Universidade do Porto
Faculdade de Engenharia
FEUP

Pesquisa aplicada ao Problema de Alocação de Lotes de Terreno

Relatório Final

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

25 de Maio de 2013

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 Funcionalidades

O programa permite ao utilizador:

- Configurar os terrenos que pretende utilizar no processo de alocação;
- Configurar as restrições para cada um dos edifícios que pretende construír;
- Alocar construções de edifícios aos terrenos cumprindo todas as restrições.

2.2 Estrutura do Programa

O programa aqui descrito possui vários módulos diferentes que se interligam entre eles de forma a atingir o objetivo principal do software. Esses módulos são:

- **Terrain** – módulo que possui a configuração do objecto Terrain, que corresponde a um lote, e os seus respetivos atributos e métodos.
- **Map** – módulo que contém a configuração do mapa que é criado após o utilizador inserir os dados sobre cada um dos lotes de terreno (Terrains) do sistema,
- **XML** – módulo que lida com a leitura e escrita em ficheiros .xml.
- **AStar –** módulo que contém a configuração do algoritmo A*.
- **TerrainIntelligentAllocation** – módulo que se liga a todos os restantes módulos e que contém toda a estrutura lógica do funcionamento do programa, desde a inserção de dados pelo utilizador, passando pela execução do algoritmo e pela apresentação dos resultados novamente ao utilizador.

2.3 Esquemas de Representação de Conhecimento

2.3.1 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>
```

```
    <id>1</id>
    <type>NOT FERTILE</type>
    <leaning>1.1</leaning>
    <width>2.2</width>
    <height>3.3</height>
    <price>4.4</price>
  </terrain>
</terrains>
```

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

```
<config>
    <buildings>
    <building>
      <name>School</name>
      <restrictions>
        <restriction>
          <type>MUST HAVE</type>
          <rule>
            <nearTerrain/>
            <measurement>MORE THAN</measurement>
            <value>0.8</value>
            <field>leaning</field>
          </rule>
        </restriction>
      </restrictions>
    </building>
    <building/>
    </buildings>
</config>
```

2.3.2 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;

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.

2.4 Análise de Complexidade

A complexidade do algoritmo A* depende em boa parte da forma como a função heurística está definida e implementada. No caso deste software, a função heurística é definida pela soma do custo da solução parcial estado-a-estado e da soma dos custos do primeiros p elementos no set Lot', sendo p o número de edifícios ainda por ser alocados e Lot' o conjunto de terrenos ainda não-alocados. Em conclusão, visto que o espaço de pesquisa acaba por ser definido por uma árvore e existe apenas um estado final definido, então a complexidade temporal do A*, neste caso, é polinomial.

2.5 Ambiente de Desenvolvimento

O trabalho foi realizado numa computador portátil com um processador Intel® Core™2 Duo CPU P8400 @ 2.26GHz × 2, com o sistema operativo Linux Ubuntu 12.10, no Eclipse e todo o software foi escrito em Java, com recurso a XML para armazenamento de dados.

3 Conclusões

O foco principal do desenvolvimento do software foi o correcto funcionamento do algoritmo A* no contexto da alocação de lotes a possíveis construções, sendo que a maior parte do tempo gasto no planeamento e desenvolvimento do mesmo foi na organização dos dados e na performance do algoritmo. Caso tivesse havido mais tempo, o software possuiria uma interface gráfica de mais fácil utilização, o que permitiria ao utilizador inserir mais rapidamente as características de cada lote assim como as restrições que pretende para as construções a alocar. Seria também mais fácil ao utilizador visualizar o funcionamento do algoritmo em tempo real e perceber quais as decisões passo-a-passo tomadas pelo mesmo.

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