TECHNICAL SCIENCES
Abbrev.: Techn. Sc., No 15(2), Y 2012

HOW TO EVALUATE OPTIMUM USAGE OF A LAND WITH LINDENMAYER'S GRAMMAR

Urszula Żukowska

Department of Mathematical Methods in Computer Science
University of Warmia and Mazury in Olsztyn

Key words: Lindenmayer's grammar, L-system, land management, land valuation, assessment of land's usage, cartographic method.

Abstract

This work presents the approach which combines Lindenmayer's grammar and Bajerowski's method for finding the optimum usage of a land. Purpose of this approach was to build quick method, which can simplify using of cartographic method and includes algorithm based on natural growth. Result of proposed method is a computer program, which supports decision-making process of analyzing usage of a land. The approach gives possibility of starting from any point from a map and finishing in any derivation step. Therefore user can generate many findings quickly, especially in the form of a map, and compare them. It is also possible to find expansion of specified function of a land. Content of the paper is presented below. First, we discuss context-free parametric L-system. Subsequently we outline the Bajerowski's method. Next we present our approach with adaptation of Bajerowski's proposal and with our description of proposed L-system. We give also example of using our approach with some explanation. Eventually, we presented advantages of our system and possible future developments.

Introduction

Initially, an L-system was designed to simulate growth of plants. Then, because of its usefulness, it was applied to many other problems such as growth of other biological organisms, creating the landscape, drawing fractals and even for composing the music. We adapted L-system's evolution mechanism to finding an optimum usage of a land, which is very vital in evaluation of land's value. The result is a computer program, in which user can choose purpose of a land and initial parameters. Subsequently, after a chosen number of iterations user receives an output as a map with charted fields and sequence of productions. It allows interpreting output in two ways and using the map in further analysis.

In our system was have used Bajerowski's (BAJEROWSKI 1996, 2003, *Podstawy teoretyczne gospodarki*... 2003) proposal for estimating the optimum usage of a land. His idea is about reading features from a cartographic map and then calculating the value for each basic field. Next step is interpretation of the value for every single field as an optimal or not optimal for specified purpose. This allows urban and spatial experts to make decision whether the land should have another purpose. Another purpose can change the value of a land; therefore it is very important to estimate its profitability.

Lindenmayer's grammar and Bajerowski's method

An L-system is a parallel rewriting system. We present basic definition. The definition with some developments can be found in works (PEITGEN et al. 1992, PRUSINKIEWICZ et al. 1995, PRUSINKIEWICZ, LINDENMAYER 1996,). The rewriting process starts from an axiom – initial module. Then it is replaced by configurations of modules taken from set of rewriting rules. Every individual module, which is replaced in an interaction, is called parent or mother or predecessor. A module, which replaces ancestor's module, is called child or daughter or successor. In the rewriting process mother module is replaced with the daughter module.

Let us denote:
– a module with letter $A \in V$ and parameters $a_1, a_2, ..., a_n$ as by $A(a_1, a_2, ..., a_n)$,
– $M = V \times \Re^*$ as a set of modules, where $\Re^*$ is the set of all finite sequences of parameters,
– $M^* = (V \times \Re^*)^*$ as a set of all strings of modules,
– $M^+ = (V \times \Re^*)^+$ as a set of all nonempty strings,
– $\mathrm{C}(\Sigma)$ as a logical expression with parameters from Σ,
– $\mathrm{E}(\Sigma)$ as an arithmetic expression with parameters from Σ.

Therefore, a parametric context-free L-system can be defined as an ordered quadruple.

$$G = \langle V, \Sigma\ \omega, P \rangle \tag{1}$$

where:
V – the alphabet of the system,
Σ – the set of formal parameters,
$\omega \in (V \times \Re^*)^+$ – a nonempty parametric word called the axiom,
$P \subset (V \times \Sigma^*) \times C(\Sigma) \times (V \times E(\Sigma))^*$ – a finite set of production.

Productions in this system have format:

Predecessor: condition $\rightarrow$ successor.

Every land has the one optimum usage, which can be achieved by re-development. The redevelopment must be done with minimal costs of it. Therefore Bajerowski (BAJEROWSKI 1996, 2003, *Podstawy teoretyczne gospodarki...* 2003) proposed a cartographic method as tool for redevelopment. We have adapted it to L-grammar; therefore we outline the method of finding the optimum usage of a land. The method assumes that we can read features from a map. We can use topographical map and map of land's sorts.

