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

14 de Abril de 2016

Conteúdo

1 Introdução

No âmbito da unidade curricular de Inteligência Artificial pretende-se desenvolver um programa que, com base em algoritmos genéticos e arrefecimento simulado, faça a gestão de um projeto balançando os elementos participantes e as tarefas a realizar do mesmo. O sistema é composto por um conjunto de tarefas que pertencem ao projeto em análise. Cada tarefa tem uma competência indispensável ao seu cumprimento e uma duração. Cada elemento tem um conjunto de competências sendo que o mesmo tem um nível de capacidade para cumprir cada uma. A gestão a ser realizada tem em vista minimizar o tempo ocupado para satisfazer todas as tarefas do projeto usando a melhor combinação de elementos para cada tarefa. Será feita uma análise comparativa entre o desempenho das solução encontradas com algoritmos genéticos e arrefecimento simulado.

Os objetivos principais do projeto passam pela exploração da implementação prática dos algoritmos genéticos e do algoritmo de arrefecimento simulado. Através dos dados obtidos, visa-se também realizar uma comparação

da solução encontrada com ambos os algoritmos. Este processo irá fomentar o conhecimento adquirido, evidenciando as vantagens principais de cada algoritmo e as suas dicotomias principais.

Espera-se que surjam dificuldades na implementação prática dos algoritmos estudados teoricamente, principalmente na construção dos cromossomas pois existem dúvidas em relação à sua influência na eficiência da solução encontrada. Para além disto, a melhor adaptação da função de avaliação ao problema por forma a obter os melhores resultados revela-se um processo tumultuoso. Os membros decidiram optar por otimizar o tempo utilizado a concluir todas as tarefas do projeto em estudo. Desta forma, a melhor solução será a que implicará um menor tempo de conclusão do projeto em questão.

2 Especificação

2.1 Problematização

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.

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 Aquitetura

O projeto foi dividido em três "packages"principais que representam os três níveis mais importantes. Um dos packages diz respeito às classes que guardam informação sobre as tarefas, os elementos e as competências. Os outros dois dizem respeito à implementação do algoritmo genético ou do arrefecimento simulado.

A arquitetura pensada para o projeto passa pela implementação de três classes principais que interagem entre si próprias. A classe "Task"representa uma tarefa e guarda a sua duração. Um elemento é representado pela classe "Element"que mantém a identificação do mesmo. A classe "Skill"corresponde a uma competência. Estas ligam-se entre si por forma a que um elemento tenha vários skills e um skill tenha várias tarefas tal como é visível na figura 1. A relação que se pretende obter será a que liga os elementos a tarefas.

