Programação Genérica Aplicada a Algoritmos Geográficos

LÚBIA VINHAS
GILBERTO RIBEIRO QUEIROZ
KARINE REIS FERREIRA
GILBERTO CÂMARA
JOÃO ARGEMIRO C. PAIVA
INPE – Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas, 1758, São José dos Campos (SP), Brasil 12227-001
{lubia, gribeiro, karine, gilberto, miro}@dpi.inpe.br

Abstract. This work discusses the application of generic programming paradigm to clustering algorithms on geographic data. The basic ideas of generic programming: (a) defines the basic steps of the algorithm independently of data structures and (b) identifies a set of requirements that the data structures must satisfy; are applied to implement a K-Means and a "Equal Step" clustering algorithms. This approach leads to a code reuse and flexibility without loss of efficiency.

1 Introdução

O paradigma de orientação a objetos tem sido largamente empregado nas arquiteturas de desenvolvimento de software, incluindo a maioria das ferramentas disponíveis para o desenvolvimento de Sistemas de Informações Geográficas (GIS). Este paradigma é baseado no princípio de agrupamento de elementos similares do mundo em classes e por tornar explícitos comportamentos e propriedades comuns através do mecanismo de herança. Através desse mecanismo, classes derivadas compartilham propriedades comuns de uma classe base, enquanto mantêm seus comportamentos específicos (Stroustrup, 1997). No desenvolvimento de aplicativos GIS, esse paradigma, freqüentemente, leva ao estabelecimento de classes diretamente relacionadas aos diferentes tipos de representações geométricas. Por exemplo, um típico GIS orientado a objeto pode ter classes básicas tais como Imagem, Polígono, Mapa de Polígonos, Grade Regular, TIN (grade irregular triangular), Ponto e Conjunto de Pontos. As classes contem tanto a estrutura de dados subjacente quanto o correspondente conjunto de algoritmos. Essa subdivisão de classes possibilita a decomposição do sistema em módulos, auxiliando os desenvolvedores de software e permite uma divisão conveniente do trabalho entre a equipe de desenvolvimento.

Existe, no entanto, uma desvantagem na aplicação direta da orientação a objetos no desenvolvimento de GIS: o acoplamento dos algoritmos às estruturas de dados. Um grande número de algoritmos não depende da implementação particular de uma estrutura de dados, mas apenas de algumas propriedades semânticas fundamentais destas. Tais propriedades podem ser – por exemplo – a habilidade de ir de um elemento da estrutura ao próximo e a de comparar dois elementos. Por exemplo, um algoritmo para computar o índice de autocorrelação espacial independe se os elementos se processa um conjunto de pontos, um conjunto de polígonos, um TIN, uma grade ou uma imagem. O que é necessário é a habilidade de procurar percorrer o conjunto de elementos, e obter, para cada elemento, seu valor e os índices dos outros elementos do conjunto que satisfazem uma certa propriedade (por exemplo, aqueles que estão mais próximos no espaço do que uma determinada distância). De maneira similar, muitos algoritmos de análise espacial podem ser abstraídos da estrutura de dados em particular e serem descritos apenas em termos de suas propriedades. Apesar disso, em muitas bibliotecas GIS, o uso dos princípios de orientação a objeto tem resultado em classes que contêm tanto a estrutura de dados subjacente quanto o conjunto de algoritmos correspondente. Ou seja, os algoritmos estão desnecessariamente acoplados a uma estrutura de dados particular, e o mesmo algoritmo (por exemplo, o cálculo de histograma) está implementado separadamente para cada estrutura de dados.

Nesse trabalho, argumenta-se que o desenvolvimento de GIS, especialmente na construção de algoritmos de análise espacial, pode ser beneficiado pela combinação da técnica de orientação a objeto e do paradigma de programação genérica. Ao invés de manipular os tipos de dados diretamente (p.ex. "classes" em C++), um algoritmo genérico trabalha sobre abstrações ou conceitos assumindo-se que esses possuem propriedades precisas. O ponto chave do projeto de algoritmos genéricos é a identificação de conjunto mínimo de requisitos impostos pelo algoritmo a fim de que possa ser executado

eficientemente. Uma vantagem da aplicação desse paradigma é que uma mesma implementação de um algoritmo se aplique a diferentes estruturas de dados.

