A Numerical Algorithm for the Optimal Placement of Inset Maps

Dimitris Varsamis
Department of Informatics Engineering
Technological Educational Institute of Central Macedonia - Serres
62124, Serres, Greece

Apostolos Papakonstantinou
Department of Geography, University of the Aegean
81100 Mytilene, Greece

Copyright © 2016 Dimitris Varsamis and Apostolos Papakonstantinou. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract
The purpose of this paper is to present an algorithmic formulation addressing the cartographic problem of siting an inset map at specific map locations under spatial and cartographic constraints. The first part of the paper aims at: (a) presenting a numerical algorithm that solves the above siting problem under such constraints, and (b) investigating the effectiveness of this numerical algorithm for a more general geographical problem, that of siting an anthropogenic structure or object of rectangular shape in suitable areas. The second part of this paper showcases the computational implementation of the above algorithm for addressing the cartographic problem of inset map placement in areas with land discontinuity.

Keywords: Numerical algorithm, Inset map placement, Complexity

1 Introduction
An often encountered spatial decision problem is to search for optimal sites to locate urban facilities. This problem is common in both public and private
sectors that form policy in landscape decision making [2]. Site selection is a crucial and complex process, in which difficulties arise due to the existence of numerous candidate sites [3]. The search for optimal location(s) is a classical problem in GIS applications, and a multiple methods exists for siting facilities in the spatial decision making literature [2], [7]. In particular, multi-criteria decision making and expert systems techniques have been routinely used in solving site selection problems, and a number of specialized tools have been developed to ensure optimal decisions in location analysis [2], [3].

Most traditional siting methods using GIS cannot easily handle thousands of candidate sites, which is often the case when raster data with many cells are involved [1]. Scale issues also arise in location science when rich and finely detailed databases are involved; in this case, the level of detail and the number of data may overwhelm many models and algorithms. The finer the scale the greater the demands on process and computation time when siting facilities [3].

The siting and placement process of a major facility, such as factory, or the siting of multiple facilities is similar to the insetting procedure in cartography where location analysis is needed in order to determine the best placement position of an inset map. In this paper, we focus on the cartographic problem of inset placement, and present a modified serial search algorithm for providing solutions to this problem.

It is a common practice for cartographers to visually inspect a map and use their judgement to determine the best placement positions for insets in order to visualize areas with land discontinuities [11]. In order to help cartographers tackle this problem [12, 13] we developed a software tool called Inset Mapper (IM), which provides optimal inset map placement positions following specific cartographic rules using a simple serial search algorithm. IM initially creates a submap with “positive areas” and “negative areas”, as many location and siting analysis algorithms do, in order to denote the existence or not of a favorable condition [2, 8, 10]. “Positive areas” are the positions that an inset can be placed and ”negative areas” are the positions that the inset cannot be placed according to specific cartographic rules.

This paper proposes an enhancement of the IM serial search algorithm, and demonstrates its implementation and computational procedure for tackling the cartographic problem of inset map placement within a map under specific constraints [13]. In particular, in Section 2 of this paper the new improved numerical algorithm is presented, where its improvement lies in the optimization of executed iterations. Furthermore the theoretical complexity of the modified algorithm is calculated in Section 2. Additionally, the empirical complexity of this algorithm is investigated in relation to (a) its theoretical complexity, and (b) the area of connected pixels that constitute regions or islands in a case study with randomly generated ”positive” and ”negative” areas. In Section
3 we present the application of the modified numerical algorithm to the Island Cartography’s Land Discontinuity problem. Furthermore, we present a comparison of the modified numerical algorithm with the old one used in first version of IM using examples with real geographical datasets. Additionally, in this section the implementation of the new modified and improved numerical algorithm into the IM software tool is presented and evaluate its performance against the existing serial search algorithm using a specific case study. Finally, Section 4 presents the conclusions and a brief discussion of future work.

2 Numerical Algorithm

In this section, we present the modified version of the original insetting numerical algorithm adopted by [13], [12] and referred in this paper as Algorithm 1. Additionally, we evaluate the computational complexity of the new modified algorithm, referred in this paper as Algorithm 2.

