Synergetics and Theory of Chaos

УДК 167.7, 517

Gritsak von Groener V. V., Gritsak-Groener J.

MATHEMATICAL METHODS OF IDENTIFICATION FOR VISUAL INFORMATION. 2.

University of Georgia, Georgia, USA; Laboratory of HRIT Corporation, Switzerland, USA, UK
e-mail: v_hrit1000000@yahoo. com

We continue grinding wooden graphic-theory for own needs, which we started in the first paper of this series. In this article we will assume that the graph is connected. We construct a minimum forest consisting of the union of the minimal spanning trees of its connected components. Because of its importance in operational research and in data analysis, many algorithms have been proposed for constructing a minimal spanning tree of a valued graph.

Keywords: graph, image, visual information.

1. Introduction

Commentary. As we have already noted in the previous article, in the future, to study the cluster structure is sufficient to consider and examine only their connected components are trees, and then successively to each connectivity component of the bush to apply the obtained results.

The main idea **MIVI** is the following. We take the coding graphs **Γ** (encoding algorithm graph is presented in the previous article in this series), which is representing a certain visual information, such as image of the painting **G̈Ɔ**. Compute a minimum spanning tree $\mathbf{\Gamma^o}$ of the graph **Γ** (an algorithm for computing the minimum spanning tree of a graph is presented in this article of the series). Further to the tree $\mathbf{\Gamma^o}$ calculate its **needles** $\{\mathbf{i_\Gamma}\}$ and **body B**. Needles are the maximum subtrees of $\mathbf{\Gamma^o}$ with all points, except for two points of $\mathbf{v(i_\Gamma)}$ and $\mathbf{o(i_\Gamma)}$, with degree equal to two: $\mathbf{st(v(i_\Gamma)) = 1}$. $\mathbf{v(i_\Gamma)}$ is the **top** of the needle $\mathbf{i_\Gamma}$, and $\mathbf{st(o(i_\Gamma))} \geq 1$ ($\mathbf{o(i_\Gamma)}$ is the basis of a needle $\mathbf{i_\Gamma}$. The **body B** is the remaining part of the tree $\mathbf{\Gamma^o}$. Thus, we turn our tree $\mathbf{\Gamma^o}$ to the **hedgehog**

Fig. 1. A. Fomenko "A hedgehog"

$$\mathbf{I(B)} = (\mathbf{B}, \{\mathbf{i_\Gamma}\}), \tag{1}$$

or **G-invariant** of graph **Γ**.

If we calculate all the hedgehogs $\mathbf{I_i(\Gamma)}$, $i = \overline{1,n}$ for all **n** spanning trees of the graph **Γ**, the following hypothesis seems to be true.

Hypothesis. The collection of all hedgehogs

$$\mathbf{I_i(\Gamma)},\ i = \overline{1,n}, \tag{2}$$

is a complete system of invariants of a plane graph **Γ**.

If we assume the validity of the above hypothesis, and that the graph **Γ** for **IVI** picture, obviously, is flat, we can assume that the hedgehog (1) is a certain individual characteristic of pattern **G̈Ɔ**.

The same goes for **IVI** of face, that actually (topologically!) is a flat pattern, the hedgehog (1) is an individual invariant.

Finally, for graphs **IVI** of any stereo **ꟷA** the hedgehog (1) is probably not enough to the qualitative identification of the image. Then, we need to calculate all the hedgehogs (2), or even all the hedgehogs of all components of a manifolds **M**, representing the flat surface of the stereo image **IA**.

2. More about the graphs

At first, there are a few necessary approvals.

Theorem 1. Let

$$\mathbf{\Gamma} = (\mathbf{V}, \mathbf{E}), \mathbf{E} \subseteq \mathbf{V}\times\mathbf{V} \tag{3}$$

be a graph. The following two assertions are equivalent:

(a) **Γ** is connected.

(b) **Γ** has a partial graph which is a tree.