Essa idéia é exemplificada na implementação de algoritmos de agrupamento de dados geográficos. Os algoritmos de agrupamento particionam os objetos em conjuntos (*clusters*) onde os elementos de um conjunto são mais semelhantes entre si, segundo algum critério, do que a elementos de outros conjuntos. Um agrupamento pode ser visto como uma partição sobre um espaço de atributos, definidos sobre algum critério (Bailey, 1995). Os algoritmos de agrupamento podem ser aplicados, por exemplo, tanto a uma imagem de sensor remoto quanto a um conjunto de objetos geográficos vetoriais com base em um de seus atributos, pois são procedimentos bem definidos que independem de uma estrutura de dados em particular.

Outros exemplos da adoção desse paradigma podem ser encontrados nas bibliotecas VIGRA – *Vision with Generic Algorithms* (Köthe,1999), que implementa um conjunto de algoritmos típicos de visão computacional. E a MTL – Matrix Template Library (Sieck, 1999) aplicada ao domínio de álgebra linear numérica.

A seção 2 aborda a programação genérica, a seção 3 mostra como a programação genérica pode ser aplicada a algoritmos de agrupamento e a seção 4 apresenta o ambiente de trabalho e os resultados obtidos com a implementação dos algoritmos genéricos de agrupamento. Por último, é apresentada a conclusão.

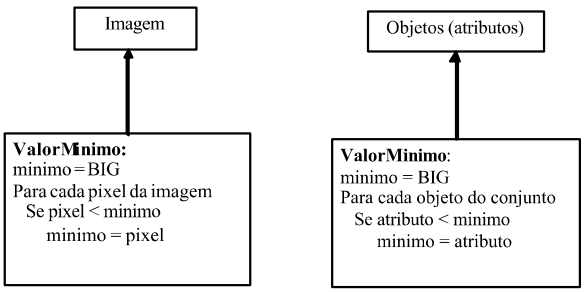
2 Programação Genérica

Programação genérica refere-se a um tipo de abstração no processo de desenvolvimento de software, voltado para a construção de algoritmos que independem de um determinado tipo ou estrutura de dados e que, no entanto, sejam tão eficientes quanto se construídos acoplados a uma determinada estrutura. Os algoritmos genéricos são escritos com base em um conjunto de requisitos sobre as estruturas que manipulam, sem explicitamente especificá-las. Sejam essas, tipos simples, compostos ou definidos pelo usuário, desde que atendam os requisitos especificados, podem ser manipuladas pelo mesmo algoritmo sem a necessidade de sua replicação para cada um deles (Austern, 1999). A chave para separação entre estruturas e algoritmos é o uso de *iteradores*, ponteiros generalizados que fornecem a ligação entre algoritmos e estruturas de dados.

Essa técnica de programação já resultou em ferramentas de programação como a STL (Standard Template Library), que é parte do padrão ISO C++ (Stroustrup 1997). Nela é fornecido um grande número de estruturas de dados, ou contêineres genéricos (tais como *list*, *set* e *map*), bem como algoritmos que operam sobre eles (tais como *sort* e *search*), de forma que seu comportamento independe de contêiner sobre o qual são aplicados.

A aplicação do paradigma de programação genérica para a construção de algoritmos genéricos em GIS pode ser descrita como um processo de quatro passos: (a) encontre padrões de manipulação de dados espaciais pelos algoritmos; (b) generalize as estruturas de dados em tipos abstratos (contêineres) que independam do tipo dos elementos; (c) forneça *iteradores* que permitam o acesso aos contêineres, atendendo aos requisitos demandados pelos algoritmos; (d) projete algoritmos que usem estes *iteradores* ao invés de acessar a estrutura diretamente.

Como exemplo, suponha um algoritmo para determinar o valor mínimo dentre os *pixels* de uma imagem, armazenada em uma dada estrutura de dados, e dentre um conjunto de objetos geográficos, representados em uma outra estrutura. *Pixels* são comparados com base em seu valor, e objetos são comparados com base em um de seus atributos. Uma abordagem dependente de tipos para o algoritmo de valor mínimo, iria escrever um algoritmo para cada estrutura de dados, em um mecanismo representado na Figura 2-1.



