DIMACS Technical Report 98-15
February 1998

Bellman Functions on Trees for Segmentation, Generalized Smoothing, Matching and Multi-Alignment in Massive Data Sets

by

Ilya Muchnik
DIMACS, Rutgers University
P.O. Box 8018, Piscataway, NJ 08854-8018, USA
muchnik@dimacs.rutgers.edu

Vadim Mottl
Tula State University, Lenin Ave. 92, 300600 Tula, Russia
mottl@atm.tsu.tula.ru

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, Bellcore and NEC Research Institute.

DIMACS is an NSF Science and Technology Center, funded under contract STC-91-19999; and also receives support from the New Jersey Commission on Science and Technology.

Abstract

A massive data set is considered as a set of experimentally acquired values of a number of variables each of which is associated with the respective node of an undirected adjacency graph that presets the fixed structure of the data set. The class of data analysis problems under consideration is outlined by the assumption that the ultimate aim of processing can be represented as a transformation of the original data array into a secondary array of the same structure but with node variables of, generally speaking, different nature, i.e. different ranges. Such a generalized problem is set as the formal problem of optimization (minimization or maximization) of a real-valued objective function of all the node variables. The objective function is assumed to consist of additive constituents of one or two arguments, respectively, node and edge functions. The former of them carry the data-dependent information on the sought-for values of the secondary variables, whereas the latter ones are meant to express the a priori model constraints. For the case when the graph of the pair-wise adjacency of the data set elements has the form of a tree, an effective global optimization procedure is proposed which is based on a recurrent decomposition of the initial optimization problem over all the node variables into a succession of partial problems each of which consists in optimization of a function of only one variable like Bellman functions in the classical dynamic programming, therefore, such intervening functions are called here extended Bellman functions. Two kinds of numerical realization of the basic optimization procedure are considered on the basis of parametric representation of the extended Bellman functions, respectively, for discretely defined and quadratic node and edge functions. The proposed theoretical approach to the analysis of massive data sets is illustrated with its applications to the problems of segmentation, smoothing, fine texture analysis and matching of visual images and geophysical explorative data, as well as to the problem of multi-alignment of long molecular sequences.

Contents

1 Introduction

Despite the dramatic variety of application problems concerned with the necessity to analyze massive data sets, it is possible to set off some single subclasses of problems which allow for treating them in unified terms of respective standard mathematical optimization problems for which there exist or can be created effective methods of solving.

One of such classes of data analysis problems is addressed in this paper. The class under consideration embraces, at least, visual images, in particular, satellite photographs, geophysical explorative data, massive sets of molecular sequences.

We will consider a data set $\mathbf{Y}$ as an experimentally acquired function y_t, $t \in T$, which takes values from an appropriate range $y \in \mathcal{Y}$, where index (argument) $t \in T$ runs through all the elements of the data set.

What is in common for the above mentioned kinds of data sets is the fact that the elementary information units y_t are ordered along the axes of one t or several arguments $\mathbf{t} = (t_1, \ldots, t_k)$, so that the data collection subject to an analysis as a whole is rather an array than set. Each element of such an array is naturally associated with a number of its neighbors which may be defined in several ways depending on the nature of the particular data source, but in any case a neighborhood relation on the set of data elements $t \in T$ is an inherent feature of the data array. It is convenient to express the neighborhood relation in the form of an undirected graph G which is understood as set of neighboring element pairs $G \subset T \times T$.

In this work, we restrict our concern to only those of data processing problems which can be represented as problems of transforming the original array $\mathbf{Y} = (y_t, t \in T)$, $y \in \mathcal{Y}$, into another function $\mathbf{X} = (x_t, t \in T)$, which would be defined on the same set of the array elements $t \in T$ and take values $x \in \mathcal{X}$ from a set specific for each particular problem.

We consider here a generalized approach to such a class of data analysis problems. The approach leans upon the natural and commonly adopted way to create algorithms of data analysis of practically any kind as those of optimization of a suitable objective function defined on a set which is meant to represent the whole variety of feasible results. For the class of problems under consideration, the result of data analysis is the sought-for secondary data array $\mathbf{X} = (x_t,\ t \in T)$, so, it must occur as argument in an objective function $J(\mathbf{X}|\mathbf{Y})$ that should assess the discrepancy between each admissible version of the result $\mathbf{X}$ and the given data array $\mathbf{Y}$. The range of definition of the objective function is completely determined once the range of the elementary variables $x_t \in \mathcal{X}$ is chosen. Thus, the result of data analysis $\hat{\mathbf{X}}(\mathbf{Y})$ is defined by the condition

$$\hat{\mathbf{X}}(\mathbf{Y}) = \arg \min_{x_t \in \mathcal{X}} J(\mathbf{X}|\mathbf{Y}).$$

Our aims will be, *first*, to outline a class of objective functions which, being consistent with the essence of data processing problems of the class under consideration, would lend themselves to an easy algorithmical solution, *second*, to study in detail the mathematical properties of objective functions of the defined class, *third*, to use the specificity of the chosen class for creating an effective generalized optimization procedure, and, *fourth*, to construct data processing algorithms or, at least, to point the ways how to construct them, for the particular applications in the above mentioned areas of image analysis, processing of seismic explorative data sets, and multi-alignment of molecular sequences.

The specificity of the optimization problems under consideration is that the variables constituting the vector to be evaluated $\mathbf{X} = (x_t,\ t \in T)$ are associated with the nodes of the neighborhood graph G on the set of data array elements T, and that the objective function $J(\mathbf{X}|\mathbf{Y})$ is separable in accordance with this graph, i.e. is sum of partial functions of one and two variables associated with, respectively, nodes and edges of graph G which represents the adjacency of variables in the structure of the objective function.

The computational complexity of the minimization problem substantially depends on the kind of the adjacency graph G. The fact itself that the objective function is separable does not contribute essentially to an improvement of the computational solvability of the optimization problem in comparison with that for an arbitrary function of many variables. An intent to create a non-iterative procedure of finding the global minimum point of a separable objective functions supported by a graph of general kind comes up against a fundamental barrier.

But in the particular case, when graph G on the set of variables T has no cycles, i.e. is a tree, there exists a highly effective global optimization procedure based on a recurrent decomposition of the initial problem of optimization of a function of $|T|$ variables, where $|T|$ is the number of elements in set T, into a succession of $|T|$ elementary problems, each of which consists in optimization of a function of only one variable. In the particular case, when tree G has no branching nodes, i.e. is a chain, the procedure is nothing else than one of the versions of the classical dynamic programming procedure. The elementary functions of one variable to be minimized at each step of minimization of a tree-supported separable function play the same role as the Bellman functions in the classical procedure and are called here extended Bellman functions.

It is just the tree-like separability of the objective functions which is systematically exploited in this work. But the typical neighborhood relation on the usual rectangular set of data array elements, that naturally arises in the data analysis problems, has the form of a lattice, and leads to the lattice-like variable adjacency graph, entirely consisting of cycles. To overcome this obstacle, we decompose the original lattice-like neighborhood relation into several tree-like ones each of which covers, nevertheless, all the elements of the data array. Respectively, instead of the overall objective function, we use several tree-supported partial functions to evaluate the goal variables at each single row of the original rectangular lattice of the neighborhood between data array elements.

The report consists of 4 Sections. First of all, in the next Section 2, we preliminarily study the few number of particular data analysis problems which appeared to the authors to bear essential common traits, typical for a much wider circle of applications, and to be worth a generalization. The problems of processing visual images and seismic explorative data arrays have been a major focus of the authors' theoretical and practical activities for the recent years, whereas the problem of multi-alignment of molecular sequences is preliminarily analyzed here as a glowing example of potential applications. Then, in Section 3, we consider theoretical and computational questions related to the problem of optimization of separable functions with tree-like separability. Finally, in Section 4, we create a generalized data analysis procedure and consider the ways of adjusting it to the particular application problems.

2 Optimization-based approach to data analysis

2.1 Some particular data analysis problems

The problems of the analysis of visual images, at least, a great part of them, appear to form the most evident class of data analysis problems of the kind under consideration. The original data set, i.e. original image, is a numerical function $\mathbf{Y} = (y_{\mathbf{t}},\, \mathbf{t} = (t_1, t_2) \in T)$, usually, darkness, which takes values from an appropriate set $y_{\mathbf{t}} \in \mathcal{Y}$ and is defined on the discrete two-dimensional grid $T = \{\mathbf{t} = (t_1, t_2)\colon\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2\}$ of image elements $\mathbf{t} = (t_1, t_2)$ usually called pixels. It is typical for low-level image processing problems that the desired result of processing is meant as another function $\mathbf{X} = (x_{\mathbf{t}},\, \mathbf{t} \in T)$ which would be defined on the same pixel grid $\mathbf{t} \in T$ and take values $x_{\mathbf{t}} \in \mathcal{X}$ from a set specific for each particular problem [20].

In the problem of segmentation of an original image into homogeneous regions of a preset number of classes (Fig. 2.1), the class allocation of the image pixels will be naturally expressed by a variable $x_{\mathbf{t}}$ taking values from a finite set of class indices $\mathcal{X} = \{1, \ldots, m\}$. So, the segmentation of the image into a set of classified regions can be sought for in the form of a class allocation map $\mathbf{X} = (x_{\mathbf{t}},\, \mathbf{t} \in T)$, $x_{\mathbf{t}} \in \mathcal{X} = \{1, \ldots, m\}$, actually, a secondary finitely valued image.

In the problem of smoothing (Fig. 2.2), the resulting function $\mathbf{X} = (x_{\mathbf{t}},\, \mathbf{t} \in T)$ is of the same nature as the original image, i.e. it is a picture. If we assume $\mathbf{X}$ to be real-valued $x_{\mathbf{t}} \in \mathsf{R}$, the set $\mathcal{X}$ the secondary picture takes values from remains the same $\mathcal{X} = \mathcal{Y} = \mathsf{R}$.

The problem of matching (Fig. 2.3) consists in finding identical points in two images of similar structure. Let two images $\mathbf{Y}^l = (y^l_{\mathbf{t}},\, \mathbf{t} \in T)$ and $\mathbf{Y}^r = (y^r_{\mathbf{t}},\, \mathbf{t} \in T)$, conventionally referred to as the left and the right one, have the same range of definition T and form an entire image pair $\mathbf{Y} = (\mathbf{Y}^l, \mathbf{Y}^r)$ to be analyzed as a whole, for instance, photos of two human faces taken at the same scale, or a stereo pair of images, and the like. Whatever goal might be pursued by the analysis of a pair of similar images, the first and algorithmically most difficult stage is matching, i.e. establishing a correspondence in the two images between the point pairs which could be treated as identical. The result of such an operation is usually represented in the form of the so-called disparity map $\mathbf{X} = (\mathbf{x}_{\mathbf{t}},\, \mathbf{t} \in T)$ which, for each point $\mathbf{t}$ of the one of the images (for instance, the left

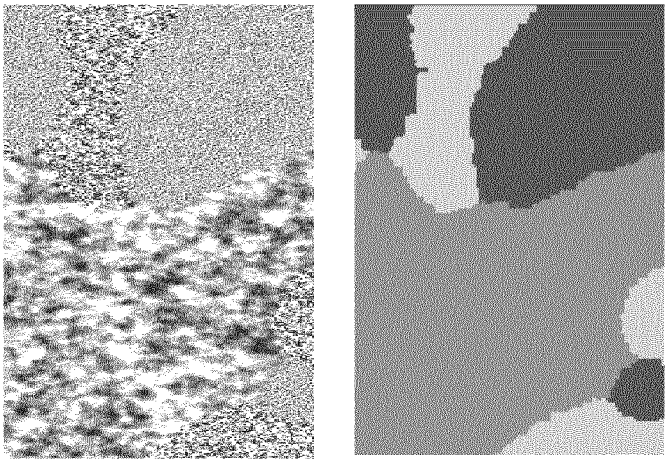

Figure 2.1: A texture image and its segmentation into homogeneous regions of three classes.

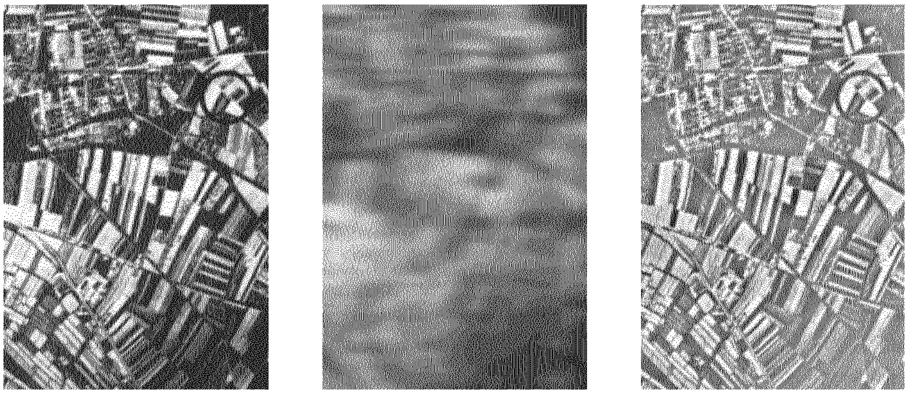

Figure 2.2: The original satellite photograph, smoothed one, and the result of darkness equalization by subtracting the smoothed picture from the original.

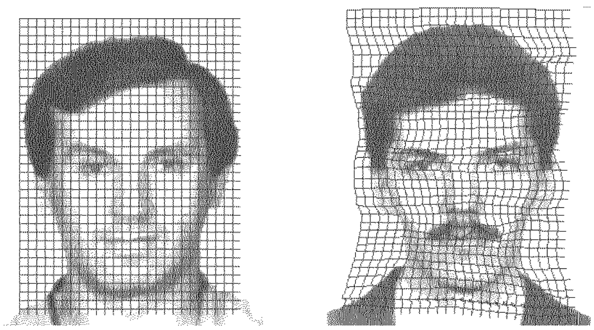

Figure 2.3: A pair of human portraits and disparity map of their identical points.

one taken as reference image) determines the vector $\mathbf{x}_\mathbf{t} = (x_\mathbf{t}^{(1)},\ x_\mathbf{t}^{(2)})$ indicating the spatial shift between $\mathbf{t}$ and its match $\mathbf{t}' = \mathbf{t} + \mathbf{x}_\mathbf{t}$ in the second (search) image. It is convenient to allow $\mathbf{x}_\mathbf{t}$ to take values from the two-dimensional real space $\mathcal{X} = \mathsf{R}^2$, and, thereby, permit inter-pixel interpolation in the discrete image grid T [18].

The geophysical explorative data sets, seismic sections and cubes, are a class of, respectively, two- and three-dimensional data arrays, which are analyzed in the course of gas and oil reserves prospecting with the purpose of studying the structure of the underground rock mass.

A two-dimensional seismic data array, the so-called *seismic section*, consists of synchronous records of reflected seismic signals $\mathbf{y}_{t_1} = (y_{t_1,t_2},\ t_2 = 1, \ldots, N_2)$ registered as real-valued functions of time t_2 by a large number of geophones (seismic sensors) $i = 1, \ldots, N_1$ placed along a straight line on the earth surface. In spite of the non-visual origin of seismic data, it is the practice to represent them in a pictorial form, frequently, by displaying the signal value at each point of the array as darkness level of the respective point in the image (Fig. 2.4,a). As the source of the initial seismic pulse, usually serves a series of explosions, responses to which are averaged in a special manner. Being placed side-by-side, the seismograms form the vertical rows of a two-dimensional data array $\mathbf{Y} = (\mathbf{y}_{t_1},\ t_1 = 1, \ldots, N_1) = (y_\mathbf{t},\ \mathbf{t} = (t_1, t_2) \in T)$, $y_\mathbf{t} \in \mathcal{Y} = \mathsf{R}$, $T = \{\mathbf{t} = (t_1, t_2)\colon\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2\}$. Despite the fact that the initial argument t_2 of seismic signals is time, a relevant preprocessing allows to identify the vertical axis with depth under the respective sensor, so that such a picture offers a vertical section through the undersurface medium.

Local peculiarities of the mode of spatial signal oscillations around each point of the seismic section are an important source of information for the judgment about the hidden structure of the rock mass. One of the aims of the data analysis is an objectification of local similarities and differences in the texture of the seismic picture, often, hardly discernible by the naked eye. In accordance with such an aim, the resulting secondary data array $\mathbf{X} = (x_\mathbf{t},\ \mathbf{t} \in T)$ should consist of estimates of a numerical $x_\mathbf{t} \in \mathcal{X} = \mathsf{R}$ or vector parameter $x_\mathbf{t} = \mathbf{x}_\mathbf{t} \in \mathcal{X} = \mathsf{R}^n$ of a local model of signal oscillation around each point of the seismic section (Fig. 2.4,b).

Usually, the ultimate goal of the analysis of a seismic section is its segmentation into a finite totality of homogeneous regions, interpreted jointly as a geological model of the rock mass. In this case, the final result will be of the same kind as in the above mentioned problem of segmentation of visual images (Fig. 2.1), i.e. a class allocation map $x_\mathbf{t} \in \mathcal{X} = \{1, \ldots, m\}$.

A seismic cube differs from a seismic section in that the seismic sensors are placed in the nodes of a rectangular lattice on the earth surface, so that the reflected seismic signals are ordered along two coordinate axes $\mathbf{y}_{t_1 t_2}$, $t_1 = 1, \ldots, N_1$, $t_2 = 1, \ldots, N_2$, and form a three-dimensional data array $\mathbf{Y} = (y_\mathbf{t},\ \mathbf{t} = (t_1, t_2, t_3) \in T)$, $T = \{(t_1, t_2, t_3)\colon\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2,\ t_3 = 1, \ldots, N_3\}$ (Fig. 2.5). Respectively, the result of the analysis, that pursues the same aims, will be a three-dimensional array $\mathbf{X} = (x_\mathbf{t},\ \mathbf{t} = (t_1, t_2, t_3) \in T)$, elements

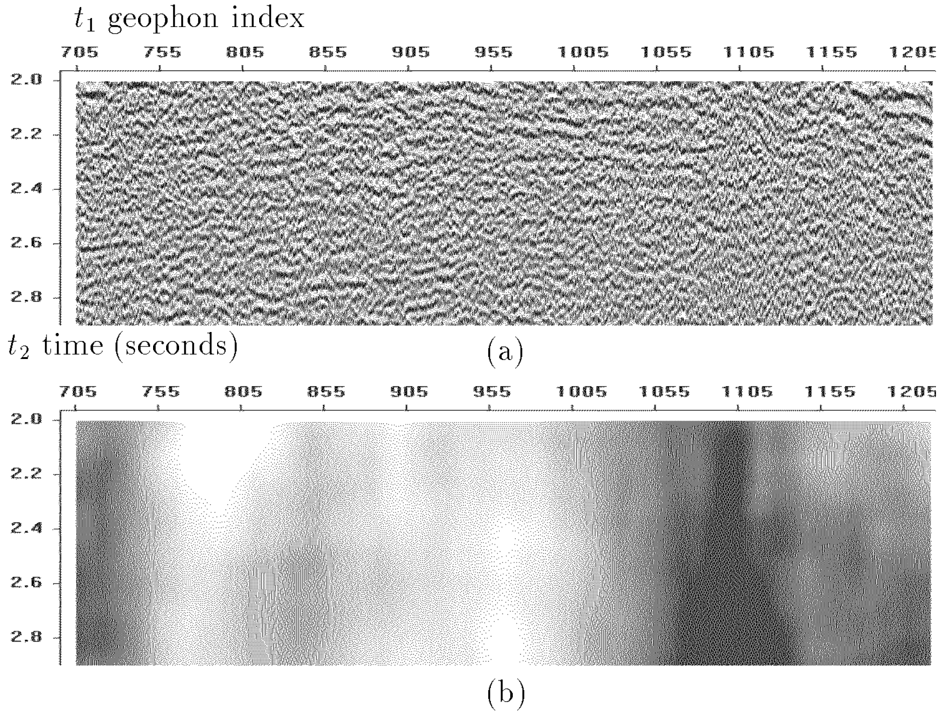

Figure 2.4: Fragment of a vertical seismic section through a granite rock mass (a), and map of the averaged local energy of signal oscillations (the darker, the greater) in a narrow band of horizontal spatial frequency (b). This local texture parameter of the seismic image of a solid body is considered as an indicator of its fracturedness around the respective point in the underground space.

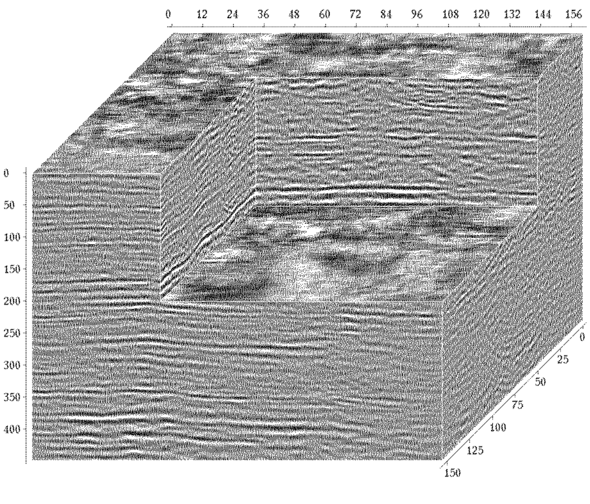

Figure 2.5: Fragment of a seismic cube.

of which have the same sense whether of estimates on the parameter of an appropriate local spatial oscillation model of the reflected seismic signals, in this case, in three dimensions, or of indices of class-membership in the estimated segmentation.

In the problem of multi-alignment of a set of proteins each row of the original data array is a sequence of amino acids forming the particular protein $\mathbf{y}_{t_2} = (y_{t_1 t_2},\ t_1 = 1, \ldots, N_{t_2})$ in the set $\mathbf{Y} = (\mathbf{y}_{t_2},\ t_2 = 1, \ldots, N_2)$ to be analyzed. Such a data set has the structure of a two-dimensional array $\mathbf{Y} = (y_{\mathbf{t}},\ \mathbf{t} = (t_1, t_2) \in T)$, $T = \{\mathbf{t} = (t_1, t_2)\colon t_1 = 1, \ldots, N_{t_2},\ t_2 = 1, \ldots, N_2\}$ whose rows are, as a rule, not completely filled up at the right because of the difference in the length N_{t_2} of amino acid sequences, as it is shown in Fig. 2.6,a. There exist strictly 20 amino acids usually denoted by capital Latin letters

$$\mathcal{Y} = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}, \tag{2.1}$$

so, the elements of the original array take values from the alphabet of amino acid symbols $y_{\mathbf{t}} = y_{t_1 t_2} \in \mathcal{Y}$.

The problem of multi-alignment is underlaid by the supposition that the particular proteins in set $\mathbf{Y}$ are homologous to each other, i.e. have their origin in a common progenitor from which they diverged, through the natural course of evolution, by different chains of amino acid substitutions, deletions and insertions without, however, affecting the basic structure [16]. The aim of multi-alignment is to find the minimal number of elementary transformations of these three kinds by which the proteins can be mutually interconverted, in other words, aligned, brought to a common length, and, therethrough, to show to the researcher what is in common in the analyzed set of proteins from the viewpoint of their amino-acid makeup.

Fig. 2.6,b shows the result of multi-alignment in the form of the collection of the amino acid sequences rearranged to a common length $N_1 \geq \max\limits_{t_2 = 1, \ldots, N_2} N_{t_2}$ (do not mind, so far, the separate sequence at the bottom marked as $\mathbf{c}$). The mutual warping of the original sequences is represented by the use of the extended alphabet

$$\bar{\mathcal{Y}} = \mathcal{Y} \bigcup \{\sqcup\}$$

that includes, in addition to the basic symbols of amino acids, also the extra null symbol $\sqcup$ meant to denote eventual gaps between actual amino acids. The problem consists in finding such an arrangement of gaps that provides the most similarity, in some sense, of the amino acids getting in each particular column of the alignment. The form of the resulting two-dimensional array $\mathbf{X} = (x_{\mathbf{t}}, \mathbf{t} = (t_1, t_2) \in \bar{T})$ is strictly rectangular $\bar{T} = \{\mathbf{t} = (t_1, t_2)\colon\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2\}$, in contrast to that of the original set of proteins $T = \{\mathbf{t} = (t_1, t_2)\colon\ t_1 = 1, \ldots, N_{t_2},\ t_2 = 1, \ldots, N_2\}$. The elements of the resulting array take values from the extended amino acid alphabet $x_{\mathbf{t}} \in \mathcal{X} = \bar{\mathcal{Y}}$.

