Revista Română de Interacţiune Om-Calculator 6 (1) 2013, 79-96

Soluție software de reprezentare a proximităților în GIS folosind matrice de adiacență

Marian Dârdală, Titus Felix Furtună, Adriana Reveiu

Academia de Studii Economice – ASE Bucureşti
Piața Romană, Nr. 6, 010374, Bucureşti
E-mail: dardala@ase.ro, titus@ase.ro , reveiua@ase.ro

Rezumat. Articolul îşi propune să prezinte modul în care s-a proiectat şi implementat un modul software, pentru a realiza o descriere alternativă a unei compoziții tematice precum şi aplicații ale acestei reprezentări. Se porneşte de la un strat tematic ce conține date spațiale de tip poligon care se descrie, alternativ, printr-o matrice de adiacență, ce reține informații de vecinătate pentru fiecare din vectorii existenți în caracteristica spațială de tip poligon. De asemenea, sunt prezentate şi aplicații ale acestei reprezentări: colorarea hărților cu un număr minim de culori şi determinarea coeficientului de corelație spațială Moran. Colorarea hărților se realizează astfel încât oricare ar fi două poligoane învecinate acestea să nu aibă aceeaşi culoare. Coeficientul de corelație spațială *Moran* este standardul *defacto* folosit pentru determinarea gradului de dependență dintre valorile unei variabile în locațiile învecinate. Acesta este folosit pentru identificarea unui şablon în distribuția spațială a valorilor unei variabile de natură economică şi a existenței unor factori comuni care influențează obținerea acestor valori.

Cuvinte cheie: matrice de adiacență, GIS, Backtracking, Greedy, simbologie, coeficientul de corelație spațială Moran.

1. Introducere

In cadrul articolului se prezintă o soluție software pentru generarea unei structuri de date capabilă să reprezinte vecinătățile unui strat tematic ce conține date spațiale de tip poligon, în cadrul unui sistem informatic geografic (GIS). Ideea articolului a pornit de la faptul că există numeroase probleme care se rezolvă luând în considerare informații cu privire la proximitatea datelor spațiale, dintr-o hartă. Răspunzând acestor cerințe, în cadrul articolului propunem o implementare obiectuală a problemei proximității datelor spațiale în GIS, astfel încât am folosit o descriere a vecinătăților prin intermediul unei matrice de adiacență. Clasa care a fost

proiectată şi implementată în acest sens poate servi ca fiind clasă de bază pentru diferite extensii care rezolvă probleme concrete de procesare în GIS.

Pentru identificarea proximității s-a utilizat operatorul spațial de identificare a caracteristicilor care partajează zone comune de graniță. Pentru a exemplifica utilitatea matricei de adiacență a fost derivată din aceasta o clasă care implementează o soluție a problemei colorării hărților cu un număr minim de culori. Astfel, s-a utilizat o metodă de determinare a soluției optime prin tehnica de programare *Backtracking* care foloseşte cel mult patru culori distincte pentru colorare cât şi o metodă euristică de colorare prin tehnica *Greedy*. Pentru o maximă generalitate a fost construită o componentă software de tip *add-in,* care să permită adăugarea acestei noi funcționalități la aplicația *ArcMap*. Modulul a fost testat pe harta României la nivel NUTS 3 (*Nomenclature of Territorial Units for Statistics*) de divizare în unități administrativ teritoriale care, pentru țara noastră, înseamnă împărțirea pe județe a teritoriului României. După cum se va observa din detaliile de proiectare şi implementare, modulul poate fi folosit pentru orice strat tematic spațial de tip poligon (de exemplu, țările unui continent, diferite niveluri de împărțire teritorială a unei țări etc.).

Prin utilizarea produsului software *ArcMap* s-a constatat că acesta are posibilitatea colorării hărții pe baza valorilor unice ale unui câmp şi fiecărei valori unice i se asociază o culoare distinctă dintr-o rampă de culoare (paletă de culori). Modul de simbolizare a hărților prin colorare nu vizează afişarea hărților folosind un număr minim de culori, lucru care a fost tratat, de noi, în acest articol.

