Universidade do Porto

Faculdade de Engenharia

FEUP

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Otimização da gestão de projetos

Authors:

Duarte Pinto
- up201304777 - up201304777@fe.up.pt

Filipa Ramos
- up201305378 - up201305378@fe.up.pt

Gustavo Silva
- up201304143 - up201304143@fe.up.pt

28 de Maio de 2016

Conteúdo

1 Introdução

«The scheduling of tasks and the allocation of resource in medium to large-scale development projects is an extremely hard problem and is one of the principal challenges of project management due to its sheer complexity.»[1]

No âmbito da unidade curricular de Inteligência Artificial foi desenvolvido um programa de otimização que tem por objetivo gerir projetos. Tendo por pressuposto a citação explicitada acima foi implementado um sistema para alocar membros a um projeto da maneira mais eficiente tendo em consideração a duração de cada tarefa e as competências de cada membro para cada tarefa. Pretende-se no presente documento avaliar comparativamente os resultados demonstrados por cada um dos algoritmos de otimização usados - algoritmos genéticos e arrefecimento simulado. Inicialmente será esmiuçado o problema sugerido e a perspetiva das soluções implementadas. Nesta especificação será englobada uma análise detalhada ao tema e aos cenários problemáticos que surgiram nessa mesma análise. A abordagem técnica adaptada de forma a solucionar estes obstáculos será pormenorizadamente descrita.

Um dos objetivos principais do presente documento é avaliar a resposta de ambos os algoritmos em cenários semelhantes por forma a melhor compreender a sua eficiência e as suas situações de risco. As conclusões retiradas das experiências efetuadas serão indispensáveis à exploração das possibilidades e fraquezas de cada algoritmo. Para além disto, surgem objetivos secundários tais

[1] Carl K. Chang et al, *Genetic Algorithms for Project Management*, (Holanda: Kluwer Academic Publishers, 2001)

como o aprofundamento do estudo de métodos de inteligência artificial e o estudo da aplicação prática da teoria abordada nas aulas.

2 Especificação

2.1 Problematização

«Um projeto é constituído por um conjunto de tarefas a desenvolver por um ou mais elementos. As tarefas podem ter precedências entre si e têm uma duração (pessoa/mês). Cada elemento candidato possui um conjunto de competências, que cobrem uma ou mais tarefas. É conhecido ainda o desempenho de um elemento em cada uma das suas competências.»[2]

Como é especificado em cima o sistema tem por objetivo otimizar a atribuição de membros por tarefas num dado projeto. Os dados do mesmo são introduzidos por input através de um ficheiro no formato json no qual estão representadas as tarefas, elementos e as competências de cada um deles.

Um projeto em análise caracteriza-se por um conjunto de tarefas (Task) a cumprir, tendo estas um nome (por motivos de identificação) e uma duração. Existe ainda um conjunto de elementos (Element) que podem ser atribuídos a essas mesmas tarefas. Um elemento é identificado por um nome e tem uma lista de competências (Skill) avaliadas em função da sua capacidade. Por exemplo, o elemento "joão"tem competências na área da informática a um nível 6 e na área da economia com nível 10.

2.2 Cenários

O programa desenvolvido teria incomensurável utilidade para qualquer tipo de empresas. Imagina-se cenários onde o software seria uma mais valia como o processo de iniciação de um projeto na qual não é claro como deve ser feita a divisão de tarefas entre os membros. Para uma empresa da área informática seria da maior importância visto que para construir software são precisos grupos de trabalho em que as competências de cada um são diferentes. É ainda aplicável no meio académico, não só como ferramenta mas também como estudo de algoritmos de otimização. Trazendo o problema para o âmbito real leva a que este possa ser aplicado em contextos mais extensivos enriquecendo o contributo dado pelo mesmo.

2.3 Dificuldades

Um dos maiores obstáculos encontrados no planeamento da implementação foi o jogo de todas as variáveis do problema da forma mais eficiente. A combinação do uso da duração das tarefas com as suas precedências e das competências dos elementos provou-se como um desafio a ultrapassar.