One of the widely adopted strategies of multi-alignment is based on the notion of the pattern sequence $\mathbf{c} = (c_{t_1},\ t_1 = 1, \ldots, N_1)$ of the same length as the alignment itself. Each element of the pattern sequence corresponds to a particular column and is meant

(a)

$y_{t_1t_2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
					t_1 original positions of amino acids in proteins															
1	T	L	T	P	L	S	T	K	D	L	T	T	Y	.	.	.	.	.	.	.
2	L	A	P	L	S	E	K	G	L	T	T	Y	G	F	G	R	Q	N	.	.
3	A	V	T	A	E	R	A	S	L	I	T	T	G	P	Y	G	H	O	S	.
4	G	V	T	E	G	R	T	D	I	T	T	M	R	T	T	G	A	F	.	.
t_2 5	G	V	T	T	T	R	S	N	I	T	T	T	G	A	F	G	Q	Q	.	.
6	T	E	P	V	I	K	K	R	K	G	D	I	K	S	Y	.	.	.	.	.
7	T	E	P	V	I	K	K	R	K	G	D	I	K	S	Y	.	.	.	.	.
8	D	R	S	I	Q	T	A	I	V	T	R	P	I	I	T	T	A	G	.	.
9	T	G	D	V	T	T	A	I	V	R	R	N	T	I	T	T	A	.	.	.
proteins																				

(b)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
						t_1 positions in the alignment														
1	T	␣	L	T	P	L	S	T	K	D	L	T	T	Y	␣	␣	␣	␣	␣	␣
2	␣	␣	L	A	P	L	S	E	K	G	L	T	T	Y	G	F	G	R	Q	N
3	A	␣	V	T	A	E	R	A	S	L	I	T	T	G	P	Y	G	H	Q	S
4	␣	␣	G	V	T	E	G	R	T	D	I	T	T	M	R	T	T	G	A	F
t_2 5	␣	␣	G	V	T	T	T	R	S	N	I	T	T	T	G	A	F	G	Q	Q
6	␣	␣	T	E	P	V	I	K	K	R	K	G	D	I	K	S	Y	␣	␣	␣
7	␣	␣	T	E	P	V	I	K	K	R	K	G	D	I	K	S	Y	␣	␣	␣
8	␣	␣	D	R	S	I	Q	T	A	I	V	T	R	P	I	I	T	T	A	G
9	T	G	D	V	T	T	A	I	V	R	R	N	T	I	T	T	A	␣	␣	␣
proteins																				
$\mathbf{c}$	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}

(c)

$c_{t_1t_2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
											t_1									
1	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
2	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
3	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
4	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
t_2 5	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
6	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
7	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
8	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
9	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}

$k_{t_1t_2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	$1\frac{1}{2}$	2	3	4	5	6	7	8	9	10	11	12	13	$13\frac{1}{2}$	$13\frac{1}{2}$	$13\frac{1}{2}$	$13\frac{1}{2}$	$13\frac{1}{2}$	$13\frac{1}{2}$
2	$\frac{1}{2}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
3	1	$1\frac{1}{2}$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
4	$\frac{1}{2}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
t_2 5	$\frac{1}{2}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
6	$\frac{1}{2}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$15\frac{1}{2}$	$15\frac{1}{2}$	$15\frac{1}{2}$
7	$\frac{1}{2}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$15\frac{1}{2}$	$15\frac{1}{2}$	$15\frac{1}{2}$
8	$\frac{1}{2}$	$\frac{1}{2}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$15\frac{1}{2}$	$15\frac{1}{2}$	$15\frac{1}{2}$
9	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	$17\frac{1}{2}$	$17\frac{1}{2}$	$17\frac{1}{2}$

Figure 2.6: Original set of protein fragments (a) and a version of its multi-alignment (b) based on the estimated pattern sequence $\mathbf{c}$ of the appropriate kind. The same multi-alignment (c) as two-component array that represents both the resulting approximation of the protein set by the many times repeated pattern sequence and the original positions of amino acids in each protein. Fractional numbers point at the imaginary gaps between amino acids including the positions immediately before the first and after the last of them.

to express the common properties of the amino acids in it. The elements of the pattern sequence take values from an appropriate set $c_i \in \mathcal{C}$ that may be different from the set $\mathcal{Y}$ of the original amino-acids and have the sense, for instance, of frequency distributions over $\mathcal{Y}$ or $\bar{\mathcal{Y}}$ [12, 15], functional descriptions of amino acid properties [12, 17], or subclasses of amino acids [16]. In accordance with the pattern-based strategy, the score of the mutual similarity of amino acids in a column should take into account the totality of their individual similarities to the respective pattern sequence element. The pattern sequence is considered as an integral part of the alignment, and is to be found along with the references in the columns to the original amino acid positions in the proteins.

Below, in Section 4.4, we will study a more general version of the pattern-based multi-alignment. In this connection, it is convenient to alter, a little, the representation of the alignment result in order to highlight the identity of the mathematical essence of the problem to that of other data analysis problems considered in this work.

Mark, that the pattern-based statement of the multi-alignment problem is aimed at finding such a pattern sequence which would be the best approximation of each of the proteins simultaneously. Thus, being repeated as many times as there are proteins under processing, the pattern sequence will form an array $\mathbf{C} = (c_{t_1 t_2},\ (t_1, t_2) \in \bar{T})$, where $c_{t_1 t_2} = \text{const}$ for all $t_2 = 1, \ldots, N_2$, which will be the best approximation of all the proteins at once among the arrays with identical elements in columns (the upper array in Fig. 2.6,c). It is natural to call such an array the pattern array of the protein set.

Each element $c_{t_1 t_2}$ of the pattern array is to be coupled with the reference $k_{t_1 t_2}$ to a particular position in the original amino acid sequence (protein) $\mathbf{y}_{t_2} = (y_{k t_2},\ k = 1, \ldots, N_{t_2})$. If the symbol at position (t_1, t_2) in the alignment is that of an actual amino acid, the reference $k_{t_1 t_2}$ takes the whole value equal to the number of the respective amino acid in the original sequence. If, otherwise, the position is occupied by null symbol $\sqcup$, it means that the alignment suggests insertion of a gap between two adjacent amino acids, let them have numbers k and $k+1$. In this case the reference $k_{t_1 t_2}$ takes fractional value $k\frac{1}{2}$ that points at the particular interface between amino acids in the original protein. If several gaps succeed each other in a row of the alignment, all of the respective references will have the same fractional value pointing at the same amino acid interface. In accordance with this rule, gaps before the first and after the last amino acid of the protein $\mathbf{y}_{t_2}$ will be indicated by references, respectively, $k_{t_1 t_2} = \frac{1}{2}$ and $k_{t_1 t_2} = N_{t_2}\frac{1}{2}$.

The range of the variable $k_{t_1 t_2}$ is set $\mathcal{K} = \{\frac{1}{2}, 1, 1\frac{1}{2}, 2, 2\frac{1}{2}, \ldots\}$. It is clear, that references in a row of the alignment cannot decrease. Moreover, their succession must obey quite rigorous restrictions: immediately adjacent fractional references are equal to each other, any whole reference following another whole one is greater than its predecessor by 1, any fractional reference having a whole predecessor is greater than it by $\frac{1}{2}$, and the same is valid for a whole reference immediately following a fractional one.

The totality of all the references makes the reference array $\mathbf{K} = (k_{t_1 t_2},\ (t_1, t_2) \in \bar{T})$, $k_{t_1 t_2} \in \mathcal{K}$, that constitutes, along with the pattern array $\mathbf{C} = (c_{t_1 t_2},\ (t_1, t_2) \in \bar{T})$, the result of multi-alignment (the lower array in Fig. 2.6,c). It is convenient to consider both variables $c_{t_1t_2}$ and $k_{t_1t_2}$ jointly as components of a combined vector variable $x_{t_1t_2} = (c_{t_1t_2}, k_{t_1t_2}) \in \mathcal{X}$ that will take values from set $\mathcal{X} = \mathcal{C} \times \mathcal{K}$. In these terms, the complete result of the alignment will be represented as unified array $\mathbf{X} = (x_{t_1t_2},\ (t_1, t_2) \in \bar{T})$.

2.2 Generalized separable objective function supported by neighborhood relation on the set of data array elements

As we see, all the above depicted particular data analysis problems allow for treating them as those of transforming the original array $\mathbf{Y} = (y_{\mathbf{t}},\ \mathbf{t} \in T)$ of experimentally acquired values of some variables $y_{\mathbf{t}} \in \mathcal{Y}$ into a resulting array $\mathbf{X} = (x_{\mathbf{t}},\ \mathbf{t} = (t_1, t_2) \in T)$, as a rule, of the same structure, whose variables would take values from an appropriate set $x_{\mathbf{t}} \in \mathcal{X}$ specific for each particular application.

The principal concept of the optimization-based approach to data analysis consists in constructing the algorithm of such a processing as that of optimization, conventionally, minimization, of an objective function over all the feasible combinations of values of the variables forming the resulting array $\mathbf{X}$. The objective function is meant to serve as a means of coordinating two kinds of information on the sought-for array, namely, the data-dependent information and the model-based constraints.

As a rule, the essence of the particular data analysis problem suggests a way to preliminarily judge, in a fuzzy form, about the value of the single hidden variable $x_{\mathbf{t}}$ from a vicinity $\mathbf{Y}_{\mathbf{t}}$ of the respective point in the original array $\mathbf{Y}$. It is convenient to express such a judgment in the form of a data-dependent function $\psi_{\mathbf{t}}(x) = \psi_{\mathbf{t}}(x|\mathbf{Y}_{\mathbf{t}})$ which must be defined over the set $\mathcal{X}$ and take the greater value the more evident is the contradiction between the hypothesis that $x \in \mathcal{X}$ is just the "correct" local value we are seeking and the current fragment $\mathbf{Y}_{\mathbf{t}}$ of the data array. Let such functions be called node functions, because each of them is associated with a node of the neighborhood graph on the set of data array elements.

However, practically inevitable interferences of various kinds usually prevent from taking the totality of locally optimal values

$$\hat{x}_{\mathbf{t}}(\mathbf{Y}_{\mathbf{t}}) = \arg\min_{x_{\mathbf{t}} \in \mathcal{X}} \psi_{\mathbf{t}}(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}}),\ \ \mathbf{t} \in T,$$

as the final decision on the sought-for secondary array $\hat{\mathbf{X}}(\mathbf{Y}) = (\hat{x}_{\mathbf{t}}(\mathbf{Y}_{\mathbf{t}}),\ \mathbf{t} \in T)$.

But the same essence of the particular application problem suggests, as a rule, also a model knowledge on the expected combination of values of the "hidden" variables forming the sought-for resulting array. This knowledge can be often expressed in local terms as a model-based function $\gamma_{\mathbf{t}'\mathbf{t}''}(x_{\mathbf{t}'},\ x_{\mathbf{t}''})$ which is defined over $\mathcal{X} \times \mathcal{X}$ at each pair of neighboring elements of the data array and takes the greater value the greater is the discrepancy, in accordance with the accepted model, between the respective neighboring hidden values. These functions will be referred to as edge functions, since each of them is associated with an edge of the neighborhood graph G.

Given a data array $\mathbf{Y}$, the decision on the secondary array $\mathbf{X} = (x_{\mathbf{t}},\ \mathbf{t} \in T)$ as a whole should be sought for as a compromise between the mutually contradictory requirements to minimize both node and edge functions that are, in the general case, different at different nodes and, respectively, edges. It is natural to take the sum

$$J(\mathbf{X}|\mathbf{Y}) = \sum_{\mathbf{t}\in T} \psi_{\mathbf{t}}(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}}) + \sum_{(\mathbf{t}',\mathbf{t}'')\in G} \gamma_{\mathbf{t}',\mathbf{t}''}(x_{\mathbf{t}'}, x_{\mathbf{t}''}) \tag{2.2}$$

as a combined criterion of decision making:

$$\hat{\mathbf{X}}(\mathbf{Y}) = \arg\min_{x_t \in \mathcal{X}} J(\mathbf{X}|\mathbf{Y}). \tag{2.3}$$

As we see, the criterion to be minimized is a sum of items each of which is function of only one or two variables. The arguments of the criterion along with data-dependent functions $\psi_{\mathbf{t}}(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}})$ are associated with elements of the data array, i.e. with nodes of the neighborhood graph G (node functions), whereas the functions of two arguments $\gamma_{\mathbf{t}'\mathbf{t}''}(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ are assigned to edges of the graph (edge functions). We will call an objective function of such a kind to be separable, pairs of its variables occurring in summands $\gamma(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ to be adjacent, and the graph G, that determines the structure of the function, its adjacency or supporting graph. We will also say that an objective function of kind (2.2) is a separable G-supported function. The number of its variables is equal to the number $|T|$ of nodes in G, i.e. the number of elements in the data array.

The manner in which the arguments of the separable objective function are adjacent to each other, i.e. the form of graph G, depends on the accepted local model of the hidden field $\mathbf{X} = (x_{\mathbf{t}}, \mathbf{t} \in T)$ to be recovered. In most cases, when the elements of the data array are associated with points of the actual or an imaginary space and assumed to be arranged at regular intervals along the respective coordinate axes, the model assumptions on the hidden field boil down to those of its smoothness, i.e. presuppose relatively small distinctions between values of the hidden variables at neighboring points. In particular, visual images (Fig. 2.1-2.3) and geophysical data arrays (Fig. 2.4, 2.5) relate just in this kind of data.

In the problems of visual data analysis, the natural neighborhood relation between pixels of an image and, thus, the appropriate adjacency graph of the separable objective function is rectangular lattice (Fig. 2.7,a). The same neighborhood relation remains to be consistent with the nature of plane seismic sections. So, for this class of data analysis problems the summands in the objective function (2.2) will be double sums over the nodes of the lattice

$$J(\mathbf{X}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2} \psi(x_{t_1 t_2}|\mathbf{Y}_{t_1 t_2}) + \sum_{t_1=1}^{N_1}\sum_{t_2=2}^{N_2} \gamma'(x_{t_1 t_2 - 1}, x_{t_1 t_2}) + \sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2} \gamma''(x_{t_1-1, t_2}, x_{t_1 t_2}).$$

The edge functions at horizontal and vertical edges are taken different, what is especially necessary for seismic sections, in which the respective axes have different physical meanings, but may turn out to be reasonable also for visual images.

Seismic cubes differ from sections only by the presence of an additional horizontal coordinate, and the natural form of the neighborhood relation and adjacency graph for them will be three-dimensional rectangular lattice (Fig. 2.7,b). The structure of the objective function remains the same, just becomes a little more bulky:

$$J(\mathbf{X}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\sum_{t_3=1}^{N_3}\psi(x_{t_1t_2t_3}|\mathbf{Y}_{t_1t_2t_3}) + \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\sum_{t_3=2}^{N_3}\gamma'(x_{t_1t_2t_3-1}, x_{t_1t_2t_3}) +$$
$$\sum_{t_1=1}^{N_1}\sum_{t_2=2}^{N_2}\sum_{t_3=1}^{N_3}\gamma''(x_{t_1t_2-1,t_3}, x_{t_1t_2t_3}) + \sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2}\sum_{t_3=1}^{N_3}\gamma''(x_{t_1-1,t_2t_3}, x_{t_1t_2t_3}).$$

We use here the same edge function in both horizontal directions t_1 and t_2 because of their identical physical meaning.

But as to arrays of molecular sequences (Fig. 2.6), the natural geometric notions are insufficient for discussion about the suitable adjacency graph on the set of their elements, namely, of amino acids forming the proteins which, in their turn, make the array to be analyzed. To be exact, the adjacency graph is to be defined on the elements of the resulting array $\mathbf{X} = (\mathbf{x_t} = (c_\mathbf{t}, k_\mathbf{t}),\ \mathbf{t} = (t_1, t_2) \in T)$ (Fig. 2.6,c), that take values from the extended set of symbols of amino acids $\bar{\mathcal{Y}}$ including the additional gap symbol.

In accordance with the principal idea of the approach to the problem of multi-sequence alignment of proteins, as it was outlined in the previous Section, there are two kinds of model constraints imposed upon the sought-for values of combined variables $\mathbf{x}_{t_1t_2} = (c_{t_1t_2}, k_{t_1t_2})$. First, all the amino acid symbols $c_{t_1t_2}$ in each column t_1 of the resulting array are assumed to take the same value, what is equivalent to the restriction $c_{t_1t_2-1} = c_{t_1t_2}$ that proclaims adjacent all the vertical pairs $(c_{t_1t_2-1}, c_{t_1t_2})$. Second, the conceived sense of the references $k_{t_1t_2}$ in horizontal rows to amino acids in the original proteins determines strict constraints on the manner of their increasing when moving from left to right, hence, all the horizontal pairs $(k_{t_1-1,t_2}, k_{t_1t_2})$ are to be considered as adjacent.

The adjacency graph for the problem of multi-sequence alignment is shown in Fig. 2.7,c. As we see, it turns out to be somewhat less straightforward than the rectangular lattice, because the elements of the secondary array are vectors whose components interact in a specific manner. The horizontal and vertical edges represent, respectively, the interdependence between sequential references to the original amino acid positions in each protein and identity the identically indexed elements in pattern sequences attributed to different proteins. The additional edges sketched as sticking out of the page express the joint dependence of each pair of variables of these two kinds on the original set of proteins. So,

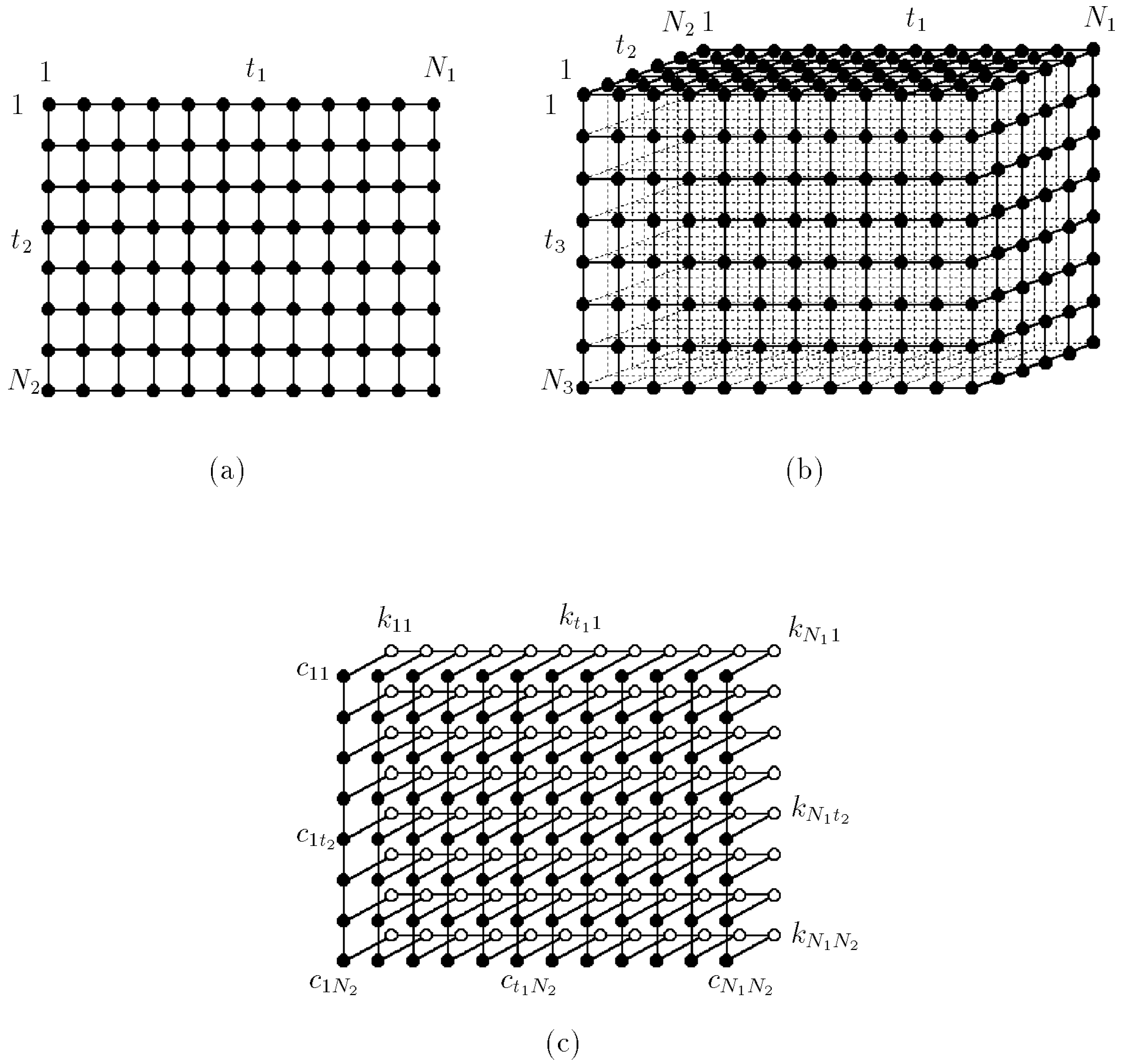

Figure 2.7: Adjacency graphs of the separable objective function in different data analysis problems: (a) two-dimensional lattice of spatial neighborhood of image pixels and seismic section elements; (b) three-dimensional lattice of spatial neighborhood of seismic cube elements; (c) adjacency of two groups of variables in the problem of multi-sequence alignment of proteins.

the objective function consists only of the edge-type elementary functions:

$$J(\mathbf{X}|\mathbf{Y}) = J(\mathbf{C},\mathbf{Z}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\psi(c_{t_1t_2},k_{t_1t_2}|\mathbf{y}_{t_2}) + \sum_{t_1=1}^{N_1}\sum_{t_2=2}^{N_2}\gamma(c_{t_1t_2-1},c_{t_1t_2}) + \sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2}\mu(k_{t_1-1,t_2},k_{t_1t_2}), \tag{2.4}$$

where each of the data-dependent functions $\psi_{t_1t_2}(c_{t_1t_2},k_{t_1t_2}) = \psi(c_{t_1t_2},k_{t_1t_2}|\mathbf{y}_{t_2})$ is meant to measure the discrepancy between the assumed value of the pattern variable $c_{t_1t_2}$ and the actual amino acid $y_{k_{t_1t_2}}$ located at the assumed position $k_{t_1t_2}$ in the protein $\mathbf{y}_{t_2} = (y_{1t_2},\dots,y_{N_{t_2}t_2})$, and to transmit, thereby, the information from the original array of proteins to the objective function.

But with vector variables $x_{t_1t_2} = (c_{t_1t_2},k_{t_1t_2})$ considered without respect to their inner structure, the adjacency graph of the resulting array is a lattice just as for a picture (Fig. 2.7,a). Respectively, the objective function will have the same structure as (2.2):

$$J(\mathbf{X}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\psi(x_{t_1t_2}|\mathbf{y}_{t_2}) + \sum_{t_1=1}^{N_1}\sum_{t_2=2}^{N_2}\gamma(x_{t_1t_2-1},x_{t_1t_2}) + \sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2}\mu(x_{t_1-1,t_2},x_{t_1t_2}),$$

where $\psi(x_{t_1t_2}|\mathbf{y}_{t_2}) = \psi(c_{t_1t_2},k_{t_1t_2}|\mathbf{y}_{t_2})$, $\gamma(x_{t_1t_2-1},x_{t_1t_2}) = \gamma(c_{t_1t_2-1},c_{t_1t_2})$, $\mu(x_{t_1-1,t_2},x_{t_1t_2}) = \mu(k_{t_1-1,t_2},k_{t_1t_2})$.

Unfortunately, the search for an optimization algorithm for separable functions of the general kind inevitably leads to computational difficulties. The nature has put a fundamental barrier to the hopes to construct a non-iterative procedure which would provide finding the global minimum or maximum of a separable function supported by an arbitrary adjacency graph, and would be less computationally dissipative than exhaustive search.

The optimization problem of such a kind has been intensively studied within the bounds of the statistical approach to image processing, where the sought-for result $\mathbf{X}$ is considered as a realization of the known, in the probabilistic sense, hidden random field having produced the picture $\mathbf{Y}$ as its only outward exhibition. As a rule, the hidden random field is assumed to possess some Markov properties, which assumption leads to the separability of the likelihood function or posterior distribution density whose maximum point is to be found. There are two basic principles which may be regarded as underlying the known algorithms of iterative optimization of separable functions.

The first of them, usually referred to as ICM (iterated conditional modes) [1, 2, 3], consists in improving the values of the goal variables $x_{\mathbf{t}}$ at each $\mathbf{t}\in T$ in turn, starting with an initial approximation. Each improvement is to be made so that the objective function would not decrease. On the force of the separability of the objective function, such an operation does not imply any computational difficulty. Not only a single variable but also a group of them may be subjected to an improvement at each iteration, if a suitable algorithm is available. For instance, the conditionally optimal values in a particular row of the image pixel grid can be easily calculated by the appropriate version of the dynamic programming algorithm. Optimization procedures of such a kind converge quickly, usually, in less than ten iterations, but the payment for the fast convergence is their tendency to be trapped in the nearest local optimum, so that the result essentially depends on the initial approximation and, as it is often the case, only slightly improves it. The algorithms based on iterative recalculation of groups of argument instead of single arguments are markedly more resistant to local extrema.

The second principle is aimed at finding the global optimum. Such a goal is achieved by a drastic aggravation of the iterative nature of search, in which the required number of iterations is measured in hundreds and even thousands. Actually, this is not so much an iterative process as a version of the method of statistical simulations, usually referred to as Monte Carlo method. In general, the search for the global optimum, let it be maximum, can be reported as combination of two principal ideas.