Studiul de caz cu privire la generarea cartogramelor prin utilizarea unei simbologii adecvate a fost realizat practic folosind *framework*-ul *ArcObjects*.NET dezvoltat de ESRI şi limbajul de programare C# din mediul Visual Studio.NET. S-a urmărit folosirea *framework*-ului *ArcObjects* pentru a dezvolta o nouă ierarhie de clase în vederea modelării datelor şi vizualizării rezultatelor în GIS.

2. Generarea matricei de adiacență

Matricea de adiacență a fost implementată sub forma unei clase care permite derivarea altor clase ce implementează procesări pe baza acesteia. In esență, clasa permite ca pentru un strat tematic având date spațiale de tip poligon să se genereze matricea de adiacență în care se stochează valori ce indică

vecinătățile poligoanelor. Astfel, matricea *M* este pătratică, având *n x n* elemente, unde *n* este numărul de vectori (poligoane) din stratul tematic analizat iar elementele au semnificația:

$$M_{i,j} = \begin{cases} 1, \textit{dacă poligonul i se învecinează cu j} \\ 0, \textit{dacă poligonul i nu se învecinează cu j} \end{cases}$$

Vecinătățile au fost obținute prin aplicarea operatorului spațial *esriSpatialRelTouches* care identifică toate poligoanele care au porțiuni comune ale marginii cu un poligon pentru care a fost aplicat.