A maior dificuldade identificada nos algoritmos genéticos foi a da construção de uma função de avaliação que tivesse em conta a precedência de tarefas. Assim, inicialmente, pensou-se ser necessária uma heurística de escolha de ordem de tarefas. Se este processo não fosse bem otimizado a solução apresentada não seria a melhor possível. Logo, esta heurística teria de ser cautelosamente planeada. Contudo, optou-se por implementar a estrutura do projeto de forma diferente sendo que a ordem das tarefas seria transmitida pelos cromossomas e seria decidida aquando da sua

[2]Enunciado do problema.

construção. A implementação será esmiuçada mais à frente. Para além disto, a escolha de um método de cruzamento provocou algumas dúvidas nos elementos porém após terem sido realizados alguns testes foi escolhido layout que produzia mais frequentemente melhores resultados.

NOS ALGORITMOS GENÉTICOS A DIFICULDADE FOI BLÁ BLÁ BLÁ...

2.4 Datasets

2.4.1 Formato do input

O input é colhido de um ficheiro de formato *json* e tem a estrutura conforme representado no exemplo apresentado a seguir.

```
{
   "skills": [
      "JQuery", "PHP", "CSS" ,"HTML"
                ],
   "tasks": [
      {"name":"Construir pagina", "duration": 8,"skill": 3,"precedences":[]},
      {"name": "Estilizar pagina", "duration": 5,"skill": 2,"precedences": [0]},
      {"name": "Fazer login", "duration": 3,"skill": 1,"precedences": [0,1]},
      {"name": "Animacoes", "duration": 7,"skill": 0,"precedences": [0,1]}
             ],
   "elements": [
      {"name": "Duarte Pinto", "skills": [[0,0.5], [2,0.3] ]},
      {"name": "Filipa Ramos", "skills": [[0,1.0], [3,0.1] ]},
      {"name": "Gustavo Silva", "skills": [[1,1.0], [2,0.1] ]}
                ]
}
```

2.5 Algoritmos

Na presente secção são analisados os algortimos usados passando pela explicação da implementação decidida e pela descrição da perspetiva tomada ao enfrentar os mesmos. Inicialmente serão esmiuçados os algoritmos genéticos, sendo explicada a função de avaliação implementada, a estrutura do cromossoma construído e os processos de seleção, cruzamento e mutação. Finalmente é explicitado o arrefecimento simulado incluindo a representação escolhida, a atribuição de tempos iniciais e a geração do próximo estado.

2.5.1 Algoritmos Genéticos

2.5.1.1 Estrutura do Cromossoma

Um cromossoma válido traduz a ordem de realização das tarefas sendo que a que aparece primeiro será realizada antes de todas as outras. Cada bloco de cromossoma contém um identificador que representa a tarefa e um bit para cada elemento do projeto. Os bits seguintes dizem respeito aos elementos. Se o elemento estiver alocado à tarefa o seu bit estará a 1. Por exemplo, se tivermos um projeto com 4 tarefas e 6 membros um dos cromossomas possiveis seria o seguinte representado

na figura **??**. Pode-se observar que o primeiro elemento trabalhará em todas as tarefas menos a de id = 011. O elemento 5 só será alocado à primeira tarefa.

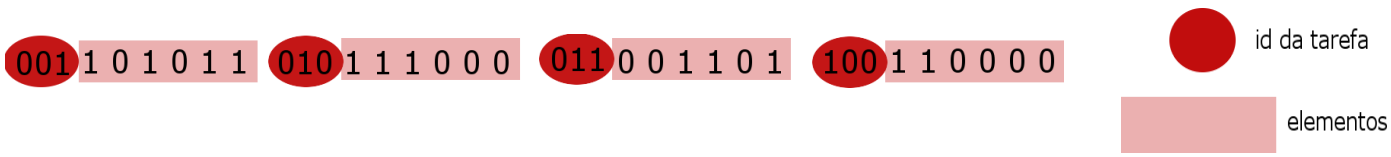

Figura 1: A figure with two subfigures

O cálculo do compimento do cromossoma depende da quantidade de membros e de tarefas do projeto. A fórmula aplicada por forma a obter o número de bits necessários à representação do identificador traduz-se na equação 1. No caso do cromossoma em cima o número de bits é 3. Assim, o comprimento de um bloco seria de 9 bits. Sendo que este tem 4 tarefas o comprimento total do cromossoma seria de 36 bits.

