EFFICIENT SPATIAL TIME-COST ANALYSIS FOR SEARCH OF LOST TOURISTS

Zsolt Magyari-Sáska[1]

Abstract
Least cost analysis for anisotropic processes rise the problem of time efficiency. That's truer in case of emergency cases as is the search and rescue for lost persons, when several minutes can be decisive. We tried to review some of the possibilities of least cost analysis. Beside fast calculi we also want to make the application in a free environment and that's why be started in R. Exploring the graph based and direct application of Dijkstra algorithm we encountered mutually exclusive problems: the fast algorithm was memory hungry, that which required reduced memory was very slow. To overcome this situation we implemented in C++ the algorithm in an environment where the two needs, speed and memory were in equilibrium.

Key Words: *R, least cost, Dijkstra, efficiency, hiking time.*

1. INTRODUCTION

Time cost analysis can be an important part of several spatial analyses. Determining the best possible location or route can help in decision making, avoiding or minimizing loss of human life or material values.

Various studies have been realized using least cost analysis. Two major assumptions are used in such studies: anisotropic or isotropic modeling. *Woood and Schemidtlein (2012)* use the anisotropic least cost analysis for determining the optimal paths for pedestrian evacuation in case of tsunami. In that research the travel time and percentage of population arriving to safe zones are calculated using both isotropic and anisotropic modeling with ArcGIS's Path Distance module. To humans safety is related *Saha et al. (2005)* study regarding the route planning in landslide-prone areas.

The anisotropic analysis is realized using a C++ program for least cost calculus. *Theobald et al. (2010)* used least cost modeling for estimating visitor's number in protected areas. In this case just isotropic modeling was used using ArcGIS's Cost Distance module. Remoteness related to accessibility is modeled in *Fritz and Carver*'s research (2000) where the effective (anisotropic) calculus is realized by an external C program based on Dijkstra algorithm. This research was lectured by *Watts et al. (2003)* whose mentioned the high computing time of Dijkstra and proposed an alternative isotropic method considering that modeling errors are not greater that walking speed estimation errors.

In all above mentioned research takes into account the land cover, estimating its influence onto travel speed. *Gitle et al. (2008)* paper make a comparison between different software's possibilities in least cost analysis, enumerating IDRISI, ArcGIS and GRASS.

Considering the rapid and efficient development and use of the free R system the aim of our research is to develop an efficient practical methodology for determining the possible locations after a given time of a lost tourist based on the digital elevation model of the region. In our conception the search area can be relatively large, 5-10 million cells.

[1] Babeş-Bolyai University, Faculty of Geography, Gheorgheni, Romania.

2. RESEARCH METHODOLOGY AND DATA

2.1. The R environment

For our research we have chose the R system based on two combined reasons. The R system offers great capabilities both in statistical and spatial analysis with no cost. The impressive number of freely available well constructed packaged makes the R to be an efficient analysis tool, also due to the fact that offers a C based programming language to construct own research algorithms. The basic puritan interface of R was face lifted with R Studio, a powerful IDE.

Recent developments (2012) also makes possible to construct R server which makes possible the Web integration of a realized a research, giving the user the possibility to repeat it with his own data. Another new development (2013) is the Shiny package which helps in the design of user friendly interfaces, such as the user can conduct an analysis without typing commands and functions.

We consider that R should be taken account in any numerical analysis as its importance is growing in many research fields. Being a collaborative environment (*Magyari-Sáska, 2007; Magyari-Sáska, Haidu, 2006*) it has an own programming language, has great speed in handling vector and matrix like data structures, but also is capable to integrate different data types (using proper packages). It can integrate inline C commands, but also launch external modules realized in other programming languages.

2.2. Hiking time calculus

The cost in our research is expressed in time, as we want to determine possible locations where a lost tourist could reach in after a given time from its lost.

Our previous research (*Magyari, Dombay, 2012*) we have shown that there's a strong correlation between Tobbler's and Naismith's time calculus methods we have chose to work with Tobbler's rule.

