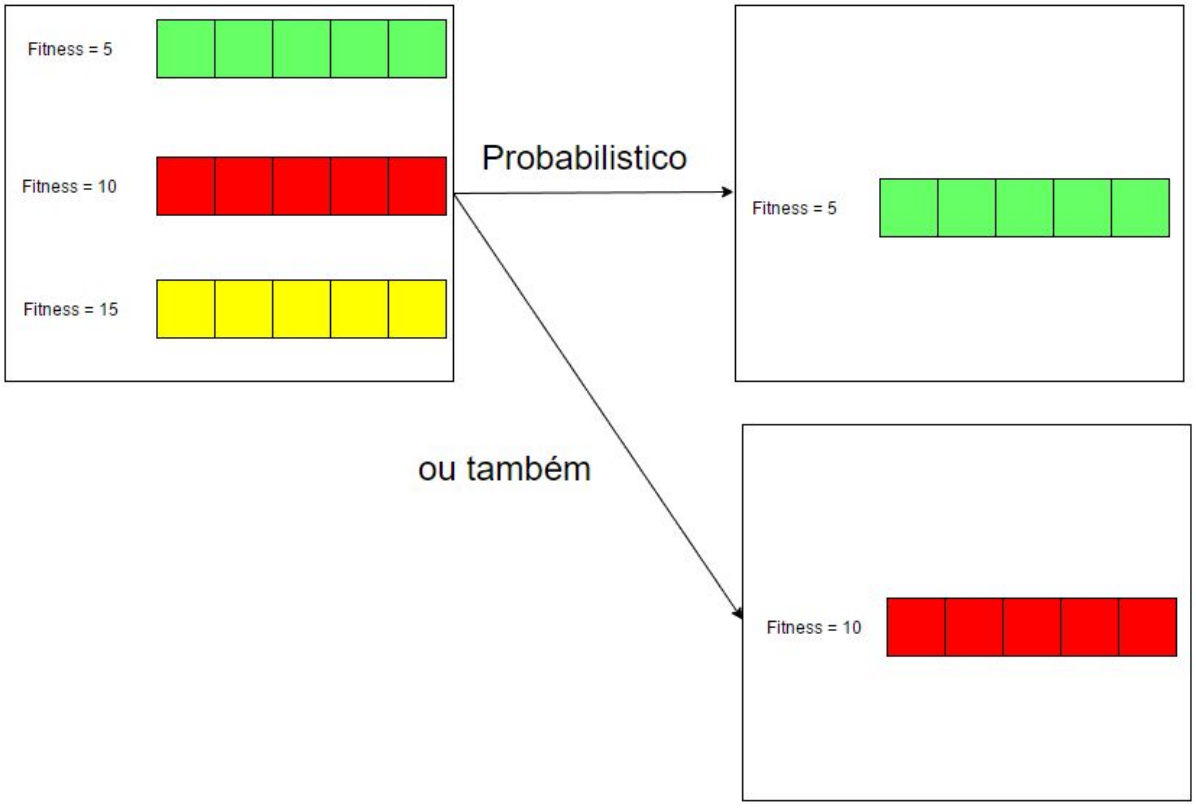
A seleção de indivíduos é uma parte na qual, como o nome indica, selecionamos indivíduos para a próxima geração. No entanto, algumas precauções têm que ser tomadas.

Umas das mais simples soluções será de organizar os indivíduos por *fitness* e escolher os melhores. Este tipo de abordagem vai de encontro àquilo que pretendemos que é encontrar rapidamente uma boa solução, em curto espaço de tempo. No entanto, esta abordagem também pode levar a uma convergência prematura e perda de diversidade nas futuras gerações.

O grupo implementou duas formas de seleção: elitista em conjunto com solução probabilística.