Let $A$ denote a $(m \times n)$ matrix with elements

$$a_{i,j} = \begin{cases} 
1, & \text{if a favorable condition exists at location } i,j \\
0, & \text{otherwise}
\end{cases}$$

and $B$ denote a $(m_{ins} \times n_{ins})$ sub-matrix of $A$, with $m_{ins}$ and $n_{ins}$ denoting the length and width, respectively, of the proposed facility. In mathematical formulation, the problem is to count $(N)$, i.e., the number of times matrix $B$ exists into matrix $A$ having all elements equal to one, and store the corresponding positions ($positions$). The modified numerical algorithm is described in Algorithm 2.

In Algorithm 2 the number of iterations depends on the size of matrix $A$ and $B$. Additionally, for both internal and external loops the number of iterations depends on the elements of matrix $B$, namely on the existence of the value one. The external double loop is defined by indices $i$ and $j$ and the internal double loop is defined by indices $b_j$ and $b_i$, respectively. In the external double loop the number of iterations for index $i$ is constant and equal to $m - m_{ins} + 1$, while the number of iterations for index $j$ is between 1 and $n - n_{ins} + 1$. Thus, the maximum number of iterations for the external loop is $(n - n_{ins} + 1) \times (m - m_{ins} + 1)$.

In the internal loop the respective number of iterations is $n_{ins} \times m_{ins}$. Therefore, the maximum total number of iterations executed in the serial algorithm, with inputs $A^{m \times n}$, $m_{ins}$ and $n_{ins}$, is given by

$$L_{max}(n, m, n_{ins}, m_{ins}) = (n - n_{ins} + 1) \cdot (m - m_{ins} + 1) \cdot n_{ins} \cdot m_{ins} \quad (2)$$
Algorithm 2 Modified Numerical Algorithm

Input: $A, m_{\text{ins}}, n_{\text{ins}}$
Output: $N, \text{positions}$

for $i = 1$ to $m - m_{\text{ins}} + 1$
  $j \leftarrow 1$
  while $j \leq n - n_{\text{ins}} + 1$
    $B \leftarrow A(i$ to $i + m_{\text{ins}} - 1, j$ to $j + n_{\text{ins}} - 1)$
    $b_j \leftarrow n_{\text{ins}}$
    done $\leftarrow$ true
    while $b_j \geq 1$
      $b_i \leftarrow 1$
      while $b_i \leq m_{\text{ins}}$
        if $b(b_i, b_j) = 1$ then
          done $\leftarrow$ false
          pos $\leftarrow b_j + 1$
          Break
        end if
        $b_i \leftarrow b_i + 1$
      end while
      $b_j \leftarrow b_j - 1$
    end while
  end for
  if done $= true$ then
    Count ($N \leftarrow N + 1$) and Store position ($\text{positions} \leftarrow i, j$)
  else
    $j \leftarrow \text{pos}$
  end if
end for

It is proven that the maximum number of iterations when $n$ and $m$ are constants is

$$L_{\text{max}}(n, m) = \left(\frac{n + 1}{2}\right)^2 \cdot \left(\frac{m + 1}{2}\right)^2$$

where

$$n_{\text{ins}} = \frac{n + 1}{2}, \quad m_{\text{ins}} = \frac{m + 1}{2}$$

Consequently, the theoretical complexity of the above algorithm is of $O(n^2m^2)$ or equivalently when $n > m$ is of $O(n^4)$.

Example 2.1 Let $A$ denote a $(100 \times 100)$ matrix with random 0/1 elements (Figure 1(a)). We define the inputs of algorithm such as
A numerical algorithm for the optimal placement of inset maps

- $A$, the random matrix
- $m_{ins} = 5$, the length of the proposed facility
- $n_{ins} = 5$, the width of the proposed facility

The execution of the algorithm returns the following outputs:

- $N = 28$, the number of the proposed facility siting locations
- positions, the matrix with the positions of all proposed facility siting locations (Figure 1(b))

![Figure 1](image)

Figure 1: Blue color denotes the non-existence of a non-favorable condition (zeros), green color denotes the existence of a favorable condition (ones), and red color denotes the solution in the location siting process, where (a) is the initial matrix and (b) is the final matrix with the solutions depicted as red areas.