The first idea is known under the name of stochastic relaxation [4]. Let the variety of combinations of argument values $\mathbf{X} = (x_{\mathbf{t}}, \mathbf{t} \in T)$ be considered as a random field, such that the greater the objective function $J(\mathbf{X})$, the greater the logarithmic probability density $\log p(\mathbf{X})$ is assumed to be at realization $\mathbf{X}$. The separability of the objective function is equivalent to the assumption that it is a Markov random field with Markov interaction of random variables given by graph G. So, we have to find the most probable realization of such a field. Let $\mathbf{X}$ be the approximation to it found at the previous iteration. Let us modify this realization by simulating the new random value of one of the variables in accordance with its local conditional distribution regarding the rest of the field, then do the same with another variable chosen by an appropriate rule, and so on. It is clear, that the random sequence of random field realizations will be Markov chain. The principal fact related to such a Markov chain is that, under some quite lenient conditions, its final distribution will be just $p(\mathbf{X})$, and, so, the sought-for most probable realization of the random field has the most chances to appear in the random sequence. Such a method of simulating random realizations of random fields or processes is known under the abbreviation MCMC (Markov Chain Monte Carlo) [5, 6].

The second idea, that procured the name of simulated annealing, consists in gradual decreasing the randomness degree when simulating new values of the random field elements, with the purpose to make the random sequence of random field realizations converge to the non-random most likely realization. Such an idea is brought into reality by a procedure named Gibbs sampler [4, 7, 8, 9]. Let us, at each step of simulating the random sequence of random field realizations, generate the random value of the next element in accordance not with its local conditional distribution but a modified one obtained by raising the conditional density to the power $1/\tau$ followed by the normalization of the resulting function to the unity integral, where τ is parameter named temperature. If $\tau \to \infty$, we obtain the uniform distribution, and with $\tau \to 0$ the new distribution will concentrate in the mode of the initial one or be the uniform distribution over the modes in case there are several of them. Let us simulate random realizations of the random field starting with a large "almost infinite" temperature, and gradually decrease it up to zero. The rate of the temperature reduction should be quite low, at least, not higher than the speed of the convergence of the Markov sequence to its final distribution. Then, in getting "less and less random", the sequence of random field realizations will approach the realization having the maximal value of the final density $p(\mathbf{X})$, i.e. the sought-for maximum point of the objective function $J(\mathbf{X})$. It is common practice to take the local conditional densities of the Markov random field from the class of the so-called Gibbs probability distributions [1, 4], in which the density formula already contains $1/\tau$ in the exponent.

Thus, the variety of known algorithmical means of finding the global optimum of a separable function of general kind is limited to the procedures of iterative random search. Iterative procedures of such a kind converge especially slowly, because the large number of random steps is just that computational expedient which prevents the search process from sticking in local extrema. The low computational speed is restricting for the range of applications where such algorithms might be used.

2.3 A series of tree-like neighborhood relations instead of the lattice-like one. Partial objective functions with tree-like separability

As it will be shown in Section 3, the computational solvability of an optimization problem gets essentially improved if the adjacency graph of the separable objective function is a tree. In this case, there exists a procedure that enables finding the global extremum without any iterations at all, in two runs through the set of variables. Therefore, it appears to be always preferable, just at the stage of formalization of the particular problem of data analysis, to set it so that the desired result would be sought for as maximum or minimum point of a tree-supported objective function.

The general characteristic feature of the class of data arrays which is considered in this work is that the desired result of processing is viewed as having an a priori determined shape in that or other space, might it be natural or imaginary. It is just the neighborhood graph on the set of the secondary data array elements which is the formal carrier of the assumed structure of such an a priori spatial model.

It is clear that there exists no way to replace a lattice-like neighborhood graph (Fig. 2.7) by a tree without the loss of the crucial property of the graph to carry the model-based information on the position of each element of the array relative to all the other elements. But the inevitable loss due to such a replacement will be much less if we concentrate our ingenuity on preserving, as much as possible, the principal traits of the entire global set of ties only for the elements within the bounds of a subset of the array, conventionally, a vertical row t_1^* or (t_1^*, t_2^*) depending on whether the lattice is plane or three-dimensional. To cover both cases, let us denote such a row $\bar{\mathbf{t}}^*$, so that $\bar{\mathbf{t}}^* = t_1^*$ or $\bar{\mathbf{t}}^* = (t_1^*, t_2^*)$.

There are several paths between any two nodes of a lattice-like graph. It is just this trivial property which is subject to a correction for the original adjacency graph would be transformed into a tree.

The first principal idea of the approach we use here consists in deleting as many edges as possible from the original graph G, so that it would become a tree in which each node in the marked row $\bar{\mathbf{t}}^*$ is connected with each of the other nodes, including those in this row itself, by exactly one path, as it is shown in Fig. 2.8,a and Fig. 2.8,b for, respectively, the plane and the three-dimensional lattice. Let us denote such a partial tree by $G_{\bar{\mathbf{t}}^*}$ and form the partial objective function $J_{\bar{\mathbf{t}}^*}(\mathbf{X}|\mathbf{Y})$ supported by it, in contrast to the full objective function (2.2) supported by the full graph G:

$$J_{\bar{\mathbf{t}}^*}(\mathbf{X}|\mathbf{Y}) = \sum_{\mathbf{t}\in T} \psi_{\mathbf{t}}(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}}) + \sum_{(\mathbf{t}',\mathbf{t}'')\in G_{\bar{\mathbf{t}}^*}} \gamma_{\mathbf{t}',\mathbf{t}''}(x_{\mathbf{t}'}, x_{\mathbf{t}''}).$$

In particular, for visual images and seismic sections, i.e. in the case of plane lattice, we will obtain N_1 partial objective functions, each marked by one-dimensional index $\bar{\mathbf{t}}^* = t_1^*$, instead of the full function (2.2)

$$\begin{aligned} J_{t_1^*}(\mathbf{X}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2} \psi(x_{t_1t_2}|\mathbf{Y}_{t_1t_2}) + \sum_{t_2=2}^{N_2} \gamma'(x_{t_1^*t_2-1}, x_{t_1^*t_2}) + \\ \sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2} \gamma''(x_{t_1-1,t_2}, x_{t_1t_2}), \quad t_1^* = 1,\ldots,N_1, \end{aligned} \tag{2.5}$$

and for seismic cubes, i.e. in the three-dimensional case, the full objective function (2.2) will be replaced by N_1N_2 partial functions marked by two-dimensional index $\bar{\mathbf{t}}^* = (t_1^*, t_2^*)$

$$\begin{aligned} J_{t_1^*t_2^*}(\mathbf{X}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\sum_{t_3=1}^{N_3} \psi(x_{t_1t_2t_3}|\mathbf{Y}_{t_1t_2t_3}) + \sum_{t_3=2}^{N_3} \gamma'(x_{t_1^*t_2^*t_3-1}, x_{t_1^*t_2^*t_3}) + \\ \sum_{t_1=1}^{N_1}\sum_{t_2=2}^{N_2}\sum_{t_3=1}^{N_3} \gamma''(x_{t_1t_2-1,t_3}, x_{t_1t_2t_3}) + \sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2}\sum_{t_3=1}^{N_3} \gamma''(x_{t_1-1,t_2t_3}, x_{t_1t_2t_3}), \\ t_1^* = 1,\ldots,N_1, \quad t_2^* = 1,\ldots,N_2. \end{aligned}$$

The optimization procedure considered below in Section 3 will easily find the optimal combination of the goal variables at all the nodes with respect to the partial objective function:

$$\hat{\mathbf{X}}_{\bar{\mathbf{t}}^*}(\mathbf{Y}) = \arg\min_{x_{\mathbf{t}}\in\mathcal{X}} J_{\bar{\mathbf{t}}^*}(\mathbf{X}|\mathbf{Y}). \tag{2.6}$$

Fig. 2.9 shows the result of segmentation obtained by this rule applied, with the respective constituents $\psi_{\mathbf{t}}(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}})$, $\gamma'(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ and $\gamma''(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ of the partial objective function, to a vertical row of the image shown in Fig. 2.1 (see Section 4.2 for detail). As wee see, such an array as a whole cannot be taken as ready result of data analysis, because the tree $G_{\bar{\mathbf{t}}^*}$ provides an approximate preservation of the entire picture of ties only for the nodes in column $\bar{\mathbf{t}}^*$. But the segmentation in the vertical stem of the pixel-adjacency tree

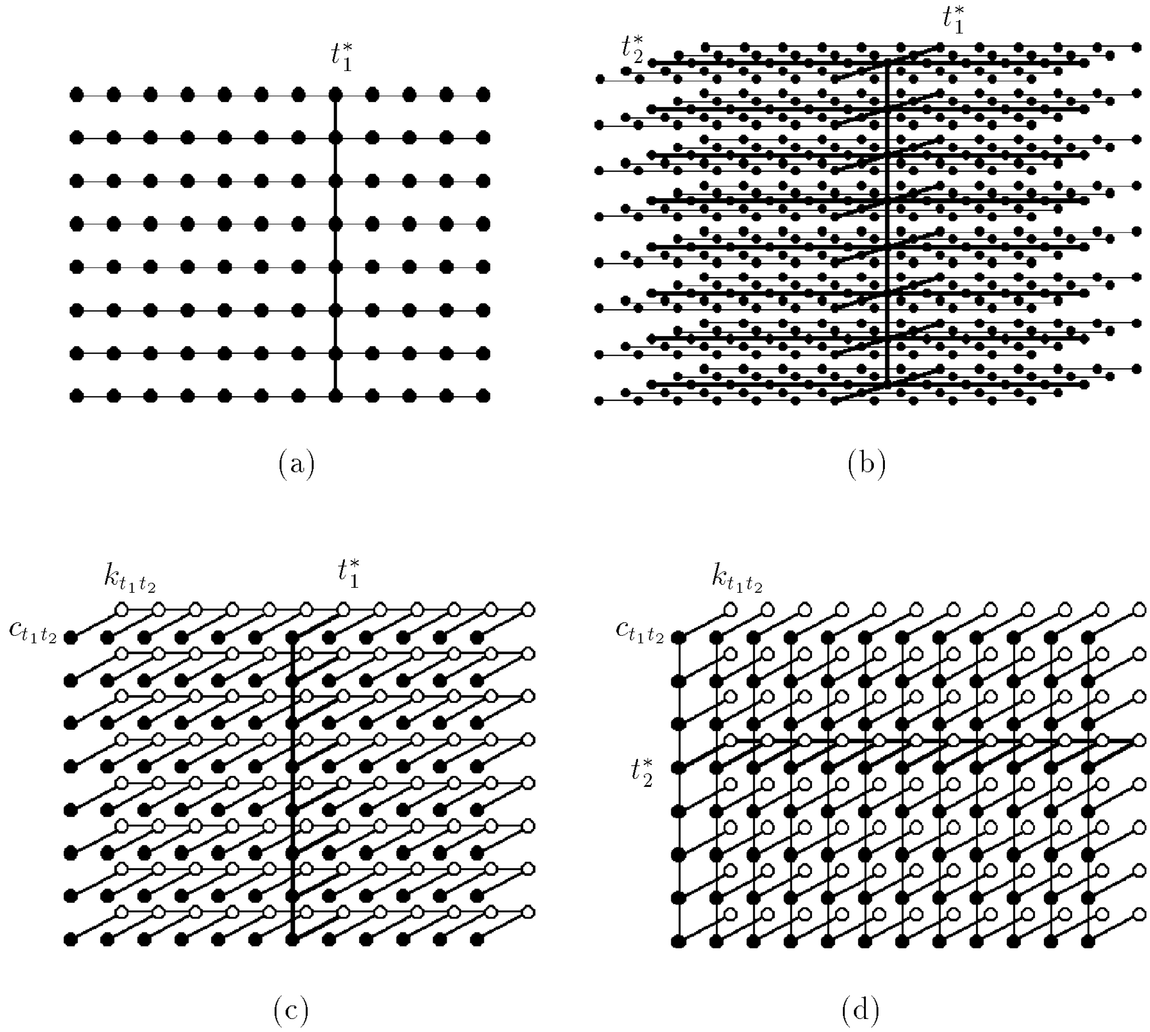

Figure 2.8: Adjacency graphs of the partial objective function with tree-like separability for evaluating the goal variables in a single vertical row of a two-dimensional (a) and three-dimensional (b) rectangular array. Two versions (c) and (d) of the tree-like adjacency graph for the partial objective function in the problem of multi-sequence alignment of proteins.

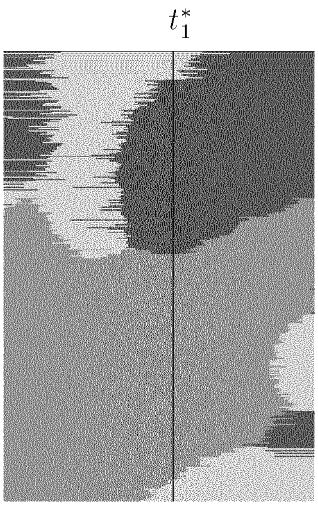

Figure 2.9: Segmentation of the image from Fig. 2.1 as result of optimization of the respective partial objective function supported by variable adjacency tree with stem in the vertical row t_1^* and horizontal lateral branches like in Fig. 2.8,a.

is absolutely sound, and there is nothing seen what could prevent us from taking it as constituent of a combined decision.

The second idea of the approach implies that the final result of data analysis should be formed as the ordered assembly of single columns $\hat{\mathbf{x}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})$ each of which is the stem column of the respective particular array $\hat{\mathbf{X}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})$. In accordance with this principle, the resulting two-dimensional array, in the plane case, must be formed as sequence of the stem columns, one from each particular two-dimensional array,

$$\hat{\mathbf{X}}(\mathbf{Y}) = \Big(\hat{\mathbf{x}}_{t_1^*}(\mathbf{Y}),\ t_1^* = 1,\ldots,N_1\Big) = \Big(\hat{x}_{t_1^* t_2}(\mathbf{Y});\ t_1^* = 1,\ldots,N_1,\ t_2 = 1,\ldots,N_2\Big), \quad (2.7)$$

and in the three-dimensional case, the resulting array should be taken as two-dimensional combination of the particular stem columns

$$\hat{\mathbf{X}}(\mathbf{Y}) = \Big(\hat{\mathbf{x}}_{t_1^* t_2^*}(\mathbf{Y});\ t_1^* = 1,\ldots,N_1,\ t_2^* = 1,\ldots,N_2\Big) =$$
$$\Big(\hat{x}_{t_1^* t_2^* t_3}(\mathbf{Y});\ t_1^* = 1,\ldots,N_1,\ t_2^* = 1,\ldots,N_2,\ t_3 = 1,\ldots,N_3\Big).$$

It is just the rule (2.7) in accordance with which the segmentation in Fig. 2.1 was assembled of the vertical stem rows of particular segmentations like that in Fig. 2.9, as well as all the results of image and data analysis in Fig. 2.2-2.4.

As it was shown in the previous subsection, the objective function in the problem of multi-sequence alignment of proteins, when it is considered as function of vector variables $x_{t_1 t_2} = (c_{t_1 t_2}, k_{t_1 t_2})$ (2.2), is supported by adjacency graph having the form of two-dimensional lattice, just like the adjacency graph in the problems of image analysis (Fig. 2.7,a). It seems to be natural to apply to this graph the same principle, and extract from it a series of trees with common horizontal or, maybe, vertical branches like the tree in Fig. 2.8,a. Such a straightforward extension of the idea, originally designed for images, onto the case of molecular sequences leads to a series of adjacency trees supporting the partial objective functions $t_1^* = 1, \dots, N_1$ with separated nodes of goal variables $c_{t_1 t_2}$ and $k_{t_1 t_2}$

$$J_{t_1^*}(\mathbf{X}|\mathbf{Y}) = J(\mathbf{C},\mathbf{Z}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\psi(c_{t_1 t_2}, k_{t_1 t_2}|\mathbf{y}_{t_2}) + \sum_{t_2=2}^{N_2}\gamma(c_{t_1^* t_2 - 1}, c_{t_1^* t_2}) + \sum_{t_2=1}^{N_2}\sum_{t_1=2}^{N_1}\mu(k_{t_1-1,t_2}, k_{t_1 t_2}), \tag{2.8}$$

as it is shown in Fig. 2.8,c, in contrast to the original full objective function (2.4) and its adjacency graph in Fig. 2.7,c.

However, there is at least one essential distinction between the problem of multisequence alignment and other problems of data analysis considered in this work.

In the cases of images and seismic data arrays, the arguments of the horizontal $\gamma'(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ and vertical constituents $\gamma''(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ of objective functions (2.2), (2.2) are of identical physical nature, and express practically identical smoothness constraints on the field of hidden variables in the respective space. Therefore, it does not matter what of directions, the vertical or horizontal one, is taken as direction of stems when the original adjacency graph is replaced by a series of partial trees. Equally well, we could turn each of the trees in Fig. 2.8a,b on the side, because there are only quantitative but no qualitative distinctions between model constraints in different directions, if any at all.

But the arguments of edge functions $\gamma(c_{\mathbf{t}'}, c_{\mathbf{t}''})$ and $\mu(k_{\mathbf{t}'}, k_{\mathbf{t}''})$ in (2.4) and (2.8) bear absolutely different meanings. The former of the edge functions is meant to impose a penalty upon any difference between pattern elements $c_{t_1 t_2 - 1}$ and $c_{t_1 t_2}$ in columns of the alignment, and to ensure, thereby, the identity or, at least, close similarity of pattern sequences associated in it with different proteins, whereas the latter one is aimed at preventing possible contradictions between neighboring references k_{t_1-1,t_2} and $k_{t_1 t_2}$ in the rows of the alignment and the natural order of original amino acids in proteins.

The fact that the stems of particular adjacency trees are formed in Fig. 2.8,c by vertical edges associated with edge functions $\gamma(c_{t_1 t_2 - 1}, c_{t_1 t_2})$ means that each of the partial objective functions (2.8) takes into account the assumed vertical identity of pattern elements in the respective stem column. As a result, the assumed identity of pattern sequences will be completely observed in the full final decision assembled of these columns

$$\hat{\mathbf{X}}(\mathbf{Y}) = \big(\hat{\mathbf{C}}(\mathbf{Y}), \hat{\mathbf{K}}(\mathbf{Y})\big) = \big(\hat{\mathbf{c}}_{t_1^*}(\mathbf{Y}), \hat{\mathbf{k}}_{t_1^*}(\mathbf{Y});\ t_1^* = 1, \dots, N_1\big) = \big(\hat{c}_{t_1^* t_2}(\mathbf{Y}), \hat{k}_{t_1^* t_2}(\mathbf{Y});\ t_1^* = 1, \dots, N_1,\ t_2 = 1, \dots, N_2\big). \tag{2.9}$$

But as to the references to actual amino acids, their ordering in the horizontal direction provided by horizontal edge functions $\mu(k_{t_1-1,t_2}, k_{t_1t_2})$ is different in different partiacular arrays $\hat{\mathbf{X}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})$. As a result, the stem columns of these arrays $\hat{\mathbf{x}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})$, when having been assembled into an entire array (2.9), will be not able to provide noncontradictory references in the resulting combined rows. So, the final array $\hat{\mathbf{X}}(\mathbf{Y})$ may turn out to be no alignment at all, unless an additional correction is undertaken.

Fig. 2.8,d shows an alternative version of the orientation of particular adjacency trees which support another, relative to (2.8), version of the series of partial objective functions

$$
\begin{aligned}
J_{t_2^*}(\mathbf{X}|\mathbf{Y}) = J(\mathbf{C},\mathbf{Z}|\mathbf{Y}) = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}\psi(c_{t_1t_2}, k_{t_1t_2}|\mathbf{y}_{t_2}) + \\
\sum_{t_1=1}^{N_1}\sum_{t_2=2}^{N_2}\gamma(c_{t_1t_2-1}, c_{t_1t_2}) + \sum_{t_1=2}^{N_1}\mu(k_{t_1-1,t_2^*}, k_{t_1t_2^*}),
\end{aligned}
\qquad (2.10)
$$

where each particular adjacency tree provides completely noncontradictory references in the respective stem row. But as to the assumed identity of pattern elements, it may get broken in the combined vertical columns of the final array formed by such rows

$$
\begin{aligned}
\hat{\mathbf{X}}(\mathbf{Y}) = \left(\hat{\mathbf{C}}(\mathbf{Y}), \hat{\mathbf{K}}(\mathbf{Y})\right) = \left(\hat{\mathbf{c}}_{t_2^*}(\mathbf{Y}), \hat{\mathbf{k}}_{t_2^*}(\mathbf{Y});\ t_2^* = 1,\ldots,N_2\right) = \\
\left(\hat{c}_{t_1t_2^*}(\mathbf{Y}), \hat{k}_{t_1t_2^*}(\mathbf{Y});\ t_1 = 1,\ldots,N_1,\ t_2^* = 1,\ldots,N_2\right),
\end{aligned}
\qquad (2.11)
$$

so that an additional correction of the alignment will be required.

The specificity of replacing of the full adjacency graph by a series of its tree-like approximations in the problem of multi-sequence alignment, as compared to those of image and seismic data analysis, do not reduce to only this intricacy. We will return to each of the above mentioned particular applications below, in Section 4, after we consider, in the next Section, the theoretical framework of optimization of separable functions with tree-like separability and the principal structure of an effective optimization procedure, making use of which is just the ultimate aim of such a replacement.

3 Optimization of separable functions with tree-like separability

3.1 Structure of the objective function. Node and edge functions

In this Section, we don't associate the variables of the objective function, i.e. the nodes of the supporting tree, with variables in any certain application problem. To denote them, we will use symbol t instead of denotations $\mathbf{t} = (t_1, t_2)$ or $\mathbf{t} = (t_1, t_2, t_3)$.

Let G be a tree on the set of nodes $T = \{t\}$ (Fig. 3.1), and

$$
J(X) = \sum_{t\in\mathsf{T}}\psi_t(x_t) + \sum_{(t',t'')\in G}\gamma_{t',t''}(x_{t'}, x_{t''}),\quad x_t \in \mathcal{X}_t, \qquad (3.1)
$$

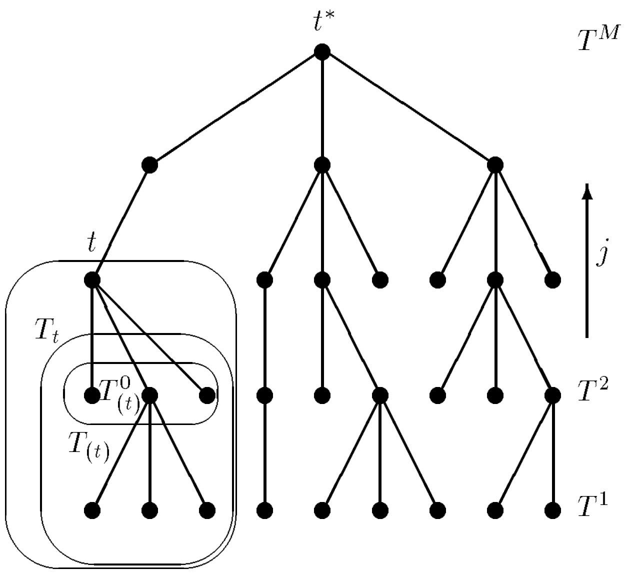

Figure 3.1: Structure of an arbitrary tree.

be a G-supported separable objective function to be minimized in an arbitrary collection of ranges of variables. Functions of one variable $\psi_t(x_t)$ at nodes of the graph will be called node functions, and functions of two variables $\gamma_{t',t''}(x_{t'}, x_{t''})$ assigned to edges will be, respectively, called edge functions.

It is out of significance which of the nodes t^* is assigned the root of the tree, but once it is done, the set of nodes gets partitioned into a number of subsets each consisting of nodes of a certain level

$$T = \bigcup_{j=1}^{M} T^j, \quad T^j \bigcap T^k = \emptyset, \quad j \neq k.$$

We will denote the set of nodes of the lowest level by T^1, and the set containing the only node of the highest level, namely, the root, by T^M.

Let, further, the tree $G_t \subset G$ formed by a node t and its descendants be called descendant tree of this node. It is clear that $G_t = \emptyset$ for leaves, in particular, for all $t \in T^1$, and $G_t = G$ for the root $t \in T^M$. The set of all the nodes in the descendant tree of a node t will be denoted by T_t, symbol $T_{(t)}$ will mean the same set without this node itself, i.e. the set of its descendants, and $T^0_{(t)} \subseteq T_{(t)}$ will denote the subset of its immediate (next) descendants. So, $T_t = \{t\}$ and $T^0_{(t)} = T_{(t)} = \emptyset$ for leaves, and $T_{t*} = T$ for the root.

Analogously, symbols $X_t = (x_s,\ s \in T_t)$ and $X_{(t)} = (x_s,\ s \in T_{(t)})$ will mean partial vectors of variables at the respective sets of nodes.