This method can be described in 4 steps:

1. We must put a square grid on a map with a mesh of a chosen size.

2. We must read features from an every single field and put them into the inventory matrix. We use "0" to denote that specified feature does not exist in basic field and "1" otherwise.

3. We must derive optimal function of a land by multiplying transposed matrix of features which can optimize usage of a land by the inventory matrix. Therefore we arrive at the matrix of optimal function of a land.

4. Interpretation of data from step 3 for optimum usage of a land.

Matrices in cartographic method are described as follows:

– The inventory matrix consists of "0" and "1" and their length and width corresponds to length and width of square grid, put on a map.

– The matrix of features for optimum functions of a land, further called the matrix of features consists of names of features, functions of terrain and parameters, which indicates usefulness of a feature for specified function. Values of parameters used in this matrix were specified by land value experts.

– The matrix of optimal function of a land consists of values which are group as less than or equal to zero or greater than zero. First group indicates squares not accepted for specified function and second group indicates squares which are optimal for specified function.

In original method all processes was done manually. Square grid was put on the whole analyzed area and therefore there were two approaches to using Bajerowski's method. First approaches used all squares in specified grid and second one used fields chosen by Monte Carlo method. Values, read from the map, were put to the inventory matrix and after that was calculated multiplication of matrices. The last step demanded user's interpretation of values, and eventually putting the outputs on a new map.

Optimum usage of a land with Lindemayer's grammar and Bajerowski's method

We use all steps from Bajerowski's method but with some changes (ŻUKOWSKA 2008). In our methods this steps can be described as follows:

1. We must put a square grid on a map with a mesh of a chosen size. Done manually or automatically if we have digital map.

2. We must read features from an every single field and put them into the inventory matrix. We arrange these features into groups therefore we use specified values for the specified feature. We accept "0" as a value, which indicates that the feature does not exist in analyzed square. Our inventory matrices are placed in proposed computer program, therefore this step is done half automatically.

3. We must derive optimal function of a land. We still use multiplication of transposed matrix of features which can optimize usage of a land by the inventory matrix, but we use it only for squares chosen by Lindenmayer's grammar.

4. Interpretation of data from step 3 for optimum usage of a land is done automatically in every rewriting of L-system and outputs are presented in a table and a map.

In our system we arrange features from the matrix of features into the followings groups:

1. Water – w,
2. Greenery – z,
3. Structure of terrain – s,
4. Another terrains – i,
5. Roads and infrastructure – dr,
6. Exposure – k,
7. Slopes – p,
8. Kind of a land – u.

Every group has feature with their values. These are:

1. Water: shorelines of a lake – 1, rivers and brooks – 2, canals and ditches – 3, bog and marshland – 4, little standing water – 5, sources – 6, marshy lands – 13.
2. Greenery: border of forests – 7, row of trees – 8, group of trees, groves – 9, single trees – 10, bushes, shrubs, hedges – 11, thickets, bush clump – 12.
3. Structure of terrain: gorges, ravines – 14, bluffs, embankments, excavations – 15, sand – 16, Rocks and boulders – 16.
4. Another terrains: devastated land -18, industrial land – 19, buildings – 20, ruins – 21, cemetery and burial ground – 29, protected area – 30, natural monuments – 31, historical monuments – 32.
5. Roads and infrastructure: power lines – 22, railway – 23, tarmac road – 24, track – 25, lane – 26, footpath – 27, fence – 28.
6. Exposure: north – 33, north-east – 34, east – 35, south-east – 36, south – 37, south-west – 38, west 39, north-west – 40.

7. Slopes: 0–3% slope – 41, 3–6% slope – 42, 6–10% slope – 43, 10–15% slope – 44, 15–25% slope – 45, 25–35% slope – 46, above 35% slope – 47.
8. Kind of a land: meadow I–III class – 48, meadow IV – V class – 49, meadow VI class – 50, pasture I–III class – 51, pasture IV–V class – 52, pasture VI–VIz class – 53, arable Land I–IIIb class – 54, arable land Iva–V class – 55, arable land VI–Viz class – 56.