Pentru o maximă generalitate, clasa *Matr_adiac* are un constructor care primeşte ca parametri: documentul hartă (*fhd*), numele caracteristicii spațiale pentru care se va genera matricea de adiacență (*ncsp*) şi un câmp de tip şir de caractere (*fet*) utilizat pentru referirea elementelor matricei, altfel decât prin (*i*,*j*). Acest câmp trebuie să eticheteze în mod unic fiecare vector din caracteristica spațială analizată. Constructorul are ca scop stocarea parametrilor ca date membre ale clasei, identificarea stratului tematic, verificare tipului acestuia (poligon), identificarea câmpului indicat pentru etichetare şi obținerea indexului lui. Pe baza numărului de vectori (*n*), din stratul tematic se calibrează vectorul de şiruri pentru etichete (*et*), matricea (*mat*), se încarcă etichetele şi apoi se apelează metoda de generare efectivă a matricei de adiacență: *genereaza_matrice*().

```
public class Matr_adiac
{
   protected IMxDocument hd; // documentul harta
   IMap h; // harta
   protected ILayer lyr=null; // stratul tematic de tip poligon
   IFeature vector; // caracteristica spatiala
   protected string numecp; // nume camp pentru etichete
   protected string[] et; // etichete
   protected byte[,] mat; // matricea de adiacență
   protected int n, nrc_et; // nr de vectori, indexul campului de
     etichete
   public Matr_adiac(IMxDocument fhd, string ncsp, string fet)
   {
      int i, t, nrc, j;
```

```
      numecp = fet;  hd = fhd;
      h = hd.FocusMap;
      for (i = 0; i < h.LayerCount; i++)
            if (h.get_Layer(i).Name == ncsp)
               {
lyr = h.get_Layer(i);
break;
}
      if (i == h.LayerCount)
throw new Mat_Ex("Nu exista layer-ul:" + ncsp);
      IFeatureLayer fl = (IFeatureLayer)lyr;
      if (fl.FeatureClass.ShapeType !=
                  esriGeometryType.esriGeometryPolygon)
                throw new Mat_Ex("Layerul nu e tip POLYGON!!");
      nrc_et = t = fl.FeatureClass.FindField(fet);
      n = nrc = fl.FeatureClass.FeatureCount(null);
      et = new string[nrc];
      IFeatureCursor ifc = fl.FeatureClass.Search(null, true);
      i=0;
      vector = ifc.NextFeature();
      while(vector != null)
         {
            et[i] = (string)vector.get_Value(t);  i++;
            vector = ifc.NextFeature();
         }
      mat = new byte[nrc, nrc];
      for (i = 0; i < nrc; i++)
for (j = 0; j < nrc; j++) mat[i, j] = 0;
      genereaza_matrice();
    }
    void genereaza_matrice()
    {
```

```
        int i = 0, j = 0;
        IQueryFilter iq = new QueryFilterClass();
        ISpatialFilter fs = new SpatialFilter();
// pentru fiecare poligon identificat prin eticheta lui
        foreach (string t in et)
        {
// se selectează vectorul
          iq.WhereClause = numecp + "='" + t + "'";
          IFeatureCursor ifc =
           ((IFeatureLayer)lyr).FeatureClass.Search(iq, true);
          vector = ifc.NextFeature();
          IGeometry geo_vec = vector.ShapeCopy;
          fs.Geometry = geo_vec;
          fs.GeometryField                                    =
     ((IFeatureClass)vector.Class).ShapeFieldName;
          fs.SpatialRel                                       =
     esriSpatialRelEnum.esriSpatialRelTouches;
// se determina pentru vectorul selectat toti vectorii care sunt
     in
// proximitatea lui
          IFeatureCursor vec_prox =
                 ((IFeatureClass)vector.Class).Search(fs, false);
          IFeature vec_crt = vec_prox.NextFeature();
// se seteaza corespunzator elementele matricei
          while (vec_crt != null)
          {
           get_indici(t,(string)vec_crt.get_Value(nrc_et), ref i,
     ref j);
            mat[i, j] = 1;
          vec_crt = vec_prox.NextFeature();
           }
        }
    }
```

```
//supraincarcarea operatorului [] pentru a referi elementele
     matricei prin
//etichete
    public byte this[string sl, string sc]
        {
            get
            {
                 int i = 0, j = 0;
  get_indici(sl, sc, ref i, ref j);
                 return mat[i, j];
            }
        }
//supraincarcarea operatorului [] pentru a referi elementele
     matricei prin
//indexul liniei şi al coloanei
    public byte this[int nl, int nc]
        {
         get
     {
             if (nl < 0 || nl >= n || nc < 0 || nc >= n)
                throw   new   ArgumentOutOfRangeException("Indice
     eronat!!");
             return mat[nl, nc];
            }
        }
//metoda privata care converteste etichetele in indecsi
   void get_indici(string sl, string sc, ref int nl, ref int nc)
        {
            int i, vf;
            for (i = 0, vf = 0; i < n; i++)
            {
                if (et[i] == sl) { nl = i; vf++; }
                if (et[i] == sc) { nc = i; vf++; }
```

```
                if (vf == 2) break;
            }
            if (vf != 2) throw new Mat_Ex("Eticheta
 inexistenta!!!");
        }
    }
```

Pentru a utiliza matricea în sensul accesării elementelor sale s-a supraîncărcat operatorul [] în două forme: una primeşte ca parametrii doi întregi pentru a da posibilitatea accesării elementelor prin indecşii lor iar cealaltă formă primeşte două şiruri de caractere pentru a se putea accesa elementele prin etichetele sale. De exemplu, dacă se consideră stratul tematic *judete* care are câmpul *sj* în care se stochează numele fiecărui județ (sub forma acronimului său), atunci o modalitate de a accesa elementele matricei de adiacență este:

```
Matr_adiac mta = new Matr_adiac(ArcMap.Document, "judete", "sj");
MessageBox.Show(mta["bv", "cv"].ToString());
```

Secvența de cod va afişa 1 pentru ca județele *Braşov (bv)* şi *Covasna (cv)* sunt adiacente. Pentru a arunca excepții specifice determinate de generarea matricei de adiacență a fost definită clasa *Mat_Ex* derivată din clasa *Exception*:

```
public class Mat_Ex : Exception
    {
        public Mat_Ex(string serr) : base(serr) { }
    }
```

