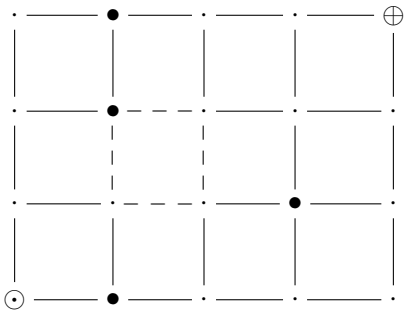
Projecto de Programação Orientada por Objectos 2009/10

MEEC – IST

1 Problema

Considere uma grelha $n \times m$. Pretende-se encontrar o melhor caminho, i.e., o caminho com menor custo, entre um ponto inicial, com coordenadas (x_i, y_i), e um ponto final, com coordenadas (x_f, y_f). O custo do caminho é determinado pelo número de arestas percorridas. Em geral, o custo de cada aresta é 1, mas podem existir zonas especiais em que o custo seja superior. Existem ainda n_{obst} obstáculos em alguns pontos da grelha pelos quais não se pode passar.

Como exemplo, considere-se a figura seguinte, representando uma grelha 5×4:



Nesta figura, o ponto inicial, assinalado por $\odot$, tem coordenadas $(1, 1)$, o ponto final, assinalado por $\oplus$, tem coordenadas $(5, 4)$, os obstáculos estão assinalados por $\bullet$, e as arestas da zona de custo especial, assinaladas a tracejado, têm custo 4. Neste caso, o melhor caminho é $(1, 1), (1, 2), (2, 2), (3, 2), (3, 1), (4, 1), (5, 1), (5, 2), (5, 3), (5, 4)$ com custo 12. Existem caminhos mais curtos mas cujo custo é superior. Por exemplo, o caminho $(1, 1), (1, 2), (2, 2), (3, 2), (3, 3), (4, 3), (5, 3), (5, 4)$ é mais curto mas o seu custo é 13.

2 Abordagem

O objectivo deste projecto é programar em Java uma solução para o problema acima descrito utilizando programação evolutiva especificada e implementada por objectos.

A ideia é gerar no instante zero uma população de ν indivíduos, todos colocados no ponto inicial, e fazê-la evoluir até ao instante final τ. A cada indivíduo z está associado um conforto

$$\varphi(z) = \left(1 - \frac{custo(z) - comp(z) + 2}{(cmax - 1) \times comp(z) + 3}\right)^k \left(1 - \frac{dist(z, (x_f, y_f))}{n + m + 1}\right)^k$$

onde:

- *cmax* é o custo máximo de uma aresta da grelha;
- $custo(z)$ é o custo do caminho do indivíduo z;
- $comp(z)$ é o comprimento desse caminho – o número de arestas percorridas;
- $dist(z, (x_f, y_f))$ é a distância do último ponto do caminho ao destino – menor número de arestas que é necessário percorrer;
- k é um parâmetro introduzido pelo utilizador de sensibilidade do conforto a pequenas variações.

Cada indivíduo z evolui de acordo com o seu conforto, através dos seguintes mecanismos aleatórios.

- **Morte**, variável exponencial com tempo médio $(1 - \log(1 - \varphi(z)))\mu$ entre eventos.
- **Reprodução**, variável exponencial com tempo médio $(1 - \log(\varphi(z)))\rho$. Da reprodução surge um novo indivíduo cujo caminho é um prefixo do caminho do progenitor. O comprimento deste prefixo é determinado pelo conforto do progenitor aproveitando sempre 90% do caminho e uma fracção $\varphi(z)$ dos restantes 10%.
- **Deslocamento** (usualmente chamado mutação em programação evolutiva), variável exponencial com tempo médio $(1 - \log(\varphi(z)))\delta$ entre eventos. O deslocamento pode ocorrer equiprovavelmente para uma posição adjacente na grelha não ocupada por um obstáculo. Se o indivíduo se deslocar para uma posição que já existe no seu caminho deve eliminar o troço compreendido entre as duas ocorrências dessa posição.

