Universidade do Porto
Faculdade de Engenharia
FEUP

Pesquisa aplicada ao Problema de Alocação de Lotes de Terreno

Relatório Intercalar

Inteligência Artificial
3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:
Rui Valente Maia – 200704519 – ei07176@fe.up.pt
Miguel Rossi Seabra – 200604224 – ei06054@fe.up.pt

27 de Março de 201

Índice

1 Objetivo

O trabalho “Pesquisa aplicada ao Problema de Alocação de Lotes de Terreno” tem como principal objetivo permitir a um utilizador definir um conjunto de terrenos e as suas características, criar ou editar restrições sobre cada uma dessas características e, de acordo com os terrenos e as respetivas restrições criadas, alocar - através da utilização do algoritmo de pesquisa A* - um conjunto de lotes de terrenos que melhor se adequem às restrições especificadas e de modo a minimizar o custo.

2 Descrição

2.1 Especificação

A especificação original do trabalho pode ser encontrada em [1], pelo que o conteúdo da subsecção 2.1.1 é a relação entre o problema proposto e a abordagem seguida. O trabalho pode ser seguido a qualquer momento em [2].

2.1.1 Problema de Alocação de Lotes de Terreno

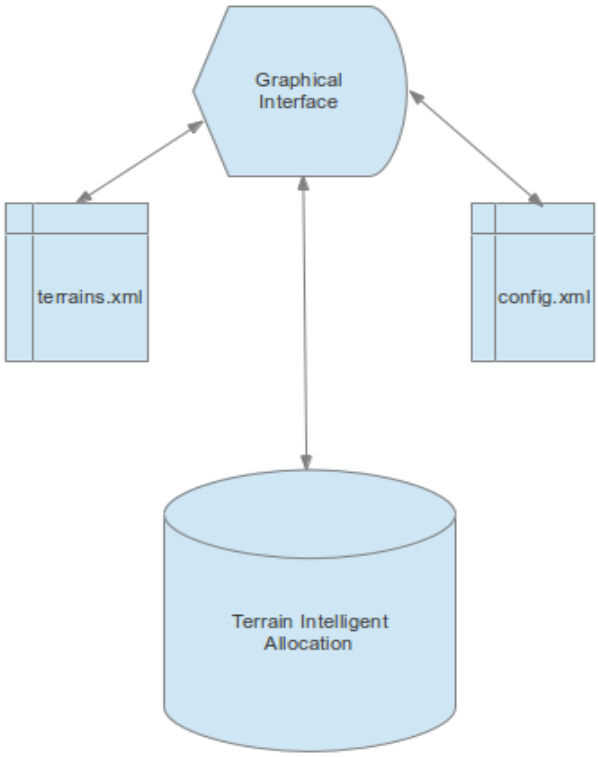
Durante a utilização do software “Pesquisa aplicada ao Problema de Alocação de Lotes de Terreno”, o utilizador terá de definir quantos lotes de terreno pretende incluir e as características correspondentes a cada um dos lotes. Caso o utilizador não pretenda definir as características lote a lote, haverá a possibilidade de atribuir valores de forma aleatória a cada um dos lotes de terreno definidos.
Posteriormente, o utilizador escolherá as restrições que pretende que a pesquisa possua. Essas restrições incidirão sobre as características dos lotes de terreno (tipo, inclinação, largura, altura, preço e forma do terreno) e serão definidas para cada um dos tipos de construção para que se pretende alocar o terreno. Cada tipo de construção é também definido e configurado pelo utilizador, bem como as restrições inerentes a cada um deles.
Por último, o sistema fará uma pesquisa recorrendo ao algoritmo A* e terá em conta as restrições adicionadas pelo utilizador, retornando como resultado final a alocação de cada uma das construções em lotes de terrenos que cumpram as restrições definidas pelo utilizador e de modo a minimizar o custo.

2.1.2 Calendário

Semana – Data Inicial a Data Final	**Objectivo**
1 – 20/02/2013 – 27/02/2013	Escolha e atribuição do trabalho prático.
2 – 27/02/2013 – 06/03/2013	Definição da arquitetura do software. Estudo do algoritmo A*. Definição das estruturas de dados. Definição dos esquemas de representação de conhecimento.
3 – 06/03/2013 – 13/03/2013	Implementação das classes “Terrain” (Terreno), “Point” (Ponto) e “Edge” (Aresta). Teste dos construtores das classes e definição e teste dos valores fronteira para cada um dos atributos. Definição da estrutura do ficheiro terrains.xml, que contém os atributos definidos pelo utilizador para cada terreno.
4 – 13/03/2013 – 20/03/2013	Implementação das classe “XML”, contendo funções de leitura de ficheiro e escrita de ficheiro. Teste de leitura e escrita de dados no ficheiro terrains.xml.
5 – 20/03/2013 – 27/03/2013	Especificação do formato das regras para as restrições e das