Figure 1(b) shows the results of the proposed modified algorithm of this example. The proposed modified algorithm determines siting positions in the favorable locations and represents these locations in the form of a matrix (red color). Note the similarity between the process of siting a location facility with that of insetting placement in cartographic design.

2.1 Empirical Complexity

In this section we study the empirical complexity of Algorithm 2. In particular, we study (a) the number of iterations ($L$) in relation with the coverage ($C$) in a random matrix, and (b) the number of iterations ($L$) in relation with the mean area of connected pixels for both Algorithm 1 and Algorithm 2.
The theoretical maximum number of iterations is defined over all iterations without taking notice of the existence of zeroes or ones.

We define the measure “Coverage” as \( C = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j}}{m \cdot n} \) where \( a_{i,j} \) is an element of matrix \( A \).

For the approximation of the number \( L \), 100 random matrices are created with the dimensions 100 × 100 to 500 × 500, with coverage from 0.1 to 0.9 (with step 0.1), thus a total of 4,500 matrices. These matrices are created in MATLAB using function “randi”. Specifically, the creation of all random matrix datasets involves a random selection of indices \( i \) and \( j \) using function “randi”, and populates a matrix with ones until the target coverage value is reached.

The performance tests of Algorithm 1 and Algorithm 2 are implemented in the above datasets with a square form facility location of dimensions 5 × 5 until 95 × 95 (with step 5). Both algorithms return the number \( N \) of the proposed facility siting locations, the matrix positions that contains all the proposed placement positions of the facility into the map, and the number of iterations \( L \). When executing both algorithms for each set of 100 random matrices with same coverage, the results follow approximately a normal distribution (Figure 2(b)). For analyzing the results for each set of 100 random matrices we use the mean of \( L \). We define the iterations number of the Algorithm 1 as \( L_1 \) and the iterations number of Algorithm 2 as \( L_2 \). For the comparison of the two algorithms we define the measures \( P_1 = \frac{L_1 - L_2}{L_1} \) and \( P_0 = \frac{L_{\text{max}} - L_2}{L_{\text{max}}} \).

Figure 2: In (a) blue color denotes the non existence of a favorable condition (zeros) and green color denotes the existence of a favorable (ones). The coverage in this particular dataset shown is \( C = 0.4 \). In (b) is the corresponding distribution of the number of iterations \( L \) for 100 matrices (not shown) of this coverage.
between Algorithm 2 and the total theoretical iterations number. In Figure 3 we present the number of iterations for Algorithm 1 and Algorithm 2, respectively. For all datasets we conclude that the number of iterations is reduced in relation to $L_{max}$ as the coverage ($C$) and inset dimensions ($m_{ins} \times n_{ins}$) are increased. Additionally, we conclude that $L_2 < L_1$ in all cases.

![Figure 3](image)

**Figure 3**: The total number of iterations when $n = 100$ and $m = 100$ vs Coverage and Inset dimensions for Algorithm 1 (a) (resp. for Algorithm 2 (b)).

From the results illustrated in Figure 3 we conclude that the values of $P_1$ and $P_0$ are minimum when the facility location dimensions are $5 \times 5$ and the coverage is $C = 0.1$. This conclusion holds for all datasets, namely, matrix dimensions $200 \times 200$, $300 \times 300$ etc.

The minimum value of $P_1$ is approximately $0.644$ which means that the minimum reduction percentage of Algorithm 2 to Algorithm 1 is $64.4\%$. Likewise, the minimum value of $P_0$ is approximately $0.867$ which means that the minimum reduction percentage of Algorithm 2 to the total theoretical iterations number is $86.7\%$.