3. Soluții pentru colorarea hărților cu un număr minim de culori

Problema a fost pentru prima dată pusă de către Francis Guthrie în 1852, încercând să coloreze comitatele Angliei şi constă în a determina numărul minim de culori necesare pentru a colora harta astfel încât două comitate care au frontieră comună să aibă culori diferite. Farancis Guthrie a considerat că numărul minim de culori este 4, lansând astfel o controversă rezolvată parțial abia în 1976 când s-a acceptat o demonstrație

computerizată a acestei afirmații. De fapt aceasta este o problemă particulară pentru problema colorării grafurilor. În 1890 Percy John Heawood formulează problema colorării grafurilor cu un număr minim de culori. În (Heawood, 1890), folosind un algoritm euristic şi formula lui Euler, el ajunge la concluzia că numărul minim de culori este 5. Problema este demonstrată de Ringel şi Youngs în 1968 (Ringel, Youngs, 1968). Problema colorării hărților în 4 culori este în cele din urmă demonstrată de către Haken şi Appel în 1976 (Appel, Haken, 1977). Folosind proceduri matematice pe baza proprietăților de reductibilitate, Appel şi Haken au identificat din infinitatea de posibile hărti un număr de 1476 configurații reductibile. Verificarea tuturor configurațiilor a necesitat încă mult timp, munca fiind finalizată abia în 1989 (Appel, Haken, 1989).

Problema colorării hărților a fost implementată în cadrul clasei *Matrice*, derivată din clasa *Matrice_adiac* deoarece soluția se bazează pe prelucrarea matricei de adiacență. Colorarea propriu-zisă s-a realizat prin doi algoritmi de colorare, unul bazat pe metoda *Backtracking* care utilizează un număr minim de culori şi unul euristic, bazat pe tehnica *Greedy* care are avantajul unei complexități liniare dar care poate să nu utilizeze numărul minim de culori.

Paşii algoritmului *Greedy* sunt următorii:

1. Este selectat un vârf necolorat;
2. Este selectată o nouă culoare pentru colorare;
3. Este colorat vârful selectat şi toate celelalte vârfuri care pot fi colorate cu aceeaşi culoare;
4. Daca mai sunt vârfuri necolorate se revine la pasul 1.

În figura 1 este prezentată o transcriere a algoritmului în pseudocod.

Procedura *Colorare1* primeşte ca parametri: matricea de adiacență (mat) şi numărul de entități (*n*) şi furnizează un vector, *c*, care reprezintă culorile asociate entităților şi un număr natural, *index*, reprezentând numărul de culori folosite.

Procedura *Select* caută şi întoarce indexul unei entități necolorate. Dacă nu există o astfel de entitate întoarce -1 şi algoritmul se termină. Prin procedura *Eliminare* culoarea aleasă, *index*, este folosită pentru a colora cât mai multe entități rămase necolorate. Algoritmul are o complexitate O(n2).

```
Procedure Colorare1(mat,n;c,index)
index=0
do
        index=index+1
        call Select(mat,n,c;k)
        if k>=0 then
                c(k)=index
                call Eliminare(mat,n,c,k,index)
        endif
while k!=-1
return
end
```

Figura 1. Algoritmul Greedy de colorare a hărților

Algoritmul *Backtracking* este prezentat în figura 2.

```
Procedure Colorare2(mat,n;c)
k=0
while k>=0 // bucla de cautare a solutiilor
        select = 0
        while c(k)<NMin && select=0 //bucla de selectie
                c(k) = c(k) +1 // selectie culoare
                call Validare(mat,c,n,k;select) // validarea selectiei
        endwile
        if select=1 then // testul de selectie valida
                if k = n-1 then return // testul de solutie valida
                else // avansul la urmatoarea entitate
                        k = k+1
                        c(k)=0
                endif
        else // revenirea la entitatea anterioara
                k = k+1
        endif
endwhile
return
end
```