The Tobler rule is in fact a formulae (equation 1) determines the hiking speed based on exponential function which takes account the slope. The dissymmetry effect of slope direction in walking speed is marked with a shift of 0.05 in the exponent (*Tobler, 1993*).

$$V = 6 \cdot e^{-3.5 \cdot \left| \frac{dh}{dx} + 0.05 \right|} \tag{1}$$

where:

- V – walking velocity (km/h)
- dh – slope high
- dx – slope base

We implemented both isotropic and anisotropic calculus. For isotropic variant we considered the minimum traveling time through a cell. We selected the half of the minimum inbound and minimum outbound travel time for every cell from all 8 adjacent cells. For anisotropic variant we implemented Dijkstra algorithm on the raster layer. In both cases we used the Queen adjacency.

The data we used was derived from SRTM 1arcdegree DEM, representing Giurgeu basin, which proved to be sufficient for our analysis (*Drachal, 2010*). Different size test areas were used to measure the efficiency of different implementations.

3. IMPLEMENTATION AND DISCUSSION

3.1. Isotropic variant

Even if travel time is not isotropic in reality some of the researches mentioned in introduction uses this variant, mainly due to the fact that is much simple and faster to calculate than isotropic one.

Our two variant of isotropic calculus show the power of R in statistical and also in spatial analysis. The first variant (**Fig. 1**) uses a classical programming method with two loops reaching all cells of the layer, and other two inner loops for calculating travel time for all adjacent cell pairs.

The data file handling is based on a matrix and for correct interpretation of file data structure we have to know its internal structure. In our case an IDRIS raster file was used with real data type.

```
      Tobbler_TimeCost1<-function(name_in,name_out,row,column,res)
      {
       p=c(1,1.4142,1,1.4142,1,1.4142,1,1.4142)
       x=c(0,1,1,1,0,-1,-1,-1)
       y=c(-1,-1,0,1,1,1,0,-1)
       total=c(rep(0,56))

       c=file(name_in,"rb")
       b=readBin(c,double(),row*column*4,size=4)
       m=matrix(b,row,column,byrow=TRUE)
       t=matrix(rep(0,row*column),row,column)

       for (i in 2:(row-1)) {
         for (j in 2:(column-1)) {
          total=c(rep(0,56))
          n=0
          for (k in 1:8) {
            for (l in 1:8) {
             if (k!=l) {
               v1=6*exp(-3.5*abs((m[i+y[k],j+x[k]]-
m[i,j])/(31.1*p[k])+0.05))
               l1=res*p[k]/1000/v1
               v2=6*exp(-3.5*abs((m[i,j]-
m[i+y[l],j+x[l]])/(31.1*p[l])+0.05))
               l2=res*p[l]/1000/v2
               n=n+1
               total[n]=(l1+l2)/2 }}}
          t[i,j]=min(total) }}
       v=unmatrix(t,byrow=TRUE)
       co=file(name_out,"wb")
       writeBin(v,co,size=4)
       close(co)
      }
```

Fig. 1. Tobbler Time-cost function, first variant.

The second variant (**Fig. 2**) takes advantage of R calculus power using a user defined sliding window function. It is an important difference between filter commands in different GIS environments and this sliding windows technique. While the filter command practically just defines weight for calculi in this case the programmer can implement different methods, like minimum calculus of individual values.

The other advantage is the use of raster package which combine with rgdal package makes possible the use of the majority of common GIS raster formats with projection information. In this case the row and column number and the raster resolution can be extracted easily. The saving of the resulted layer in a desired format is also easily performed.

```
Tobbler_TimeCost2<-function(name_in,name_out)
{
 p=c(1.4142,1,1.4142,1,1,1.4142,1,1.4142)
 mycalculus=function(x)
 {
  y=x[-5]
  t1=r*p/1000/(6*exp(-3.5*abs((y-x[5])/(r*p)+0.05)))
  t2=r*p/1000/(6*exp(-3.5*abs(((rep(x[5],8)-y)/(r*p)+0.05))))
  return((min(t1)+min(t2))/2)
 }
 f=raster(name_in)
 r=xres(f)
 m=focal(f,w=3,fun=mycalculus)
 writeRaster(m,name_out,"IDRISI",overwrite=TRUE)
}
```

Fig. 2. Tobbler Time-cost function, second variant.