Proof. (a) implies (b). We construct a sequence $\mathbf{\Gamma} = \mathbf{\Gamma}_0, \mathbf{\Gamma}_1, ..., \mathbf{\Gamma}_m$ of partial graphs of Γ defined as follows: Γ_i is a connected graph obtained by removing one edge from

$\mathbf{\Gamma}_{i-1}$. Because of the finiteness of **Γ** graph, we eventually come to a tree $\mathbf{\Gamma}_m$.. In an extreme case, $\mathbf{\Gamma}_m$. will be the usual edge. The converse is obvious, even more.

When the partial graph **H** of **Γ** is a tree, we say that **H** is a **tree** of **Γ**. If **H** is a tree of a sub-graph of **Γ** we say that **H** is a **partial sub-tree** of **Γ**. Now consider a connected valued **G**. The **length** $\mathbf{dL_H}$ of a tree **H** of **Γ** is defined to be the sum of the lengths of the edges of **H**. A tree **H** of **Γ** such that $\mathbf{dL_H}$ is a minimum is called a **minimal spanning tree** of **Γ**. Before indicating a solution, we will show that when the lengths of the edges are all different this tree is unique.

Theorem 2. Let **G = (V, E)** be a connected graph valued by an injective edge's function $\mathscr{J}$

$$\mathscr{J} : \mathbf{E} \to \mathbb{R}^+ . \tag{4}$$

Then the minimum spanning tree **T** of **(G,** $\mathscr{J}$**)** is unique.

Proof. Let $\mathbf{T_1}$ and $\mathbf{T_2}$ be two distinct minimal spanning trees of **G**. Let **U** be the set whose elements are the edges of $\mathbf{T_1}$ which are not edges of $\mathbf{T_2}$ and the edges of $\mathbf{T_2}$ which are not edges of $\mathbf{T_1}$. Let **(uv)** be the edge of **U** (unique because of the injectivity of $\mathscr{J}$) of smallest length. Assume that **(uv)** is an edge of $\mathbf{T_1}$. In $\mathbf{T_2}$ the vertices **u** and **v** are joined by a path:

$$\mathbf{u} = \mathbf{u_0 u_l \ldots u_m} = \mathbf{v}.$$

Since $\mathbf{T_1}$ is acyclic and since **(uv)** is an edge of $\mathbf{T_1}$, at least one of the edges in this path is in **U**. If $\mathscr{J}(\mathbf{u_i u_{i+1}}) < \mathscr{J}(\mathbf{uv})$, for all i, $\mathbf{0 \le i \le m - 1}$, then $\mathscr{J}(\mathbf{uv})$ is not minimal in **U**. Thus, there exists an edge $(\mathbf{u_i u_{i+1}})$ in (3) such that $\mathscr{J}(\mathbf{u_i u_{i+1}}) > \mathscr{J}(\mathbf{uv})$. Now consider the partial graph $\mathbf{T_3}$ of **G** obtained from $\mathbf{T_2}$ by removing the edge $(\mathbf{u_i u_{i+1}})$ and adding the edge (**uv**). $\mathbf{T_3}$ is clearly connected. Moreover, it has no cycle not containing (**uv**) (for this would be a cycle of $\mathbf{T_2}$) and every cycle of $\mathbf{T_3}$ containing (**uv**) would define a cycle of $\mathbf{T_2}$ by using the path (3). Thus $\mathbf{T_2}$ is a tree of **G** and, by construction,

$$\mathscr{J}(\mathbf{T_3}) = \mathscr{J}(\mathbf{T_2}) + \mathscr{J}(\mathbf{uv}) - \mathscr{J}(\mathbf{u_i u_{i+1}}) < \mathscr{J}(\mathbf{T_2})$$

which contradicts the minimality of $\mathbf{T_2}$.