Finally, let us associate with each node t a partial objective function

$$J_t(X_t) = \sum_{s \in T_t} \psi_s(x_s) + \sum_{(s',s'') \in G_t} \gamma_{s',s''}(x_{s'}, x_{s''}) \tag{3.2}$$

which has absolutely the same structure as the full objective function (3.1) but is defined on the respective partial descendant tree. It is easy to see that

$$J_t(X_t) = J_t(x_t, X_{(t)}) = \psi_t(x_t) + \sum_{s \in T^0_{(t)}} \gamma_{t,s}(x_t, x_s) + \sum_{s \in T^0_{(t)}} J_s(X_s) =$$

$$\psi_t(x_t) + \sum_{s \in T^0_{(t)}} \left\{ \gamma_{t,s}(x_t, x_s) + J_s(x_s, X_{(s)}) \right\}.$$

In particular, for the nodes of the lowest level $t \in T^1$, as well as for all other leaves, the partial objective function gets trivial $J_t(X_t) = J_t(x_t) = \psi_t(x_t)$, and for the root the partial function is nothing else than the full objective function $J_{t^*}(X_{t^*}) = J(X)$ (3.1).

3.2 Bellman functions and the basic optimization procedure

The principal idea of the optimization procedure for separable objective functions supported by trees is based, like the classical dynamic programming procedure for chains, on the notion of Bellman function

$$\tilde{J}_t(x_t) = \min_{X_{(t)}} J(x_t, X_{(t)}) = \psi_t(x_t) + \min_{X_s,\, s \in T^0_{(t)}} \sum_{s \in T^0_{(t)}} \left\{ \gamma_{t,s}(x_t, x_s) + J_s(X_s) \right\}, \quad x_t \in \mathcal{X}_t. \tag{3.3}$$

Theorem 1 *For all the nodes of the tree, the Bellman functions satisfy the recurrent relation*

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \sum_{s \in T^0_{(t)}} \min_{x_s \in \mathcal{X}_s} \left\{ \gamma_{t,s}(x_t, x_s) + \tilde{J}_s(x_s) \right\}. \tag{3.4}$$

Proof It immediately follows from the definition of Bellman function (3.4) that

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \min_{X_s,\, s \in T^0_{(t)}} \sum_{s \in T^0_{(t)}} \{ \gamma_{t,s}(x_t, x_s) + J_s(X_s) \}.$$

Each of the vector variables $X_s = (x_s, X_{(s)})$, $s \in T^0_{(t)}$, occurs only in one summand $\{\gamma_{t,s}(x_t, x_s) + J_s(X_s)\} = \{\gamma_{t,s}(x_t, x_s) + J_s(x_s, X_{(s)})\}$, therefore, the operations of minimization with respect to $(x_s, X_{(s)})$ can be fulfilled independently of each other before summation:

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \sum_{s \in T^0_{(t)}} \min_{x_s \in \mathcal{X}_s} \{ \gamma_{t,s}(x_t, x_s) + J_s(X_s) \} =$$

$$\psi_t(x_t) + \sum_{s \in T^0_{(t)}} \min_{x_s \in \mathcal{X}_s} \{ \gamma_{t,s}(x_t, x_s) + \min_{X_{(s)}} J(X_{(s)}) \}.$$

Here, in their turn, $\min\limits_{X_{(s)}} J(X_{(s)})$ are Bellman functions $\tilde{J}_s(x_s)$ of the node variables x_s, whence (3.4) immediately follows.

Q.E.D.

The fundamental property of the Bellman function (3.4) established by Theorem 1 will be called the upward recurrent relation. Once the Bellman functions are known for all the immediate descendants $s \in T^0_{(t)}$ of a node t, the relation defines the Bellman function at this node. The inverted form of the upward recurrent relation

$$\tilde{x}_s(x_t) = \arg \min_{x_s \in \mathcal{X}_s} \left\{ \gamma_{t,s}(x_t, x_s) + \tilde{J}_s(x_s) \right\}, \quad s \in T^0_{(t)}, \tag{3.5}$$

will be referred to as the downward recurrent relation.

The optimization procedure is based on the assumption that there exists a sufficiently effective way of solving the partial optimization problems occurring in (3.4), and storing, in a compact form, the Bellman functions or downward relations.

The procedure runs twice through all the nodes of the graph in order of their membership in the hierarchy levels T^j, first upwards $j = 1, \ldots, M$, and then downwards $j = M, \ldots, 2$.

It starts with the nodes of the lowest (first) hierarchy level $t \in T^1$, $j = 1$. As $T^0_{(t)} = \emptyset$ for these nodes,

$$\tilde{J}_t(x_t) = \psi_t(x_t), \quad x_t \in \mathcal{X}_t, \quad t \in T^1. \tag{3.6}$$

Thereupon, the upward recurrent relation (3.4) is applied sequentially to the nodes of the higher levels $j = 2, \ldots, M$. The Bellman functions $\tilde{J}_t(x_t)$ or downward relations $\tilde{x}_s(x_t)$, $x_t \in \mathcal{X}$, are to be stored for all the nodes of the tree except those of the first level.

The Bellman function at the root $\tilde{J}_{t^*}(x_{t^*})$, $x_{t^*} \in \mathcal{X}$, that is obtained at the last step of the upward pass, immediately gives the optimal value of the root variable:

$$\hat{x}_{t^*} = \arg \min_{x_{t^*} \in \mathcal{X}_{t^*}} \tilde{J}_{t^*}(x_{t^*}). \tag{3.7}$$

On the downward pass, as the procedure descends the hierarchy levels $j = M, \ldots, 2$, the already found optimal values of the current level determine optimal values of the variables at the immediately underlying level in accordance with the downward recurrent rules (3.5) stored as the result of the upward pass or determined by the stored Bellman functions (3.4)

$$\hat{x}_s = \tilde{x}_s(\hat{x}_t), \quad s \in {T_{(t)}}^0. \tag{3.8}$$

The degenerate particular case of a tree is tree without branching nodes, i.e. a chain. In this case, the above described optimization procedure becomes the classical dynamic programming procedure that will be considered in the next Section 3.3.

We introduce here an additional denotation $\hat{J}_t(x_t)$ for the so-called marginal node functions each of which shows how the full objective function depends on the value of a single variable at the respective node t, if other node variables take conditionally optimal values in accordance with x_t:

$$\hat{J}_t(x_t) = \min_{x_s,\, s \in T,\, s \neq t} J(X). \tag{3.9}$$

At the root, the marginal node function is found at the last step of the upward pass $\hat{J}_{t^*}(x_{t^*})$. The following theorem gives a downward recurrent relation that allows to derive all the remaining marginal node functions from the obvious one at the root.

Theorem 2 *Let the marginal node function be known for an inner (not-leaf) node t of the tree. Then, for each of its immediately descendant nodes the marginal node function satisfies the relation*

$$\hat{J}_s(x_s) = \min_{x_t \in \mathcal{X}_t} \left\{ \hat{J}_t(x_t) + \left[\tilde{J}_s(x_s) + \gamma_{t,s}(x_t, x_s)\right] - \min_{x'_s \in \mathcal{X}_s} \left[\tilde{J}_s(x'_s) + \gamma_{t,s}(x_t, x'_s)\right] \right\},\ s \in T^0_{(t)}. \tag{3.10}$$

Proof Let t be an arbitrary not-leaf node. Each of such nodes partitions the set of all the nodes of the tree into three non-overlapping subsets: the node t itself, the set of its descendants $T_{(t)}$, and the set of remaining nodes $T^+_{(t)}$. Respectively, in addition to the subtree G_t defined previously on the set $T_t = \{t\} \bigcap T_{(t)}$, let us define subtree G^+_t on $T^+_t = \{t\} \bigcap T^+_{(t)}$. By analogy with the partial objective function $J_t(X_t) = J_t(x_t, X_{(t)})$ (3.2) that leans upon x_t along with the set of descendant variables $X_{(t)}$, we define here a function $J^+_t(X^+_t) = J_t(x_t, X^+_{(t)})$ of the same structure but supported by the set of remaining variables $\bar{X}_{(t)}$:

$$J^+_t(X^+_t) = J_t(x_t, X^+_{(t)}) = \sum_{s \in T_t{}^+} \psi_s(x_s) + \sum_{(s',s'') \in G^+_t} \gamma(x_{s'}, x_{s''}).$$

Finally, by analogy with the previously defined (lower) Bellman function $\tilde{J}_t(x_t)$ (3.3), we define the alternative (upper) Bellman function

$$\tilde{J}^+_t(x_t) = \min_{X^+_{(t)}} J^+_t(X^+_t) = \min_{X^+_{(t)}} J^+_t(x_t, X^+_{(t)}),\ \ x_t \in \mathcal{X}.$$

Let t and $s \in T^0_{(t)}$ be, respectively, a node and one of its immediate descendants. It is obvious that $J(X) = J_s(X_s) + \gamma_{t,s}(x_t, x_s) + J^+_t(X^+_t)$. Hence, in accordance with (3.9) $\hat{J}_t(x_t) = \tilde{J}^+_t(x_t) + \min_{x_s}\{\gamma_{t,s}(x_t, x_s) + \tilde{J}_s(x_s)\}$, and

$$\tilde{J}^+_t(x_t) = \hat{J}_t(x_t) - \min_{x_s}\{\gamma_{t,s}(x_t, x_s) + \tilde{J}_s(x_s)\}. \tag{3.11}$$

Likewise, $\hat{J}_s(x_s) = \min_{x_t}\{\gamma_{t,s}(x_t, x_s) + \tilde{J}^+_t(x_t)\} + \tilde{J}_s(x_s)$. Substitution of (3.11) in the last equation immediately gives (3.10).

Q.E.D.

3.3 Dynamic programming procedure for optimization problems with chain-like adjacency of variables

As we saw in Section 2.2, the rectangular lattice (Fig. 2.7) is a typical kind of variable adjacency in separable objective functions which is consistent with many engineering and scientific applications. Although no specificity is seen that could set off the lattice out of the general kind of adjacency graphs from the standpoint of computational solvability of the optimization problem, we managed to find a heuristic way to decompose the initial problem over the entire lattice-like array of goal variables into a succession of partial optimization problems, each over a single row of variables with tree-like picture of ties to the rest of the array.

In the simplest case, if a tree has no branching nodes, it turns into a chain, in which each node, except two of them at the opposite ends, has strictly two neighbors, as it is shown in Fig. 3.2. As we shall see below in Section 4.1, the partial optimization problems with tree-like separability of the objective function, that result from the heuristic decomposition of the original problem, are supported by especially simple trees, and allow for a further decomposition into a succession of still more simple optimization problems with chain-like adjacency of variables, for this time, without any heuristic indulgence.

It is natural to number the nodes of a chain in accordance with their adjacency $t = 1, \dots, N$. In contrast to the separable objective function of general kind (2.2), (3.1), and that with lattice-like separability of arguments (2.2), node and edge functions assembled into a chain occur in a separable objective function with chain-like separability as single-indexed summands:

$$J(\mathbf{X}) = \sum_{t=1}^{N} \psi_t(x_t) + \sum_{t=2}^{N} \gamma_t(x_{t-1}, x_t). \tag{3.12}$$

It is obvious that all the assertions proven for objective functions with tree-like separability are valid also for chains, as they are particular case of trees. To retain as much as possible the structure of the basic optimization procedure built in the previous Section, let us assign the last node $t = N$ the "root" of the chain considered as degenerate tree, then the first node $t = 1$ will be its only leaf.

As before, the optimization procedure will run twice through all the nodes, first forward, namely, from the leaf to the root $t = 1, \dots, N$ (or, maybe, upward from bottom

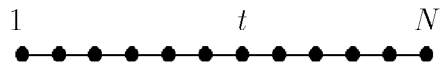

Figure 3.2: Chain, the simplest kind of tree-like adjacency of variables in a separable objective function.

to top, as previously, if we rearrange the nodes in Fig. 3.2 vertically with $t = 1$ at the bottom, like the hierarchy levels in Fig. 3.1), and then back (downward), from the root to the leaf $t = N, \dots, 1$.

The procedure starts at $t = 1$ by the initial assignment of the first Bellman function (3.6)

$$\tilde{J}_1(x_1) = \psi_1(x_1). \tag{3.13}$$

Thereupon, the Bellman functions are recurrently recalculated, without storing, in full accordance with (3.4)

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \min_{x_{t-1}\in\mathcal{X}_{t-1}} \left\{\gamma_t(x_{t-1}, x_t) + \tilde{J}_{t-1}(x_{t-1})\right\},\ x_t \in \mathcal{X},\ \ t = 2, \dots, N. \tag{3.14}$$

But the backward recurrent relations, like downward recurrent relations (3.5),

$$\tilde{x}_{t-1}(x_t) = \arg\min_{x_{t-1}\in\mathcal{X}_{t-1}} \left\{\gamma_t(x_{t-1}, x_t) + \tilde{J}_{t-1}(x_{t-1})\right\},\ x_t \in \mathcal{X}, \tag{3.15}$$

should be stored at all the nodes $t = 2, \dots, N$.

The minimum point of the last Bellman function is, according to (3.7), the sought-for optimal value of the last variable $\hat{x}_N$, and this function itself is the respective marginal node function $\hat{J}_N(x_N)$:

$$\tilde{x}_N = \arg\min_{x_N\in\mathcal{X}_N} \tilde{J}_N(x_N), \quad \hat{J}_N(x_N) = \tilde{J}_N(x_N),\ x_N \in \mathcal{X}. \tag{3.16}$$

On the backward pass, the procedure recurrently finds, in full accordance with (3.8), the optimal values and, if required, defines the marginal node function (3.10) for all the remaining variables:

$$\begin{aligned}
&\hat{x}_{t-1} = \tilde{x}_{t-1}(\hat{x}_t),\\
&\hat{J}_{t-1}(x_{t-1}) = \min_{x_t\in\mathcal{X}_t}\Big\{\hat{J}_t(x_t) + \left[\tilde{J}_{t-1}(x_{t-1}) + \gamma_t(x_{t-1}, x_t)\right] - \\
&\qquad\qquad\qquad \min_{x'_{t-1}\in\mathcal{X}_{t-1}}\left[\tilde{J}_{t-1}(x'_{t-1}) + \gamma_t(x'_{t-1}, x_t)\right]\Big\},\\
&x_t \in \mathcal{X}_t,\ \ t = N, \dots, 2.
\end{aligned} \tag{3.17}$$

Such an algorithm of finding the optimal values $\hat{x}_1, \dots, \hat{x}_N$ is nothing else than a version of the classical dynamic programming procedure [10] widely adopted as a means of optimization of multi-stage processes. What is new here is the succession of marginal node functions $\hat{J}_1(x_1), \dots, \hat{J}_N(x_N)$ which are, actually, the main result of our procedure, because the optimal value of each of the goal variables is the minimum point of the respective marginal function. It is just this procedure which will underlie all the algorithms of data analysis considered below in Section 4.

3.4 Numerical realization of the optimization procedures for discretely defined and quadratic node and edge functions

A numerical realization of the basic procedure for a separable function supported by a tree, as well as the classical dynamic programming procedure, is possible only in case there exists a finitely parametrized function family $\tilde{J}(x; \mathbf{a})$ concordant with node functions $\psi_t(x)$ and edge functions $\gamma_{t',t''}(x', x'')$ in the sense that Bellman functions (3.4) belong to this family at each step. In this case, the upward pass of the procedure consists in a recurrent re-evaluating parameters $\tilde{\mathbf{a}}_t$ that completely represent the Bellman functions.

We consider here two kinds of such a parametrization which present, in some sense, two extremes of the range of possible ways to define node and edge functions.

The evident case of a trivial parametrization is the case of a discrete range of node variables $\mathcal{X}_t = \mathcal{X} = \{1, \ldots, m\}$. In this case, Bellman functions (3.4) are m-dimensional vectors, downward relations (3.5) are also m-dimensional vectors, and the elementary enumeration gives an easy solution of the partial optimization problems.

Another extreme case is the assumption that the range of node variables is n-dimensional real space $\mathbf{x}_t \in \mathcal{X}_t = \mathcal{X} = \mathsf{R}^n$, and both node and edge functions are quadratic.

Theorem 3 *Let the node and edge functions be quadratic functions of the structure*

$$\psi_t(\mathbf{x}_t) = b_t^0 + (\mathbf{x}_t - \mathbf{x}_t^0)^T \mathbf{Q}_t^0 (\mathbf{x}_t - \mathbf{x}_t^0), \quad \gamma_{t',t''}(\mathbf{x}_{t'}, \mathbf{x}_{t''}) = (\mathbf{x}_{t'} - \mathbf{x}_{t''})^T \mathbf{U}_{t',t''} (\mathbf{x}_{t'} - \mathbf{x}_{t''}), \tag{3.18}$$

defined, respectively, by minimum values b_t^0, minimum points $\mathbf{x}_t^0 \in \mathsf{R}^n$ and arbitrary Hessian matrices $\mathbf{Q}_t^0$, and by positive definite Hessian matrices $\mathbf{U}_{t',t''}$. Then, each of the Bellman functions will be a quadratic function

$$\tilde{J}_t(\mathbf{x}_t) = \tilde{b}_t + (\mathbf{x}_t - \tilde{\mathbf{x}}_t)^T \tilde{\mathbf{Q}}_t (\mathbf{x}_t - \tilde{\mathbf{x}}_t), \quad \mathbf{x}_t \in \mathsf{R}^n,$$

with parameters $\tilde{b}_t \in \mathsf{R}$, $\tilde{\mathbf{x}}_t \in \mathsf{R}^n$ and $\tilde{\mathbf{Q}}_t$ $(n \times n)$ satisfying the upward recurrent relation

$$\begin{gathered}
\tilde{\mathbf{Q}}_t = \mathbf{Q}_t^0 + \sum_{s \in T_{(t)}^0} \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1}, \ \tilde{\mathbf{x}}_t = \tilde{\mathbf{Q}}_t^{-1} \left\{ \mathbf{Q}_t^0 \mathbf{x}_t^0 + \sum_{s \in T_{(t)}^0} \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1} \tilde{\mathbf{x}}_s \right\}, \\
\tilde{b}_t = b_t^0 + (\mathbf{x}_t^0 - \tilde{\mathbf{x}}_t)^T \mathbf{Q}_t^0 \mathbf{x}_t^0 + \sum_{s \in T_{(t)}^0} \left\{ \tilde{b}_s + (\tilde{\mathbf{x}}_s - \tilde{\mathbf{x}}_t)^T \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1} \tilde{\mathbf{x}}_s \right\}.
\end{gathered} \tag{3.19}$$

The proof of this and the next theorem will lean upon the following lemma.

Lemma 4 *For any quadratic function*

$$L(\mathbf{x}) = \sum_{i=1}^{M} (\mathbf{D}_i \mathbf{x} - \mathbf{z}_i)^T \mathbf{B}_i (\mathbf{D}_i \mathbf{x} - \mathbf{z}_i), \quad \mathbf{x} \in \mathsf{R}^n, \tag{3.20}$$

determined by vectors $\mathbf{z}_i \in \mathsf{R}^{k_i}$*,* $k_i \le n$*,* $i = 1, \dots, N$*, non-negative definite matrices* $\mathbf{B}_i$ $(n \times n)$*, and arbitrary matrices* $\mathbf{D}_i$ $(k_i \times n)$*, where, at least, one of the matrices* $\mathbf{B}_i$ *is positive definite* $\mathbf{B}_i > 0$ *and the respective matrix* $\mathbf{D}_i$ *is of the full rank* $\mathrm{Rank}(\mathbf{D}_i) = k_i$*, the following identity holds:*

$$L(\mathbf{x}) = (\mathbf{x} - \mathbf{z})^T \mathbf{B}(\mathbf{x} - \mathbf{z}) + d, \tag{3.21}$$

where

$$\mathbf{B} = \sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{D}_i > 0,$$

$$\mathbf{z} = \arg\min_{\mathbf{x}\in\mathsf{R}^n} L(\mathbf{x}) = \left(\sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{D}_i\right)^{-1} \sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{z}_i, \tag{3.22}$$

$$d = \min_{\mathbf{x}\in\mathsf{R}^n} L(\mathbf{x}) = \sum_{i=1}^{M} (\mathbf{D}_i \mathbf{z} - \mathbf{z}_i)^T \mathbf{B}_i (\mathbf{D}_i \mathbf{z} - \mathbf{z}_i) = \sum_{i=1}^{M} (\mathbf{z}_i - \mathbf{D}_i \mathbf{z})^T \mathbf{B}_i \mathbf{z}_i \ge 0. \tag{3.23}$$

In particular, if $N = 2$*,* $k_1 = k_2 = n$*, and square matrices* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *are nonsingular,*

$$d = \min_{\mathbf{x}\in\mathsf{R}^n} L(\mathbf{x}) = (\mathbf{D}_1^{-1}\mathbf{z}_1 - \mathbf{D}_2^{-1}\mathbf{z}_2)^T \Big(\mathbf{D}_1^{-1}\mathbf{B}_1^{-1}(\mathbf{D}_1^{-1})^T + \mathbf{D}_2^{-1}\mathbf{B}_2^{-1}(\mathbf{D}_2^{-1})^T\Big)^{-1} (\mathbf{D}_1^{-1}\mathbf{z}_1 - \mathbf{D}_2^{-1}\mathbf{z}_2). \tag{3.24}$$

Proof The representation (3.21) follows from the very fact that the function (3.20) is quadratic. It is obvious that $\nabla L(\mathbf{x}) = 0$ when $\mathbf{x} = \mathbf{z}$, therefore,

$$\sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i (\mathbf{D}_i \mathbf{z} - \mathbf{z}_i) = 0, \ \text{ i.e. } \left(\sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{D}_i\right) \mathbf{z} = \sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{z}_i. \tag{3.25}$$

Taking into account that, at least, one of the matrices $\mathbf{B}_i$ is positive definite and the respective matrix $\mathbf{D}_i$ is of the full rank and, hence, at least, one of the summands $\mathbf{D}_i^T \mathbf{B}_i \mathbf{D}_i$ is positive definite, we obtain (3.22). The first identity in (3.23) immediately follows from (3.20)-(3.22). As to the second identity, the substitution of (3.22) into (3.20) gives

$$d = \sum_{i=1}^{M} (\mathbf{z}_i - \mathbf{D}_i \mathbf{z})^T \mathbf{B}_i (\mathbf{z}_i - \mathbf{D}_i \mathbf{z}) = \sum_{i=1}^{M} (\mathbf{z}_i - \mathbf{D}_i \mathbf{z})^T \mathbf{B}_i \mathbf{z}_i - \sum_{i=1}^{M} (\mathbf{z}_i - \mathbf{D}_i \mathbf{z})^T \mathbf{B}_i \mathbf{D}_i \mathbf{z} =$$
$$\sum_{i=1}^{M} (\mathbf{z}_i - \mathbf{D}_i \mathbf{z})^T \mathbf{B}_i \mathbf{z}_i - \mathbf{z}^T \sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i (\mathbf{z}_i - \mathbf{D}_i \mathbf{z}) =$$
$$\sum_{i=1}^{M} (\mathbf{z}_i - \mathbf{D}_i \mathbf{z})^T \mathbf{B}_i \mathbf{z}_i - \mathbf{z}^T \left[\sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{z}_i - \left(\sum_{i=1}^{M} \mathbf{D}_i^T \mathbf{B}_i \mathbf{D}_i\right) \mathbf{z}\right].$$

The difference in the square brackets equals zero on the force of (3.25), what proves (3.23). The non-negative value of (3.23) follows from the fact that the function (3.20) under consideration is a sum of non-negative quadratic forms.