In our computer system, we used context-free parametric L-system. Our system is defined as follows:

$$G = \langle V, \Sigma\ \omega, P \rangle \tag{2}$$

where:

V – as the alphabet of the system, consists of numbers 0...9, letters – A...Z, a...z,

Σ – the set of formal parameters, consist of parameters: v, w, z, s, dr, i, k, p, u, nr.x, nr.y, f, d, licz,

$\omega \in (V \times \Re^*)^+$ – is a nonempty parametric word called the axiom chosen by a user,,

$P \subset (V \times \Sigma^*) \times C(\Sigma) \times (V \times E(\Sigma))^*$ – is a finite set of production.

Below are described all used parameters,

v – variable where the total sum of values is stored;

w – variable where the value from water group is stored;

z – variable where the value from greenery group is stored;

s – variable where the value from structure of a terrain group is stored;

dr – variable where the value from roads and technical infrastructure group is stored;

i – variable where the value from another terrains group is stored;

k – variable where the value from exposure group is stored;

p – variable where the value from slopes group is stored;

u – variable where the value from kind of land group is stored;

Parameters which assume an integer value dependent on a size of a map and mesh: nr.x nr.y – variable where the coordinates of a specified basic field are stored, respectively x and y.

f – variable where the purpose of a land is stored. Possible values are: 1- agricultural – arable land, 2 – agricultural – pasture, 3 – agricultural – meadow, 4 – forestry – productive, 5 – forestry – ecological, 6 – recreational – individual recreation, 7 – recreational – collective recreation, 8 – recreational without legal possibility of building development, 9 – B – housing estate areas, 10 – P – industrial infrastructure.

d – variable where randomly chosen direction to the next square is stored,

Here possible variables are: dn = 0 (north), ds = 1 (south), dw = 2 (west), de = 3 (east), dne = 4 (north-east), dnw = 5 (north-west), dse = 6; (south-east), dsw = 7 (south-west).

licz – boolean variable where information is stored whether the total sum was calculated for the square (true = 1) or not (false = 0).

ω as an axiom, is a single basic square chosen by a user.

In the proposed system, every single square is stored as an object with all operations done on it.

There is one important function, which prints squares with its parameters as follows:

$$SQ([nr.x,nr.y],\ d,f,v,w,z,s,dr,i,k,p,u) \tag{3}$$

where:
SQ means square and other symbols are described above.

There are a few groups of productions in our system. The First group consist of productions which determine direction of a new square. The second group consists of productions which calculate optimal usage of a land. The third group consist of productions calculates value v for a specified function of terrain.

Productions from the first group are as follows:

1. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=0
   →SQ([nr.x -1,nr.y], d, f, v,w, z, s, dr, i, k, p, u),
2. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=1
   →SQ([nr.x +1,nr.y], d, f, v,w, z, s, dr, i, k, p, u),
3. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=2
   → SQ([nr.x,nr.y-1], d, f, v,w, z, s, dr, i, k, p, u),
4. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=3
   → SQ([nr.x,nr.y+1], d, f, v,w, z, s, dr, i, k, p, u),
5. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=4
   →SQ([nr.x-1,nr.y+1], d, f, v,w, z, s, dr, i, k, p, u),
6. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=5
   →SQ([nr.x-1,nr.y-1], d, f, v,w, z, s, dr, i, k, p, u),
7. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : d=6
   →SQ([nr.x-1,nr.y+1], d, f, v,w, z, s, dr, i, k, p, u).

Productions from the second group are as follows (vv is a parameter, which is used to denote usefulness of o square):

8. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) :v>0 ∧ licz=1 →
   SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) :vv='X'
   This means "denote square as optimal for function f".

9. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) : v<=0 ∧ licz=1 →
   SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) :vv='0'
   This means "denote square as not optimal for function f".
10. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) :v=0 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, p, u) :vv='+'
   This means "count the value v using the productions from third group".