3. The idea of the algorithm

It is an example of a **greet-algorithm**, see 5.algorithm 2, which operates on an ordered list of edges **G** given in order of increasing length. The **greet-algorithm** is an approximation algorithm computes a spanning tree for very large graphs. The two smallest edges necessarily occur in the minimal tree. At each stage, we consider the next edge. If it does not create a cycle, we add this edge to the tree, otherwise we pass on to the next step. The **greet-algorithm** stops when all the vertices have been included. This procedure is particularly simple to implement because an edge does not create a cycle unless its two end-points belong to the same connected component. It therefore suffices for each edge remaining to ascertain the number of the connected component corresponding to each endpoint. Since we examine at most $n(n-1)/2$ edges (the number of edges in a complete graph), the complexity of this algorithm is **O(n^2)** once the values of the edges have been arranged in order of increasing value. But this preliminary procedure, if it is not required for other analyses, is rather lengthy and more rapid algorithms are known. In any case we will encounter it again when we present the algorithm for enumerating the connected components of the threshold graphs of a complete valued graph.

4. G-trees (Hedgehogs)

Let **T = (V , E)** be a tree, for example obtained by **greet-algorithm**. A **needle ɩ** of the tree **T** is a subtree **ɩ = (V$_ɩ$, E$_ɩ$)**, **V$_ɩ$** ⊆ **V**, **E$_ɩ$** ⊆ **E**, in which all the vertices $\mathbf{V_ɩ}/\{\mathbf{v(ɩ)}\ ,\ \mathbf{o(ɩ)}\}$ have a degree equal to two, **st(v(ɩ)) =1**, **st(o(ɩ))** ≥ 1. A vertex **v(ɩ)** is called the **edge** (or **top**), and a vertex **o(ɩ)** is called a **nest of needle ɩ**. A **needle ɩ** is double-edged, if **st(o(ɩ))** = 1. A **sharpening** of needle **ɩ**, denoted as **ɩ°**, is a subgraph, which coincides with **ɩ** on all edges and vertices except the nest, **o(°ɩ)** = 1. The **body** of the tree **T** is a subgraph $\mathscr{B}$(**T**), which removed all the sharpenings of all its needles. Suppose that the set {**°ɩ$_j$**}, **j**∈**J** is the set of all the sharpenings of the tree **T**. Then the pair

$$\mathbf{I(B)} = (\mathscr{B}(\mathbf{T}), \{°ɩ_j\}, \mathbf{j} \in \mathbf{J}\ ), \tag{5}$$

is called the hedgehog of the tree **T**.

In the rest of this article we will consider trees (valued or not) some of whose vertices (called real vertices) are labelled and some of which (called latent vertices) are not. The 'labels' are elements of a set **X** which is fixed once and for all. The idea is that, taking account of the data relating to the real vertices, the latent vertices play the role of 'intermediary' vertices which are required to obtain a tree structure. Before proceeding further we will indicate a number of situations in which one encounters this requirement.

Let **T = (V, E)** be a tree. Further, **D$_{max}$** denote the length of the longest edge **v*** plus **1**. At each iteration we simply adjoin a vertex outside the tree **T** (the nearest) and update the distances of the outside vertices from the tree **T**. This is implemented by a very simple data structure: we use two arrays ℵ and ℑ of dimension **n**. If the vertex **v** is in the tree, ℵ(**v**) is the number of the vertex to which **v** has been previously attached, and ℑ(**v**) = **D$_{max}$**, otherwise T(**v**) is the tree vertex closest to **v** and ℑ(**v**) is the length of the edge **v** -ℵ(**v**). At each iteration,we look for a vertex **w** such that ℑ (**w**) is minimal and we add this vertex to the tree. We then assign V(**w**) = Draax. Then for each vertex s outside the tree (for which ℑ (**s**) < **D$_{max}$** , we compare its distance from the previous tree to the length of **s** – **w**. If the latter length is smaller, we put ℵ(s) = **w** and equate ℑ (**s**) to this length.

Initially the last vertex **n** is the only one located in the tree, and it is therefore the closest vertex to all the others. At each iteration we scan the array ℵ of dimension **n** and we adjoin one vertex and one edge. After **n** – 1 iterations, the **n** – i edges in the tree are **v** – ℵ(**v**). This algorithm has complexity O(**n**^{2}). The minimal spanning tree can be stored using a file structure which is found in all of our programs. Among other things, this allows us to draw it using the methods for drawing trees that we will meet in other part of article.