A população evolui em função da evolução individual dos seus elementos e ainda por ocorrência de **epidemias**. Quando o número de indivíduos excede um máximo ν_{max}, ocorre uma epidemia. À epidemia sobrevivem sempre os cinco indivíduos com maior conforto. Para cada um dos restantes, a probabilidade de sobrevivência é $\varphi(z)$.

A evolução da população deverá ser regida por simulação estocástica discreta, isto é, baseada numa cadeia de acontecimentos pendentes.

3 Parâmetros

O programa deve receber o conjunto seguinte de dados:

- dimensões n e m da grelha;
- as coordenadas (x_i, y_i) do ponto inicial e (x_f, y_f) do ponto final;

- uma lista de tuplos (coordenada, coordenada, custo)

$$\{((x_1, y_1), (x'_1, y'_1), c_1), \ldots, ((x_k, y_k), (x'_k, y'_k), c_k)\}$$

  para especificar as zonas rectangulares de custo especial;

- o número de obstáculos n_{obst} e as coordenadas (x_i, y_i), para $i = 1, \ldots, n_{obst}$, desses obstáculos;

- instante final $\tau(> 0)$ da evolução;

- lista de parâmetros $k, \nu, \nu_{max}, \mu, \delta, \rho$.

3.1 Formato do ficheiro de dados

O ficheiro que descreve os parâmetros de entrada da simulação é um ficheiro XML. A simulação é um elemento `simulacao` cujos atributos indicam o instante final (`instfinal`), população inicial (`popinicial`) e população máxima (`popmaxima`) da simulação, assim como a sensibilidade do conforto a variações (`sensconforto`). O elemento `simulacao` contém seis elementos:

- O elemento vazio `grelha` cujos atributos indicam a sua dimensão (`numcolunas` e `numlinhas`).

- O elemento vazio `pontoinicial`, cujos atributos indicam as coordenadas iniciais (`xinicial` e `yinicial`).

- O elemento vazio `pontofinal`, cujos atributos indicam as coordenadas finais `xfinal` e `yfinal`).

- O elemento vazio `zonascustoespecial` contém por sua vez uma lista de elementos `zona` com atributos que indicam a sua posição na grelha (`xinicial`, `yinicial`, `xfinal`, `yfinal`) e cujo conteúdo indica o respectivo custo.

- O elemento `obstaculos`, cujo único atributo descreve o número de obstaculos (`num`), contém por sua vez uma lista de elementos vazios `obstaculo` com atributos que indicam a sua posição na grelha (`xpos`, `ypos`).

- O elemento `eventos`, que por sua vez contém três novos elementos vazios (`morte`, `reproducao` e `deslocamento`) cujos atributos (`param`) descrevem os parâmetros relativos aos eventos morte, reprodução e mutação.

3.2 Exemplo

Como ilustração considere o exemplo descrito na Secção 1:

- dimensões da grelha: $n = 5$ e $m = 4$;

- ponto inicial: $(x_i, y_i) = (1, 1)$;

- ponto final: $(x_f, y_f) = (5, 4)$;

- zona de custo especial: $(x_1, y_1) = (2, 2)$, $(x'_1, y'_1) = (3, 3)$ e $c_1 = 4$;

- obstáculos: $n_{obst} = 4$ e coordenadas $(2, 1), (2, 3), (2, 4), (4, 2)$;

- instante final da evolução: $\tau = 100$;

- população inicial: $\nu = 10$;
- população máxima: $\nu_{max} = 100$;
- sensibilidade do conforto: $k = 3$;
- parâmetros relacionados com os eventos: $\mu = 10, \delta = 1, \rho = 1$.

Seguem os mesmo parâmetros em XML:

```
<simulacao instfinal="100" popinicial="10" popmaxima="500" sensconforto="3" >
  <grelha numcolunas="5" numlinhas="4"/>
  <pontoinicial xinicial="1" yinicial="1" >
  <pontofinal xfinal="5" yfinal="4" >
  <zonascustoespecial>
    <zona xinicial="2" yinicial="2" xfinal="3" yfinal="3" >4</zona>
  </zonascustoespecial>
  <obstaculos num="4" >
    <obstaculo xpos="2" ypos="1"/>
    <obstaculo xpos="2" ypos="3"/>
    <obstaculo xpos="2" ypos="4"/>
    <obstaculo xpos="4" ypos="2"/>
  </obstaculos>
  <eventos>
    <morte param="10"/>
    <reproducao param="1"/>
    <mutacao param="1"/>
  </eventos>
</simulacao>
```