In the particular case when $N = 2$, $k_1 = k_2 = n$, and the square matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ are nonsingular, it is possible to transform (3.20):

$$L(\mathbf{x}) = (\mathbf{x} - \bar{\mathbf{z}}_1)^T \bar{\mathbf{B}}_1 (\mathbf{x} - \bar{\mathbf{z}}_1) + (\mathbf{x} - \bar{\mathbf{z}}_2)^T \bar{\mathbf{B}}_2 (\mathbf{x} - \bar{\mathbf{z}}_2),$$

where

$$\bar{\mathbf{z}}_1 = \mathbf{D}_1^{-1}\mathbf{z}_1, \ \ \bar{\mathbf{z}}_2 = \mathbf{D}_2^{-1}\mathbf{z}_2, \ \ \bar{\mathbf{B}}_1 = \mathbf{D}_1^T \mathbf{B}_1 \mathbf{D}_1, \ \ \bar{\mathbf{B}}_2 = \mathbf{D}_2^T \mathbf{B}_2 \mathbf{D}_2. \tag{3.26}$$

For such a representation of $L(\mathbf{x})$, in accordance with (3.22) and (3.23), we have:

$$\begin{aligned}
d = &\left[\bar{\mathbf{z}}_1 - (\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}(\bar{\mathbf{B}}_1\bar{\mathbf{z}}_1 + \bar{\mathbf{B}}_2\bar{\mathbf{z}}_2)\right]^T \bar{\mathbf{B}}_1\bar{\mathbf{z}}_1 + \left[\bar{\mathbf{z}}_2 - (\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}(\bar{\mathbf{B}}_1\bar{\mathbf{z}}_1 + \bar{\mathbf{B}}_2\bar{\mathbf{z}}_2)\right]^T \bar{\mathbf{B}}_2\bar{\mathbf{z}}_2 = \\
&\bar{\mathbf{z}}_1^T\bar{\mathbf{B}}_1\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_1^T\bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_2^T\bar{\mathbf{B}}_2(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1\bar{\mathbf{z}}_1 + \bar{\mathbf{z}}_2^T\bar{\mathbf{B}}_2\bar{\mathbf{z}}_2 - \\
&\qquad \bar{\mathbf{z}}_1^T\bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2\bar{\mathbf{z}}_2 - \bar{\mathbf{z}}_2^T\bar{\mathbf{B}}_2(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2\bar{\mathbf{z}}_2 = \\
&\bar{\mathbf{z}}_1^T\left(\bar{\mathbf{B}}_1 - \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1\right)\bar{\mathbf{z}}_1 - 2\bar{\mathbf{z}}_1^T\bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2\bar{\mathbf{z}}_2 + \bar{\mathbf{z}}_2^T\left(\bar{\mathbf{B}}_2 - \bar{\mathbf{B}}_2(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2\right)\bar{\mathbf{z}}_2.
\end{aligned}$$

It is easy to see that

$$\begin{aligned}
&\bar{\mathbf{B}}_1 - \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1 = \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2) - \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1 = \\
&\qquad \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2 - \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_1 = \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2.
\end{aligned}$$

Likewise,

$$\bar{\mathbf{B}}_2 - \bar{\mathbf{B}}_2(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2 = \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2.$$

So, we have

$$d = (\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_2)^T \bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2(\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_2).$$

In its turn,

$$\begin{aligned}
&\bar{\mathbf{B}}_1(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)^{-1}\bar{\mathbf{B}}_2 = \bar{\mathbf{B}}_1\left[\bar{\mathbf{B}}_2^{-1}(\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_2)\right]^{-1} = \bar{\mathbf{B}}_1\left(\bar{\mathbf{B}}_2^{-1}\bar{\mathbf{B}}_1 + \mathbf{I}\right)^{-1} = \bar{\mathbf{B}}_1\left(\bar{\mathbf{B}}_2^{-1}\bar{\mathbf{B}}_1 + \bar{\mathbf{B}}_1\bar{\mathbf{B}}_1^{-1}\bar{\mathbf{B}}_1\right)^{-1} = \\
&\quad \bar{\mathbf{B}}_1\left[\left(\bar{\mathbf{B}}_1^{-1} + \bar{\mathbf{B}}_2^{-1}\right)\bar{\mathbf{B}}_1\right]^{-1} = \bar{\mathbf{B}}_1\bar{\mathbf{B}}_1^{-1}\left(\bar{\mathbf{B}}_1^{-1} + \bar{\mathbf{B}}_2^{-1}\right)^{-1} = \left(\bar{\mathbf{B}}_1^{-1} + \bar{\mathbf{B}}_2^{-1}\right)^{-1},
\end{aligned}$$

whence we obtain

$$d = (\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_2)^T \left(\bar{\mathbf{B}}_1^{-1} + \bar{\mathbf{B}}_2^{-1}\right)^{-1}(\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_2),$$

what is equivalent to (3.24) with respect to (3.26).

Q.E.D.

Proof of Theorem 3 It follows from (3.6) and (3.18), that the Bellman functions at the leaves, in particular, at the nodes of the lowest level $t \in T^1$, will be quadratic

$$\tilde{J}_t(x_t) = \tilde{b}_t + (\mathbf{x_t} - \tilde{\mathbf{x}}_t)^T \tilde{\mathbf{Q}}_t(\mathbf{x_t} - \tilde{\mathbf{x}}_t), \ \tilde{b}_t = b_t^0, \ \tilde{\mathbf{x}}_t = \mathbf{x}_t^0, \ \tilde{\mathbf{Q}}_t = \mathbf{Q}_t^0.$$

If we assume, in accordance with this fact, the Bellman functions $\tilde{J}_s(x_s)$ at all the immediate descendants $s \in T^0_{(t)}$ of a node t to be quadratic, we will have, on the force of (3.4):

$$\tilde{J}_t(\mathbf{x}_t) = b^0_t + (\mathbf{x}_t - \mathbf{x}^0_t)^T \mathbf{Q}^0_t (\mathbf{x}_t - \mathbf{x}^0_t) + \sum_{s \in T^0_{(t)}} \min_{\mathbf{x}_s \in \mathsf{R}^n} \left\{ (\mathbf{x}_t - \mathbf{x}_s)^T \mathbf{U}_{t,s} (\mathbf{x}_t - \mathbf{x}_s) + \tilde{b}_s + (\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) \right\} =$$

$$b^0_t + \sum_{s \in T^0_{(t)}} \tilde{b}_s + (\mathbf{x}_t - \mathbf{x}^0_t)^T \mathbf{Q}^0_t (\mathbf{x}_t - \mathbf{x}^0_t) + \sum_{s \in T^0_{(t)}} \min_{\mathbf{x}_s \in \mathsf{R}^n} \left\{ (\mathbf{x}_t - \mathbf{x}_s)^T \mathbf{U}_{t,s} (\mathbf{x}_t - \mathbf{x}_s) + (\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) \right\}.$$

Application of (3.24) to each summand in the right sum gives

$$\tilde{J}_t(\mathbf{x}_t) = b^0_t + \sum_{s \in T^0_{(t)}} \tilde{b}_s + (\mathbf{x}_t - \mathbf{x}^0_t)^T \mathbf{Q}^0_t (\mathbf{x}_t - \mathbf{x}^0_t) + \sum_{s \in T^0_{(t)}} (\mathbf{x}_t - \tilde{\mathbf{x}}_s)^T \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1} (\mathbf{x}_t - \tilde{\mathbf{x}}_s),$$

whence it follows, on the force of (3.21) and (3.22), that

$$\tilde{J}_t(\mathbf{x}_t) = b^0_t + \sum_{s \in T^0_{(t)}} \tilde{b}_s + d + (\mathbf{x}_t - \tilde{\mathbf{x}}_t)^T \tilde{\mathbf{Q}}_t (\mathbf{x}_t - \tilde{\mathbf{x}}_t),$$

where

$$\tilde{\mathbf{Q}}_t = \mathbf{Q}^0_t + \sum_{s \in T^0_{(t)}} \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1}, \ \tilde{\mathbf{x}}_t = \tilde{\mathbf{Q}}_t^{-1} \left\{ \mathbf{Q}^0_t \mathbf{x}^0_t + \sum_{s \in T^0_{(t)}} \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1} \tilde{\mathbf{x}}_s \right\},$$

$$d = (\mathbf{x}^0_t - \tilde{\mathbf{x}}_t)^T \mathbf{Q}^0_t \mathbf{x}^0_t + \sum_{s \in T^0_{(t)}} (\tilde{\mathbf{x}}_s - \tilde{\mathbf{x}}_t)^T \left(\tilde{\mathbf{Q}}_s^{-1} + \mathbf{U}_{t,s}^{-1} \right)^{-1} \tilde{\mathbf{x}}_s,$$

what is equivalent to (3.19).

Q.E.D.

Theorem 5 *If the node and edge functions are quadratic functions of the structure (3.18) and satisfy the conditions of Theorem 3, the downward recurrent relations (3.5) are linear*

$$\tilde{\mathbf{x}}_s(\mathbf{x}_t) = (\mathbf{I} - \tilde{\mathbf{H}}_s)\tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s \mathbf{x}_t = \tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s (\mathbf{x}_t - \tilde{\mathbf{x}}_s), \ s \in T^0_{(t)}, \tag{3.27}$$

with transition matrices

$$\tilde{\mathbf{H}}_s = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} = \left(\mathbf{U}_{t,s}^{-1} \tilde{\mathbf{Q}}_s + \mathbf{I} \right)^{-1}. \tag{3.28}$$

Proof According to (3.5) and (3.18),

$$\tilde{\mathbf{x}}_s(\mathbf{x}_t) = \arg\min_{\mathbf{x}_s \in \mathsf{R}^n} \left\{ (\mathbf{x}_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}_s - \mathbf{x}_t) + (\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) \right\}.$$

Hence, it follows from (3.22) that

$$\tilde{\mathbf{x}}_s(\mathbf{x}_t) = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{Q}}_s \tilde{\mathbf{x}}_s + \mathbf{U}_{t,s} \mathbf{x}_t) = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \tilde{\mathbf{Q}}_s \tilde{\mathbf{x}}_s + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} \mathbf{x}_t = \left(\mathbf{I} - (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} \right) \tilde{\mathbf{x}}_s + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} \mathbf{x}_t.$$

The denotation $\tilde{\mathbf{H}}_s = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s}$ (3.28) gives (3.27). Finally, $(\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} = \left[\mathbf{U}_{t,s}^{-1} (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s}) \right]^{-1} = \left(\mathbf{U}_{t,s}^{-1} \tilde{\mathbf{Q}}_s + \mathbf{I} \right)^{-1}$, what proves the second identity in (3.28).

Q.E.D.

The first (upward) pass of the procedure starts with the initial assignment in accordance with (3.4) and (3.8) $\tilde{\mathbf{x}}_t = \mathbf{x}_t^0$, $\tilde{b}_t = b_t^0$, $\tilde{\mathbf{Q}}_t = \mathbf{Q}_t^0$, $t \in T^1$, $j = 1$, whereupon vectors $\tilde{\mathbf{x}}_t$, real numbers $\tilde{b}_t$ and matrices $\tilde{\mathbf{Q}}_t$ are recurrently recalculated for $t \in T^j$ at levels $j = 2, \ldots, M$ according to (3.10) and stored. The last step $j = M$ gives the final result for the root $\hat{\mathbf{x}}_{t^*} = \tilde{\mathbf{x}}_{t^*}$ (3.5). On the second (downward) pass $t \in T^j$, $j = 2, \ldots, M$, the set of nodes for which the final values of variables are already found propagates downwards in accordance with recurrent rule $\hat{\mathbf{x}}_s = \tilde{\mathbf{x}}_s(\hat{\mathbf{x}}_t)$, i.e.

$$\hat{\mathbf{x}}_s = (\mathbf{I} - \tilde{\mathbf{H}}_s) \tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s \hat{\mathbf{x}}_t = \tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s (\hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_s), \;\; s \in T_{(t)}^0, \tag{3.29}$$

which follows from (3.27).

It remains to specify the parametric version of the downward recurrent procedure of recalculating the marginal node functions (3.10) given by Theorem 2. The initial marginal node function at the root $\hat{J}_{t^*}(\mathbf{x}_{t^*})$, $\mathbf{x}_{t^*} \in \mathsf{R}^n$, is quadratic because it coincides with the quadratic Bellman function $\tilde{J}_{t^*}(\mathbf{x}_{t^*})$:

$$\hat{J}_{t^*}(\mathbf{x}_{t^*}) = \hat{b}_{t^*} + (\mathbf{x}_{t^*} - \hat{\mathbf{x}}_{t^*})^T \hat{\mathbf{Q}}_{t^*} (\mathbf{x}_{t^*} - \hat{\mathbf{x}}_{t^*}), \;\; \hat{b}_{t^*} = \tilde{b}_{t^*}, \;\; \hat{\mathbf{x}}_{t^*} = \tilde{\mathbf{x}}_{t^*}, \;\; \hat{\mathbf{Q}}_{t^*} = \tilde{\mathbf{Q}}_{t^*} \tag{3.30}$$

The following theorem states that the marginal node functions at all the descendant nodes remain likewise quadratic, and gives a recurrent rule of recalculating their parameters.

Theorem 6 *If the node and edge functions are quadratic functions of the structure (3.18) and meet the conditions of Theorem 3, all the marginal node functions are quadratic functions*

$$\hat{J}_t(\mathbf{x}_t) = \hat{b} + (\mathbf{x}_t - \hat{\mathbf{x}}_t)^T \hat{\mathbf{Q}}_t (\mathbf{x}_t - \hat{\mathbf{x}}_t) \tag{3.31}$$

with constant minimum value $\hat{b} \in \mathsf{R}$ found as that of the Bellman function at the root $\hat{b} = \hat{b}_{t^*} = \tilde{b}_{t^*}$ (3.30), and parameters $\hat{\mathbf{x}}_t \in \mathsf{R}^n$ and $\hat{\mathbf{Q}}_t$ $(n \times n)$ satisfying the downward recurrent relations, respectively, (3.29) and

$$\hat{\mathbf{Q}}_s = \left[\tilde{\mathbf{H}}_s \hat{\mathbf{Q}}_t^{-1} \tilde{\mathbf{H}}_s^T + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \right]^{-1}, \;\; s \in T_{(t)}^0, \tag{3.32}$$

which start at the root t^* with the initial values (3.30).

Proof It follows from the very definition of the marginal node functions (3.9) that the minimum value remains the same for all of them, and equals, for instance, the minimum value of the marginal node function and, hence, the Bellman function at the root $\hat{b} = \hat{b}_{t*} = \tilde{b}_{t*}$.

Since the marginal node function at the root is evidently quadratic, the quadraticity of all the marginal functions will be proven if we prove that it follows from the quadraticity of the marginal function at a node t that the marginal function at each of its immediate descendants $s \in T^0_{(t)}$ remains quadratic. Let the marginal node function at a node t be quadratic (3.31) and consider the marginal node function at one of the nodes $s \in T^0_{(t)}$. It follows from (3.10), (3.21) and (3.23) that

$$\begin{aligned} \hat{J}_s(\mathbf{x}_s) = \min_{\mathbf{x}_t \in \mathsf{R}^n} \Big\{ & \hat{b} + (\mathbf{x}_t - \hat{\mathbf{x}}_t)^T \hat{\mathbf{Q}}_t (\mathbf{x}_t - \hat{\mathbf{x}}_t) + \\ & \Big[\Big(\tilde{b}_s + (\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}_s - \mathbf{x}_t) \Big) - \\ & \min_{\mathbf{x}'_s \in \mathsf{R}^n} \Big(\tilde{b}_s + (\mathbf{x}'_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}'_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}'_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}'_s - \mathbf{x}_t) \Big) \Big] \Big\}, \end{aligned}$$

and, hence,

$$\begin{aligned} \hat{J}_s(\mathbf{x}_s) = \hat{b} + \min_{\mathbf{x}_t \in \mathsf{R}^n} \Big\{ & (\mathbf{x}_t - \hat{\mathbf{x}}_t)^T \hat{\mathbf{Q}}_t (\mathbf{x}_t - \hat{\mathbf{x}}_t) + \\ & \Big[\Big((\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}_s - \mathbf{x}_t) \Big) - \\ & \min_{\mathbf{x}'_s \in \mathsf{R}^n} \Big((\mathbf{x}'_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}'_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}'_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}'_s - \mathbf{x}_t) \Big) \Big] \Big\}, \end{aligned} \tag{3.33}$$

If we denote $L_{t,s}(\mathbf{x}_s) = (\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}_s - \mathbf{x}_t)$, we obtain from (3.20), (3.21) and (3.22) that the expression in the square brackets in (3.33) is quadratic form

$$\begin{aligned} & \Big[\Big((\mathbf{x}_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}_s - \mathbf{x}_t) \Big) - \\ & \min_{\mathbf{x}'_s \in \mathsf{R}^n} \Big((\mathbf{x}'_s - \tilde{\mathbf{x}}_s)^T \tilde{\mathbf{Q}}_s (\mathbf{x}'_s - \tilde{\mathbf{x}}_s) + (\mathbf{x}'_s - \mathbf{x}_t)^T \mathbf{U}_{t,s} (\mathbf{x}'_s - \mathbf{x}_t) \Big) \Big] = \\ & \qquad (\mathbf{x}_s - \mathbf{z}_{t,s})^T \mathbf{B}_{t,s} (\mathbf{x}_s - \mathbf{z}_{t,s}), \end{aligned} \tag{3.34}$$

where $\mathbf{B}_{t,s} = \tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s}$, $\mathbf{z}_{t,s} = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{Q}}_s \tilde{\mathbf{x}}_s + \mathbf{U}_{t,s} \mathbf{x}_t)$. So, we have for the argument of this quadratic form:

$$\mathbf{x}_s - \mathbf{z}_{t,s} = \mathbf{x}_s - (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{Q}}_s \tilde{\mathbf{x}}_s + \mathbf{U}_{t,s} \mathbf{x}_t) = \mathbf{x}_s - (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \tilde{\mathbf{Q}}_s \tilde{\mathbf{x}}_s - (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} \mathbf{x}_t.$$

The matrix coefficient at $\mathbf{x}_t$ is nothing else than $\tilde{\mathbf{H}}_s$ (3.28), and that at $\tilde{\mathbf{x}}_s$ equals $\mathbf{I} - \tilde{\mathbf{H}}_s$, because $(\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \tilde{\mathbf{Q}}_s + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \mathbf{U}_{t,s} = (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s}) = \mathbf{I}$. Thus,

$$\mathbf{x}_s - \mathbf{z}_{t,s} = \mathbf{x}_s - (\mathbf{I} - \tilde{\mathbf{H}}_s) \tilde{\mathbf{x}}_s - \tilde{\mathbf{H}}_s \mathbf{x}_t = \mathbf{x}_s - \tilde{\mathbf{x}}_s - \tilde{\mathbf{H}}_s (\mathbf{x}_t - \tilde{\mathbf{x}}_s).$$

The successive changes of variables

$$\mathbf{z}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t,$$

then

$$\hat{\mathbf{x}}_s = \tilde{\mathbf{x}}_s + \tilde{\mathbf{H}}_s(\hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_s), \tag{3.35}$$

and, finally,

$$\mathbf{z}_s = \mathbf{x}_s - \hat{\mathbf{x}}_s, \tag{3.36}$$

give

$$\mathbf{x}_s - \mathbf{z}_{t,s} = \mathbf{z}_s - \tilde{\mathbf{H}}_s\mathbf{z}_t.$$

The substitution of this identity into (3.34) and then into (3.33) with respect to the fact, that, in accordance with (3.28), matrix $\tilde{\mathbf{H}}_s$ is nonsingular, gives

$$\hat{J}_s(\mathbf{x}_s) = \hat{b} + \min_{\mathbf{z}_t \in \mathsf{R}^n} \left\{ \mathbf{z}_t^T \hat{\mathbf{Q}}_t \mathbf{z}_t + (\mathbf{z}_s - \tilde{\mathbf{H}}_s\mathbf{z}_t)^T (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})(\mathbf{z}_s - \tilde{\mathbf{H}}_s\mathbf{z}_t) \right\} =$$
$$\hat{b} + \min_{\mathbf{z}_t \in \mathsf{R}^n} \left\{ (\mathbf{z}_t - \mathbf{0})^T \hat{\mathbf{Q}}_t (\mathbf{z}_t - \mathbf{0}) + (\tilde{\mathbf{H}}_s\mathbf{z}_t - \mathbf{z}_s)^T (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})(\tilde{\mathbf{H}}_s\mathbf{z}_t - \mathbf{z}_s) \right\},$$

and, according to (3.24),

$$\hat{J}_s(\mathbf{x}_s) = \hat{b} + \mathbf{z}_s^T (\tilde{\mathbf{H}}_s^{-1})^T \left[\hat{\mathbf{Q}}_t^{-1} + \tilde{\mathbf{H}}_s^{-1} (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{H}}_s^{-1})^T \right]^{-1} \tilde{\mathbf{H}}_s^{-1} \mathbf{z}_s.$$

The matrix of this quadratic form can be identically transformed with respect to the fact that $(\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1}$ for any nonsingular matrix, because $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$, whence we have $(\mathbf{A}^{-1}\mathbf{A})^T = \mathbf{I}$ and $\mathbf{A}^T(\mathbf{A}^{-1})^T = \mathbf{I}$, i.e. $(\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1}$. We obtain:

$$(\tilde{\mathbf{H}}_s^{-1})^T \left\{ \left[\hat{\mathbf{Q}}_t^{-1} + \tilde{\mathbf{H}}_s^{-1} (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{H}}_s^{-1})^T \right]^{-1} \tilde{\mathbf{H}}_s^{-1} \right\} =$$
$$(\tilde{\mathbf{H}}_s^{-1})^T \left\{ \tilde{\mathbf{H}}_s \left[\hat{\mathbf{Q}}_t^{-1} + \tilde{\mathbf{H}}_s^{-1} (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{H}}_s^{-1})^T \right] \right\}^{-1} = (\tilde{\mathbf{H}}_s^T)^{-1} \left[\tilde{\mathbf{H}}_s \hat{\mathbf{Q}}_t^{-1} + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{H}}_s^{-1})^T \right]^{-1} =$$
$$\left\{ \left[\tilde{\mathbf{H}}_s \hat{\mathbf{Q}}_t^{-1} + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} (\tilde{\mathbf{H}}_s^T)^{-1} \right] \tilde{\mathbf{H}}_s^T \right\}^{-1} = \left[\tilde{\mathbf{H}}_s \hat{\mathbf{Q}}_t^{-1} \tilde{\mathbf{H}}_s^T + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \right]^{-1}.$$

Hence, with respect to (3.36) we come to the identity

$$\hat{J}_s(\mathbf{x}_s) = \hat{b} + (\mathbf{x}_s - \hat{\mathbf{x}}_s)^T \hat{\mathbf{Q}}_s (\mathbf{x}_s - \hat{\mathbf{x}}_s),$$

where $\hat{\mathbf{Q}}_s = \left[\tilde{\mathbf{H}}_s \hat{\mathbf{Q}}_t^{-1} \tilde{\mathbf{H}}_s + (\tilde{\mathbf{Q}}_s + \mathbf{U}_{t,s})^{-1} \right]^{-1}$, and $\hat{\mathbf{x}}_s$ (3.35) satisfies (3.29).

Q.E.D.

Up to now, we studied the ways of numerical realization of the basic optimization procedure built in Section 3.2 for separable functions with tree-like separability of the general kind. It remains to apply the results obtained to its particular case (3.13)-(3.17) considered in Section 3.3.

Just as in the general case, no assumptions on the node and edge functions are required if the arguments of the objective function take values from a finite set $\mathcal{X} = \{1, \ldots, m\}$. In this case, the Bellman functions $\tilde{J}_t(x_t)$ and marginal node functions $\hat{J}_t(x_t)$ are completely representable as vectors with m elements each, and the backward recurrent relations $\tilde{x}_{t-1}(x_t)$ can be stored as vectors of the same structure. On the forward pass of the procedure, it is required to store $N-1$ such matrices, which will be used on the backward pass for recalculating the optimal values $\hat{x}_t$ and, if required, vector representations of the marginal functions.

In the next Section, the discrete version of the dynamic programming procedure will be involved in the algorithms of image segmentation and multi-alignment of molecular sequences.

In the case of real-valued vector variables $\mathbf{x}_t \in \mathsf{R}^n$ and quadratic node and edge functions of the same structure as in (3.18)

$$\psi_t(\mathbf{x}_t) = b_t^0 + (\mathbf{x}_t - \mathbf{x}_t^0)^T \mathbf{Q}_t^0 (\mathbf{x}_t - \mathbf{x}_t^0), \quad \gamma_t(\mathbf{x}_{t-1}, \mathbf{x}_t) = (\mathbf{x}_{t-1} - \mathbf{x}_t)^T \mathbf{U}_t (\mathbf{x}_{t-1} - \mathbf{x}_t),$$

in accordance with Theorems 3, 5 and 6, the Bellman functions, backward recurrent relations and marginal node functions will be quadratic, too.

Starting with the initial assignment (3.13)

$$\tilde{b}_1 = b_1^0, \quad \tilde{\mathbf{x}}_1 = \mathbf{x}_1^0, \quad \tilde{\mathbf{Q}}_1 = \mathbf{Q}_1^0,$$

the forward pass of the procedure will consist in recurrent recalculation of matrix representations of Bellman functions $\tilde{J}_t(\mathbf{x}_t) = \tilde{b}_t + (\mathbf{x}_t - \tilde{\mathbf{x}}_t)^T \tilde{\mathbf{Q}}_t (\mathbf{x}_t - \tilde{\mathbf{x}}_t)$ (3.14), (3.19)

$$\begin{aligned}
&\tilde{\mathbf{Q}}_t = \mathbf{Q}_t^0 + \left(\tilde{\mathbf{Q}}_{t-1}^{-1} + \mathbf{U}_t^{-1}\right)^{-1}, \quad \tilde{\mathbf{x}}_t = \tilde{\mathbf{Q}}_t^{-1} \left\{ \mathbf{Q}_t^0 \mathbf{x}_t^0 + \left(\tilde{\mathbf{Q}}_{t-1}^{-1} + \mathbf{U}_t^{-1}\right)^{-1} \tilde{\mathbf{x}}_{t-1} \right\}, \\
&\tilde{b}_t = b_t^0 + (\mathbf{x}_t^0 - \tilde{\mathbf{x}}_t)^T \mathbf{Q}_t^0 \mathbf{x}_t^0 + \left\{ \tilde{b}_{t-1} + (\tilde{\mathbf{x}}_{t-1} - \tilde{\mathbf{x}}_t)^T \left(\tilde{\mathbf{Q}}_{t-1}^{-1} + \mathbf{U}_t^{-1}\right)^{-1} \tilde{\mathbf{x}}_{t-1} \right\}, \\
&t = 2, \ldots, N,
\end{aligned}$$

and recalculation and storing of the backward recurrent relations $\tilde{\mathbf{x}}_{t-1}(\mathbf{x}_t)$ (3.15), (3.27), (3.28)

$$\tilde{\mathbf{x}}_{t-1}(\mathbf{x}_t) = (\mathbf{I} - \tilde{\mathbf{H}}_{t-1}) \tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{H}}_{t-1} \mathbf{x}_t = \tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{H}}_{t-1} (\mathbf{x}_t - \tilde{\mathbf{x}}_{t-1}),$$

$$\tilde{\mathbf{H}}_{t-1} = (\tilde{\mathbf{Q}}_{t-1} + \mathbf{U}_t)^{-1} \mathbf{U}_t = \left(\mathbf{U}_t^{-1} \tilde{\mathbf{Q}}_{t-1} + \mathbf{I}\right)^{-1}.$$

The result of the last step immediately gives the optimal value of the last variable $\hat{\mathbf{x}}_N$ along with the matrix representation of its marginal function $\hat{J}_N(\mathbf{x}_N) = \hat{b} + (\mathbf{x}_N - \hat{\mathbf{x}}_N)^T \hat{\mathbf{Q}}_N (\mathbf{x}_N - \hat{\mathbf{x}}_N)$ (3.16), (3.30):

$$\hat{b} = \tilde{b}_N, \ \hat{\mathbf{x}}_N = \tilde{\mathbf{x}}_N, \ \hat{\mathbf{Q}}_N = \tilde{\mathbf{Q}}_N.$$

On the backward pass, the procedure recurrently finds the final values of all the variables as well as parameters of their marginal functions $\hat{J}_t(\mathbf{x}_t) = \hat{b} + (\mathbf{x}_t - \hat{\mathbf{x}}_t)^T \hat{\mathbf{Q}}_t (\mathbf{x}_t - \hat{\mathbf{x}}_t)$ in accordance with (3.17), (3.29), (3.32):

$$\hat{\mathbf{x}}_{t-1} = (\mathbf{I} - \tilde{\mathbf{H}}_{t-1})\tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{H}}_{t-1}\hat{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{H}}_{t-1}(\hat{\mathbf{x}}_t - \tilde{\mathbf{x}}_{t-1}),$$

$$\hat{\mathbf{Q}}_{t-1} = \left[\tilde{\mathbf{H}}_{t-1}\hat{\mathbf{Q}}_t^{-1}\tilde{\mathbf{H}}_{t-1}^T + (\tilde{\mathbf{Q}}_{t-1} + \mathbf{U}_t)^{-1}\right]^{-1},$$

$$t = N, \ldots, 2.$$

It is just this parametric version of the dynamic programming procedure that will be immediately used in the next Section in the algorithms of image smoothing and matching.