Beside the code simplicity the main advantage of this second variant resides in calculi time. Testing the two functions on the same computer configuration we obtained the results show in **Table no. 1** and **Fig. 3**.

Table no. 1. Calculi times for Tobler time-cost function

Raster size [cells]	With loops [sec]	Without loops [sec]
100x100	14.47	0.37
200x200	58.14	1.08
300x300	132.35	2.28
400x400	235.42	4.03
500x500	369.27	6.21
600x600	529.75	9.04
700x700	720.19	12.23
800x800	943.81	15.76

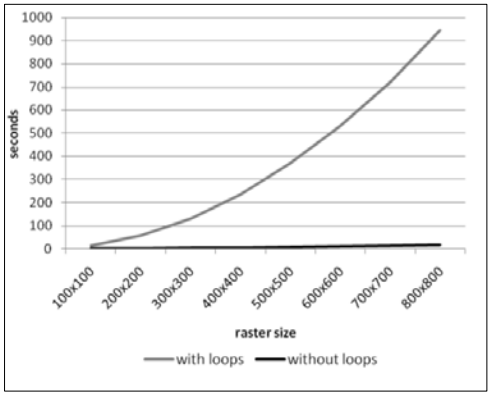

Fig. 3. Comparison of calculi times for Tobler time-cost functions.

As the result of such analysis we got the time-cost surface of the terrain, which needs additional least cost analysis to get the final result. As our goal was to try a more realistic modeling we discarded further analysis for isotropic modeling and continued with anisotropic variant.

3.2. Isotropic variant

The isotropic analysis is based on bidirectional weighted graphs. In this case every cell of the raster layer becomes a graph node, while between the adjacent cells a connection will be formed having the travel time as cost. In the scientific literature there are three well known adjacency versions: the Rook, the Queen and the Knight connections (*Pingel, 2009*). In our study we use the Queen connection, in which case a center cell has 8 adjacency cells (**Fig. 4**).

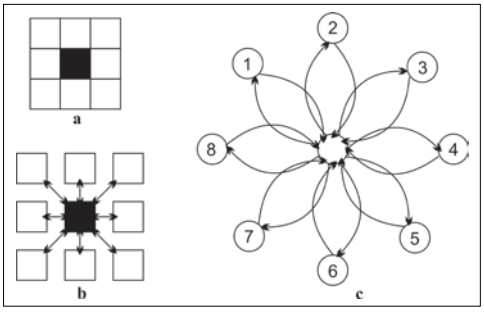

Fig. 4. Fig. 4 Queen adjacency connection
(a) its decomposition (b) and graph based representation (c)

We used the igraph package for graph operations. There exists other packages for graph based analysis such as gdistance, spatgraphs, but these operate with unidirectional graphs. The graph construction function is presented in **Fig. 5**.

```
      Construct1<-function(name_in)
      {
       p=c(1.4142,1,1.4142,1,1,1.4142,1,1.4142)
       x=c(-1,0,1,-1,1,-1,0,1)
       y=c(-1,-1,-1,0,0,1,1,1)
       f=raster(name_in)
       r=xres(f)
       nr=nrow(f)
       nc=ncol(f)
       gr=graph.empty(nr*nc)
       m=matrix(data=f[],nrow=nr,ncol=nc,byrow=TRUE)
       for (i in 2:(nr-1)) {
        for (j in 2:(nc-1)) {
           for (k in 1:8) {
              t=r*p[k]/1000/(6*exp(-3.5*abs((m[i,j]-
m[i+y[k],j+x[k]])/(r*p[k])+0.05)))
              gr=add.edges(gr,c((i-1)*nc+j,(i-
1+y[k])*nc+(j+x[k])),weight=t) }}}
       return (gr)
      }
```

Fig. 5. Constructing a bi-direction weighted graph form raster, first variant.