Figura 2. Algoritmul Backtracking de colorare a hărților

Semnificația variabilelor este aceeaşi. Selecția este limitată la numărul minim de culori în care se poate colora o hartă, NMin = 4 (Appel, Haken, 1968). În procedura *Validare* este impus testul de vecinătate. Algoritmul se opreşte după găsirea primei soluții (*return* după soluție validă), dar poate fi lăsat şi să genereze toate soluțiile. În contextul colorării hărților în *ArcGIS* generarea tuturor soluțiilor nu are sens. Alegerea rampei de culori pe baza vectorului index, *c*, este identică în toate situațiile. Ca orice algoritm *Backtracking* şi algoritmul de colorare utilizat este unul de complexitate exponențială. Ținând cont însă de particularitățile utilizării lui, limitându-ne

la numărul minim de culori şi la o singură soluție, algoritmul permite colorarea hărților în timp real pentru majoritatea situațiilor practice (hărți posibile).

```
public class Matrice : Matr_adiac
 {
   const int NMax = 4; // nr. maxim de culori in varianta
     Backtracking
   RgbColorClass[] colors = null; // culorile utilizate
   int[] c; // vectorul de culori
   int colN = 0; // numarul de culori utilizate
   public const int BKSOLUTION = 1, GREEDYSOLUTION = 2;

   public Matrice(IMxDocument fhd, string ncsp, string fet) :
                        base(fhd, ncsp, fet)  {   }
// determinarea numarului maxim de culori folosite
   void MaxColor()
        {
            colN = -1;
            for (int i = 0; i < n; i++)
if (c[i] > colN)
colN = c[i];
        }
//  metoda care genereaza vectorul de culori ce sunt asociate
     fiecarui
// element al stratului tematic – primeste metoda de colorare
   public int[] GetSolution(int method)
        {
            if (method == BKSOLUTION)
            {
               BackSolution();
 MaxColor();
            }
            else  GreedySolution();
```

```
            colors = new RgbColorClass[colN];
            return c;
        }
   void GreedySolution()
        {
// a fost implementat algoritmul descris în figura 1
        }
   void BackSolution()
        {
// a fost implementat algoritmul descris în figura 2
        }
//metoda de validare a selectiei
   bool Validation(int[] c, int k)
        {
            for (int i = 0; i < k; i++)
            {
                if (this.mat[i, k] == 1 && c[i] == c[k]) return
     false;
            }
            return true;
        }
//metoda de atribuire a culorii in reprezentare RGB
   public void SetColor(int k, RgbColorClass color)
        {
            if (k < 0 || k >= colN)
throw new Exception("Invalid color index!");
            colors[k] = color;
        }
//metoda de atribuire a culorii prin furnizarea componentelor RGB
//in mod individual
   public void SetColor(int k, int R, int G, int B)
        {
```

```
            if (k < 0 || k >= colN)
throw new Exception("Invalid color index!");
            colors[k] = new RgbColorClass();
            colors[k].Red=R;
            colors[k].Green=G;
            colors[k].Blue=B;
        }
//  obtine  numarul  de  culori  distincte  folosite  la  colorarea
     hartii
   public int GetColorNumber() { return colN; }
//metoda  care  coloreaza  efectiv  harta  pe  baza  vectorului  de
     culori
//asociat
   public void Colorare()
        {
            int z = GetColorNumber(), i;
            IGeoFeatureLayer igfl = (IGeoFeatureLayer)lyr;
            IUniqueValueRenderer uvr = new UniqueValueRenderer();
            uvr.FieldCount = 1;
            uvr.set_Field(0, numecp);
            for (i = 0; i < n; i++)
            {
                ISimpleFillSymbol fs = new SimpleFillSymbol();
                fs.Color = colors[c[i]-1];
                uvr.AddValue(et[i], numecp, (ISymbol)fs);
            }
            igfl.Renderer = (IFeatureRenderer)uvr;
            hd.ActiveView.Refresh();
            hd.UpdateContents();
       }
  }
```

Colorarea propriu-zisă a hărții se realizează prin metoda *Colorare*() care utilizează vectorul de culori asociat elementelor spațiale de tip poligon (*c*) şi

culorile efective stocate în vectorul *colors*. Pentru vizualizare, harta a fost trasată folosind o simbologie de tipul umplere uniformă a poligoanelor cu culoare (Dârdală, Reveiu, 2011), fiecare poligon fiind colorat de sine stătător şi identificat prin câmpul de etichetare indicat în constructorul clasei de bază.