4 Algorithms of data analysis

4.1 The generalized data analysis procedure

Let the particular data analysis problem from the class outlined in Introduction and Section 2.1 be already reduced to the formal problem of minimization of a separable objective function with two-dimensional (2.2) or three-dimensional (2.2) lattice-like separability. In accordance with the optimization-based approach to data analysis (Section 2.2), it means that the specificity of both the original data array $\mathbf{Y}$ and the secondary resulting array $\mathbf{X}$ is completely expressed by the appropriate choice of, first, the range of the hidden variable $x \in \mathcal{X}$ and, second, the node and edge functions which occur in the objective function as its summands, respectively, $\psi(x_{\mathbf{t}}|\mathbf{Y}_{\mathbf{t}})$, $\gamma'(x_{\mathbf{t}'}, x_{\mathbf{t}''})$ and $\gamma''(x_{\mathbf{t}'}, x_{\mathbf{t}''})$, where $\mathbf{t} = (t_1, t_2)$ or $\mathbf{t} = (t_1, t_2, t_3)$.

Having replaced the adjacency graph of the full objective function by a series of trees (Fig. 2.8), we have substituted the original optimization problem by a series of partial problems (2.6), each meant to evaluate only one row of the goal variables in the resulting array with the hope that their assembly will be a satisfactory approximation of the "ideal" decision (2.3).

The theoretical framework developed in the previous Section suggests a computationally effective procedure of optimization of a separable function with tree-like separability of arguments. It appears to be natural to construct the algorithm of data analysis as plain repetition of the basic optimization procedure being applied, in turn, to each of the partial objective functions. But such a straightforward algorithm would be no good decision, because, to form the resulting array, it would be necessary to evaluate, every time, all the variables of the particular array $\hat{\mathbf{X}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})$ in accordance with (2.6), whereas only those of the stem row $\hat{\mathbf{x}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})$ participate in the final combined result of data processing $\hat{\mathbf{X}}(\mathbf{Y}) = [\hat{\mathbf{x}}_{\bar{\mathbf{t}}^*}(\mathbf{Y})]$.

The notion of marginal node functions (3.9) introduced in Section 3.2 gives a possibility to create a procedure which, having scanned a two- or three-dimensional rectangular array only twice, would find the solutions of the partial optimization problems (2.6) for all the approximating trees with common horizontal branches as in Fig. 2.8, and escape evaluating redundant variables in their branches in addition to those in the stems.

To start with, consider a plane rectangular array of the structure as in Fig. 2.7,a. In accordance with (2.7), to form the column t_1^* of the final resulting array, we have to find the optimal particular array $\hat{\mathbf{X}}_{t_1^*}(\mathbf{Y})$ (2.6), cut the respective column $\hat{\mathbf{x}}_{t_1^*}(\mathbf{Y})$ out of it, and put that column in the final array. We will avoid the necessity of evaluating the unneeded variables, if find a way to calculate the stem elements of the arrays $\hat{\mathbf{X}}_{t_1^*}(\mathbf{Y})$ immediately, without those beyond the stems.

It is easy to see that, in accordance with (2.5) and (2.6), each of the sought-for stem columns is determined by the condition

$$\hat{\mathbf{x}}_{t_1^*}(\mathbf{Y}) = \Big(\hat{x}_{t_1^*1}(\mathbf{Y}), \dots, \hat{x}_{t_1^*N_2}(\mathbf{Y})\Big) = \arg\min_{x_{t_1^*t_2}} \left(\min_{x_{t_1t_2},\, t_1\neq t_1^*} J_{t_1^*}(\mathbf{X}|\mathbf{Y}) \right),$$

or, what is the same,

$$\begin{aligned} &\hat{\mathbf{x}}_{t_1^*}(\mathbf{Y}) = \Big(\hat{x}_{t_1^*1}(\mathbf{Y}), \dots, \hat{x}_{t_1^*N_2}(\mathbf{Y})\Big) = \arg\min \hat{J}_{t_1^*}(x_{t_1^*1}, \dots, x_{t_1^*N_2}|\mathbf{Y}), \\ &\hat{J}_{t_1^*}(x_{t_1^*1}, \dots, x_{t_1^*N_2}|\mathbf{Y}) = \sum_{t_2=1}^{N_2} \hat{J}_{t_1^*t_2}(x_{t_1^*t_2}|\mathbf{Y}_{1t_2}, \dots, \mathbf{Y}_{N_1t_2}) + \sum_{t_2=2}^{N_2} \gamma'(x_{t_1^*t_2-1}, x_{t_1^*t_2}), \end{aligned} \tag{4.1}$$

where

$$\hat{J}_{t_1^*t_2}(x_{t_1^*t_2}|\mathbf{Y}_{1t_2}, \dots, \mathbf{Y}_{N_1t_2}) = \min_{x_{t_1t_2},\, t_1\neq t_1^*} J_{t_2}(x_{1t_2}, \dots, x_{N_1t_2}|\mathbf{Y}_{1t_2}, \dots, \mathbf{Y}_{N_1t_2}),$$

and, in its turn,

$$J_{t_2}(x_{1t_2}, \dots, x_{N_1t_2}|\mathbf{Y}_{1t_2}, \dots, \mathbf{Y}_{N_1t_2}) = \sum_{t_1=1}^{N_1} \psi(x_{t_1t_2}|\mathbf{Y}_{t_1t_2}) + \sum_{t_1=2}^{N_1} \gamma''(x_{t_1-1,t_2}, x_{t_1t_2}). \tag{4.2}$$

In what follows, we will omit the upper asterisk t_1^* at indices t_1, because it is no longer required to mark a single column out of the rest of the array.

So, each of the stem columns $\hat{\mathbf{x}}_{t_1}(\mathbf{Y}) = \Big(\hat{x}_{t_11}(\mathbf{Y}), \dots, \hat{x}_{t_1N_2}(\mathbf{Y})\Big)$, $t_1 = 1, \dots, N_1$, can be found independently of the others as result of minimization of the respective objective function $\hat{J}_{t_1}(x_{t_11}, \dots, x_{t_1N_2}|\mathbf{Y})$ (4.1). Each of these functions is separable with adjacency graph having the form of a chain, i.e. the simplest tree without branching nodes. The basic optimization procedure which follows from the theoretical framework developed in Section 3, to be exact, its simplest version reported in Section 3.3, will find the optimal values of all the variables forming the resulting array, after it runs twice through each column t_1, for instance, first from top to bottom $t_2 = 1, \dots, N_2$, and then from bottom to top $t_2 = N_2, \dots, 1$. The only thing we have to do before is to find, for all the elements of the resulting array (t_1, t_2), $t_1 = 1, \dots, N_1$, $t_2 = 1, \dots, N_2$, the single-variable functions $\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2}, \dots, \mathbf{Y}_{N_1t_2})$ (4.1) that occur in (4.1) as data-dependent node functions.

Each of these functions is the conditional minimum value of function (4.2) when one of its arguments, namely, that in the respective column t_1, is fixed. So, in accordance with the definition introduced in Section 3.2, function $\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2})$ is one of the marginal functions of the separable function $J_t(x_{1t_2},\ldots,x_{N_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2})$. Mark, that its adjacency graph is the chain formed by horizontal branch t_2 which is common for all the adjacency trees (Fig. 2.8) of the partial objective functions $J_{t_1^*}(\mathbf{X}|\mathbf{Y})$. Thus, all the marginal functions in a horizontal row $\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2})$ (4.1), $t_1=1,\ldots,N_1$, can be found at once by the same simplest version of the basic optimization procedure for separable functions with chain-like adjacency of variables (Section 3.3). Having run twice through each row t_2 independently, for instance, from left to right $t_1=1,\ldots,N_1$ and from right to left $t_1=N_1,\ldots,1$, the procedure will define all the single-variable functions $\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2})$ (4.1), $t_1=1,\ldots,N_1$, $t_2=1,\ldots,N_2$, required for the final evaluation of the goal variables $\hat{x}_{t_1t_2}(\mathbf{Y})$.

Now, we are ready to put down the generalized data analysis procedure, so far, for two-dimensional arrays. Let $\mathbf{Y}=(y_{t_1t_2},\, t_1=1\ldots,N_1,\, t_2=1,\ldots,N_2)$ be such an array. It does not matter what is the nature of its elements, and what is that of the desired result of processing $\mathbf{X}=(x_{t_1t_2},\, t_1=1\ldots,N_1,\, t_2=1,\ldots,N_2)$, it is enough to assume the range of the goal variables $x_{t_1t_2}\in\mathcal{X}$ and the respective node $\psi(x_{t_1t_2}|\mathbf{Y})$ and edge functions $\gamma'(x_{t_1t_2-1},x_{t_1t_2})$, $\gamma''(x_{t_1-1,t_2},x_{t_1t_2})$ occurring in the separable objective function (2.2) which will be, therethrough, completely defined.

The array is to be processed in two stages, the first of which will be called the horizontal processing and the second the vertical one. However, this order is conventional to some extent, and may be inverted if this does not come into conflict with the nature of the original data array and that of the conceived result of its processing. The first stage is applied to the original array $\mathbf{Y}=(y_{t_1t_2};\ t_1=1,\ldots,N_1,\ t_2=1,\ldots,N_2)$ and results in the intervening array of marginal functions

$$\hat{\mathbf{J}}(\mathbf{Y})=\Big(\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2});\, t_1=1,\ldots,N_1,\ t_2=1,\ldots,N_2\Big), \tag{4.3}$$

which is the input for the second stage resulting in the final array $\hat{\mathbf{X}}(\mathbf{Y})=\Big(\hat{x}_{t_1t_2}(\mathbf{Y});\, t_1=1,\ldots,N_1,\ t_2=1,\ldots,N_2\Big)$.

The horizontal processing consists in that the dynamic programming procedure (3.13)-(3.17) is applied independently to each horizontal row $(1t_2,\ldots,N_1t_2)$, $t_2=1,\ldots,N_2$, but without calculating the optimal values $\hat{x}_{t_1t_2}$ on the backward pass, so that only marginal functions $\hat{J}_{t_1t_2}(x_{t_1t_2})=\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2})$, $x_{t_1t_2}\in\mathcal{X}$, are stored at each position (t_1,t_2).

In the course of *the vertical processing*, the same dynamic programming procedure runs independently through each vertical row $(t_11,\ldots,t_1N_2)$, $t_1=1,\ldots,N_1$, with two alterations relative to (3.13)-(3.17): first, the marginal functions $\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2},\ldots,\mathbf{Y}_{N_1t_2})$ found as result of the first stage should occur in (3.13) and (3.14) instead of $\psi(x_{t_1t_2}|\mathbf{Y}_{t_1t_2})$, and, second, on the backward pass it is enough to calculate only the optimal values of variables $\hat{x}_{t_1t_2}$ and take them as elements $\hat{x}_{t_1t_2}(\mathbf{Y})$ of the final result.

However, if having been calculated, each of the marginal functions $\hat{J}_{t_1t_2}(x_{t_1t_2}) = \hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y})$ would be marginal function of one of the stem variables of the respective partial objective function (2.5)

$$\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}) = \min_{x_{t_1' t_2}, t_1' \neq t_1} J_{t_1}(\mathbf{X}|\mathbf{Y}).$$

Thus, the array of such functions would show, in addition to the optimal values of variables, the manner in which each of them affects the respective partial criterion when being individually varied with optimal values of other variables.

For cubic data arrays of the structure shown in Fig. 2.7,b, the algorithm will be little more complicated than for the plane case. The only difference is that there will be two horizontal runs instead of one, let them be called the lateral and the frontal processing. As a result, three stages of processing are required instead of two. If the adjacency trees substituting the full graph are chosen as in Fig. 2.8,b, the lateral processing must be done first, the processing that follows must be the frontal one, and the vertical processing will be the final stage. However, any inversions are possible in the general case.

The argumentation which leads to the generalized procedure of the analysis of cubic data arrays remains practically the same as that set out above in this Section for plane arrays. Therefore, let us pass over its repeating once again and go directly to the presentation of the procedure.

At the stage of *lateral processing*, the dynamic programming procedure (3.13)-(3.17) is applied independently to each lateral horizontal row $(1t_2t_3, \ldots, N_1t_2t_3)$ with storing marginal functions found at the backward pass. As a result, the original array $\mathbf{Y} = (y_{t_1t_2t_3};\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2,\ t_3 = 1, \ldots, N_3)$ will be transformed into the first intervening array of marginal functions

$$\hat{\mathbf{J}}^*(\mathbf{Y}) = \left(\hat{J}^*_{t_1t_2t_3}(x_{t_1t_2t_3}|\mathbf{Y}_{t_2t_3});\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2,\ t_3 = 1, \ldots, N_3\right) \qquad (4.4)$$

where $\mathbf{Y}_{t_2t_3} = (\mathbf{Y}_{t_1t_2t_3};\ t_1 = 1, \ldots, N_1)$ is tunnel-like union of elementary fragments of the original data cube along the respective horizontal row (t_2, t_3).

At the stage of *frontal processing* the same procedure, running in the same manner independently along each frontal row $(t_11t_3, \ldots, t_1N_2t_3)$ and using the respective functions from (4.4) instead of $\psi(x_{t_1t_2t_3}|\mathbf{Y}_{t_1t_2t_3})$ in (3.13) and (3.14), transforms the array $\hat{\mathbf{J}}^*$, in its turn, into the second intervening array of marginal functions

$$\hat{\mathbf{J}}^{**}(\mathbf{Y}) = \left(\hat{J}^{**}_{t_1t_2t_3}(x_{t_1t_2t_3}|\mathbf{Y}_t);\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2,\ t_3 = 1, \ldots, N_3\right) \qquad (4.5)$$

where $\mathbf{Y}_{t_3} = (\mathbf{Y}_{t_1t_2t_3};\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2)$ is, for this time, layer-like union of elementary fragments of the cube at level t_3.

Finally, at the stage of *vertical processing* the dynamic programming procedure runs independently along each vertical row $(t_1t_21, \ldots, t_1t_2N_3)$ with the respective elements of (4.5) instead of $\psi(x_{t_1t_2t_3}|\mathbf{Y}_{t_1t_2t_3})$ and stores, at the backward pass, the optimal values of variables as elements of the resulting array $\hat{\mathbf{X}}(\mathbf{Y})$.

To create a particular algorithm of data analysis, it remains only to choose the node and edge functions along with the range of the goal variables in accordance with the nature of the respective application, and, on the basis of this choice, to specify the appropriate parametric representation of the Bellman functions, i.e., what is the same, parametric representation of the marginal functions forming the intervening data arrays between the processing stages. It is just this circle of questions which will be the subject of our consideration in the next Section.

4.2 Visual data analysis

In the problem of segmentation of a texture image into homogeneous regions of a preset number of classes, the goal variable $x_{t_1t_2}$ takes values from the finite set of class indices $\mathcal{X} = \{1, \dots, m\}$. The simplest statement of the segmentation problem we consider here consists in the requirement to partition the pixel grid into an appropriate number of sufficiently prolonged regions of m classes each of which is associated with a known kind of texture [19, 20, 21].

Just as in all the optimization-based statements of data analysis problems, the node function $\psi(x|\mathbf{Y}_{t_1t_2})$ is meant to provide immediate mining of the data-dependent information in the form of a measure of discrepancy between the assumed value of the hidden variable at point (t_1, t_2), in this case, texture class $x = 1, \dots, m$, and the current fragment of the original data array $\mathbf{Y}_{t_1t_2}$, in this case, of the image. For instance, it is convenient to construct such a function on the basis of treating a homogeneous texture as a realization of a two-dimensional linear autoregressive random field

$$y_{t_1t_2} - \left(a^x_{00} + \sum_{(i_1,i_2)\in\mathcal{F}} a^x_{i_1i_2} y_{t_1+i_1,t_2+i_2} \right) = \sigma^x \xi_{t_1t_2}, \tag{4.6}$$

where autoregression mask $\mathcal{F}$, i.e. set of admissible shifts from the current point in the pixel grid, determines the form $\mathbf{Y}_{t_1t_2}$ of the elementary image fragment, $\xi_{t_1t_2}$ is discrete normal white noise with zero mathematical expectation and variance taken to be equal to unity, and $\mathbf{c}^x = (a^x_{00},\, a^x_{i_1i_2},\, \sigma^x)$ is autoregression parameter vector meant to be known for each texture class $x \in \mathcal{X} = \{1, \dots, m\}$. In accordance with such a model, the conditional distribution of the "random" value $y_{t_1t_2}$ of the image darkness at the current point with respect to the rest $\mathbf{y}_{t_1t_2}$ of its elementary vicinity $\mathbf{Y}_{t_1t_2}$ will be also normal

$$\varphi^x(y_{t_1t_2}|\mathbf{y}_{t_1t_2}) = \frac{1}{\sigma^x\sqrt{2\pi}} \exp\Big[-\frac{1}{2(\sigma^x)^2}\Big(y_{t_1t_2} - a^x_{00} - \sum_{(i_1,i_2)\in\mathcal{F}} a^x_{i_1i_2} y_{t_1+i_1,t_2+i_2} \Big)^2 \Big],$$

and the posterior distribution over the finite set of hypotheses on the texture class without taking into account any prior information will be just the appropriate node function:

$$\psi_{t_1t_2}(x) = \psi(x|\mathbf{Y}_{t_1t_2}) = \mathrm{P}(x_{t_1t_2} = x|\mathbf{Y}_{t_1t_2}) = \frac{\varphi^x(y_{t_1t_2}|\mathbf{y}_{t_1t_2})}{\sum_{x'=1}^m \varphi^{x'}(y_{t_1t_2}|\mathbf{y}_{t_1t_2})}, \quad x \in \mathcal{X} = \{1, \ldots, m\}.$$

The purpose of the edge functions $\gamma'(x_{t_1-1,t_2}, x_{t_1t_2})$ and $\gamma''(x_{t_1,t_2-1}, x_{t_1t_2})$ is to prevent too detailed segmentation, respectively, in the horizontal and vertical direction in the image plane. It is hardly reasonable to distinguish between the two directions in the case of visual images. As the simplest version of the model-based edge function, meant to express the discrepancy between two neighboring values of the discrete hidden class-membership index, we used coincidence function

$$\gamma(x_{t'_1t'_2}, x_{t''_1t''_2}) = \begin{cases} 0, & \text{if } x_{t'_1t'_2} = x_{t''_1t''_2}, \\ \alpha, & \text{if } x_{t'_1t'_2} \neq x_{t''_1t''_2}, \end{cases}$$

where the constant $\alpha > 0$ must be taken the greater, the more prolonged the sought-for areas of homogeneous texture are expected to be.

Once the texture parameters $\mathbf{c}^x$ are known for all the classes of homogeneous regions $x = 1, \ldots, m$, segmentation of an image can be easily obtained by the two-dimensional version of the generalized data-analysis procedure described above in Section 4.1. Because of the discrete range of the goal variables, the Bellman function and marginal node functions are completely represented as vectors of m real numbers, and, so, numerical realization of the generalized procedure is not concerned with the problem of finding an appropriate parametric representation for them. At the first stage, the image to be segmented $\mathbf{Y} = (y_{t_1t_2};\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2)$ is transformed into the array (4.3) $\hat{\mathbf{J}}(\mathbf{Y})$ of marginal functions $\hat{J}_{t_1t_2}(x_{t_1t_2}|\mathbf{Y}_{1t_2}, \ldots, \mathbf{Y}_{N_1t_2})$ represented each by m-dimensional real vector of its values, and, so, the intervening information to be stored is m times as voluminous as the original image. The final segmentation is obtained from $\hat{\mathbf{J}}(\mathbf{Y})$ at the second stage as discretely valued array $\hat{\mathbf{X}}(\mathbf{Y}) = \big(\hat{x}_{t_1t_2}(\mathbf{Y});\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2\big)$.

However, it is not quite realistic to assume the "tentative" parameters of the texture types $\mathbf{c}^x$ to be chosen in advance. These parameters should be rather estimated immediately in the course of image analysis. We don't study this question in this work, and restrict us only to the remark, that the respective iterative technique is described in [19]. It is just this algorithm by which the segmentation of the texture image in Fig. 2.1 was obtained.

The problem of smoothing appears to be the simplest example of a data analysis problem for which quadratic node and edge functions are evidently suitable bases of its optimization-based mathematical statement [20]. If the image darkness is represented by real numbers $y_{t_1t_2} \in \mathcal{Y} = \mathsf{R}$, the result of smoothing will consist of real numbers, too, $x_{t_1t_2} \in \mathcal{X} = \mathsf{R}$. Smoothing of a picture does not pursue the aim of any substantial revision of its essence but estimating the average darkness in a vicinity of each pixel. So, the smoothed picture must be as close to the original as possible and, simultaneously, as smooth as possible. In the version of smoothing technique proposed here, we allocate the responsibility for these two mutually contradictory requirements, respectively, to the node and the edge functions.

From the viewpoint of the intent to retain the original image, the node function $\psi_{t_1t_2}(x)$ should reach its minimum value when x coincides with the actual darkness at the current pixel $y_{t_1t_2}$. Because no further preferences are meant to be acquirable from the original image, the node function must be symmetric relative to this point. The simplest quadratic function will be the appropriate choice:

$$\psi_{t_1t_2}(x) = \psi(x|y_{t_1t_2}) = (x - y_{t_1t_2})^2, \quad x \in \mathcal{X} = \mathsf{R}. \tag{4.7}$$

So, the elementary vicinity of the original image consists, in this case, of only one image element $\mathbf{Y}_{t_1t_2} = y_{t_1t_2}$.

Like as in other problems of visual image analysis, there is no reason to take different edge functions for different directions in the image plane. It is just the purpose of the edge function $\gamma(x_{t'_1t'_2}, x_{t''_1t''_2})$ to provide the desired smoothing effect by imposing a penalty upon the eventual difference of neighboring goal variables $x_{t'_1t'_2}$ and $x_{t''_1t''_2}$. It can be taken in the simplest form of a quadratic function with an additional coefficient:

$$\gamma(x_{t'_1t'_2}, x_{t''_1t''_2}) = \alpha\,(x_{t'_1t'_2} - x_{t''_1t''_2})^2. \tag{4.8}$$

The greater is coefficient $\alpha > 0$, the deeper will be the resulting smoothing.

To smooth an image $\mathbf{Y} = (y_{t_1t_2};\ t_1 = 1,\dots,N_1,\, t_2 = 1,\dots,N_2)$, it is enough to apply to it the two-dimensional version of the generalized procedure with node and edge functions (4.7), (4.8), and with quadratic representation of the Bellman and marginal node functions. Each quadratic marginal function $\hat{J}_{t_1t_2}(x_{t_1t_2}|y_{1t_1},\dots,y_{N_1t_2})$ in the intervening array (4.3) obtained at the first stage of processing will be represented by two real numbers, namely, its minimum point $\hat{x}_{t_1t_2}(y_{1t_2},\dots,y_{N_1t_2})$ and second derivative $\hat{q}_{t_1t_2}(y_{1t_2},\dots,y_{N_1t_2})$ that is one-dimensional analog of the Hessian matrix in (3.31). Thus, the array $\hat{\mathbf{J}}(\mathbf{Y})$ to be stored is twice as large as the original image. At the second stage, this array is transformed into the result of smoothing $\hat{\mathbf{X}}(\mathbf{Y}) = \big(\hat{x}_{t_1t_2}(\mathbf{Y});\ t_1 = 1,\dots,N_1,\, t_2 = 1,\dots,N_2\big)$.