The main problem we encountered was the slowness of the function. Trying to optimize the process we used the power of R in matrix and vector calculus. Instead of accessing each cell of the layer, with the loops as in figure 6, and with the third getting all neighbors, we duplicated the layer we have to analyze and shifted one of the matrix in such a way that the neighbor cells got the same layer position as the analyzed cell. In this way the calculus between adjacent cells was transformed in calculus between corresponding cells in two layers. As R has the power of operating very efficient between same sized even complex data structures, the calculi became much faster. The only additional problem that had to be resolved was to eliminate the marginal cells, as they don't get values in this manner.

Making the same comparison as for the isotropic variant we observed a great performance in calculus time (**Table no. 2**), but also a great inconvenient appear: the hungry for memory space.

Over 64000 cells in most cases we got the Not enough memory error message from R, and we never could not pass over 100000 cells. The main problem was not the duplication of the initial raster layer but the impressive number of graph connection. In case of 800x800 cell which means 64000x64000 possible adjacencies modeling in the graph. Even if R uses efficient memory management not storing all 0 values between non adjacent cells from the raster, the continuous growing of the graph structure by adding new connections, the whole structure exceeds the limit of continuously free memory part form where it has been started.

```
      Construct2<-function(name_in)
      {
       f=raster(name_in)
       r=xres(f)
       nr=nrow(f)
       nc=ncol(f)
       gr=graph.empty(nr*nc)
       p=c(1.4142,1,1.4142,1,1,1.14142,1,1.4142)

pm=matrix(data=seq(1:(nr*nc)),nrow=nr,ncol=nc,byrow=TRUE)
       m=matrix(data=f[],nrow=nr,ncol=nc,byrow=TRUE)
       for (i in 1:8){
        if (i==1) { pm1=shift.right(shift.down(pm))
                    m1=shift.right(shift.down(m)) }
      ..................................................................
        if (i==8) { pm1=shift.left(shift.up(pm))
                    m1=shift.left(shift.up(m)) }
        t=r*p[i]/1000/(6*exp(-3.5*abs((m-m1)/(r*p[i])+0.05)))
        pmr=pm[-nr,]    pm1r=pm1[-nr,]      t=t[-nr,]
        pmr=pmr[,-nc]   pm1r=pm1r[,-nc]     t=t[,-nc]
        pmr=pmr[-1,]    pm1r=pm1r[-1,]      t=t[-1,]
        pmr=pmr[,-1]    pm1r=pm1r[,-1]      t=t[,-1]
        pmr=c(pmr)      pm1r=c(pm1r)        t=c(t)
        df=data.frame(pmr,pm1r,t)
        names(df)=NULL
        dm=data.matrix(df)
        gr=add.edges(gr,c(t(dm[,1:2])),weight=dm[,3])
       }
      return (gr)
      }
```

Fig. 6. Constructing a bi-direction weighted graph form raster, second variant.

Table no. 2. Calculi times for bi-directional weighted graphs.

Raster size [cells]	Construct1 [sec]	Construct2 [sec]
100x100	0.35	0.25
200x200	1.39	0.26
300x300	5.98	0.26
400x400	19.42	0.26
500x500	53.01	0.27
600x600	131.68	0.28
700x700	325.09	0.28
800x800	595.09	0.28