Desta forma, iremos tomar como ponto de partida a geração atual e iremos tentar selecionar por roleta metade dos indivíduos. Caso o número de indivíduos selecionados seja inferior a metade da população inicial iremos, por seleção elitista, obter os restantes indivíduos a partir dos iniciais, sem incluir os já retirados. Caso o número de indivíduos selecionados seja superior a metade da população inicial, iremos, por solução elitista, selecionar a metade da população inicial, mas apenas dos que foram selecionados previamente. Caso o número de indivíduos selecionados seja igual a metade da população passamos à fase seguinte do algoritmo, *crossover* e mutação.



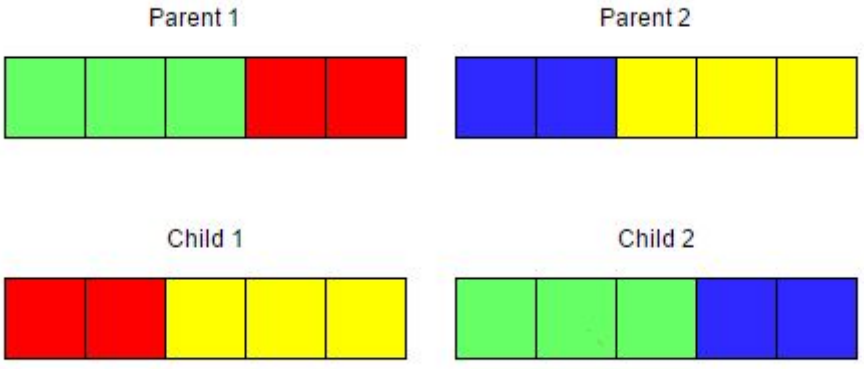


2.4 *Crossover*

A função de *crossover* é uma analogia à biologia quando se dá a mistura de genes na reprodução de um novo ser. Na nossa implementação resolvemos implementar um *Multi-Point Crossover*.

Aquando a seleção de indivíduos, nós selecionamos sempre metade da população, ou seja, nesta fase de *crossover* temos que repôr os números de população inicial.

Portanto, a partir da seleção, a qual tem metade dos indivíduos da população inicial iremos, aleatoriamente, selecionar indivíduos para reprodução (dois). Desses dois, iremos selecionar um número aleatório desde 0 até ao tamanho do cromossoma e iremos "dividir" os cromossomas dos dois "pais" em duas partes. Desde 0 até ao número gerado que constituirá a segunda parte do cromossoma do filho 1 e o restante, ou seja, desde o número gerado até ao tamanho do cromossoma fará parte da primeira parte do cromossoma do filho 2. O método funcionará em vice-versa para o segundo parente.

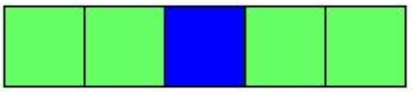


2.5 Mutação

As mutações são pequenas alterações que são introduzidas na população com o objetivo de manter ou introduzir alguma diversidade na população genética e, geralmente, é de baixa probabilidade para que o problema não se reduza a um problema de procura aleatória.

Na nossa solução implementamos uma solução modificada do *Bit Flip*, apesar da nossa solução não puder ser ser representada em bits.

Desta forma, a cada há um *bit* que será mutado para uma outra altura do calendário.



2.6 Condição ou Condições de Paragem

Como qualquer algoritmo, os algoritmos genéticos requerem que se cesse o seu procedimento. Este é um passo importante porque quando este tende para a solução ótima, ou apenas uma boa solução, pequenas alterações são feitas e este podia entrar em ciclo infinito. Assim, as mais comuns condições de paragem são: quando não houver melhoramentos há X iterações, quando se atingir um número absoluto de gerações ou quando a função de otimização retornar valores perto ou superiores a um valor já pré definido.