Examples 1

We are concerned here with the problem of grouping the elements of a set X into **homogeneous classes** according to certain criteria. This set may be structured in various ways: by the observation or calculation of measures of proximity among its elements, using values taken by one or several variables etc. In the hierarchical model the classes so formed may be compared by order of inclusion and two incomparable classes are always disjoint. The classes can be located at the nodes of a tree whose edges represent direct comparisons in order of inclusion. The leaves of this tree are occupied by the individuals that we wish to classify (i.e. the elements of X or the species): lion, cheetah.... The nodes (i.e. the classes, orders, families and genders) therefore arise as intermediaries which are necessary to obtain a tree classification of X.

Examples 2

The biological literature abounds in **phylogenetic trees** of extremely varied types. We will only mention non-rooted trees. Figure 2 exhibits two simplified examples. Here X is the set {man, chimpanzee, gorilla, gibbon, orangutan}. Each of these trees interprets an evolutionary hypothesis. We remark that they differ from classificatory trees in two respects: while the tree in examples 1, in a natural manner, from bottom to top (corresponding to inclusion between classes) no directional reading is assumed a priori in phylogenetic trees (such a direction, if it were given, would indicate a hypothesis about the direction of evolution). On the other hand, in classification trees it is only the leaves which are labeled (since they represent the objects to be classified while the nodes correspond to the classes obtained). In phylogenetic trees, although the leaves are still labeled, some nodes may represent some known common ancestors. We remark that this type of tree is not the exclusive preserve of naturalists, it is also used in other domains, such as the filing of manuscripts, the psychology of memory etc.

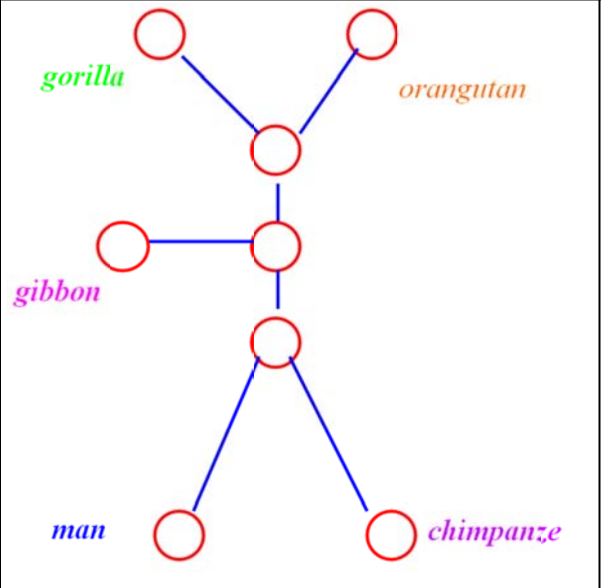

Fig. 2.

Examples 3

An **additive tree** is a tree of the phylogenetic type whose edges are valued. They correspond to the following problem: given a set X and an index of proximity 5 on X, determine a valued tree, whose set of leaves is contained in X, such that the length of the paths joining two elements of X constitute a 'good approximation' to 5. The real vertices are the elements of X; the latent vertices correspond to the nodes which need to be added in order to obtain the 'best possible' approximation. After being used in Operational Research and the Analysis of Information, the utilisation of these trees was revivified by Mathematical Psychology. Indeed, they supply a method for analysing and representing similarities in accordance with structured hypotheses (e.g. about memory organization and certain types of representations of the universe of knowledge) concerning the objects between which these similarities are evaluated. For a similar reason (evolutionary hypotheses) they are commonly used in 'systematic biology'. Picture 1 represents an 'additive' tree calculated on the basis of similarity data perceived between various perceivable.

Examples 4

These correspond originally to the following problem: given n points in the arline plane, find a tree of minimal length containing all these points. Figure 3 shows a simple example of a Steiner tree on three points. The real vertices are the n initial points, the latent vertices are those which it is necessary to add in order to obtain a tree of minimal length. This problem has many variants, for example the Steiner problem in a graph: given a valued graph (G, L) and a set X of vertices of G (the real vertices), determine a partial subtree of G whose vertices contain X and whose length is minimal. We remark that in a Steiner tree, each leaf is a real vertex.