Thus, we can bound the iterations number of Algorithm 2 from the approximation of the maximum iterations number with the minimum value of measure $P_0$ for all matrix dimensions. Based on the results of Table 1 we conclude that the maximum iterations number of Algorithm 2 is given by

$$\max\{L_2\} \lessapprox L_{max} \cdot (1 - 0.865)$$

Similarly, the minimum value of measure $P_1$ for all matrix dimensions is

$$P_1 = 0.644 \implies \frac{L_1 - L_2}{L_1} = 0.644 \implies \frac{L_2}{L_1} = 0.356$$

From above ratio, we conclude that Algorithm 2 outperforms Algorithm 1, in terms of the iterations number.
Table 1: The minimum values of measures $P_1$ and $P_0$.

<table>
<thead>
<tr>
<th>Matrix size</th>
<th>$P_1$</th>
<th>$P_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100 \times 100$</td>
<td>0.646</td>
<td>0.869</td>
</tr>
<tr>
<td>$200 \times 200$</td>
<td>0.643</td>
<td>0.867</td>
</tr>
<tr>
<td>$300 \times 300$</td>
<td>0.642</td>
<td>0.867</td>
</tr>
<tr>
<td>$400 \times 400$</td>
<td>0.643</td>
<td>0.867</td>
</tr>
<tr>
<td>$500 \times 500$</td>
<td>0.644</td>
<td>0.867</td>
</tr>
</tbody>
</table>

As a next analysis step, we investigate some aspects of the spatial patterns of the random matrices generated. In particular, we define all the areas that consist favorable conditions regions for all matrices and all cases. From this definition, we compute the number ($N_{CP}$) of favorable conditions regions of a random matrix, the number of connected pixels that comprise each such region using a four (4) connectivity rule [6, 14].

We also calculate the minimum number of connected pixels ($min_{CP}$), the maximum number of connected pixels ($max_{CP}$), as well as the mean and the standard deviation of the number of connected pixels ($mean_{CP}$, $std_{CP}$).

The results of this spatial analysis over all random matrices generated are given in Table 2. Additionally, in Table 2 we present the corresponding iterations numbers of Algorithm 1 and 2.

Table 2: Spatial analysis results.

<table>
<thead>
<tr>
<th>Coverage</th>
<th>$min_{CP}$</th>
<th>$max_{CP}$</th>
<th>$mean_{CP}$</th>
<th>$std_{CP}$</th>
<th>$N_{CP}$</th>
<th>A1</th>
<th>A2</th>
<th>A1/A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>1.00</td>
<td>5.98</td>
<td>1.24</td>
<td>0.61</td>
<td>803.98</td>
<td>85493</td>
<td>30648</td>
<td>2.7895</td>
</tr>
<tr>
<td>0.20</td>
<td>1.00</td>
<td>12.19</td>
<td>1.63</td>
<td>1.24</td>
<td>1224.74</td>
<td>45892</td>
<td>10706</td>
<td>4.2866</td>
</tr>
<tr>
<td>0.30</td>
<td>1.00</td>
<td>24.49</td>
<td>2.31</td>
<td>2.43</td>
<td>1297.35</td>
<td>30725</td>
<td>6538</td>
<td>4.6994</td>
</tr>
<tr>
<td>0.40</td>
<td>1.00</td>
<td>59.18</td>
<td>3.69</td>
<td>5.41</td>
<td>1085.38</td>
<td>23038</td>
<td>4802</td>
<td>4.7976</td>
</tr>
<tr>
<td>0.50</td>
<td>1.00</td>
<td>238.57</td>
<td>7.24</td>
<td>18.10</td>
<td>691.86</td>
<td>18436</td>
<td>3839</td>
<td>4.8023</td>
</tr>
</tbody>
</table>

From Table 2 it is evident that the increase of $max_{CP}$, $mean_{CP}$ and $std_{CP}$ denotes an increase of the connected pixels area. We therefore conclude that the iterations number of Algorithm 2 is decreased significantly in relation to Algorithm 1, while the connected pixels area is increased.

3 Application to Cartographical problems

In this section, we incorporate the new search algorithm in the Inset Mapper (IM) software tool [13, 12]. IM is developed in MATLAB in order to use the software’s capability to manipulate numerical matrices [9], to implement numerical algorithms [4], and to handle mathematical representations [5]. The
use of the new search algorithm in IM reduces computational cost when dealing with large geographical datasets, thus making IM more efficient to use in real-world applications. In what follows, we particularly focus on the effect of the reduction in the total number of iterations $L$ on the execution time of the IM software tool.