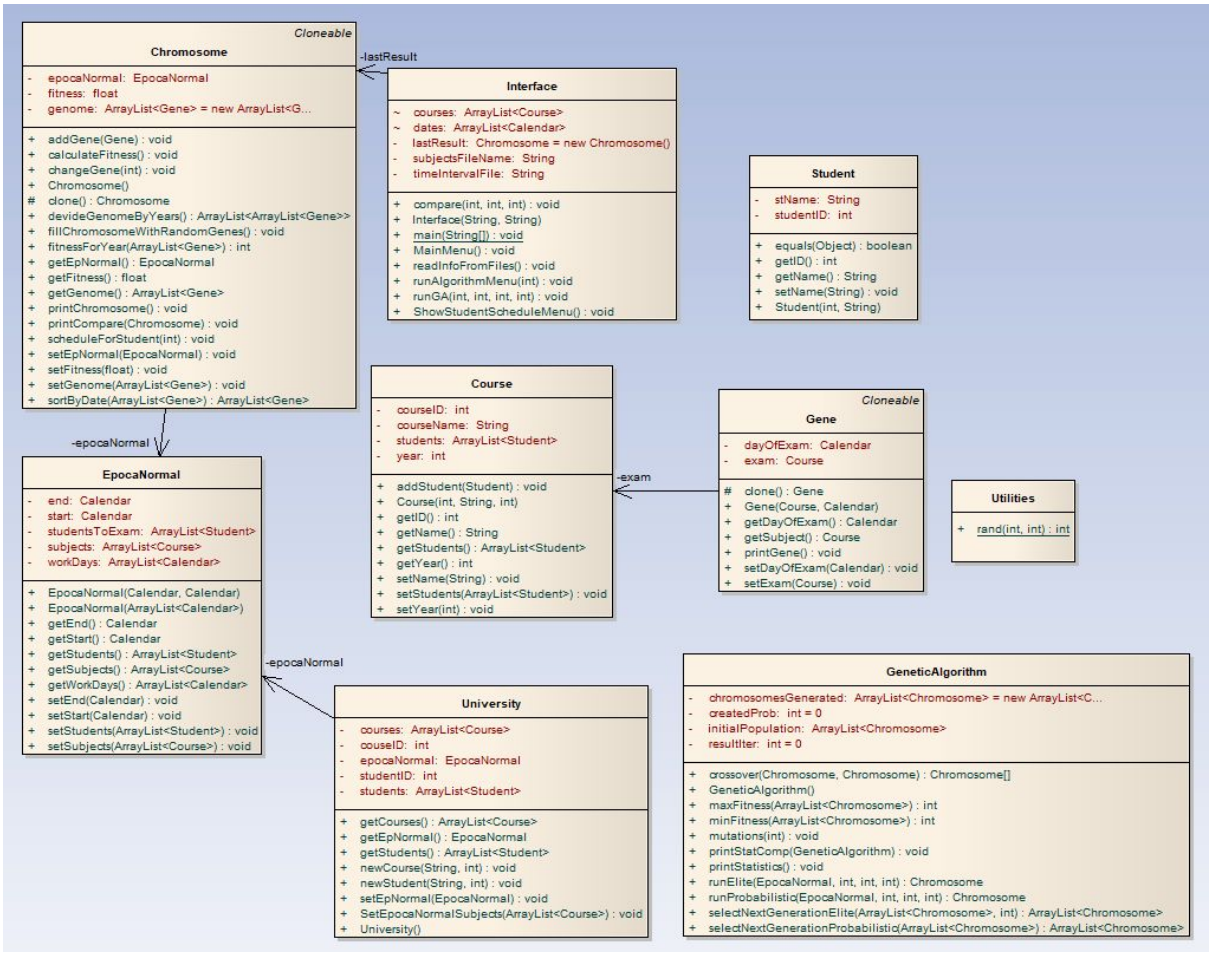
Na nossa implementação do algoritmo genético, o utilizador vai poder escolher qual o número de iterações que o algoritmo irá correr.