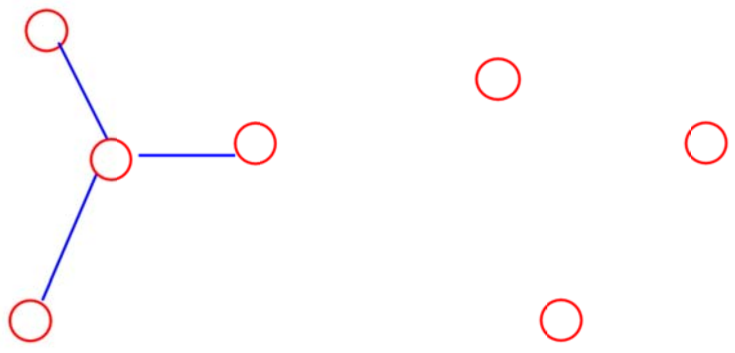

Fig. 3.

Let **O** be a finite set (discrete graph!) and **T = (V, E)**, **D = (V$_1$, E$_1$)** are trees. **I(B)** = ($\mathscr{B}$(**T**) , {ᶅ$_j$}, **j**∈**J**) is hedgehog дерева **T**. A **simple O-hedgehog T** (we will simply say — **O-hedgehog**, or a **hedgehog**, if clearly defined **O**, although in the future, the term **O-hedgehog** will mean a generalization, when **O** is an arbitrary finite graph) is a triple

$$\mathbb{P}_{\mathbb{H}}(\mathbf{T}) = (\mathbf{T}\, ,\, \mathbb{H}\, ,\, \mathbf{O}) \qquad (6)$$

formed by a tree **T** and a homomorphism ℍ from **O** to **T**

$$\mathbb{H} : \mathbf{O} \Rightarrow \mathbf{T} \qquad (7)$$

such that ℍ(**O**) are isomorphic to Ω

$$\mathbb{H}(\mathbf{O}) \approx \Omega \qquad (8)$$

where $\Omega = \bigcup\limits_{j \in J}$ ᶅ$_j$.

The homomorphism ℍ is the labeling of the $\mathbb{P}_{\mathbb{H}}$(**T**). The vertices in ℍ(**O**) are called **real** vertices and the vertices in **V** - ℍ(**O**) are called **latent** vertices.

Two **O-hedgehog** $\mathbb{P}_{\mathbb{H}}$(**T**) and $\mathbb{P}_{\mathbb{Q}}$(**D**) are **isomorphic** ⇔ if there exists a bijection φ from **V** to **V$_1$**

$$\varphi : \mathbf{V} \rightarrow \mathbf{V}_1 \qquad (9)$$

such that:

(i) an edge **<vu>** ∈ **V** ⇔ φ (**v**)φ(**u**) ∈ **V$_1$**;

(ii) $\varphi \circ \mathbb{H} = \mathbb{Q}$

where “∘” is composition of maps.

Thus two isomorphic hedgehog only differ in the 'names' given to the latent vertices; in the following we will consider them to be equal.

We distinguish between different types of **O**-hedgehog $\mathbb{P}_{\mathbb{H}}$(**T**) according to the nature of the function ℍ. When ℍ(**O**) is the set of leaves of **T**, we say that the **O**- hedgehog $\mathbb{P}_{\mathbb{H}}$(**T**) is **free**. When ℍ (**O**) is the set **V** of all vertices of **T** (i.e. when ℍ is surjective), we say that $\mathbb{P}_{\mathbb{H}}$(**T**) is **constrained**. When the homomorphism ℍ is injective (i.e. when there is no multiple labeling), we say that hedgehog $\mathbb{P}_{\mathbb{H}}$(**T**) is **separated**.

For example, the **O**-hedgehog in Figure 1 is separated, **O**-hedgehog in Figure 1 is free.