(a) valor mínimo de imagem

(b) valor mínimo de conjunto de objetos

Figura 2-1 Representação de algoritmo acoplado a estrutura de dados

Segundo a abordagem de programação genérica, o algoritmo de valor mínimo seria definido em termos do conjunto de passos que o descrevem:

- percorra toda a estrutura
- para cada elemento, compare com o valor mínimo atual

- substitua o valor mínimo atual pelo valor do elemento caso esse seja menor

Com base na definição de passos do algoritmo, é feito um levantamento dos requisitos que uma estrutura de dados deve atender a fim de poder ser manipulada por esse algoritmo (o resultado dessa abordagem é ilustrado na Figura 2-2). Nesse exemplo, observa-se a seguinte lista de requisitos:

- a estrutura deve oferecer algum mecanismo de percorrimento de seus elementos um a um, do primeiro ao último;
- os elementos devem poder ser comparados entre si.

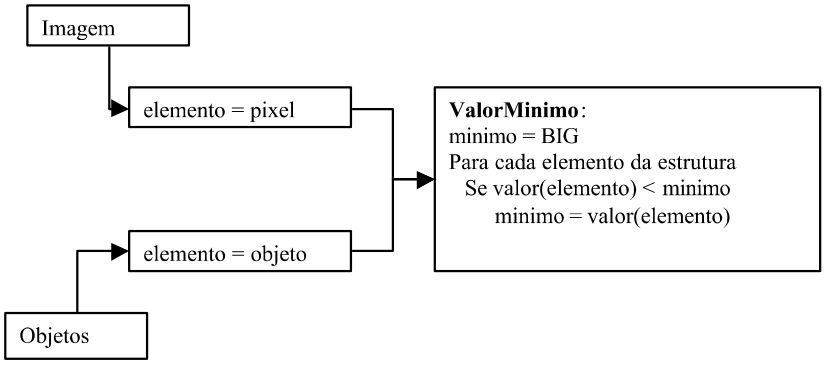


Figura 2-2 Representação de algoritmo desacoplado de estrutra de dados

De maneira geral, o conceito de programação genérica consiste, portanto, em se encontrar abstrações em dois níveis (Austern, 1999):

1. as instruções que representam o algoritmo e
2. o conjunto de requisitos sobre as estruturas de dados e tipos, demandados pelos algoritmos.

3 Programação Genérica Aplicada a Algoritmos de Agrupamento

Os algoritmos de agrupamento particionam os objetos em conjuntos (*clusters*) onde os elementos de um conjunto são mais semelhantes, segundo algum critério, do que de elementos de outros conjuntos. Um agrupamento pode ser visto como uma partição sobre um espaço de atributos, definidos sobre algum critério (Bailey,1995).

Como exemplos, apresenta-se os algoritmos de agrupamento **k-médias** e **passo igual**, descritos a seguir em termos dos passos de cada um.

3.1 K-Médias

Informalmente, o agrupamento por k-médias irá encontrar k grupos (k é escolhido a priori) onde a variância dos objetos dentro de cada grupo é mínima. Cada grupo é identificado por um valor chamado de centróide do grupo. Abaixo são apresentados os passos envolvidos no algoritmo K-Médias:

- Determinar os k centróides iniciais dos grupos, segundo algum critério, por exemplo, aleatoriamente dentro do espaço dos objetos ou de acordo com alguma heurística;
- Calcular a distância entre cada objeto e o centróide de cada grupo já definido;
- Alocar o objeto ao grupo cuja distância ao seu centróide seja a menor;
- Recalcular os centróides dos grupos considerando os objetos re-alocados;
- Repetir os passos de 2 a 4 até que algum critério de convergência seja atingido.

3.2 Passos Iguais

O agrupamento em k grupos, por passos iguais, de um conjunto de valores consiste em dividir o intervalo determinado pelo menor até o maior valor em k fatias, onde cada fatia possui o mesmo comprimento. Cada fatia é identificada pelo seu limite inferior e limite superior. Cada elemento é atribuído à fatia cujo intervalo contém seu valor. Abaixo são apresentados os passos envolvidos no algoritmo de agrupamento em passos iguais:

1. Encontrar o intervalo global dos dados, definido pelo menor e o maior valor dentre o conjunto de valores;
2. Dividir o intervalo global em k intervalos. Cada intervalo define um grupo;
3. Associar cada elemento do conjunto ao grupo cujo intervalo o contém seu valor.