3. Desenvolvimento

3.1 Ferramentas utilizadas

O projeto foi desenvolvido em sistema operativo *Windows*. Foi utilizada a linguagem de programação *Java*, com a ajuda do ambiente de desenvolvimento *Eclipse Neon*.

3.2 Estrutura e Diagrama de Classes



4. Experiências

Realizamos uma experiência na qual corremos o algoritmo um elevado número de vezes nas duas soluções que implementamos: elitista, elitista+probabilística. Os dados de *input* foram exatamente os mesmos.

Número iterações	População Inicial	Bit de mutação
200	20	2

No final construímos os resultados:

	Elitista	Elitista + Probabilística
Média de fitness	37420	35671
Média de Iterações até atingir solução	58,8	71

Com isto, o grupo concluiu que, de facto, a solução elitista é uma forma de atingir mais rápido a solução. Além de ser mais rápido a atingir a solução, atinge, em média, uma solução melhor, com um valor superior de *fitness*.

No entanto, é de frisar que a média é um método estatístico um pouco falacioso na medida em que é muito “abalada” por valores extremos mas, de facto, algumas soluções obtidas pelo método Elitista + Probabilística obtiveram valores de *fitness* muito elevados, concluindo que a utilização de probabilidades contribui para a introdução de diversificação de resultados.

5. Conclusões

5.1 Algoritmica

O grupo concluiu que os algoritmos genéticos são uma simples, no entanto eficiente, solução para problemas do mundo real.

De facto, os algoritmos genéticos oferecem um “leque” grande de soluções não se prendendo apenas a uma como alguns algoritmos clássicos. Além disso, oferecem sempre uma boa solução, poderá não ser ótima, mas, de certo, será melhor que a solução inicial e, ainda muito importante, são muito úteis quando o espaço pesquisa é grande e com grande número de parâmetros envolvidos.

5.2 Desenvolvimento do projeto

O desenvolvimento do projeto não foi, provavelmente, tão sequencial como se poderia imaginar à partida. A carga de trabalho num tema ligado à inteligência artificial mostrou-se bastante distinto a muitos dos projetos desenvolvidos noutras unidades curriculares do curso.

Conforme os elementos do grupo foram começando a pensar como seria implementado o algoritmo - em especial o cálculo do fitness de cada solução - verificou-se que estaria desde logo aí o maior entrave do projeto.

Como se pode imaginar foi nessa parte do trabalho que o grupo despendeu mais tempo porque, a partir desse momento, a lógica imperou e o desenvolvimento do projeto tornou-se quase instintivo e bastante rápido no seu desenrolar.

6. Melhoramentos

De facto, há poucos aspetos onde o projeto poderá melhorar, mas existe sempre espaço para eles.

O grupo pensa que a introdução de uma *interface* gráfica que permitisse a visualização do calendário geral de exames, e em particular de cada aluno, aliado às já implementadas estatísticas seria uma mais-valia.

O principal problema deste projeto é, realmente, a avaliação da qualidade de uma solução. Por outras palavras, o cálculo do *fitness* é sempre um pouco subjetivo e é realmente difícil encontrar uma solução que seja sempre unânime.

Além disso, poderiam ter sido implementadas mais opções de como o algoritmo iria trabalhar, por exemplo, utilizar diferentes formas de mutação e de *crossover*.

7. Recursos

7.1 Bibliografia

1. Oliveira, Eugénio: Métodos de Resolução de Problemas e Algoritmos para a Evolução, Ano Lectivo 2016/2017
2. https://www.tutorialspoint.com/genetic_algorithms/index.htm
3. Jha, Sujit Kumar - Exam Timetabling problem using genetic algorithm, eISSN: 2319-1163 | pISSN: 2321-7308

7.2 Software

1. Windows, Sistema Operativo
2. Eclipse Neon, IDE

7.3 Percentagem de Contribuição dos Elementos do Grupo

1. Hugo Cunha - ⅓
2. Ilona Generalova - ⅓
3. João Castro - ⅓