$$\lfloor(\frac{\log_{10}(size-1)}{log_{10}(2)+1}\rfloor \tag{1}$$

Figura 2: Fórmula para calcular o número de bits máximo para representar com um id binário todas as tarefas (size equivale ao número de tarefas).

2.5.1.2 Função de Avaliação

A função de avaliação verifica se o cromossoma é válido em termos de precedência de tarefas. Para além disto, calcula os tempos de início e de fim para cada membro alocado à tarefa. Verifica também se outra tarefa pode ser feita ao mesmo tempo. Se nenhuma tarefa puder ser feita, aumenta o tempo até à primeira a poder ser executada. Caso as condições não sejam cumpridas é aplicada uma penalização que se verifica pelo aumento do tempo. Assim, quanto menor o valor de **fitness** melhor o cromossoma.

2.5.1.3 Seleção

A seleção é feita por uma roleta aleatória que gera valores. Estes valores implicam um valor aleatório multiplicado pelo *fitness* total da população. A este valor é subtraído o do *fitness* individual de cada cromossoma. Os selecionados são os que têm menor valor final. A seleção elitista pode implicar um cromossoma ou mais, sendo este valor escolhido conforme o que se pretende testar.

2.5.1.4 Cruzamento

O cruzamento que se revelou mais eficiente foi o de trocar membros dentro de um cromossoma. É trocado apenas um bit dentro de tarefas aleatoriamente. Se os bits forem iguais é escolhido um novo bit para a troca.

2.5.1.5 Mutações

A mutação ocorre a uma taxa variável, sendo esta selecionada conforme os testes pretendidos. A mutação exclui os cromossomas elitistas, tendo por base o princípio de que um cromossoma elitista tem a melhor seleção de genes e, após uma mutação, essa combinação pode ser alterada para uma pior. São gerados valores aleatórios para cada cromossoma numa roleta e, se este for menor que o valor da taxa de mutação, é trocado o bit que resulta do resto da divisão da posição do mesmo no cromossoma pelo comprimento total do cromossoma.

2.5.2 Arrefecimento Simulado

3 Desenvolvimento

3.1 Ferramentas, linguagens e ambientes

Para tornar mais fácil a visualização dos resultados foi usada uma API de visualização de grafos chamada NOME DA API. A linguagem usada foi Java.

3.2 Aquitetura

As classes indispensáveis à resolução do problema através de algortimos genéticos são as presentes no package `optimizer.solver.genetic_algorithm` e as suas relações e métodos principais estão descritos na figura 3.