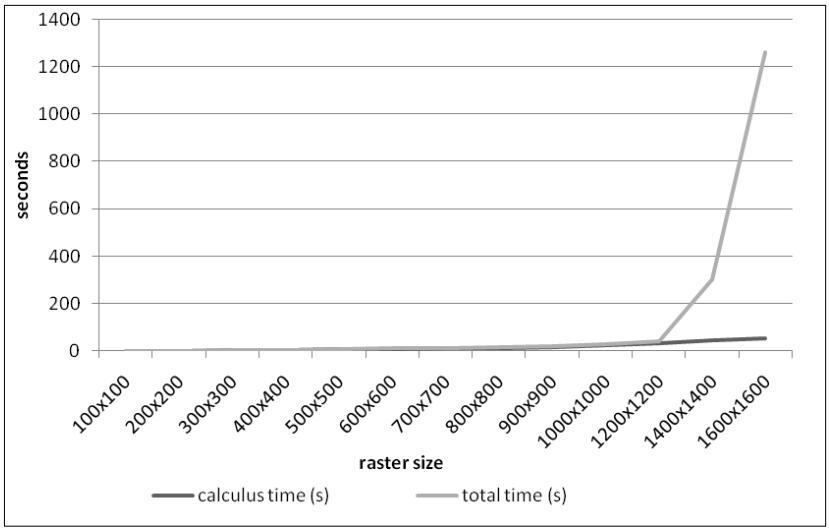

Fig. 7. Comparison of effective calculi time and total user time for Construct2 in a 64 bit environment.

Trying to resolve this problem we studied the bigmemory and ff packages, but in finally we decided to give a try in a 64 bit environment. In this case the problem of memory reallocation was resolved automatically but the swapping of memory content resulted in huge additional time consumption, making this option unreliable (**Fig. 7**).

For getting the final result, a layer that represent the minimum time for reach every cell for a given initial position an addition command has to be performed, which time consumption was 3 seconds for an 10000 cell raster. This command uses the Dijkstra algorithm in a constructed graph for getting the least cost cell values. The main problem in this case was not the efficiency of effective calculus time but the fact that over a given raster size the memory handling was poor.

The next step was to try to implement Dijkstra algorithm directly on raster without transforming it into a graph. We tried two variants the native and an optimized one, but even for a 90000 cells we've got over 5 minutes for calculus. Maintaining the idea to use free solution we used Code Blocks to implement Dijkstra algorithm in C++, knowing that R offers the possibility to call external programs, and enforcing the idea that GIS is an interdisciplinary science (*Iosifecu, Hurni, 2010*).

The main idea of Dijkstra shortest path algorithm is:

- all cells has an infinite cost value, excepting the start cell which has 0
- all cell are elements of a list
- while exists cells in the list
  - extract the lowest cost from it
  - consider all their neighbors
    - calculate the total cost to the neighbor
    - if the total cost to it is lower than the actual update the cost

To get an efficient calculus we use the implemented STD priority queue. Using it our calculi time does not reach 10 seconds for 1000x1000 cells and maintain its performance even for higher cell numbers, but due to the fact that priority queue doesn't offer the possibility to address the queue elements by their position the minimal distance update could not be realized correctly. Some internet sources suggested using heap vectors to construct a priority queue but in this case the need to rearrange the elements after every cost update slows down the whole process. That's why we return to the existing priority queue and inserting every new lowest cost for a cell, but being attentive at the extraction to take into account just elements cost which hasn't been taking account yet.

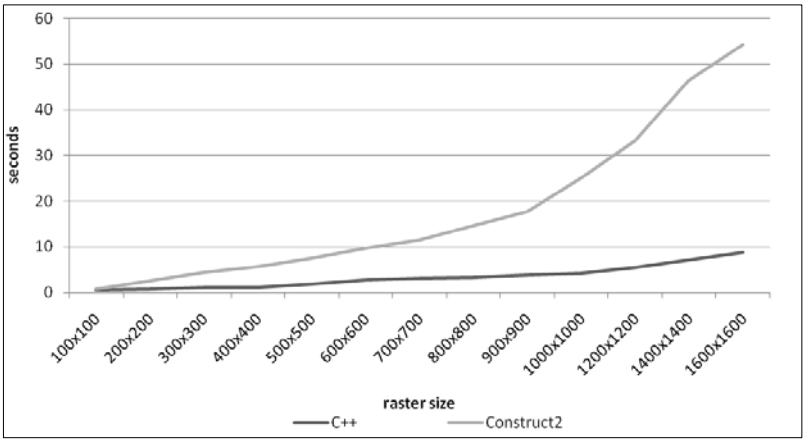

Fig. 8. Comparison of calculi time for C++ and R implementation of Dijkstra algorithm.