4 Resultados

O programa deve imprimir para o terminal no final da simulação o caminho do indivíduo mais adaptado ao longo de toda a simulação. Por caminho do indivíduo mais adaptado entende-se:

- caso algum indivíduo atinja o ponto final, o caminho do indivíduo z com menor custo, independentemente de o indivíduo z ter ou não sobrevivido até ao fim da simulação;
- caso nenhum indivíduo atinja o ponto final, o caminho do indivíduo z com maior conforto, independentemente de o indivíduo z ter ou não sobrevivido até ao fim da simulação;

Durante a simulação o programa deve ainda imprimir para o terminal o resultado de observações da população, realizadas de $\tau/20$ em $\tau/20$ unidades de tempo. Cada observação deve incluir o instante actual, o número de eventos já realizados, a dimensão da população, o caminho do melhor indivíduo e o respectivo custo/conforto (custo no caso de ter sido atingido o ponto final e conforto caso contrário), segundo o seguinte formato:

Observacao x:

Instante actual:	instante
Numero de eventos realizados:	eventos
Dimensao da populacao:	dimensao
Foi atingido o ponto final:	sim/nao
Caminho do individuo mais adaptado:	$\{(x_1, y_1), \ldots, (x_j, y_j)\}$
Custo/Conforto:	custo/conforto

Qualquer outra impressão para o terminal, ou uma impressão deste conteúdo fora deste formato, incorre em penalização na nota do projecto.

5 Simulação

O simulador deve executar os seguintes passos:

1. Ler o ficheiro que descreve os parâmetros de entrada da simulação, e guardar/criar os valores/objectos necessários. O ficheiro com os dados de entrada é um ficheiro XML, cujo formato é descrito na Secção 3.1, e deve ser validado através de um DTD apropriado.
2. Executar o ciclo de simulação até que: (i) o tempo limite de simulação seja atingido; ou (ii) não haja mais eventos a simular. Durante a simulação devem ser impressos para o terminal as observações da população descritas na Secção 4.
3. No final da simulação deve ser imprimido para o terminal a informação pedida na Secção 4.

6 Avaliação

O projecto vale 8 valores da nota final que se distribuem da seguinte forma:

1. **(2 val) Relatório intercalar: 27 Abril 2010**
   - Nesta fase deverá ser entregue a especificação em UML do diagrama de classes e pacotes (tão detalhado quanto possível).
   - O diagrama UML deve ser entregue via fenix.
2. **(6 val) Relatório final: 27 Maio 2010**
   - Nesta fase deverá ser entregue as fontes do programa, o respectivo executável .jar (com os ficheiros .java, .class, e MANIFEST.MF organizados correctamente em directorias), e a documentação (gerada pela ferramenta javadoc) da aplicação.
   - As fontes, executável e documentação da aplicação devem ser entregues via fenix.
3. **Discussão final: 1 e 8 Junho 2010**
   A distribuição dos grupos para a discussão final será disponibilizada oportunamente. Todos os membros do grupo devem estar presentes na discussão. A nota final do projecto dependerá desta discussão, e não será necessariamente a mesma para todos os membros do grupo.

Tanto para o relatório intercalar como para o final, projectos entregues após a data estabelecida terão a seguinte penalização: por cada dia de atraso haverá uma penalização de 2^n valores da

nota, onde n é o número de dias em atraso. Ou seja, relatórios entregues com 1 dia de atraso serão penalizados em $2^1 = 2$ valores (o relatório intercalar incorre numa penalização de 0.2 valores da nota final, o relatório final incorre numa penalização de 0.6 valores da nota final), relatórios entregues com 2 dias de atraso serão pensalizados em $2^2 = 4$ valores (o relatório intercalar incorre numa penalização de 0.4 valores da nota final, o relatório final incorre numa penalização de 1.2 valores da nota final), etc. Por dia de atraso entende-se ciclos de 24h a partir do dia estabelecido para a entrega.