A separated, constrained **O**- hedgehog $\mathbb{P}_{\mathbb{H}}$(**T**) may simply be identified with a tree. When there is no ambiguity we may simply denote the $\mathbb{P}_{\mathbb{H}}$(**T**) by **T**.

6. Algorithm 2

The algorithm implemented below is a recursive procedure (**greet-algorithm**) due to V. V. Gritsak (1980). We start with a tree consisting of a single vertex and we adjoin the nearest neighbour to this vertex (and the edge which joins them). This subtree with two vertices is minimal. Now, if we have a minimal partial subtree and if we add to it an edge, with one endpoint in the tree and one endpoint outside it, and if this edge is minimal in the set of edges having this property, we obtain another minimal sub-tree. It is easy to show that the minimal spanning tree is obtained after *n–i* iterations.

```
10 INPUT "File name";NS : NN$=N$.".DIS"
20 OPEN "I",#I,N : INPUT #1,N :INPUT #1,N2
30 DIM D(N2) ,T%(N) ,V(N) ,L(N)
40 Ni=N-1 : FOR I=l TO N2 : INPUT $1,D(I)
50 IF D(I)>DMAX THEN DMAX=D(I)
60 NEXT I : CLOSE #1 : DMAX=DMAX+i
100 PRINT " separate vertex";NS : PRINT : PRINT .... ;
110 FOR I=l TO N1 : PRINT USING "####..;I; : NEXT I : PRINT : PRINT
120 FOR J=2 TO N : PRINT USING "##";J; : pRINT ..... ;
130 FOR I=l TO J-1 : GOSUB 2120
140 PRINT USING "###.#";D(P); : NEXT I : PRINT
150 NEXT J : PRINT
500 REM  The main algorithm
510 J=N : FOR I=1 TO N1 : T%(I)=N : GOSUB 2120
520 V(I)=D(P) : NEXT I : T%(N)=0 : SOM=0
530 FOR ITER=I TO N1 : DMIN=DMAX
540 FOR K=i TO N1 : IF V(K)<DMIN THEN DMIN=V(K) : MIN=K
550 NEXT K : V(MIN)=DMAX : SC4=SC4+DMIN : L(MIN)=DMIN
600 REM received a neighborhood tree
610 FOR K=i TO N1 : IF V(K)=DMAX THEN 640
620 I=K : J=MIN : GosUB 2110
630 IF D(P)<V(K) THEN T%(K)=MIN : V(K)=D(P)
640 NEXT K : NEXT ITER
700 PRINT " Minimal spanning tree " : PRINT
710 FOR I=1 TO N1 : PRINT "Edge ";I; ...... ;T%(I);
720 PRINT " length ";L(I) : NEXT I : PRINT
730 PRINT " the sum of the lengths of edges ";Sf4
740 PRINT : GOSUB 3400 : END
2100 REM P := (I,J)
2110 IF I>J TEN S$%P I,J
2120 P=(I-1)*(N-I/2)+J-I : RETURN
3400 INPUT " You get a tree (Y/N) ";CS
3405 IF C$<>"Y" THEN RETURN
3410 INPUT " File name ";NS : NS=NS+".ARB"
3420 OPEN "O",#1,N$ : PRINT #1,N : FOR I=1 TON1
3430 PRINT #l,I : PRINT #1,T%(I) : PRINT #i,L(I)
3440 NEXT I : CLOSE #1 : RETURN
```

Статья поступила в редакцию 24.06.2011 г.

Гритсак-Грёнер В. В., Гритсак-Грёнер Ю.

Математические методы идентификации визуальной информации. 2.

Мы продолжаем шлифование теории графов-деревьев, начатое в первой статье этой серии. В этой статье мы будем считать, что граф связный. Построим минимальный лес, состоящий объединения минимальных остовных деревьев его связных компонент. Из-за важности этих понятий в исследованиях операций и для анализа данных, предложены алгоритмы для построения минимального остовного дерева взвешенного графа.

Ключевые слова: граф, изображение, визуальная информация.