	várias restrições possíveis para cada um dos tipos de Terreno. Implementação das classes “Constraint” (Restrição) e “Rule” (Regra). Teste da aplicação das restrições a um terreno.
6 – 27/03/2013 – 03/04/2013	Férias da Páscoa.
7 – 03/04/2013 – 10/04/2013	Composição do relatório intercalar. Criação do mockup da interface gráfica.
8 – 10/04/2013 – 17/04/2013	Finalização do relatório intercalar. Início da construção da interface gráfica. Testes intensivos da aplicação de restrições.
9 – 17/04/2013 – 24/04/2013	Início da implementação do algoritmo de pesquisa A*. Teste intensivo da interface gráfica.
10 – 24/04/2013 – 01/05/2013	Finalização da implementação do algoritmo de pesquisa A*. Criação de testes unitários de forma a testar a validade das soluções da aplicação.
11 – 01/05/2013 – 08/05/2013	Queima das Fitas.
12 – 08/05/2013 – 15/05/2013	Queima das Fitas. Continuação da criação dos testes unitários.
13 – 15/05/2013 – 22/05/2013	Continuação da criação dos testes unitários. Composição do relatório final.
14 – 22/05/2013 – 29/05/2013	Composição do relatório final. Apresentação final do projeto.

2.1.3 Proposta de Arquitetura



A interface gráfica do sistema recebe informação do ficheiro config.xml – informação relativa às preferências do utilizador - e do ficheiro terrains.xml – informação relativa às propriedades de cada um dos lotes de terreno definidos pelo utilizador. O utilizador pode mudar atributos dos lotes de terreno ou configurações das restrições do sistema e guardar essas modificações, pelo que a interface gráfica também envia informação quer ao ficheiro terrains.xml quer ao ficheiro config.xml.

Posteriormente, esta envia ao sistema os dados dos lotes de terreno e das configurações inseridas/modificadas pelo utilizador, que processa essa informação e realiza a pesquisa tendo em conta esses dados. Por último, o sistema responde novamente à interface gráfica com o resultado da pesquisa e outros dados sobre a mesma.

2.1.4 Esquemas de Representação de Conhecimento

2.1.4.1 Ficheiros .xml

Existe dois ficheiros de formato XML diferentes que estão previstos: o ficheiro terrains.xml, que contém a informação relativa aos terrenos bem como os seus atributos e o ficheiro config.xml, que possui informação das configurações do utilizador para a pesquisa que está a realizar. Durante a realização deste relatório está a discussão de um terceiro ficheiro de formato XML que guarda informação relativa aos resultados da pesquisa, no entanto, até à conclusão da realização deste documento, não foi ainda tomada uma decisão definitiva.
O ficheiro terrains.xml possui o seguinte formato:

```
<terrains>
<terrain>
    <type>NOT FERTILE</type>
    <leaning>1.1</leaning>
    <width>2.2</width>
    <height>3.3</height>
    <price>4.4</price>
    <edges>
      <edge>
        <x1>1.0</x1>
        <y1>1.1</y1>
        <x2>1.2</x2>
        <y2>1.3</y2>
      </edge>
      <edge>
        <x1>1.2</x1>
        <y1>1.3</y1>
        <x2>1.4</x2>
        <y2>1.5</y2>
      </edge>
      <edge>
        <x1>1.4</x1>
        <y1>1.5</y1>
        <x2>1.6</x2>
        <y2>1.7</y2>
      </edge>
      <edge>
        <x1>1.0</x1>
        <y1>1.1</y1>
        <x2>1.6</x2>
        <y2>1.7</y2>
      </edge>
    </edges>
  </terrain>
</terrains>
```

O ficheiro config.xml, por sua vez, possui o formato:

```
<config>
    <numberTerrains>15</numberTerrains>
```

```
    <buildingTypes>
       <building>
          <name>SCHOOL</name>
          <restrictions>
              <r1>
                <type>MUST HAVE</type>
                <rule>
                  <nearTerrain>0</nearTerrain>
                  <measurement>MORE THAN</measurement>
                  <value>1.0</value>
                  <field>leaning</field>
                </rule>
              </r1>
              <r2>
                 <type></type>
                 <rule>
                   <nearTerrain></nearTerrain>
                   <measurement></measurement>
                   <value></value>
                   <field></field>
                 </rule>
              </r2>
         </restrictions>
       </building>
    </buildingTypes>
</config>
```

2.1.4.2. Classes e atributos

A classe *XML* contem um único atributo, denominado *file*, que é um atributo do tipo File com o objectivo de saber sobre que ficheiro .xml os seus métodos operam.

A classe *Point*, que é a classe que define o que é um ponto, contém dois atributos do tipo double: x1 e y1. A classe *Edge*, sendo a classe que define a ligação entre dois objetos do tipo *Point*, contém dois atributos do tipo *Point*: p1 e p2.