Third group of productions are too long for this paper therefore we present only first ten of productions for agricultural function – arable land. These are:

11. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, R, u) :f=1 ∧ w=1 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-5,w, z, s, dr, i, k, p, u)
12. SQ([nr.x,nr.y], d, f, v,w, z, s, dr, i, k, R, u) :f=1 ∧ w=2 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-5,w, z, s, dr, i, k, p, u)
13. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ w=3 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-1,w, z, s, dr, i, k, p, u)
14. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ w=4 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-1,w, z, s, dr, i, k, p, u)
15. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ w=5 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-3,w, z, s, dr, i, k, p, u)
16. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ w=6 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-5,w, z, s, dr, i, k, p, u)
17. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ w=13 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-1,w, z, s, dr, i, k, p, u)
18. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ z=7 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-5,w, z, s, dr, i, k, p, u)
19. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ z=8 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-3,w, z, s, dr, i, k,p, u)
20. SQ(nr, d, f, v,w, z, s, i, dr, k, R, u) :f=1 ∧ z=9 ∧ licz=0 →
   SQ([nr.x,nr.y], d, f, v-3,w, z, s, dr, i, k, p, u)

In our computer program, every single square is represented as an object, which is stored in a dynamic array of objects. Initially, program initiates the specified axiom in the dynamic array. Constructor creates this object using coordinates, randomizing direction for the next square, using chosen function, reading values for certain variables from inventory matrix and marking the counting flag on "0" value (because the firs iteration does not count the total value). Next few steps are repeated in every iteration:

– one function searches every object in the dynamic array and calculates value v if licz=0, and after calculating it sets licz for 1,

– another functions searches dynamic array and make new objects using the direction written in the parameter d (it is counted accordingly to coordinates of checked square) it also checks whether the new square exists in the dynamic array, if it exists there is no initiations of this square again,

– function for printing the output; put the new production on the screen.

To make a new iteration user must push the button; therefore user decides how much iteration will be made.

The method for calculating the value v uses the matrix of features, which is constant in the program. It initiates indices for the matrix of features using the function (indicates a column in the matrix) and value from every parameter (indicates a row in the matrix). Using the indices, method reads values from the matrix of features and sums the values and store the result in the parameter v.

Example of using our system

Using the computer program we must follow the 4 steps:

1. Preparing data such as maps (scanning) and then inventorying features from a map and write them to the computer program. In our example it is fragment of the topographical map of Olsztyn and its outskirts. Single square has side of 500 meters, therefore its area is 250000 square meters=25 hectares which is between 4 and 30 hectares for village or district as suggested in literature (SENETRA, CIEŚLAK 2004).

2. Choosing coordinates of an initial square (axiom) and the number of appropriate functions of the terrain. In our example these are: coordinates – [4,4] and function – 1, that is agricultural – arable land.

3. Making certain iterations. In our example it is five.

4. Visualize the output. We have three forms of output: sequences of squares, tables and map, which visualize the same results in a different way.

After putting the grid on a map we make the inventory of all features, which can be read from a map, and write them into inventory matrices . Then we chose axiom.
The chosen axiom is : SQ([4,4],5,1,0,2,10,0,24,20,36,42,0)
Values from inventory matrices are represented in table 1.

Table 1

Inventory Matrices

w – water								
	1	2	3	4	5	6	7	8
1	0	0	2	2	0	0	5	0
2	5	2	2	5	0	0	1	1
3	2	0	2	2	5	4	1	1
4	0	5	5	2	2	2	5	1
5	5	0	2	5	0	0	0	1
6	5	4	4	2	2	2	1	1
7	5	4	4	5	2	2	1	0
8	0	4	2	2	2	4	1	1

z – greenery								
	1	2	3	4	5	6	7	8
1	7	7	7	7	7	7	7	7
2	7	7	7	7	7	7	7	7
3	0	7	7	7	10	10	7	7
4	0	7	7	10	10	12	7	7
5	7	7	10	10	10	10	10	10
6	7	10	10	10	10	10	0	10
7	7	7	9	10	7	8	12	0
8	7	12	7	7	7	7	12	12

cont. table 1

s – structure of a terrain								
	1	2	3	4	5	6	7	8
1	0	0	15	15	14	14	0	0
2	0	14	14	0	15	0	15	0
3	14	15	15	0	0	0	15	0
4	0	0	15	0	15	0	15	15
5	0	0	0	0	0	0	15	15
6	0	0	0	0	15	14	15	15
7	0	0	0	0	15	14	15	0
8	0	0	15	15	0	15	15	15