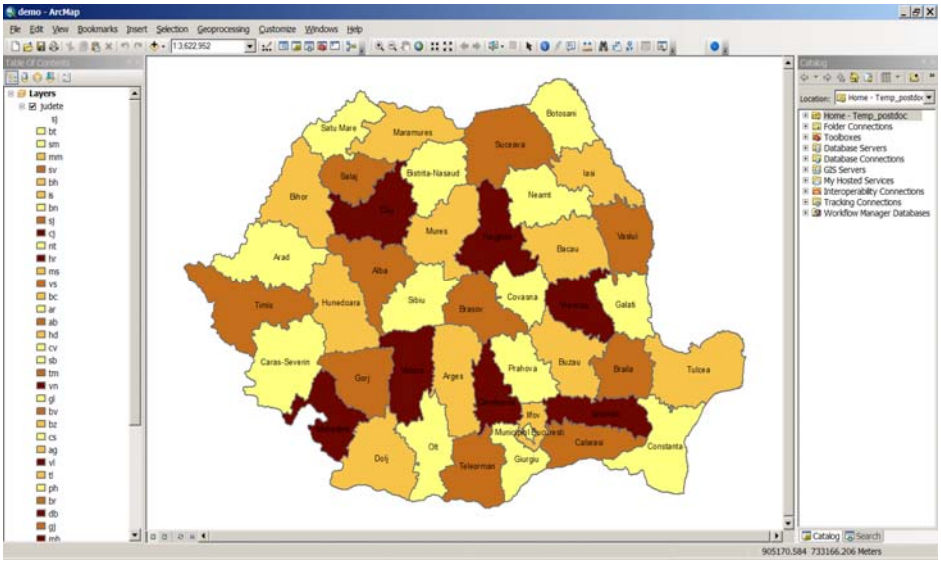

Figura 3. Colorarea hărții prin metoda Backtracking

Pentru exemplificare s-a construit o extensie de tip *add-in* la aplicația *ArcMap* care realizează colorarea hărții cu județele din România ținând cont de matricea de adiacență a acestora. Se poate observa, în figura 3, că prin metoda de colorare *Backtracking* s-au utilizat patru culori distincte în timp ce prin metoda *Greedy* au fost utilizate cinci culori distincte (figura 4).