### 3.1 Algorithms calculation time in geographical datasets

We compare the two algorithms with regards to their calculation time using geographical datasets with land discontinuity problems, which cover local, regional and continental geographical scales. The comparison for each scale was realized in maps with resolutions 50, 100, 150 and 200 ppi.

The inset dimensions in each example are proportional to the dimensions of the map in each resolution. Also, the area of each inset is $\frac{1}{10}$ of the main map area, in order to fulfill the cartographic rules mentioned in Subsection 3.2. Performance tests are implemented on a personal computer having an Intel Core Quad CPU (Q9400) at 2600 GHz with 3.5 Gb RAM.

For the local scale, the island of Samos was selected in scale 1:160,000 (Figure 4(a)), for the regional scale, the region of Chalkidiki was selected in 1:500,000 scale (Figure 4(b)) and for the continental scale, Europe was selected in (1:30,000,000) (Figure 4(c)).

![Maps](image)

(a) Island of Samos  
(b) Region of Chalkidiki  
(c) Europe

Figure 4: Maps in local (a), regional (b) and continental scale (c).

The percentage coverage is given by

$$\text{Cover} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{i,j}}{m \cdot n} \cdot 100\%$$

The ratio of inset area to map area is given by

$$\text{Ratio} = \frac{m_{\text{ins}} \times n_{\text{ins}}}{m \times n}$$

The above measurements for local scale (Samos Island) are $\text{Cover} = 20.34\%$ and $\text{Ratio} \approx \frac{1}{10}$, for regional scale (region of Chalkidiki) are $\text{Cover} = 37.85\%$.
and \( \text{Ratio} \approx \frac{1}{10} \) and for continental scale (Europe) with \( \text{Cover} = 22.22\% \) and \( \text{Ratio} \approx \frac{1}{10} \). The results of these performance tests are presented in Table 3.

Table 3: Calculation time for Algorithms 1 and 2 in local, regional and continental scale

<table>
<thead>
<tr>
<th>Resolution (ppi)</th>
<th>Map-matrix Size ((m \times n))</th>
<th>Inset Size ((m_{ins} \times n_{ins}))</th>
<th>Time in sec Algorithm 1</th>
<th>Time in sec Algorithm 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Local Scale</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>320x486</td>
<td>100x150</td>
<td>3.2344</td>
<td>0.51563</td>
</tr>
<tr>
<td>100</td>
<td>640x972</td>
<td>200x300</td>
<td>49.9530</td>
<td>8.31250</td>
</tr>
<tr>
<td>150</td>
<td>961x1458</td>
<td>300x450</td>
<td>273.7700</td>
<td>41.20300</td>
</tr>
<tr>
<td>200</td>
<td>1281x1944</td>
<td>400x600</td>
<td>906.1400</td>
<td>131.27000</td>
</tr>
<tr>
<td><strong>Regional Scale</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>320x486</td>
<td>100x150</td>
<td>0.515625</td>
<td>0.015625</td>
</tr>
<tr>
<td>100</td>
<td>640x972</td>
<td>200x300</td>
<td>7.765625</td>
<td>0.296875</td>
</tr>
<tr>
<td>150</td>
<td>961x1458</td>
<td>300x450</td>
<td>44.343750</td>
<td>1.359375</td>
</tr>
<tr>
<td>200</td>
<td>1281x1944</td>
<td>400x600</td>
<td>148.343800</td>
<td>4.078125</td>
</tr>
<tr>
<td><strong>Continental Scale</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>320x486</td>
<td>100x150</td>
<td>1.7813</td>
<td>0.1875</td>
</tr>
<tr>
<td>100</td>
<td>640x972</td>
<td>200x300</td>
<td>24.6090</td>
<td>2.1875</td>
</tr>
<tr>
<td>150</td>
<td>961x1458</td>
<td>300x450</td>
<td>138.3600</td>
<td>10.5000</td>
</tr>
<tr>
<td>200</td>
<td>1281x1944</td>
<td>400x600</td>
<td>457.4100</td>
<td>32.5470</td>
</tr>
</tbody>
</table>