dr – roads and technical infrastructure								
	1	2	3	4	5	6	7	8
1	26	23	26	26	26	26	26	26
2	23	26	26	24	24	26	26	24
3	24	24	24	28	24	26	26	26
4	24	27	26	24	24	23	23	25
5	26	26	0	25	24	24	24	23
6	25	25	25	25	24	26	24	24
7	26	26	26	25	25	26	26	0
8	26	26	25	25	26	26	26	0

i – another terrains								
	1	2	3	4	5	6	7	8
1	0	0	0	20	20	0	0	0
2	20	20	0	0	20	20	20	0
3	0	20	20	20	20	20	20	0
4	0	0	20	20	20	20	20	0
5	20	20	0	20	0	20	20	20
6	20	20	20	20	20	20	20	20
7	20	20	0	20	20	20	0	0
8	20	0	0	20	0	0	0	20

k – exposure								
	1	2	3	4	5	6	7	8
1	35	35	36	35	33	33	34	36
2	34	37	40	36	40	34	35	37
3	34	33	34	35	38	36	34	33
4	40	39	40	36	38	38	35	35
5	35	33	40	40	34	37	37	35
6	40	34	40	40	35	36	36	35
7	38	37	36	36	37	37	39	0
8	38	37	36	36	35	34	35	34

p – slopes								
	1	2	3	4	5	6	7	8
1	0	0	42	42	0	0	45	0
2	45	42	42	45	0	0	41	41
3	42	0	42	42	45	44	41	41
4	0	45	45	42	42	42	45	41
5	45	0	42	45	0	0	0	41
6	45	44	44	42	42	42	41	41
7	45	44	44	45	42	42	44	0
8	0	44	42	42	42	44	41	41

u – kind of land								
	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

We chose function agricultural – arable land and we arrive at sequences, which are presented below.

After first rewriting:

SQ([4,4],4,1,8,2,10,0,24,20,36,42,0) SQ([3,4],4,1,0,2,7,0,28,20,35,42,0)

After second rewriting:

SQ([4,4],4,1,8,2,10,0,24,20,36,42,0) SQ([3,4],2,1,4,2,7,0,28,20,35,42,0)

SQ([3,5],6,1,0,5,10,0,24,20,38,45,0) SQ([2,5],5,1,0,0,7,15,24,20,40,0,0)

After third rewriting:
SQ([4,4],5,1,8,2,10,0,24,20,36,42,0)
SQ([3,5],0,1,-9,5,10,0,24,20,38,45,0)
SQ([3,3],6,1,0,2,7,15,24,20,34,42,0)
SQ([1,4],2,1,0,2,7,15,26,20,35,42,0)
SQ([3,4],2,1,4,2,7,0,28,20,35,42,0)
SQ([2,5],0,1,-13,0,7,15,24,20,40,0,0)
SQ([4,6],1,1,0,2,12,0,23,20,38,42,0)
After fourth rewriting:
SQ([4,4],2,1,8,2,10,0,24,20,36,42,0)
SQ([3,5],1,1,-9,5,10,0,24,20,38,45,0)
SQ([3,3],3,1,-3,2,7,15,24,20,34,42,0)
SQ([1,4],7,1,20,2,7,15,26,20,35,42,0)
SQ([5,6],1,1,0,0,10,0,24,20,37,0,0)
SQ([3,4],7,1,4,2,7,0,28,20,35,42,0)
SQ([2,5],6,1,-13,0,7,15,24,20,40,0,0)
SQ([4,6],2,1,6,2,12,0,23,20,38,42,0)
SQ([1,5],0,1,0,0,7,14,26,20,33,0,0)
SQ([1,3],3,1,0,2,7,15,26,0,36,42,0)
After fifth rewriting:
SQ([4,4],1,1,8,2,10,0,24,20,36,42,0)
SQ([3,5],2,1,-9,5,10,0,24,20,38,45,0)
SQ([3,3],6,1,-3,2,7,15,24,20,34,42,0)
SQ([1,4],5,1,20,2,7,15,26,20,35,42,0)
SQ([5,6],1,1,-4,0,10,0,24,20,37,0,0)
SQ([4,3],2,1,0,5,7,15,26,20,40,45,0)
SQ([3,6],2,1,0,4,10,0,26,20,36,44,0)
SQ([6,6],4,1,0,2,10,14,26,20,36,42,0)
SQ([3,4],0,1,4,2,7,0,28,20,35,42,0)
SQ([2,5],4,1,-13,0,7,15,24,20,40,0,0)
SQ([4,6],6,1,6,2,12,0,23,20,38,42,0)
SQ([1,5],6,1,5,0,7,14,26,20,33,0,0)
SQ([1,3],2,1,22,2,7,15,26,0,36,42,0)
SQ([4,5],4,1,0,2,10,15,24,20,38,42,0)
SQ([2,3],1,1,0,2,7,14,26,0,40,42,0)