3.3 Levantamento de Requisitos

Uma vez definidos os passos de cada algoritmo, desacoplados de estruturas de dados específicas, o próximo passo é identificar quais requisitos os algoritmos demandam das possíveis estruturas de dados sobre as quais irá atuar. Observa-se que:

1. Ambos os algoritmos necessitam que a estrutura sobre a qual sejam aplicados, forneça um mecanismo de percorrimento de cada um de seus elementos.
2. Os elementos das estruturas devem possuir uma forma de comparação entre si.

Um conceito bem definido que permite o desacoplamento do algoritmo dos detalhes de percorrimento de cada estrutura específica é o iterador. O iterador trata a estrutura de dados como uma seqüência de elementos, referindo-se a um dos elementos e fornecendo operações necessárias para iterar nessa seqüência. Do ponto de vista do algoritmo o iterador permite o percorrimento de estruturas sem o conhecimento exato de qual tipo seja a estrutura (Stroustrup, 1997).

4 Resultados

Os algoritmos foram implementados no ambiente TerraLib (Câmara et al., 2001), que é uma biblioteca de classes para o desenvolvimento de aplicações GIS, em linguagem de programação C++ (Meyers, 1997) utilizando o compilador Microsoft Visual C++ 6.0, na plataforma PC/Windows. A TerraLib oferece uma classe chamada `TeRaster` para a representação de dados matriciais, a qual pode ser instanciada a partir de imagens em diferentes formatos, como GeoTIFF e JPEG (TerraLib, 2002). Para a aplicação dos algoritmos genéricos de agrupamento, foi implementado na classe `TeRaster` o conceito de iterador, atendendo aos requisitos do algoritmo, conforme definido na seção 3.

A biblioteca possui também suporte para a manipulação de objetos geográficos discretos, com uma representação vetorial para a sua componente espacial e com atributos descritivos os quais podem ser usados em critérios de agrupamento. A classe `TeSelectedObject` é a estrutura de dados fornecida para armazenar um conjunto desses objetos e sobre ela podem ser aplicados os algoritmos de agrupamento. Nessa classe também foi implementado o conceito de iterador sobre os objetos.

A implementação dos algoritmos genéricos de agrupamento, em função de iteradores, é mostrada na Figura 4-1.

A aplicação do K-Médias a uma imagem, tem por objetivo agrupar *pixels* de valor de intensidade próximos, o que corresponde, no caso de imagens de sensores remotos, a um tipo de classificação (Mather, 1999). A Figura 4-2 mostra em (a) a utilização do algoritmo K-Médias sobre uma imagem, e em (b) o resultado do agrupamento onde o tom de cinza de cada *pixel* da imagem resultante corresponde ao grupo ao qual pertence.

A aplicação do K-Médias a um conjunto de objetos vetoriais tem por objetivo agrupar objetos segundo o critério de semelhança sobre um de seus atributos descritivos. A Figura 4-3 mostra o resultado da aplicação do mesmo algoritmo de k-médias ao atributo ÁREA dos objetos relativos aos setores censitários do município de São Paulo.

5 Conclusões

Esse trabalho apresentou um exemplo de aplicação de programação genérica a algoritmos geográficos de agrupamento. Mostrou-se que usando esse paradigma obtem-se maior eficiência no desenvolvimento de GIS, uma vez que os algoritmos não precisam ser reimplementados em diferentes estruturas de dados. Ou seja, o código dos algoritmos fica mais modular e reusável, pois alterações nas estruturas de dados não se refletem na reimplementação dos algoritmos e vice-versa.

6 Referências Bibliográficas

Austern, M. H. *Generic Programming and STL.Using and Extending the C++ Standard Template Library.* Canada, 1999

Câmara, G., Souza, R., Pedrosa, B. and et.al. *Terralib: Technology in Support of GIS Innovation*. II Workshop Brasileiro de Geoinformática, GEOInfo 2000.

INPE – Divisão de Processamento de Imagens *TerraLib* [online] <www.dpi.inpe.br/ terralib> Agosto, 2002.

Köthe, U. Reusable Software in Computer Vision In: Jähne, B. Haußecker, H., Geißler P. (eds.) *Handbook on Computer Vision and Applications*, Vol. 3, Acadamic Press, 1999.