3.2 Inset Mapper computational procedure

In this subsection, we the computational procedure implemented in IM for selecting optimal scale. This procedure utilizes mathematical descriptions of cartographic rules that control inset map placement in cartographic design. The objective here is to find positions on the map for the placement of insets with maximum dimensions. In order mathematically formulate the above objective, the following are defined:

- \( A \in \mathbb{R}^{m \times n} \): The geographical dataset (map) is transformed to a two-dimensional \((m \times n)\) numerical matrix \( A \), where \( n \) is the length and \( m \) the width of the dataset in pixels. This matrix has binary values: zero for every element corresponding to sea and one to land (note the inverse definition with respect to (1)):

\[
a_{i,j} = \begin{cases} 
1, & \text{land} \\
0, & \text{sea} 
\end{cases}
\]
• $b_x, b_y$: the length and the width, respectively, of the region which will become inset.

• $q$: the multiplier of inset dimensions.

• $q_{\text{max}}$: the maximum multiplier of inset dimensions.

• $B \in \mathbb{R}^{m_{\text{ins}} \times n_{\text{ins}}}$: the map-matrix that corresponds to inset, where $m_{\text{ins}}$ is the width of inset, $n_{\text{ins}}$ is the length of inset and

$$m_{\text{ins}} = b_y \cdot q, \ n_{\text{ins}} = b_x \cdot q$$

• $N$: the number of positions at which the inset can be placed.

Additionally, the following four cartographic rules should be applied:

1. the width of an inset should be less than or equal to the width of the map, that is, $m_{\text{ins}} \leq m$.

2. the length of an inset should be less than or equal to the length of the map, that is, $n_{i} \leq n$.

3. the area of an inset should be less than the $\frac{1}{4}$ area of the map, that is,

$$m_{\text{ins}} \cdot n_{\text{ins}} < \frac{m \cdot n}{4}$$

4. the inset dimensions should be at least two times the dimensions of the selected area

$$m_{\text{ins}} \geq 2 \cdot b_y$$
$$n_{\text{ins}} \geq 2 \cdot b_x$$

$\Rightarrow q \geq 2$

Accordingly, the following definition is in order:

**Maximum multiplier ($q_{\text{max}}$)** is the maximum number with which we multiply the inset dimensions in order to follow the above cartographic rules.

Therefore, we have

$$m_i \leq m \Rightarrow q \cdot b_y \leq m \Rightarrow q \leq \frac{m}{b_y}$$

from cartographic rule 1

$$n_i \leq n \Rightarrow q \cdot b_x \leq n \Rightarrow q \leq \frac{n}{b_x}$$

from cartographic rule 2

$$m_i \cdot n_i \leq \frac{m \cdot n}{4} \Rightarrow q \cdot b_x \cdot q \cdot b_y \leq \frac{m \cdot n}{4} \Rightarrow q^2 \leq \frac{m \cdot n}{4 \cdot b_x \cdot b_y} \Rightarrow q \leq \sqrt{\frac{m \cdot n}{4 \cdot b_x \cdot b_y}}$$
from cartographic rule 3. Thus, maximum multiplier is given by

\[ q_{\text{max}} = \min \left\{ \frac{m}{2 \cdot b_y}, \frac{n}{2 \cdot b_x}, \sqrt{\frac{m \cdot n}{4 \cdot b_x \cdot b_y}} \right\} \]  

(4)

The objective of the computational procedure is to find the maximum value of \( q \) such that matrix \( B \) is contained into matrix \( A \), with the corresponding elements of \( A \), being zero (the inset occupies marine space).

The computational procedure starts with multiplier \( q \) equal to the maximum multiplier \( q_{\text{max}} \), and calculates the number \( N \) using the proposed search algorithm. While \( N \) is equal to zero, multiplier \( q \) is decreased by step 1 \((q = q - 1)\) and \( N \) is recalculated. If \( N \) is not equal to zero then multiplier \( q \) is increased by step 0.1 \((q = q + 0.1)\) and \( N \) is recalculated. The computational procedure stops when \( N \in (0, N_s] \), where \( N_s \) is a user-defined threshold.