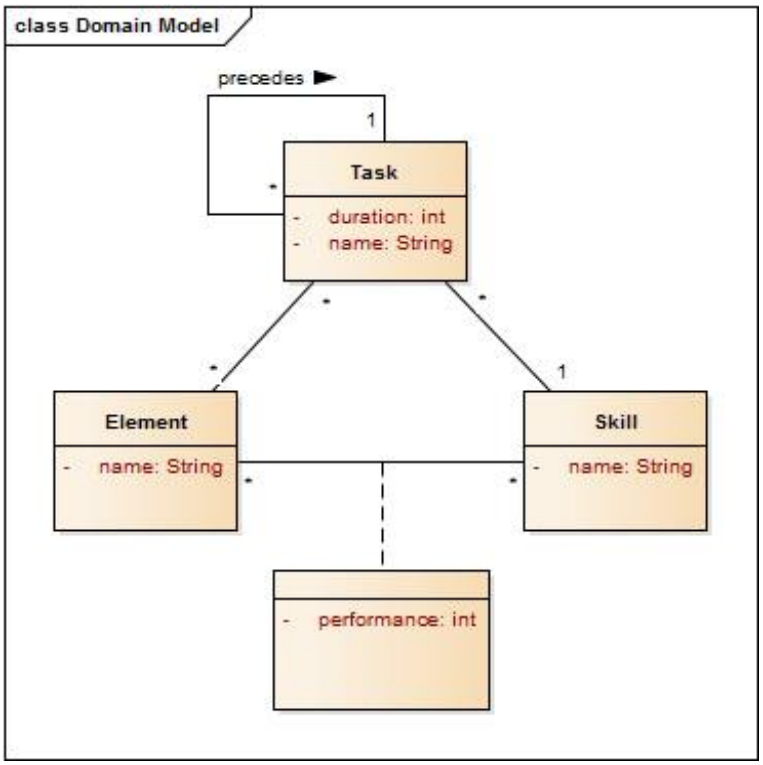

Figura 1: Diagrama de classes UML

2.3 Formato do input

O input será colhido de um ficheiro de formato *json* e será estrutrurado conforme está representado no exemplo apresentado a seguir. Espera-se receber:

- uma lista de *skills*
- uma lista de *tasks* sendo que cada uma tem um **nome** e uma **competência**
- uma lista de *elements* sendo que cada um tem um **nome** e um array de **competências** com as respetivas **capacidades**

```
{{
  "skills": [
    "Z",
    "Y",
    "Z"
  ], "tasks": [
    {
      "name": "A",
      "skill": 0
    },
    {
      "name": "B",
      "skill": 1
    }],
      "elements": [
     {
      "name": "Duarte Pinto",
      "skills": [
        [0, 0.5],
        [2, 0.1]
       ]},
    {
      "name": "Filipa Ramos",
      "skills": [
        [0,1]
       ]},
    {
      "name": "Gustavo Silva",
      "skills": [
        [1, 0.5],
        [2, 0.1]
      ]}]}
}
```

2.4 Fases

O projeto será dividido em fases de trabalho. A inicial passa pelo desenvolvimento da arquitetura supracitada. Seguidamente, proceder-se-á implementação dos algoritmos genéticos. Finalmente, será desenvolvido o arrefecimento simulado. As fases abordadas até ao presente relatório foram as duas primeiras. A arquitetura explicitada pelo diagrama de classes uml já foi implementada e a obtenção de solução por algoritmos genéticos encontra-se numa fase avançada.

2.5 Algoritmos Genéticos

Explicar o algoritmo usado: - estrutura do cromossoma - função de avaliação - seleção - cruzamento - mutações

2.6 Arrefecimento Simulado

Quando adaptado de maneira eficiente o algorítmo de Arrefecimento Simulado é característico pela facilidade de implementação e pela rapidez de convergência do resultado.

Para este projecto optamos por representar a nossa solução através de uma lista de tasks.

2.6.1 Representação

A representação da solução é importante pois tem que permitir a geração rápida do próximo estado e rápido calculo do ΔE.

Para tal colocamos as tarefas numa lista ordenada de tasks onde cada task aparece numa posição depois de todos os seus antecessores e antes dos seus sucessores e onde as tasks mais pequenas têm prioridade(aparecem primeiro na lista) sobre as tasks mais pesadas.

2.6.2 Atribuição dos tempos iniciais

Para atribuir os tempos iniciais de cada tarefa executamos um ciclo onde cada iteração representa a passagem de um *sprint*, de duração é varíavel. Em cada iteração, i, percorremos a lista e tentamos atribuir o tempo inicial à task j. Se as tasks antecessoras de j já tiverem terminado e houverem

pessoas com as *skills* para executar a tarefa, então o $j_{start\_time} = i$. Caso contrário, não atribui tempo inicial e continua.

```
int i = 0;
while(!allTimesAssigned){
   for(Task j : tasks){
      if(j.assigned)
         continue;
      if(j.predecessorsFinished() &&
          workers.hasAvailableResources(j)){
         j.startTime = i;
         workers.assign(j,i);  //Assigns resources(people) to the
            task j from i until the end of the task
      }else{
         continue;
      }
   }
   i++;
}
end;
```

3 Trabalho Realizado

Incluir trechos do código já implementado.

4 Testes

Explicar os testes planeados.

5 Conclusões

Conclusões retiradas.

Palavras Chave