Knowing at the beginning the necessary continuous space for data structures and the dynamic allocation of data structures in C++ resolves the memory management problem, and the priority queue resolves the inefficient calculi (Fig. 8). We don't abandoned the R environment, as specific raster layer data management remains in R, but the effective calculi are made by an external application call from R.

4. CONCLUSION

Spatial analysis in general (*Haidu I. & Haidu C., 1998*) and least cost analysis for anisotropic processes in special rise the problem of time efficiency. That's truer in case of emergency cases as is the search and rescue for lost persons, when several minutes can be decisive. We tried to review some of the possibilities of least cost analysis. Beside fast calculi we also want to make the application in a free environment and that's why be started

in R. Exploring the graph based and direct application of Dijkstra algorithm we encountered mutually exclusive problems: the fast algorithm was memory hungry, that which required reduced memory was very slow. To overcome this situation we implemented in C++ the algorithm in an environment where the two needs, speed and memory were in equilibrium.

Nevertheless we don't abandoned the R environment, the realized C++ program is called from it as spatial data handling is very easy and efficient in R and further operations, calibrating hiking time based on land cover and the lost persons characteristics can be easily perform in it.

REFERENCES

Drachal J., (2010). Small scale mapping of the mountains with the use of SRTM and MODIS data, Geographia Technica, Special Issue, 2010

Fritz S., Carver S., (2000). Modelling remoteness in roadless areas using GIS.Parks BO, Clarke KM, Crane MP, editors. 4th International Conference on Integrating GIS and Environmental Modeling (GIS/EM4): Problems, Prospects and Research Needs. September 2–8, 2000, Banff, Alberta, Canada.

Gietl R., Doneus M., Fera M., (2008). Cost distance analysis in an alpine environment: comparison of different cost surface modules, in Posluschny, A., Lambers, K. and Herzog, I. (eds) Layers of perception. Proceedings of the 35th international conference on computer applications and quantitative methods in archaeology (CAA 2007). Berlin, Germany, April 2–6,2007.

Haidu I., Haidu C., (1998). S.I.G. – Analiza Spatiala, Editura *H*G*A* Bucuresti.

Iosifescu I., Hurni L., (2010). GIS platform for interdisciplinary environmental research, Geographia Technica, S.I. 2010.

Magyari-Sáska Zs., Dombay Ş., (2012). Determining minimum hiking time using DEM, Geographia Napocensis, anul VI. Nr. 2, Cluj-Napoca.

Magyari-Sáska Zs., (2007). ArcGIS software module for calculating the S.P.I. value, Geographia Technica, No. 2, 2007, Presa Universitară Clujeană.

Magyari-Sáska Zs., Haidu I., (2006). Posibilităţi de modelare spaţială în mediu programat, Analele ştiinţifice ale Universităţii „Al. I. Cuza" din Iaşi, Geografie, Tomul LII.

Pingel T.J., (2009). Modeling slope as a contributor to route selection in mountainous areas, UCGIS 2009 Summer Assembly.

Saha A.K., Arora M.K., Gupta R.P., Virdi M.L., Csaplovics E., (2005). GIS-based route planning in landslide-prone areas, International Journal of Geographical Information Sciences, vol. 19, No. 10.

Tobler W., (1993). Non-isotropic geographic modeling, Technical Report No. 93-1, Santa Barbara, CA: National Center for Geographic Information and Analysis.

Watt R.D., Compton R.W., McCammon J.H., Ouren D.S., (2003). Intensity of human use, backcountry roads, and analysis of human accessibility, ICOET 2003 Proceedings, making Connections.

Wood N.J., Schmidtlein M.C., (2012). Anisotropic path modeling to assess pedestrian-evacuation potential from Cascadia-related tsunamis in the US Pacific Northwest, Natural Hazards, vol. 62, Springer.

Acknowledgments:
This research and paper has been created within a project funded by the Domus Program of Hungarian Academy of Sciences.