In Figure 1, are presented output in table, where: + – denotes square as optimal for specified function, 0 – denotes square as non optimal for specified function and X – denotes square, for which value v will be calculated in next iteration.

Figure 2 presents the map with results after 5 iterations. Chequered squares indicate fields, which are optimal for agricultural function – arable land. Those squares are located near the road and buildings, which is convenient for this function of land. It is because all special machines used in agriculture can be easily moved from garages to arable lands and crops can be quickly transported to silos or to other parts of the region.

In the Figure 2 squares with big cross indicate area which is not optimal for arable land. In this example most of these squares are located in the forest. The forest is not convenient to arable land, because it demands a lot of additional work of preparing soil. This work increases costs of redevelopment, which should be as little as possible. Therefore that land is not optimal for arable land.

a

	1	2	3	4	5	6	7	8
1								
2								
3				X				
4				+				
5								
6								
7								

b

	1	2	3	4	5	6	7	8
1								
2					X			
3				+	X			
4				+				
5								
6								
7								

c

	1	2	3	4	5	6	7	8
1				X				
2					0			
3			X	+	0			
4				+		X		
5								
6								
7								

d

	1	2	3	4	5	6	7	8
1			X	+	X			
2					0			
3			0	+	0			
4				+		+		
5						X		
6								
7								

e

	1	2	3	4	5	6	7	8
1			+	+	+			
2			X		0			
3			0	+	0	X		
4			X	+	X	+		
5						0		
6						X		
7								

Fig. 1. Output of (*a*) first, (*b*) second, (*c*) third, (*d*) fourth and (*e*) fifth rewriting

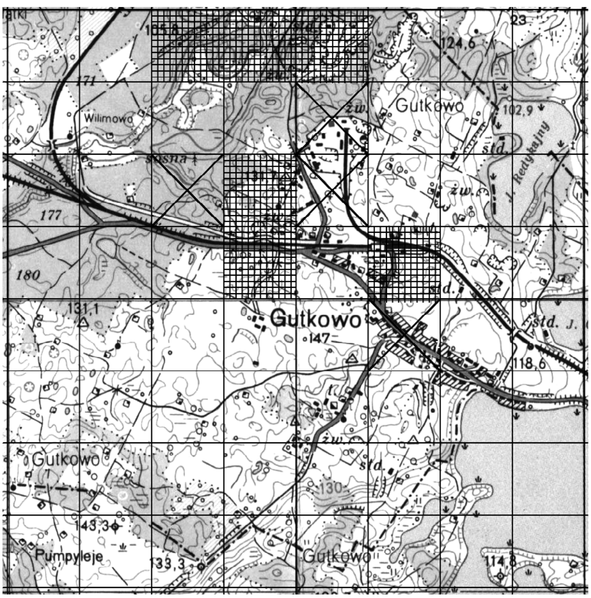

Fig. 2. Output of fifth rewriting in the map of Olsztyn, side of a single square corresponds to 500 meters

Conclusions

The presented approach uses Bajerowski's method of finding the optimum usage of a land and Lindenmayer's grammar. The approach was programmed in Delphi 6.0 as a computer program. This facilitates and quickens the whole process of finding the optimum usage of a land instead of doing the whole process manually.

Moreover, computer program generates the output very quickly whereas with classic approach we have drawn the output results on a map. With our solution we can generate the results even after every rewriting.

In classic method we should calculates all values and we should also decide which value indicates squares which are optimal or not optimal for specified function. In our proposition it is done automatically.