Mather, P. M. *Computer Processing of Remotely-Sensed Images.* Wiley, 1999.

Meyers, S. *Effective C++: 50 Specific Ways to Improve Your Programs and Design.* Addison-Wesley, 1997.

Remy, N. *The Geostatistics Template Library* [online] <http://pangea.stanford.edu/~nremy/GTL/GsTL_home.html, Agosto, 2002.

Sieck, J. G; Lurnsdane, A. The Matrix Template Library: A Generic Programming Approach to High Performance Numerical Linear Algebra. *In Proc.1999 International Symposium in Computing in Object-Oriented Parallel Environments*, Springer Lecture Notes in Computer Science, pp. 59–70. 1999.

*S*troustrup, B. *The C++ Programming Language*. Massachusetts, 1997.

```
template<typename It, typename C> void
TeKMeans(It begin, It end, C& result,
         double conv)
{
  C centroides, centr_aux;
  MinimumValue(begin, end, minVal);
  MaximumValue(begin, end, maxVal);
  Initialize(centroides, minVal, maxVal);
  centr_aux = centroides;
  do
  {
      for(It it = begin; it != end; it++)
      {
         j=MinimumDistance(it, centroides);
         ReCalculate(it,centr_aux[j]);
      }
      centroides = centr_aux;
  } while (CheckConvergence(centroides,
           centr_aux,conv));
  result = centroides;
  return;
}
```

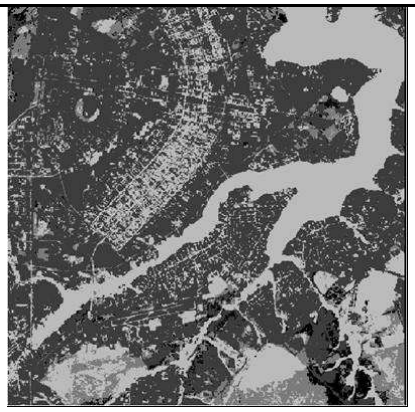
(a) Algoritmo K-Médias

```
template<class It> void
TeDefineEqualStep(It begin, It end,
         int nstep, TeSliceVector& result)
{
    MinimumValue(begin, end, minVal);
    MaximumValue(begin, end, maxVal);
   double slice = (maxValue -
                   minValue)/double(nstep);
   int ns;
   for(ns=0; ns<nstep; ns++)
   {
       TeSlice ps;
       ps.from_= minVal+double(ns)*slice;
       ps.to_ = minVal+double(ns+1)*slice;
       result.push_back(ps);
    }
    return;
}
```

(b) Algoritmo Passo Igual

Figura 4-1 Algoritmos genéricos de agrupamento

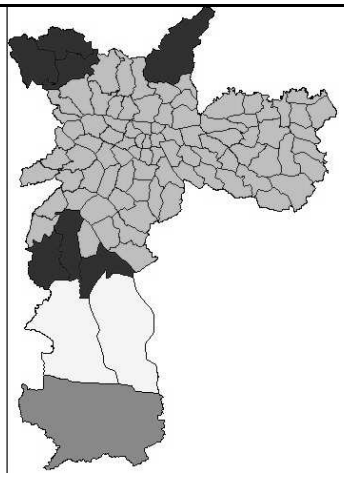
```
main()
{
  TeRaster ima("Brasilia.tif");
  TeRaster ima2("BrasClass.raw");
  vector<TeCentroid> res(4);
  TeKMeans(ima.begin(),ima.end(),res,0.01);
  TeClassify(ima.begin(),ima.end(),res,
              ima2.begin());
}
```



(a) Utilização do algoritmo (b) Imagem agrupada em 4 grupos

Figura 4-2 Aplicação do algoritmo K-Médias genérico em imagem

```
main()
{
  TeObjectMap objs("sampa.mdb", "AREA_");
  TeObjectMap objs2;
  vector<TeCentroid> res(4);
  TeKMeans(objs.begin(),objs.end(),res,0.01);
TeClassify(objs.begin(),objs.end(),res,objs2.begin());
}
```



(a) Utilização do algoritmo (b) Objetos agrupados em 4 grupos

Figura 4-3 Aplicação do algoritmo K-Médias genérico em um conjunto de objetos