The problem of matching is another example of image analysis problem for that the quadratic form of node and edge functions is completely consistent with the nature of the designed result of processing [18, 20, 21]. But the structure of the node functions required for mining the data-dependent information from the original image pair is inevitably much more complicated than in the case of smoothing.

The principal idea outlined in Section 2.1 suggests that the desired result of matching of an image pair $\mathbf{Y} = (\mathbf{Y}^l, \mathbf{Y}^r)$ should be represented by the so-called disparity map, an array of real-valued vector variables $\mathbf{x}_{t_1t_2} = (x^1_{t_1t_2}, x^2_{t_1t_2}) \in \mathcal{X} = \mathsf{R}^2$ having the sense of local spatial shifts between the respective elements (t_1, t_2) of the reference image $\mathbf{Y}^l$ and their best matches $(t_1 + x^{(1)}_{t_1t_2}, t_2 + x^{(2)}_{t_1t_2})$ in the search image $\mathbf{Y}^r$. As we see, the particular disparity value $\mathbf{x}_{t_1t_2}$ is assumed to point, as a rule, not exactly at an element of the discrete image-pixel grid, but at an imaginary point of the two-dimensional real plane associated with the search image, so, an inter-pixel interpolation is permitted in the course of matching. Below, we describe a way of representing the elementary units of image-dependent information on the sought-for disparity map in the form of a quadratic node functions in full accordance with (3.18)

$$\psi_{t_1t_2}(\mathbf{x}) = \psi(\mathbf{x}|\mathbf{Y}_{t_1t_2}) = (\mathbf{x} - \mathbf{x}^0_{t_1t_2})^T \mathbf{Q}^0_{t_1t_2}(\mathbf{x} - \mathbf{x}^0_{t_1t_2}), \ \ \mathbf{x} \in \mathcal{X} = \mathsf{R}^2, \tag{4.9}$$

where $\mathbf{Y}_{t_1t_2}$ is the respective elementary fragment of the image pair $\mathbf{Y}$.

In accordance with this convention, the node functions meant to carry the elementary units of information on the sought-for disparity map should be defined at pixels of the left (reference) image. Let $T^l = \{\mathbf{t} = (t_1, t_2);\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2\}$ be the rectangular discrete grid formed by these pixels. The pixel grid T^r of the right (search) image is to be taken greater at each side by the respective whole number restricting the finite rectangular set of discrete tentative disparity values that will participate in forming the image-dependent node functions:

$$\mathcal{V} = \left\{\mathbf{v} = (v^{(1)}, v^{(2)}):\ v^{(1)} = -k^{(1)}, \ldots, k^{(1)},\ v^{(2)} = -k^{(2)}, \ldots, k^{(2)}\right\} \subset \mathcal{X} = \mathsf{R}^2. \tag{4.10}$$

Let $\mathbf{t} \in T^l$ be an element of the left pixel grid. Consider in the right pixel grid all the elements $(\mathbf{t} + \mathbf{v}) \in T^r$ reachable from $\mathbf{t}$ by admissible discrete tentative shifts $\mathbf{v} \in \mathcal{V}$. It appears to be natural that the more similar the search image fragment $\mathbf{Y}^r_{\mathbf{t}+\mathbf{v}}$ within a sufficiently small vicinity of the point $\mathbf{t} + \mathbf{v}$ is to the analogous fragment $\mathbf{Y}^l_{\mathbf{t}}$ of the reference image, the more likely is the hypothesis that the $\mathbf{x} = \mathbf{v}$ is just the correct value of the sought-for disparity at point $\mathbf{t}$. Let consider an appropriate measure of distance between the two image fragments $\eta(\mathbf{Y}^l_{\mathbf{t}}, \mathbf{Y}^r_{\mathbf{t}+\mathbf{v}})$ as function of the discrete tentative shift $\mathbf{v}$. As image-dependent node function (4.9) at point $\mathbf{t} = (t_1, t_2)$ of the reference image (node of the pixel-neighborhood graph), we use here the quadratic approximation of this distance measure over all the admissible shifts:

$$\left(\mathbf{x}^0_{\mathbf{t}}, \mathbf{Q}^0_{\mathbf{t}}\right) = \arg \min_{\mathbf{x}\in\mathsf{R}^2,\, \mathbf{Q}(2\times 2)} \sum_{\mathbf{v}\in\mathcal{V}} \left[\eta(\mathbf{Y}^l_{\mathbf{t}}, \mathbf{Y}^r_{\mathbf{t}+\mathbf{v}}) - (\mathbf{v} - \mathbf{x})^T \mathbf{Q}(\mathbf{v} - \mathbf{x})\right]^2.$$

So, to find the node function at a pixel $\mathbf{t}$ of the reference image grid, we have to calculate $(2k^{(1)}+1) \times (2k^{(2)}+1)$ values of the distance measure $\eta(\mathbf{Y}^l_{\mathbf{t}}, \mathbf{Y}^r_{\mathbf{t}+\mathbf{v}})$ for all the admissible tentative shifts $\mathbf{v} \in \mathcal{V}$ and approximate the obtained lattice function on the set of discrete points by a quadratic function whose range of definition is the two-dimensional real space as a whole. It is clear that the parameters of the node function will be real-valued. Vector $\mathbf{x}^0_{\mathbf{t}}$ is the local estimate on the disparity value with eventual inter-pixel interpolation, and the nonnegative definite matrix $\mathbf{Q}^0_{\mathbf{t}}$ carries the information on the reliability of this estimate in different directions in the image plane.

If point $\mathbf{t}$ is a salient point of the reference image and the similar peculiarity is present in the search one, it is very likely that $\mathbf{x}_{\mathbf{t}}^{0}$ will be approximately equal to the spatial shift between them, and both positive eigenvalues of $\mathbf{Q}_{\mathbf{t}}^{0}$ will be large in value, i.e. the second derivatives of the node function will be large in all directions from its minimum point, what means a very high reliability level of the local estimate on the disparity value.

In case $\mathbf{t}$ belongs to a ridge in the reference image and the analogous ridge is found also in the search image, it should be expected that $\mathbf{x}_{\mathbf{t}}^{0}$ would show an occasional point of it, and the two eigenvalues of $\mathbf{Q}_{\mathbf{t}}^{0}$ would be different. The eigenvector with the small eigenvalue will be directed along the ridge, so, the respective directional second derivative at the minimum point of the node function will be small, what means that there is no information on the disparity value in this direction. On the contrary, the large eigenvalue of the orthogonal eigenvector will make the second derivative large in the transversal direction, and indicate thereby a very high reliability level of the local estimate across the ridge.

Finally, if there are no informative peculiarities of the reference image in the vicinity of point $\mathbf{t}$, or the search image is noninformative in the area of admissible tentative shifts (4.10), both eigenvalues of $\mathbf{Q}_{\mathbf{t}}^{0}$ will be small. In this case the small second derivatives of the node function in all directions from its occasional minimum point will indicate that the local estimate $\mathbf{x}_{\mathbf{t}}^{0}$ does not deserve any reliance.

Formally speaking, the role of the edge function remains the same as that in the problem of smoothing, namely, to smooth the local estimates on the neighboring goal variables $\mathbf{x}_{\mathbf{t}'}^{0}$, $\mathbf{x}_{\mathbf{t}''}^{0}$. The natural symmetry of the image analysis problems suggests the identity of the vertical and horizontal edge functions. But as to the identity of the degree of smoothing for the two components of the vector disparity $\mathbf{x}_{\mathbf{t}} = (x_{\mathbf{t}}^{1}, x_{\mathbf{t}}^{2})$, the question is not so simple. If the pictures to be analyzed are images of two objects of similar structure (Fig. 2.3), and matching is undertaken with the purpose of immediate comparing them pixel-by-pixel, for instance, with the ultimate aim of recognition, there is no reason to set different smoothing degrees for the horizontal and the vertical disparity. But in case of matching a stereo pair of images of the same object (Fig. 4.1), the expected local shifts in the two directions in the image plane are not identical. As a rule, the horizontal axis is assumed to coincide, maybe, only approximately, with the so-called epipolar line, i.e. the line at which the two cameras having taken the pictures were located [18]. In this case, the total amount of information on the distances between the cameras' plane and the particular points on the object surface is carried by the horizontal components $x_{\mathbf{t}}^{(1)}$ of disparity vectors, whereas the vertical components $x_{\mathbf{t}}^{(2)}$ are expected to equal zero or almost zero, at least, to be uniform or almost uniform in value. It is sufficient to take a quadratic edge function defined by diagonal Hessian matrix 2×2 with, generally speaking, different positive values $\alpha^{(1)} > 0$ and $\alpha^{(2)} > 0$ in the main diagonal:

$$\gamma(\mathbf{x}_{\mathbf{t}'}, \mathbf{x}_{\mathbf{t}''}) = (\mathbf{x}_{\mathbf{t}'} - \mathbf{x}_{\mathbf{t}''})^T \mathbf{A} (\mathbf{x}_{\mathbf{t}'} - \mathbf{x}_{\mathbf{t}''}), \quad \mathbf{A} = \begin{pmatrix} \alpha^{(1)} & 0 \\ 0 & \alpha^{(2)} \end{pmatrix}, \tag{4.11}$$

where $\alpha^{(1)} = \alpha^{(2)}$ in the case of matching usual images, and $\alpha^{(1)} \ll \alpha^{(2)}$ for stereo pairs.

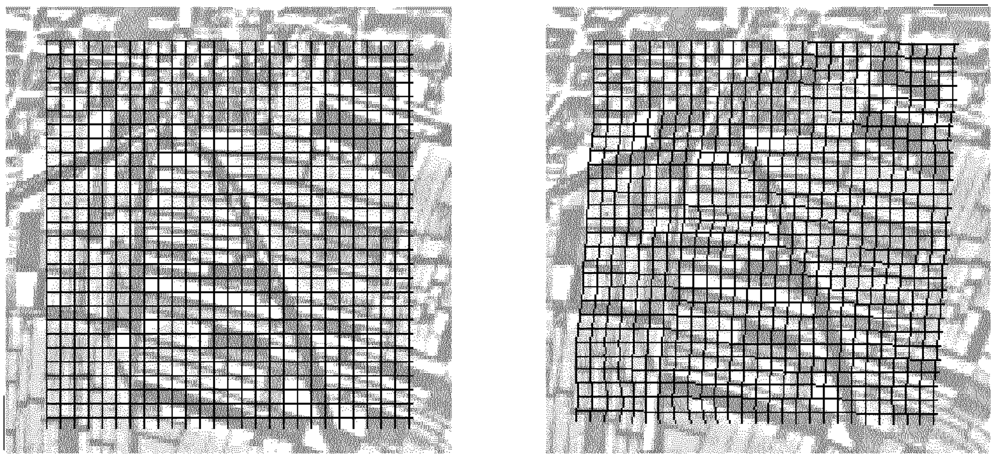

Figure 4.1: Disparity map of a stereo pair of satellite photographs.

However, the fact that the variables to be smoothed are vectors and, especially, the presence of matrix coefficients in the node functions makes the effect of co-ordinating the image-dependent and model-based information much more complicated than plain smoothing. It is hardly probable that the local estimate $\mathbf{x}_\mathbf{t}^0$ at a point where both eigenvalues of $\mathbf{Q}_\mathbf{t}^0$ are large will be significantly changed in the course of smoothing. On the contrary, if both eigenvalues are small, the vague local estimate will practically not affect the result, and the final disparity value at such a point will be actually interpolated from more firm local estimates at the surrounding points. The most sophisticated effect will occur at the ridge and edge points of the pictures, where the two eigenvalues of matrices $\mathbf{Q}_\mathbf{t}^0$ are substantially different. The final disparity values will offer a complicated compromise between different kinds of relative freedom of alterations in one direction specific for each point and almost prohibited changes in the orthogonal one.

Matching of two images is provided by the same two-dimensional version of the generalized data analysis procedure. The Bellman and marginal node functions will be quadratic functions of two variables due to the quadraticity of the node and edge functions (4.9), (4.11). Each quadratic marginal function in (4.3) will be represented by its minimum point $\hat{\mathbf{x}}_{t_1 t_2}(\mathbf{Y}_{1t_2}, \ldots, \mathbf{Y}_{N_1 t_2})$ and Hessian matrix $\hat{\mathbf{Q}}_{t_1 t_2}(\mathbf{Y}_{1t_2}, \ldots, \mathbf{Y}_{N_1 t_2})$. It is just this amount of intervening information, namely, six real numbers, that is to be stored in each element of $\hat{\mathbf{J}}(\mathbf{Y})$ after the first stage of processing. As we see, the intervening array is thrice as large as the original image pair. The second stage of processing will result in the sought-for disparity map $\hat{\mathbf{X}}(\mathbf{Y}) = \big(\hat{\mathbf{x}}_{t_1 t_2}(\mathbf{Y});\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2\big)$.

4.3 Analysis of geophysical explorative data

A two-dimensional seismic section (Fig. 2.4) manifests, despite its non-visual origin, all the principal traits of a texture image. The aim of processing is to make visible fine distinctions in the spatial oscillation mode of the reflected seismic signals forming the section to be analyzed, i.e. local texture properties of the section considered as a texture image. As it was mentioned in Section 2.1, the final result of processing should be represented in the form of a two-dimensional array $\mathbf{X} = (\mathbf{x_t},\, \mathbf{t} = (t_1, t_2) \in T)$ of estimated vector parameters $\mathbf{x_t} \in \mathsf{R}^n$ of an appropriate local oscillation model.

When in the previous subsection we considered the problem of texture image segmentation, we used the statistical approach to extracting the original image-dependent information on the local signal-oscillation mode in the vicinity of each point $\mathbf{t} = (t_1, t_2)$. The principal idea of the approach was treating a texture image as a realization of a random field. The same approach remains completely suitable also in the problem of seismic section analysis with the only difference that segmentation is not required, it is enough to estimate the local model parameters.

As the basis of the local spatial model of signal oscillations in a seismic section, we will use the same autoregressive model (4.6) which underlies the texture segmentation technique, but the model will be subjected here to three alterations. First, the vector of autoregression parameters will be considered as a variable and, so, will characterize the local properties of texture at each point $\mathbf{t} = (t_1, t_2)$ individually, instead of a constant which was previously assumed to depend only on the index of the texture class. Second, seismic signals are usually registered so that their mean value equals zero, and the presence of coefficient a_{00} in the autoregression equation is needless. Third, it will be much more easy to estimate at each point not the white noise intensity $\sigma_{t_1 t_2}$ but the full variance of the random field or, in other words, the full local intensity of seismic signals oscillations, let it be denoted by $d_{t_1 t_2}$. The latter parameter is proportional to $\sigma^2_{t_1 t_2}$, but the proportionality coefficient is quite a complicated function of autoregression coefficients $a_{t_1 t_2, i_1 i_2}$. Thus, the instantaneous local parameters of the autoregression equation will form vector $\mathbf{x}_{t_1 t_2} = \big(a_{t_1 t_2, i_1 i_2},\, (i_1, i_2) \in \mathcal{F};\; d_{t_1 t_2}\big)$:

$$y_{t_1 t_2} - \sum_{(i_1, i_2) \in \mathcal{F}} a_{t_1 t_2, i_1 i_2} y_{t_1 + i_1, t_2 + i_2} = \xi_{t_1 t_2}, \tag{4.12}$$

where $\xi_{t_1 t_2}$ is the white noise of just that intensity $\sigma_{t_1 t_2}$ which is required for the variance of the resulting random field would be equal to $d_{t_1 t_2}$.

Let, just as in (4.6), $\mathbf{Y}_{t_1 t_2}$ be the elementary fragment cut out of the seismic section by the autoregression mask $\mathcal{F}$, and $\mathbf{y}_{t_1 t_2} = \big(y_{t_1 + i_1, t_2 + i_2},\, (i_1, i_2) \in \mathcal{F}\big)$ be the same fragment without the current point. We will assume that $\mathcal{F}$ consists of n elements ordered in an appropriate manner and consider $\mathbf{y}_{t_1 t_2} \in \mathsf{R}^n$ as a real-valued column vector. Respectively, the autoregression coefficients ordered in the same way will be considered as column vector $\mathbf{a} = \big(a_{i_1 i_2},\, (i_1, i_2) \in \mathcal{F}\big) \in \mathsf{R}^n$ forming, along with the full intensity of signal oscillations, the full vector of texture parameters at each point $\mathbf{x}_{t_1 t_2} = (\mathbf{a}_{t_1 t_2},\, d_{t_1 t_2})$.

The node function associated with point $\mathbf{t} = (t_1, t_2)$ is intended to be an appropriate measure of mismatch between a hypothesis $\mathbf{x} = (\mathbf{a},\, d)$ on the hidden value of the autoregression parameter and the respective elementary fragment $\mathbf{Y}_{t_1 t_2} = (y_{t_1 t_2}, \mathbf{y}_{t_1 t_2})$. To start with, it is convenient to consider two different measures of such a kind, separately for autoregression coefficient vector $\mathbf{a}$ and signal oscillation intensity d.

The squared left part of the autoregression equation (4.12) is a function which shows to what extent the hypothesis that $\mathbf{a}$ is just the current autoregression coefficient vector $\mathbf{a}_{t_1t_2}$ contradicts the elementary fragment $\mathbf{Y}_{t_1t_2}$:

$$\psi^{(\mathbf{a})}(\mathbf{a}|\mathbf{Y}_{t_1t_2}) = \left(y_{t_1t_2} - \sum_{(i_1,i_2)\in\mathcal{F}} a_{i_1i_2}y_{t_1+i_1,t_2+i_2}\right)^2 = \left(y_{t_1t_2} - \mathbf{y}^T_{t_1t_2}\mathbf{a}\right)^2. \qquad (4.13)$$

It is easy to see that the degenerate matrix $\mathbf{B}^0_{t_1t_2} = \mathbf{y}_{t_1t_2}\mathbf{y}^T_{t_1t_2}$ is Hessian matrix of the degenerate quadratic function $\psi^{(\mathbf{a})}(\mathbf{a}|\mathbf{Y}_{t_1t_2})$, and each vector satisfying the condition $\mathbf{x}^T_{t_1t_2}\mathbf{a} = y_{t_1t_2}$ is its minimum point. The shortest of them in the sense of Euclidean norm $\|\mathbf{a}\|$ is vector $\mathbf{a}^0_{t_1t_2} = \left(y_{t_1t_2}/\mathbf{y}^T_{t_1t_2}\mathbf{y}_{t_1t_2}\right)\mathbf{y}_{t_1t_2}$. So, function (4.13) can be represented in the form

$$\psi^{(\mathbf{a})}(\mathbf{a}|\mathbf{Y}_{t_1t_2}) = \left(\mathbf{a} - \mathbf{a}^0_{t_1t_2}\right)^T \mathbf{B}^0_{t_1t_2}\left(\mathbf{a} - \mathbf{a}^0_{t_1t_2}\right), \quad \mathbf{a} \in \mathsf{R}^n,$$

$$\mathbf{a}^0_{t_1t_2} = \frac{y_{t_1t_2}}{\mathbf{y}^T_{t_1t_2}\mathbf{y}_{t_1t_2}}\mathbf{y}_{t_1t_2} \in \mathsf{R}^n, \quad \mathbf{B}^0_{t_1t_2} = \mathbf{y}_{t_1t_2}\mathbf{y}^T_{t_1t_2}\ (n\times n).$$

Analogously, quadratic function

$$\psi^{(d)}(d|\mathbf{Y}_{t_1t_2}) = \left(d - y^2_{t_1t_2}\right)^2, \quad d \in \mathsf{R},$$

shows the extent to which real number d is contradictory, when judging from the single observation $y_{t_1t_2}$, to the hypothesis that it coincides with the current variance of the random field $d_{t_1t_2}$.

Thus, all the requirements placed upon a data-dependent node function in the generalized optimization-based statement of the data analysis problem are met by the sum of $\psi^{(\mathbf{a})}(\mathbf{a}|\mathbf{Y}_{t_1t_2})$ and $\psi^{(d)}(d|\mathbf{Y}_{t_1t_2})$. If we consider the full parameter vector $\mathbf{x} = (\mathbf{a}^T,\ d)^T$ as column vector in which d is the last element, the desired quadratic data-dependent node function will obtain the form

$$\psi(\mathbf{x}|\mathbf{Y}_{t_1t_2}) = \left(\mathbf{x} - \mathbf{x}^0_{t_1t_2}\right)^T \mathbf{Q}^0_{t_1t_2}\left(\mathbf{x} - \mathbf{x}^0_{t_1t_2}\right), \quad \mathbf{x} \in \mathsf{R}^{n+1},$$

$$\mathbf{x}^0_{t_1t_2} = \begin{pmatrix} \dfrac{y_{t_1t_2}}{\mathbf{y}^T_{t_1t_2}\mathbf{y}_{t_1t_2}}\mathbf{y}_{t_1t_2} \\ y^2_{t_1t_2} \end{pmatrix} \in \mathsf{R}^{n+1}, \quad \mathbf{Q}^0_{t_1t_2} = \begin{pmatrix} \mathbf{y}_{t_1t_2}\mathbf{y}^T_{t_1t_2} & \mathbf{0} \\ \mathbf{0}^T & y^2_{t_1t_2} \end{pmatrix} [(n+1)\times(n+1)], \qquad (4.14)$$

where $\mathbf{0}$ is column vector of n zero elements.

The fact that a seismic data array is no visual image reveals itself in the fundamental distinction between the nature of the vertical axis interpreted as time and that of the horizontal one formed by the increasing geophone index. Whereas the purpose of edge functions remains the same, namely, to smooth or, in other terms, co-ordinate fuzzy local decisions given by the node functions, the degree of smoothing should be taken different in the two directions. In addition, because the autoregression coefficient vector $\mathbf{a}$ and oscillation intensity d occur in the node functions (4.14) in different manners, it may turn out to be needed to set a specific smoothing degree for each of these variables. Two identical quadratic functions taken with different coefficients would be just the appropriate decision:

$$\begin{aligned}\gamma'(\mathbf{x}_{t_1,t_2-1},\mathbf{x}_{t_1t_2}) &= \beta'\,(\mathbf{x}_{t_1,t_2-1}-\mathbf{x}_{t_1t_2})^T\mathbf{A}(\mathbf{x}_{t_1,t_2-1}-\mathbf{x}_{t_1t_2}),\\ \gamma''(\mathbf{x}_{t_1-1,t_2},\mathbf{x}_{t_1t_2}) &= \beta''\,(\mathbf{x}_{t_1-1,t_2}-\mathbf{x}_{t_1t_2})^T\mathbf{A}(\mathbf{x}_{t_1-1,t_2}-\mathbf{x}_{t_1t_2}),\end{aligned}\qquad \mathbf{A}=\begin{pmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0}^T & \alpha\end{pmatrix}, \quad (4.15)$$

where $\mathbf{I}$ is identity matrix $n\times n$. Coefficient $\alpha>0$ is intended to align the smoothing degree of d relative to that of $\mathbf{a}$, and coefficients $\beta'>0$ and $\beta''>0$ determine two different smoothing degrees in, respectively, the vertical and the horizontal direction.

At the first stage of the generalized data-analysis procedure, the original seismic section

$$\mathbf{Y}=(y_{t_1t_2};\ t_1=1,\dots,N_1,\ t_2=1,\dots,N_2)$$

will be transformed into the intervening data array $\hat{\mathbf{J}}(\mathbf{Y})$ (4.3) consisting, just as in the matching problem considered in the previous Section, of quadratic marginal functions each represented by its minimum point $\hat{\mathbf{x}}_{t_1t_2}(\mathbf{Y}_{1t_2},\dots,\mathbf{Y}_{N_1t_2})\in\mathsf{R}^{n+1}$ and Hessian matrix $\hat{\mathbf{Q}}_{t_1t_2}(\mathbf{Y}_{1t_2},\dots,\mathbf{Y}_{N_1t_2})$ $(n\times n)$, i.e. by $(n+1)+(n+1)^2$ real numbers. So, the amount of information to be stored is determined by number n of elements in the autoregression mask $\mathcal{F}$ accepted as local texture model of the seismic section. The resulting array

$$\hat{\mathbf{X}}(\mathbf{Y})=\big(\hat{\mathbf{x}}_{t_1t_2}(\mathbf{Y});\ t_1=1,\dots,N_1,\ t_2=1,\dots,N_2\big)$$

obtained at the second stage of processing consists of $(n+1)$-dimensional vectors $\hat{\mathbf{x}}_{t_1t_2}=(\hat{\mathbf{a}}_{t_1t_2},d_{t_1t_2})$ each of which represents the local spatial autoregression properties and local intensity of signal oscillations.