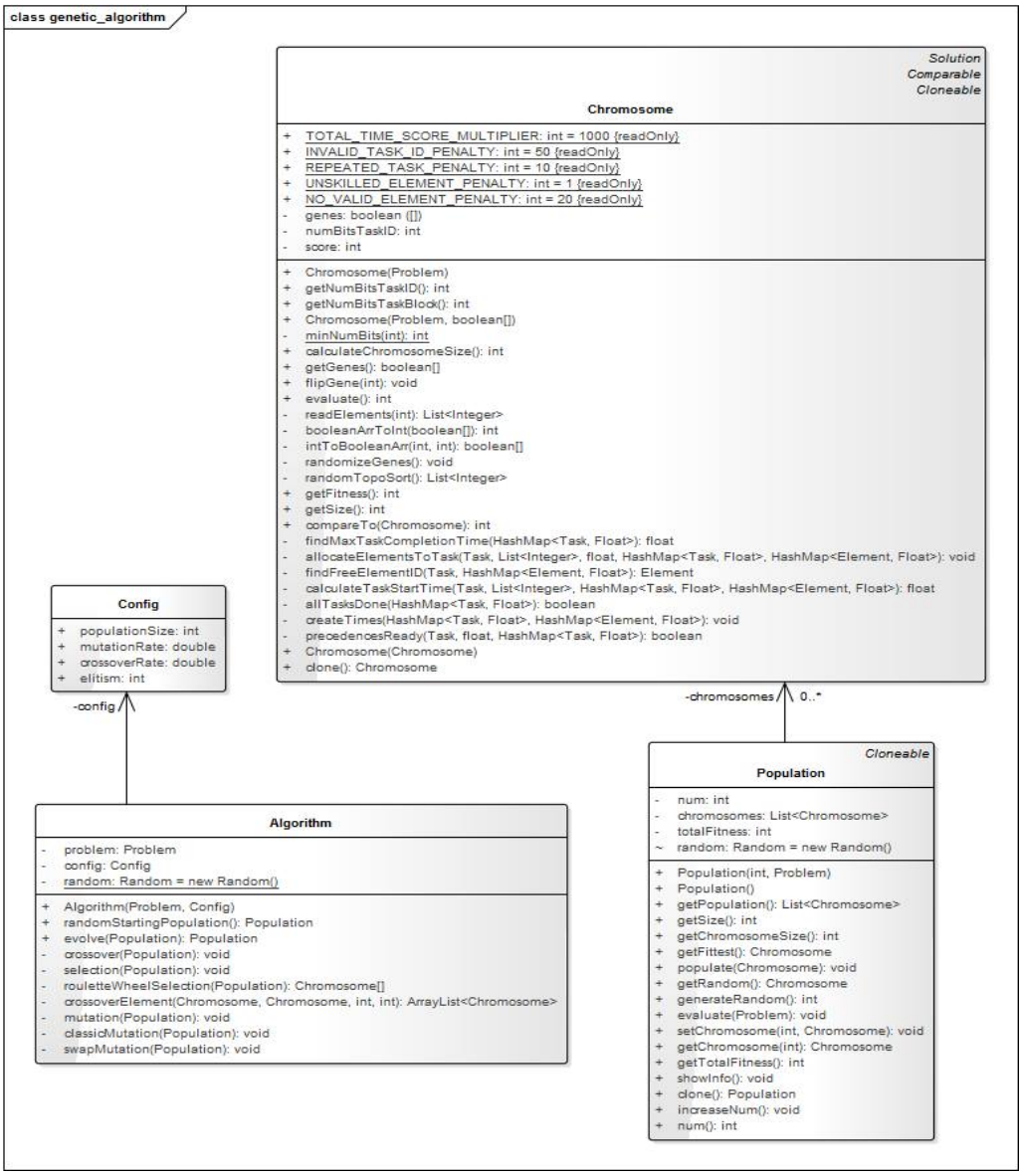

Figura 3: Diagrama de classes UML do package `optimizer.solver.genetic_algorithm`.

O projeto foi dividido em packages de forma hierárquica para representar os níveis da solução implementada. Um dos packages serve o propósito de guardar a informação relativa aos membros, tarefas e competências. Outro engloba dois outros packages onde está implementada a solução usando cada um dos algoritmos. O package .gui contém todas as classes que permitem a visualização gráfica da solução encontrada. Esta arquitetura de subpackages foi escolhida por forma a melhor representar a dimensão do problema e da implementação. Na realidade, a hierarquia é o que melhor representa a ligação entre os packages e as suas classes. Esta relação pode ser visualizada na figura 4.