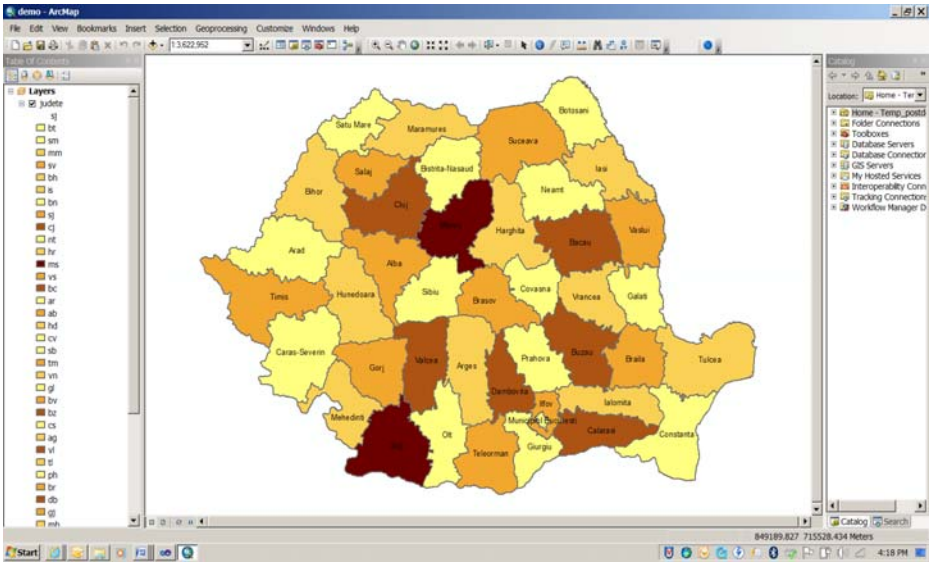

Figura 4. Colorarea hărții prin metoda Greedy

4. Determinarea coeficientul de autocorelație spațială Moran

Autocorelația spațială există pentru că fenomenele din lumea reală se desfăşoară în spațiu, fiind mai degrabă ordonate, reprezentând concentrări sistematice decât cu o distribuție aleatorie. Conform lui Tobler (1970) „în sistemele geografice totul este în legătură cu orice altceva, dar lucrurile aflate în apropiere sunt mai legate între ele decât lucrurile îndepărtate." Autocorelația spațială identifică dependența dintre valorile unei variabile, aşa cum sunt identificate în locații învecinate. De asemenea permite identificarea existenței unui model de distribuție a valorilor unei variabile în locațiile din proximitate, în funcție de distribuția acestora pe hartă, datorită existenței unor factori comuni (Reveiu, Dârdală, 2011).

Coeficientul de autocorelație spațială Moran (Moran, 1950) se calculează pe baza coeficientului localizării, determinat pentru fiecare domeniu de activitate dorit şi pentru fiecare regiune sau județ.

Conform cu Porter (1998), coeficientul localizării se calculează cu următoarea formulă:

$$\text{Coeficientul localizării} = \frac{\frac{\text{Număr angajați în industria A, dintr-o regiune}}{\text{Număr total angajați ai regiunii}}}{\frac{\text{Număr angajați în industria A, la nivel național}}{\text{Număr total angajați la nivel național}}}$$

Pentru a permite comparații între diverse regiuni sau județe, coeficientul Moran este normalizat. Din punct de vedere statistic se compară valoarea unei variabile continue, în fiecare locație, cu valoarea aceleiași variabile pentru locațiile din proximitate. Cu alte cuvinte, este necesar a se determina vecinătățile spațiale pentru fiecare element spațial de pe hartă (regiune, județ etc.).

Un coeficient de corelație spațială nu este o bună măsură pentru determinarea concentrării spațiale. El este proiectat să identifice *pattern*-uri spațiale privind distribuția variabilei analizate. Formula de determinare a coeficientului Moran (*I*) este:

$$I = \left(\frac{1}{s_y^2}\right)\frac{\sum_i^N \sum_{(j:i\neq j)}^N w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_i^N \sum_{(j:i\neq j)}^N w_{ij}}$$

Unde:

$$\bar{y} = \sum_i {}^{y_i}/_N \ ,$$

$$s_y^2 = \frac{1}{N}\sum_{i=1}^N (y_i - \bar{y})^2 \text{ și}$$

$$w_{ij} = \begin{cases} 1, \textit{daca } i,j \textit{ sunt vecini} \\ 0, \textit{ altfel} \end{cases}$$

N – reprezintă numărul total de vectori ai caracteristicii spațiale.

Valoarea lui *I* se poate situa în intervalul [-1, 1], valoarea de referință fiind calculată astfel:

$$E(I) = -{}^1/_{N-1}.$$

Valori pozitive ale lui *I* indică un puternic *pattern* geografic pentru clusterul spațial, valorile negative sunt asociate cu un patern normal şi valorile apropiate de 0 indică schimbări spațiale.

Din formula prezentată se poate observa că un rol important în calculul coeficientului (*I*) îl are matricea de adiacență, care a fost generată pornind de la datele spațiale analizate.

Metoda a fost aplicată pe un set de date statistice referitoare la distribuția teritorială şi pe domenii de activitate a forței de muncă din România, în anul 2011, pentru toate județele din România şi pentru principalele domenii de activitate aşa cum sunt ele prezentate în Anuarul statistic al României, din 2012.

In tabelul 1 se prezintă valoarea coeficientului lui Moran calculat pentru mai multe activități CAEN. Valoarea de referință calculată este E(I) = 0,0244.