In summary, the procedure starts with the maximum \( q \) value, which is decreased rapidly (step 1) up to finding positions and then it increases slowly (step 0.1) in order to minimize the number of attempted positions.

The outputs of the above computational procedure are:

1. multiplier \( (q) \)
2. number \( N \) of insets
3. the proposed inset map placement positions in matrix \( A \)

The calculation of \( N \) uses the proposed search algorithm, and benefits from the execution speed of the latter.

### 3.3 IM calculation times

In this section, both Algorithm 1 and Algorithm 2 are applied within IM, and their performance comparison is undertaken. For this comparison, focused on calculation time, the six preselected cartographic datasets used are maps with land discontinuity problem (Figure 5).

The preselected maps are created in scale 1:500,000 and with the resolution of 100 ppi. The corresponding map-matrix dimensions are 972 × 641, and the initial inset dimensions are 10 × 15, which means that the maximum multiplier is the same for all maps and \( q_{\text{max}} = 32 \).

The results of IM are illustrated in Table 4, where the calculation times for both algorithms are presented.

From the above performance tests it becomes evident that the use of Algorithm 2 in the IM computational procedure brings a significant reduction in the calculation time of optimal scale and inset placement position, compared to the performance of Algorithm 1.
Figure 5: The six preselected maps for the comparison of the IM computational procedure.

Table 4: IM computational procedure results for each map and calculation times for Algorithms 1 and 2 for each map.

<table>
<thead>
<tr>
<th>Map</th>
<th>$q$</th>
<th>$N$</th>
<th>Cover%</th>
<th>Ratio</th>
<th>Times in sec Algorithm 1</th>
<th>Algorithm 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21.2</td>
<td>35</td>
<td>40.07</td>
<td>0.109911</td>
<td>15.828130</td>
<td>1.109375</td>
</tr>
<tr>
<td>2</td>
<td>23.1</td>
<td>54</td>
<td>6.84</td>
<td>0.130514</td>
<td>389.296900</td>
<td>2.171875</td>
</tr>
<tr>
<td>3</td>
<td>23.7</td>
<td>6</td>
<td>6.56</td>
<td>0.135989</td>
<td>588.687500</td>
<td>3.359375</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>402</td>
<td>7.37</td>
<td>0.246528</td>
<td>116.375000</td>
<td>1.546875</td>
</tr>
<tr>
<td>5</td>
<td>24.7</td>
<td>9</td>
<td>31.84</td>
<td>0.147673</td>
<td>431.671900</td>
<td>3.859375</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>426</td>
<td>25.90</td>
<td>0.246528</td>
<td>5.359375</td>
<td>1.515625</td>
</tr>
</tbody>
</table>

4 Conclusions and Future work

In this paper we presented a numerical algorithmic implementation for the solution of the land discontinuity problem arising in cartographic design. This algorithm enables cartographers to address the problem of siting an inset map at specific map locations resolving spatial and cartographic restrictions. In particular, we developed a new numerical algorithm improving the one in [13, 12], which now leads to significant reduction in the number of executed iterations, as well as computation time. We computed its theoretical complexity, and evaluated its empirical performance using randomly generated datasets as well as real geographical datasets. Finally, we presented the implementation of this new numerical algorithm in the Inset Mapper (IM) software tool, which converts the complicated insetting placement procedure to an easy automated
one.

Future research will focus on incorporating more cartographic rules into the computational procedures involved in IM using mathematical representations. We will also investigate the effectiveness of a dynamic version of the above algorithm in reducing computational time when working with very large geographical datasets.

Acknowledgements. The authors wish to acknowledge financial support provided by the Research Committee of the Technological Education Institute of Central Macedonia, under grant SAT/IC/10042014-78/6.

References


A numerical algorithm for the optimal placement of inset maps


Received: September 2, 2016; Published: October 21, 2016