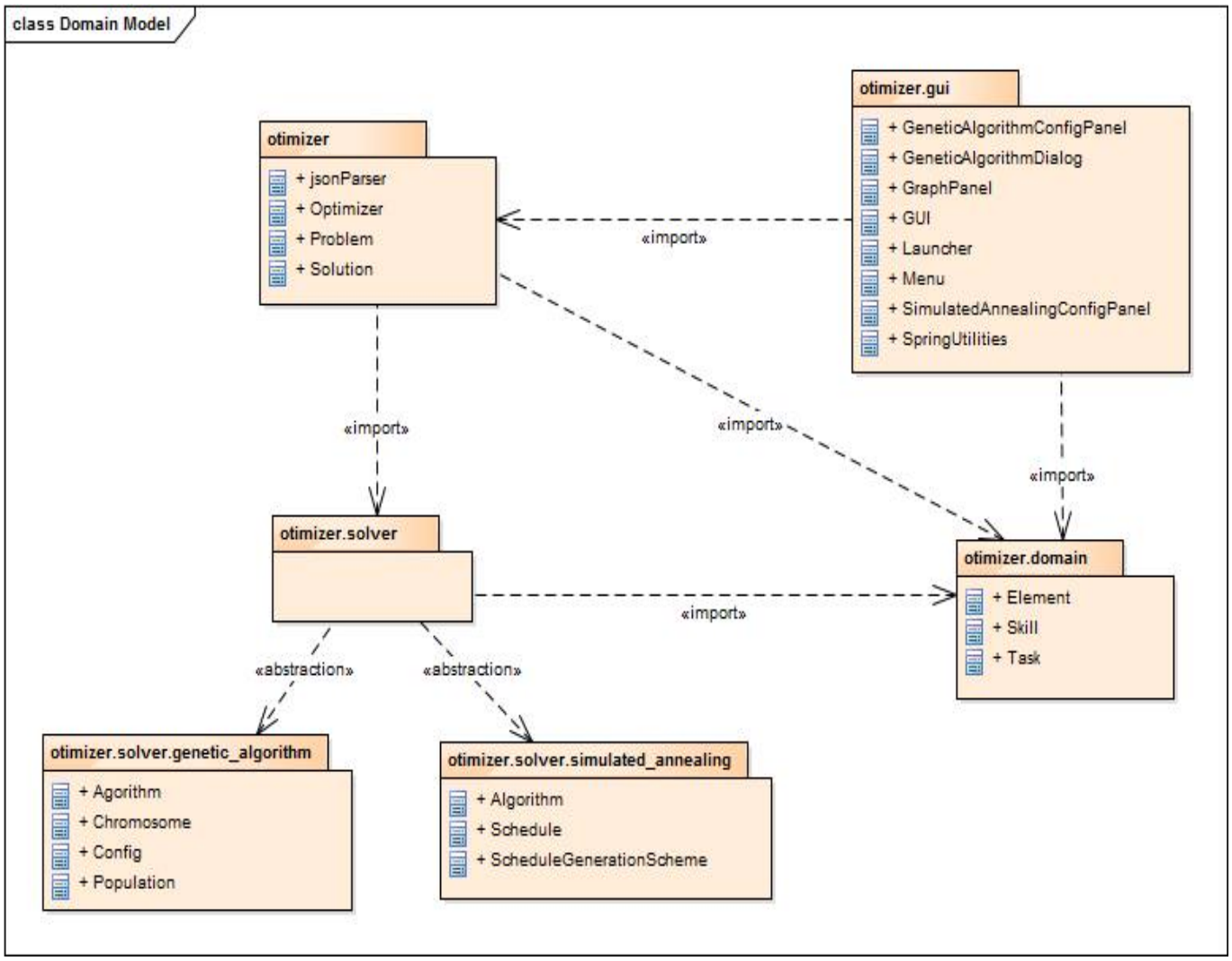

Figura 4: Diagrama de packages UML

4 Experiências

INSERIR EXPERIÊNCIAS AQUI

5 Conclusões

INSERIR CONCLUSÕES AQUI

6 Melhoramentos

NÃO HÁ MELHORAMENTOS PORQUE ESTÁ DO CRLHHHHHHHH

Referências

[1] Tao Zhang Carl K. Chang, Mark J. Christensen. Genetic algorithms for project management, 2001.

[2] H. Lecocq K. Bouleimen. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, 2002.