Tabelul 1. Valoarea coeficientului Moran pentru principalele domenii de activitate, din România

CAEN	Coeficientul Moran	Tipul autocorelației spațiale
Industrie prelucrătoare	0,343	Puternică
Industrie	0,341	Puternică
Producția şi furnizarea de energie electrică şi termică, gaze, apă caldă şi aer condiționat	0,337	Puternică
Învățământ	0,307	Puternică
Administrație publică şi apărare; asigurări sociale din sistemul public	0,233	Puternică
Activități profesionale, ştiințifice şi tehnice	0,186	Puternică
Agricultură, silvicultură şi pescuit	0,172	Puternică
Alte activități de servicii	0,169	Puternică
Distribuția apei; salubritate, gestionarea deşeurilor, activități de decontaminare	0,168	Puternică
Hoteluri şi restaurante	0,137	Puternică
Transport şi depozitare	0,131	Puternică

Sănătate şi asistență socială	0,113	Puternică
Comerț cu ridicata şi cu amănuntul; repararea autovehiculelor şi motocicletelor	0,089	Puternică
Activități de servicii administrative şi activități de servicii suport	0,072	Puternică
Tranzacții imobiliare	0,010	Modificare spațială
Intermedieri financiare şi asigurări	0,002	Modificare spațială
Informații şi comunicații	0,001	Modificare spațială
Activități de spectacole, culturale şi recreative	-0,011	Model obişnuit
Industrie extractivă	-0,027	Model obişnuit
Construcții	-0,147	Model obişnuit

Domeniile de activitate analizate pot fi considerate ca având o puternică autocorelație spațială, dacă valoarea coeficientului lui Moran calculată pentru activitățile desfăşurate în profil teritorial în acele domenii sunt mai mari decât valoarea de referință calculată E(I). Pentru aceste domenii componenta spațială poate fi considerată importantă în analiza distribuției spațiale a activității economice. Pentru domeniile în care valoarea coeficientului Moran este pozitivă, dar sub valoarea de referință şi pentru valori negative ale coeficientului lui Moran, nu se poate stabili existența unei autocorelări spațiale.

5. Concluzii

In cadrul articolului s-a prezentat o variantă de implementare pentru reprezentarea vecinătăților datelor unui strat tematic de tip poligon printr-o matrice de adiacență, precum şi două aplicații ale acesteia. Totodată, s-a pus accentul şi pe partea de proiectare a soluției software prin construirea de

clase, care prin specializare formează o ierarhie în vederea soluționării de probleme particulare. După cum s-a putut observa, matricea de adiacență este un element cheie pentru diverse probleme particulare, în articol a fost prezentat un mod de colorare a hărților dar şi o aplicabilitate economică prin calculul coeficientului de corelație spațială Moran.

Referințe

Appel, K, Haken, W. (1977) *Solution of the Four Color Map Problem*, Scientific American 237 (4);

Appel, K, Haken, W. (1989) *Every Planar Map is Four-Colorable*, Providence, RI: American Mathematical Society;

Chang K.-T. (2008) *Programming ArcObjects with VBA*, CRC Press;

Dârdală, M., Reveiu, A. (2011) *Soluție software de reprezentare a datelor statistice în GIS, prin combinarea simbologiilor*, Revista Română de Interacțiune Om-Calculator, vol 4, nr 1, Editura MATRIX ROM, Bucureşti;

Heawood, P. J. (1890) *Map colour theorem*, Quarterly Journal of Mathematics 24;

Moran, P. A. P. (1950) *Notes on Continuous Stochastic Phenomena*,. Biometrika 37 (1): 17–23;

Porter M. (1998) *The Competitive Advantage of the Inner City, In: On Competition*, Boston, Harvard Business School Press;

Reveiu, A., Dârdală, M. (2011) *Spatial Distribution of Economic Activities in Romania*, în volumul celei de-a şasea ediție a conferinței internaționale "Business Excellence", Universitatea Transilvania, Braşov, Octombrie 2011, publicat la editura Universității Transilvania Braşov;

Ringel, G., Youngs, J. W. T. (1968) *Solution of the Heawood map-coloring problem*, Proceedings of the National Academy of Sciences of the United States of America 60;

Tobler W. (1970) *A computer movie simulating urban growth in the Detroit region*. Economic Geography, 46(2): 234-240