Being done manually, cartographic method can only use a small number of squares. This means using this method for a small area or chose some squares from the whole area. Our approach solves the problem of squares in a different way. User must choose first square, which is an axiom, and the other squares are chosen by the algorithm. In every rewriting process there are some new squares added. We can continue the process till we have the whole area covered but we can stop in every moment. This means that we decide how long the algorithm will work. This solution shows also the possible expansion of specified function of terrain. This point was not considered in discussed method.

Our computer program can be included in GIS as an innovative tool, which uses simulation of natural growth in the contrast of deterministic tools.

Our approach has many advantages but it can be also developed. Today we use digital maps therefore it is possible to add a new feature to our program. The feature could allow automatisation of reading map and generating the inventory matrix. The program was intended for people responsible for land planning or land management therefore it is possible to add features of decision making.

Further developments can also consist of elements from genetic programming, which describe the simulation process more precisely.

Translated by Author

Accepted for print 18.10.2012

References

Bajerowski T. 1996. *Metodyka optymalnego użytkowania ziemi na obszarach wiejskich*. Acta Acad. Agricult. Tech. Olst., Geod. Ruris Regulat., 26, suppl B. Olsztyn.

Bajerowski T. 2003. *Niepewność w dynamicznych układach przestrzennych*. Wydawnictwo Uniwersytetu Warminsko-Mazurskiego, Olsztyn.

MARTYN T. 1996. *Fraktale i obiektowe algorytmy ich wizualizacji*. Nakom, Poznań.

PEITGEN H.-O., JÜRGENS H., SAUPE D. 1992. *Granice chaosu fraktale,* cz. 2. Wydawnictwa Naukowe PWN, Warszawa.

Podstawy teoretyczne gospodarki przestrzennej zarządzania przestrzenią. 2003. Ed. T. BAJEROWSKI, Wydawnictwo UniwersytetuWarminsko-Mazurskiego, Olsztyn.

PRUSINKIEWICZ P. 2004. *Art and science for life: Designing and growing virtual plants with L-systems*. In: *Nursery Crops: Development, Evaluation*. Eds. C. Davidson, T. Fernandez. Production and Use proc XXVI International Horticultural Congress., Acta Horticulturae 630: 15–28.

PRUSINKIEWICZ P., HAMMEL M., HANAN J., MECH R. 1995. *The artificial life of plants. In: Artificial life for graphics, animation, and virtual reality*. Volume 7 of SIGGRAPH '95 Course Notes, ACM Press, pp. 1–38.

PRUSINKIEWICZ P., HAMMEL M., HANAN J., MECH R. 1996. *L-systems: from the theory to visual models of plants*. In: *Proc the 2nd CSIRO Symposium on Computational Challanges in Life Sciences*, Ed. M.T. Michalewicz. CSIRO Publishing,

PRUSINKIEWICZ P., HAMMEL M., MJOLSNESS E. 1993. *Animation of plant development*. Proc SIGGRAPH 93, in Computer Graphics Proceedings, Annual Conference Series, Anaheim – California, pp. 351–360.

PRUSINKIEWICZ P. KARWOSKI R., LANE B. 2007. *The L+C plant-modelling language*. In: *Functional-Structural Plant Modelling in Crop Production*. Springer.

PRUSINKIEWICZ P., LINDENMAYER A. 1996. *The algorithmic beauty of plants*. Springer-Verlag, New York.

PRUSINKIEWICZ P., PALUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MECH R. 2009. *Self-organizing tree models for image synthesis*. ACM Transactions on Graphics 28(3): 58: 1–10.

SENETRA A., CIEŚLAK I. 2004. *Kartograficzne aspekty oceny i waloryzacji przestrzeni*. Wydawnictwo Uniwersytetu Warminsko-Mazurskiego, Olsztyn

STREIT L., FEDERL P., SOUSA M.C. 2005. *Modelling Plant Variation Through Growth*. Computer Graphics Forum, 24(3): 497–506.

TRACZ W., KLAPEC M. 2007. *Using an expert system for housing utility assessment of terrain*. Proceedings of Artificial Intelligence Studies, vol 4(27): pp. 157–163.

ŻUKOWSKA U. 2008. *Planning of Optimum Usage of a Land with Lindenmayer's Grammar*. Proc IEEE Conference on Human System Interaction, Cracow, Poland, pp. 409–414.