A classe *Terrain*, por sua vez, define um terreno – ou lote, no contexto do problema – e contém os seguintes atributos:

- *type*, do tipo *String*, que pode assumir apenas dois valores possíveis (“Fertile” e “Not Fertile”) e serve para definir se o terreno é fértil ou não;
- *leaning*, do tipo *double*, que assume apenas valores maiores ou iguais a 0 e serve para definir qual o valor da inclinação do terreno;
- *width*, do tipo *double*, que assume apenas valores maiores do que 0 e serve para definir qual o valor da largura do terreno;
- *height*, do tipo *double*, que assume apenas valores maiores do que 0 e serve para definir qual o valor da altura do terreno;
- *price*, do tipo *double,* que assume apenas valores maiores do que 0 e serve para definir qual o preço do terreno;
- *edges*, do tipo *List<Edge>*, que sendo uma lista de objetos do tipo *Edge*, guarda toda a informação sobre quais os quatro pontos diferentes do terreno e das arestas que os ligam.

As classes *Rule* e *Constraint* são classes que definem a estrutura das restrições. Uma restrição é

definida pelo formato <tipo_de_terreno> <conector> <Regra>, sendo que a classe *Constraint* possui os seguintes atributos *terrainType*, do tipo *String*; *connector,* do tipo *String,* que pode assumir um de dois valores possíveis "MUST HAVE" e "MUST NOT HAVE", sendo o primeiro uma obrigação positiva a ser cumprida pelo *terrainType* e indicada pela regra e o segundo uma obrigação negativa; *rule,* do tipo *Rule*. A classe Rule, que completa o formato de uma restrição, contém os atributos:

- *nearTerrain*, do tipo *String*, que pode ser uma *String* vazia ou com um tipo de terreno e serve para definir que um terreno deve ter um tipo de terreno na sua vizinhança mais próxima;
- *measurement*, do tipo *String*, que pode assumir um de cinco valores possíveis ("*MORE THAN*" - corresponde ao sinal '>' -, "*MORE OR THE SAME AS*" - correspondente ao sinal '>=' -, "*LESS THAN*" - correspondente ao sinal '<' -, "*LESS OR THE SAME AS*" - sinal '<=' -, "*EXACTLY*" - sinal '==') e serve para definir qual o tipo de comparação a ser feita na regra que é definida;
- *number*, do tipo *double*, que pode assumir valores maiores ou iguais a zero e serve para definir o valor a ser comparado pelo atributo *measurement*;
- *field*, do tipo *String*, que pode assumir valores iguais a qualquer um dos atributos da classe Terrain, expecto o atributo *edges*, e serve para definir qual é esse atributo que servirá como regra da restrição.

2.2 Trabalho efetuado

O trabalho efetuado, até ao momento da elaboração do presente documento, consistiu na implementação das classes representativas do conhecimento e de manipulação do ficheiros *XML*.

As classes em questão são:

- *XML*
    - *readFile(List<details.Edge> edges)* – Lê o ficheiro *terrains.xml* e adiciona a *edges* os terrenos lidos, criando um mapa dos terrenos.
    - *addTerrainToFile(Terrain t)* – Adiciona um terreno ao ficheiro *terrains.xml*.
    - *readConfigurationFile()* - Lê o ficheiro *config.xml* e retorna um objeto do tipo *Map*.
- *Map*
    - *print()* - Escreve na consola os dados dos terrenos e as suas restrições.
    - *updateConstraint(Constraint cOld, Constraint cNew)* – Substitui uma restrição antiga por uma nova.
    - *updateTerrain(Terrain tOld, Terrain tNew)* - Substitui um terreno antigo por um novo.
- *Constraint*
- *Rule*
- *Terrain*
    - *print()* - Escreve no ecrã a informação do terreno.

- *applyUserSelectedConstraint(Constraint c)* – Verifica se é possível aplicar a restrição escolhida.

- *Point*
- *Edge*

2.3 Resultados esperados

Devido à arquitetura enunciada espera-se que a aplicação do algoritmo *A** seja relativamente simples e permita uma execução eficiente. Para garantir o correto funcionamento da aplicação, serão desenvolvidos testes unitários que testem funcionalidade a funcionalidade. De modo a garantir que o algoritmo A* está bem implementado, serão testados ficheiros de configuração de terrenos com vários terrenos e vários edifícios que terão de ser alocados, sendo que nos primeiros testes o número de terrenos e edifícios a alocar será menor do que nos testes seguintes. Os resultados obtidos desses testes serão apresentados no relatório final do projeto.

3 Conclusões

Para obter um sistema eficiente é imperativo conseguir uma arquitetura flexível com uma boa representação dos dados. Dado o problema em questão, pode-se concluir que a representação dos dados é a chave para que a arquitetura possa ser flexível.

4 Recursos

4.1 Bibliografia

[1] «Pesquisa aplicada ao Problema de Alocação de Lotes de Terreno», http://paginas.fe.up.pt/~eol/IA/1213/TRABALHOS/pesqsol_alocacaoLotes.html.
[2] «ruivalentemaia/TerrainIntelligentAllocation», https://github.com/ruivalentemaia/TerrainIntelligentAllocation .

4.2 Software

1. Eclipse IDE
2. LibreOffice Writer