In particular, if the autoregression mask $\mathcal{F}$ completely concentrates in one of the image lines, namely, in the horizontal $\dots,t_1{-}1,t_1,t_1{+}1,\dots$ or vertical direction $\dots,t_2{-}1,t_2,t_2{+}1,\dots$, each vector of texture parameters $\mathbf{x}_{t_1t_2}=(\mathbf{a}_{t_1t_2},d_{t_1t_2})$ will actually represent spatially smoothed local properties of the respective one-dimensional random process formed by a single horizontal slice of the seismic section $y_{t_1t_2}$, $t_2=\text{const}$, or a single seismic signal $y_{t_1t_2}$, $t_1=\text{const}$. In the case of one-dimensional models, it is sufficient to use one-sided masks, i.e. masks concentrated before the current point according to the direction in which the respective index increases, then the local autoregression model (4.12) becomes more simple:

$$y_{t_1t_2}-\sum_{i_1=1}^{n}a_{t_1t_2,i_1}y_{t_1-i_1,t_2}=\xi_{t_1t_2},\qquad y_{t_1t_2}-\sum_{i_2=1}^{n}a_{t_1t_2,i_2}y_{t_1,t_2-i_2}=\xi_{t_1t_2}. \qquad (4.16)$$

Models of such a kind describe the texture through describing its one-dimensional horizontal or vertical cross-sections, so, these are actually models of the respective one-dimensional random processes.

For seismic sections through some kinds of rock mass, it is important to study local spectral properties of such one-dimensional processes. It is well known that the autoregression model of a random process is completely equivalent to its frequency spectrum, or, in other terms, spectral density of its variance [22].

Variance (dispersion) of a stationary random process on the discrete argument axis $y_t,\ t=\dots,1,2,3,\dots$ is mathematical expectation of its squared random values $d=\mathrm{E}(y_t^2)$, assumed that the mean value of the process equals zero. Spectral density of variance is a function of frequency $S(f)\geq 0$ which is defined in the range $0\leq f\leq 1/2$, because frequencies higher than $1/2$ periods per sample are senseless in the case of discrete argument. Integral of the spectral density over the whole range of definition gives the variance of the random process, and integral over a frequency interval $f_{\min}\leq f\leq f_{\max}$, where $f_{\min}\geq 0$ and $f_{\max}\leq 1/2$, shows what partial variance is carried by harmonic constituents within this frequency band:

$$d=\int_0^{1/2} S(f)df, \qquad d(f_{\min},f_{\max})=\int_{f_{\min}}^{f_{\max}} S(f)df.$$

Spectral density of an autoregressive random process

$$y_t=\sum_{k=1}^{n} a_k y_{t-k}+\xi_t$$

is a function $S(f;\mathbf{x})$ of the vector of autoregression parameters $\mathbf{x}=(\mathbf{a},d)$, namely, of the autoregression coefficients $\mathbf{a}=(a_1,\dots,a_n)$ and full variance of the process d. This function is especially simple in the case of second-order autoregression $n=2$:

$$S(f)=\frac{(1+a_2)\Big((1-a_2)^2-a_1^2\Big)}{(1-a_2)}\,\frac{2d}{1+a_1^2+a_2^2-2a_1(1-a_2)\cos 2\pi f-2a_2\cos 4\pi f}\,.$$

One-dimensional local texture models (4.16) are based on just the opposite assumption than that the random process being analyzed is stationary, namely, it is assumed that the estimated autoregression parameters $\hat{\mathbf{x}}_{t_1t_2}(\mathbf{Y})$ of a horizontal or vertical process cut out of a seismic section $\mathbf{Y}$ are different at different points (t_1,t_2) as along the axis of this process as well as across it. A series of spectra representing gradually changing oscillation mode of a signal on the time axis is usually referred to as result of its time-frequency analysis. In the case of a seismic section, we will obtain a two-dimensional array of horizontal or vertical spectra $S\big(f;\hat{\mathbf{x}}_{t_1t_2}(\mathbf{Y})\big)$, $0\leq f\leq 1/2$, but the term "time-frequency analysis" is applicable in full sense only to the vertical direction, because it is just the vertical axis of a seismic section which is associated with time.

Of course, a series of spectral densities is a too immense representation of the local oscillation modes in a nonstationary signal, the more so, in a seismic section, but it is required, as a rule, to know the partial intensity of oscillations only in one sufficiently narrow frequency band:

$$\hat{d}_{t_1 t_2}(f_{\min}, f_{\max}; \mathbf{Y}) = \int\limits_{f_{\min}}^{f_{\max}} S\big(f; \hat{\mathbf{x}}_{t_1 t_2}(\mathbf{Y})\big) df.$$

It is just this kind of processing result representation that is illustrated by the example in Fig. 2.4.

A seismic cube (Fig. 2.5) is a kind of data arrays for which the immediate visual analogy to texture images gets lost because of the presence of the third coordinate. The semantic analogy, nevertheless, remains, and it appears natural to consider the local oscillation mode of reflected seismic signals forming a three-dimensional array as its three-dimensional texture at the respective point. The mathematical model of texture in the form of autoregression equation (4.12) also remains quite correct with the only difference that argument $\mathbf{t}$ is tree-dimensional vector $\mathbf{t} = (t_1, t_2, t_3)$ and, instead of a plane mask $\mathcal{F}$, its three-dimensional analogue should be used.

The structure of the node functions consistent with such a local spatial model of signal oscillations completely coincides with that for two-dimensional seismic sections (4.14), it is enough to reinterpret $\mathbf{t}$ as vector of coordinates in the discrete three-dimensional space. As before (4.15), coefficients at the quadratic edge functions should be taken different in the horizontal and the vertical directions, but, for the horizontal symmetry of a seismic cube, there is no reason to distinguish between two horizontal axes:

$$\gamma'(\mathbf{x}_{t_1-1,t_2t_3}, \mathbf{x}_{t_1t_2t_3}) = \beta'\,(\mathbf{x}_{t_1-1,t_2t_3} - \mathbf{x}_{t_1t_2t_3})^T \mathbf{A}(\mathbf{x}_{t_1-1,t_2t_3} - \mathbf{x}_{t_1t_2t_3}),$$

$$\gamma''(\mathbf{x}_{t_1t_2-1,t_3}, \mathbf{x}_{t_1t_2t_3}) = \beta'\,(\mathbf{x}_{t_1t_2-1,t_3} - \mathbf{x}_{t_1t_2t_3})^T \mathbf{A}(\mathbf{x}_{t_1t_2-1,t_3} - \mathbf{x}_{t_1t_2t_3}),$$

$$\gamma'''(\mathbf{x}_{t_1t_2t_3-1}, \mathbf{x}_{t_1t_2t_3}) = \beta''\,(\mathbf{x}_{t_1t_2t_3-1} - \mathbf{x}_{t_1t_2t_3})^T \mathbf{A}(\mathbf{x}_{t_1t_2t_3-1} - \mathbf{x}_{t_1t_2t_3}),$$

where the structure of matrix $\mathbf{A}$ is the same as in (4.15).

One-dimensional masks like (4.16) and spectral representation of the respective one-dimensional autoregression models remain useful for revealing local peculiarities in some kinds of rock mass through studying their three-dimensional seismic images. True, there are many horizontal directions in this case, and some additional averaging over them may be required, but the vertical direction t_3 retains the sense of time, and it is often very important to study local spectral properties of the reflected seismic signals along this axis.

Processing of a seismic cube

$$\mathbf{Y} = (y_{t_1t_2t_3};\ t_1 = 1, \ldots, N_1,\ t_2 = 1, \ldots, N_2,\ t_3 = 1, \ldots, N_3)$$

is performed in three stages, instead of two ones as in the case of plane sections, by the three-dimensional version of the generalized data analysis procedure. The elements of the first $\hat{\mathbf{J}}^{*}(\mathbf{Y})$ and the second $\hat{\mathbf{J}}^{**}(\mathbf{Y})$ intervening array of quadratic marginal functions are represented by their $(n+1)$-dimensional minimum points and Hessian matrices $(n+1)\times(n+1)$, respectively, $\hat{\mathbf{x}}_{t_1t_2t_3}(\mathbf{Y}_{t_2t_3})$, $\hat{\mathbf{Q}}_{t_1t_2t_3}(\mathbf{Y}_{t_2t_3})$, and $\hat{\mathbf{x}}_{t_1t_2t_3}(\mathbf{Y}_{t_3})$, $\hat{\mathbf{Q}}_{t_1t_2t_3}(\mathbf{Y}_{t_3})$, where the cube's fragments $\mathbf{Y}_{t_2t_3}$ and $\mathbf{Y}_{t_3}$ are defined by (4.4) and (4.5). Just as in the plane case, the elements of the resulting array

$$\hat{\mathbf{X}}(\mathbf{Y}) = \big(\hat{\mathbf{x}}_{t_1t_2t_3}(\mathbf{Y});\ t_1=1,\dots,N_1,\ t_2=1,\dots,N_2,\ t_3=1,\dots,N_3\big)$$

have the sense of local parameters of the spatial texture $\hat{\mathbf{x}}_{t_1t_2t_3} = (\hat{\mathbf{a}}_{t_1t_2t_3},\, d_{t_1t_2t_3})$, the three-dimensional one for this time.

4.4 Remarks on multi-sequence alignment of proteins

In contrast to the problems of processing visual images and geophysical data arrays considered in the previous Sections 4.2 and 4.3, the idea of creating algorithms of multi-alignment of molecular sequences as those of finding the minimum point of a separable objective function is not yet brought into reality. In this Section, we consider some details of such a project.

The nature of a set of proteins (Fig. 2.6,a) as a data array is absolutely different from that of images, as visual as well as seismic ones. From the formal point of view, this fact reveals itself in that there are two different kinds of variables $c_{t_1t_2}$ and $k_{t_1t_2}$ occurring in the original separable objective function of the multi-alignment problem (2.4) or (2.2), and, respectively, two different kinds of neighborhood between them, as it is shown in Fig. 2.7,c.

The structure of the data-dependent functions $\psi(c_{t_1t_2}, k_{t_1t_2}|\mathbf{y}_{t_2})$ in (2.4) and $\psi(x_{t_1t_2}|\mathbf{y}_{t_2})$ in (2.2), which are, respectively, edge-type and node-type functions in the former and the latter case, as well as that of the model-based ones $\gamma(c_{t_1t_2-1}, c_{t_1t_2})$ and $\gamma(x_{t_1t_2-1}, x_{t_1t_2})$ depends on the nature of the pattern variable $c_{t_1t_2}$ chosen to represent common properties of amino acids having fallen in the same column in an alignment (Fig. 2.6,b,c).

In any case, the data-dependent elementary functions in (2.4) and (2.2) are completely defined by the choice on set $\mathcal{C}$ and dissimilarity function $\varphi(c,y)$ assumed to measure the residual discrepancy between a particular element of the extended set of amino acids $y\in\bar{\mathcal{Y}}$ and its approximate representation in the form of the pattern element $c\in\mathcal{C}$:

$$\psi_{t_1t_2}(c,k) = \psi(c,k|\mathbf{y}_{t_2}) = \psi(x|\mathbf{y}_{t_2}) = \varphi(c, y_{kt_2}),\ \ c\in\mathcal{C},\ k\in\mathcal{K},\ x\in\mathcal{C}\times\mathcal{K}.$$

Mark, that the data-dependent function $\psi_{t_1t_2}(c,k)$ does not depend on t_1, because it does not matter which column of the alignment the reference to amino acid y_{kt_2} is issued from.

In this work, we restrict our consideration only to the case, when pattern elements c belong to a finite set $\mathcal{C}$. The most straightforward kind of a discrete pattern element meant to represent a collection of amino acids would be simply one of them $c\in\mathcal{C}=\mathcal{Y}$ (2.1). But a more reasonable way of forming a discrete set of pattern elements is to interpret them as fuzzy subclasses of amino acids possessing some common properties $c\subseteq\bar{\mathcal{Y}}$ [16].

In the latter case, the dissimilarity function $\varphi(c, y)$ should be chosen so that it takes the greater value the smaller is the membership degree $y \in c$ measured, for instance, by fuzzy membership index $0 \le I(y \in c) \le 1$. Then, it appears to be reasonable to take the dissimilarity function in the form

$$\varphi(c, y) = 1 - I(y \in c),$$

and the data-dependent function can be formed as

$$\psi_{t_1 t_2}(c, k) = \psi(c, k|\mathbf{y}_{t_2}) = 1 - I(y_{kt_2} \in c). \tag{4.17}$$

In accordance with the principle accepted in Section 2.1, the discrete variables $k_{t_1 t_2}$ take values from set $\mathcal{K} = \{\frac{1}{2}, 1, 1\frac{1}{2}, 2, 2\frac{1}{2}, \ldots\}$. A whole value $k_{t_1 t_2} = i = 1, 2, 3, \ldots$ has the sense of the ordinal number of the amino acid y_{it_2} which element $c_{t_1 t_2}$ of the pattern sequence in row t_2 of the alignment is associated with, and a fractional value $k_{t_1 t_2} = i\frac{1}{2}$ points at the interface between the respective two neighboring amino acids y_{it_2} and y_{i+1,t_2}.

Actually, each variable $k_{t_1 t_2}$ in the row t_2 of the alignment takes values from the specific set $\mathcal{K}_{t_2} = \{\frac{1}{2}, 1, 1\frac{1}{2}, \ldots, N_{t_2}, N_{t_2}\frac{1}{2}\}$ of possible positions of amino acids and eventual gaps between them. The number of elements M_{t_2} in such a set depends on the number of amino acids in the respective protein $\mathbf{y}_{t_2} = (y_{1t_2}, \ldots, y_{N_{t_2} t_2})$ and equals $M_{t_2} = 2N_{t_2} + 1$.

If, in addition, we denote by m the number of pattern elements in the set $\mathcal{C}$, each of the combined variables $x_{t_1 t_2} = (c_{t_1 t_2}, k_{t_1 t_2})$ will take values from the set $\mathcal{X} = \mathcal{C} \times \mathcal{K}_{t_2}$ that numbers $m(2N_{t_2} + 1)$ elements.

Two kinds of neighborhood between the respective two kinds of variables $c_{t_1 t_2}$ and $k_{t_1 t_2}$ are represented in the objective function (2.4) by elementary edge functions $\gamma(c_{t_1 t_2 - 1}, c_{t_1 t_2})$, corresponding to edges of the adjacency graph conventionally shown in Fig. 2.7,c as vertical ones, and $\mu(k_{t_1 - 1, t_2}, k_{t_1 t_2})$, corresponding to horizontal edges in that picture.

In the version of the multi-alignment problem considered in this work, the only purpose of the horizontal edge functions is to impose a capital penalty upon inadmissible pairs of adjacent values $(k_{t_1 - 1, t_2}, k_{t_1 t_2})$ in order to provide that the sequence of references $k_{1t_2}, \ldots, k_{N_1 t_2}$ in each row to actual amino acid positions in the respective protein complies with the ultimate aim of forming a correct alignment. The sequence of references in a row is correct if immediately adjacent fractional references are equal to each other, any whole reference following another whole one is greater than its predecessor by 1, any fractional reference having a whole predecessor is greater than it by $\frac{1}{2}$, and the same is valid for a whole reference immediately following a fractional one. These rigid restrictions are provided by the following values of the horizontal penalty:

$$\mu(k_{t_1-1,t_2}, k_{t_1 t_2}) = \begin{cases} \left.\begin{array}{l} 0 \text{ if } k_{t_1 t_2} > k_{t_1-1,t_2} \text{ with whole } k_{t_1-1,t_2} \\ 0 \text{ if } k_{t_1 t_2} = k_{t_1-1,t_2} + \frac{1}{2} \text{ with fract. } k_{t_1-1,t_2} \end{array}\right\} \text{with whole } k_{t_1 t_2}, \\ \left.\begin{array}{l} 0 \text{ if } k_{t_1 t_2} = k_{t_1-1,t_2} + \frac{1}{2} \text{ with whole } k_{t_1-1,t_2} \\ 0 \text{ if } k_{t_1 t_2} = k_{t_1-1,t_2} \text{ with fract. } k_{t_1 t_2} \end{array}\right\} \text{with fract. } k_{t_1 t_2}, \\ \infty \text{ otherwise.} \end{cases}$$

In the case of discrete pattern variables (4.17), it is enough to choose the simplest vertical edge functions

$$\gamma(c_{t_1 t_2 - 1}, c_{t_1 t_2}) = \begin{cases} 0 & \text{if} \quad c_{t_1 t_2 - 1} = c_{t_1 t_2}, \\ \alpha & \text{if} \quad c_{t_1 t_2 - 1} \neq c_{t_1 t_2}, \end{cases} \tag{4.18}$$

where $\alpha > 0$ is assumed to be so large, theoretically, $\alpha \to \infty$, that in case $c_{t_1 t_2 - 1} \neq c_{t_1 t_2}$ the penalty dwarfs the values of other functions $\psi(c_{t'_1 t'_2}, k_{t'_1 t'_2} | \mathbf{y}_{t'_2})$ and $\mu(k_{t'_1 - 1, t'_2}, k_{t'_1 t'_2})$ with any values of arguments.

With increasing penalty $\alpha \to \infty$, the final values of pattern variables found as result of minimization of the objective function (2.4) will be, in the limit, equal to each other in columns of the alignment $\hat{c}_{t_1 1}(\mathbf{Y}) = \ldots = \hat{c}_{t_1 N_2}(\mathbf{Y}) = \hat{c}_{t_1}(\mathbf{Y})$. The elements of the pattern sequence $\hat{\mathbf{c}} = \big(\hat{c}_1(\mathbf{Y}), \ldots, \hat{c}_{N_1}(\mathbf{Y})\big)$ will not depend, in the limit again, on the order in which the proteins occur in the set $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{N_2})$ (Fig. 2.6). It is just this reason which justifies the lattice-like adjacency of variables in the separable objective function of the multi-alignment problem (Fig. 2.7,c).

The formal problem of minimization of such an objective function will remain absolutely sound from the mathematical point of view even if we take a moderate value of the penalty $0 < \alpha \ll \infty$. True, the problem of protein sequence analysis we obtain in this case will differ from that of multi-alignment as it was set in Section 2.1, because each protein will be assigned its own pattern sequence $\hat{\mathbf{c}}_{t_2}(\mathbf{Y}) = \big(\hat{c}_{1t_2}(\mathbf{Y}), \ldots, \hat{c}_{N_1 t_2}(\mathbf{Y})\big)$ and the array of such sequences $t_2 = 1, \ldots, N_2$ will depend on the order of proteins in the original set. The greater the penalty α, the closer will be the pattern sequences $\big(\hat{\mathbf{c}}_{t_2 - 1}(\mathbf{Y}), \hat{\mathbf{c}}_{t_2}(\mathbf{Y})\big)$ associated with neighboring proteins $\big(\mathbf{y}_{t_2 - 1}, \mathbf{y}_{t_2}\big)$ in the original ordered set, and with $\alpha \to \infty$ we obtain $\hat{\mathbf{c}}_{t_2 - 1}(\mathbf{Y}) \to \hat{\mathbf{c}}_{t_2}(\mathbf{Y})$, i.e. approach the result of the usual multi-alignment.

Thus, the problem of protein sequence analysis considered in this work and formally expressed by the separable objective function (2.4) is essentially wider than that of multi-sequence alignment of an unordered set of proteins, and includes the latter problem as particular case with infinitely increasing penalty on the vertical neighborhood of pattern elements $\alpha \to \infty$. With moderate values of the vertical penalty $\alpha \ll \infty$, the minimization of the objective function for an ordered sequence of proteins $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_{N_2})$ is actually a kind of its "generalized smoothing", in which the role of the "smoothed" representation of the original sequence in the respective terms is played by the sequence of pattern sequences $\hat{\mathbf{C}}(\mathbf{Y}) = \big(\hat{\mathbf{c}}_1(\mathbf{Y}), \ldots, \hat{\mathbf{c}}_{N_2}(\mathbf{Y})\big)$ associated with each of the original proteins (Fig. 2.6,c). The soundness of such an analysis is concerned with the reason that, in an ordered chain of proteins, it may be senseless to align proteins $\mathbf{y}_{t'_2}$ and $\mathbf{y}_{t''_2}$ with quite distant indices t'_2 and t''_2.

Just as for other data analysis problems formulated in this work as those of optimization of respective separable objective functions, the way of creating effective algorithms of processing protein sequences

$$\mathbf{Y} = (y_{t_1 t_2}; \;\; t_1 = 1, \ldots, N_{t_2}, \; t_2 = 1, \ldots, N_2)$$

is based on replacing the original graph of lattice-like adjacency of variables (Fig. 2.7,c) by a series of tree-like ones (Fig. 2.8, c or d). The principal scheme of processing remains the same as it was depicted in Section 4.1 under the name of a generalized data analysis procedure.

In the case of a discrete set of pattern elements (4.17), (4.18), the combined variables $x_{t_1t_2} = (c_{t_1t_2}, k_{t_1t_2})$ take values from the finite sets $\mathcal{X} = \mathcal{C}\times\mathcal{K}_{t_2}$. Once one of the two version Fig. 2.8,c or Fig. 2.8,d of the tree-like decomposition of the lattice-like variable adjacency graph is chosen, there is no specificity in the numerical realization of the procedure in comparison, for instance, with that of image segmentation (Section 4.2), except the fact that the number of elements in the range of the goal variables $m(2N_{t_2}+1)$ is specific for each horizontal row of the alignment t_2 and depends on the number N_{t_2} of amino acids in the respective protein.

However, in contrast to the problem of image segmentation, the horizontal Fig. 2.8,c and the vertical version Fig. 2.8,d are far of being equivalent to each other from the viewpoint of the expected result, and it is an open question which of them will be preferable.

But what both versions have in common is the fact that an additional correction of the resulting array

$$\hat{\mathbf{X}}(\mathbf{Y}) = \Big(\hat{\mathbf{C}}(\mathbf{Y}), \hat{\mathbf{K}}(\mathbf{Y})\Big) = \Big(\hat{c}_{t_1t_2}(\mathbf{Y}), \hat{k}_{t_1t_2}(\mathbf{Y});\ t_1 = 1,\dots,N_1,\ t_2 = 1,\dots,N_2\Big)$$

will be inevitably required, might it be assembled of stem columns (2.9) or stem rows (2.11) of the respective partial variable-adjacency trees. As it was already mentioned in Section 2.3, the point is that each of the two versions provides observance of only one of two kinds of constraints on the values of neighboring goal variables, namely, that expressed by the edges forming the stems of the partial trees.

Specifically, in the vertical version (Fig. 2.8,c), the identity or, at least, smoothness constraints on the pattern variables in each of the vertical columns of the final array $\Big(\hat{c}_{t_1t_2-1}(\mathbf{Y}),\ \hat{c}_{t_1t_2}(\mathbf{Y})\Big)$ will be observed on the force of the fact that the values of these variables are found jointly as result of minimization of the respective partial objective function (2.8). But the values of neighboring references in horizontal rows $\Big(\hat{k}_{t_1-1,t_2}(\mathbf{Y}),\ \hat{k}_{t_1t_2}(\mathbf{Y})\Big)$ will be obtained from different objective functions and may turn out to be mutually contradictory.

One of the reasonable ways of overcoming this inconsistency would be to re-evaluate the references independently in each row t_2 with respect to the fixed values of pattern variables $\hat{c}_{t_1t_2}(\mathbf{Y})$ and in accordance with the condition

$$\Big(\hat{k}_{1t_2}(\mathbf{Y}),\dots,\hat{k}_{N_1t_2}(\mathbf{Y})\Big) = \arg\min_{k_{t_1t_2}\in\mathcal{K}_{t_2}} J'_{t_2}\Big(k_{1t_2},\dots,k_{N_1t_2}|\hat{c}_{1t_2}(\mathbf{Y}),\dots,\hat{c}_{N_1t_2}(\mathbf{Y})\Big),$$

$$J'_{t_2}\Big(k_{1t_2},\dots,k_{N_1t_2}|\hat{c}_{1t_2}(\mathbf{Y}),\dots,\hat{c}_{N_1t_2}(\mathbf{Y})\Big) = \sum_{t_1=1}^{N_1}\psi\left(\hat{c}_{t_1t_2}(\mathbf{Y}), k_{t_1t_2}|\mathbf{y}_{t_2}\right) + \sum_{t_1=2}^{N_1}\mu(k_{t_1-1,t_2}, k_{t_1t_2}).$$

The objective function of such a re-evaluation is separable with chain-like adjacency of variables. To find the sought-for corrected sequence of references in each row t_2, it is enough to apply to this row, once again, the same dynamic programming procedure (3.13)-(3.17) which is the main constituent part of the generalized data analysis procedure, for this time, with $k_{t_1t_2} \in \mathcal{K}_{t_2}$, $t_1 = 1, \ldots, N_1$, $t_2 = \text{const}$, instead of $x_t \in \mathcal{X}$, $t = 1, \ldots, N$.

Analogously, in the horizontal version (Fig. 2.8,d), minimization of each of the partial objective functions (2.10) results in the complete correctness of adjacent references $\big(\hat{k}_{t_1-1,t_2}(\mathbf{Y}),\, \hat{k}_{t_1t_2}(\mathbf{Y})\big)$ in the horizontal rows, whereas the vertically neighboring values of pattern variables $\big(\hat{c}_{t_1t_2-1}(\mathbf{Y}),\ \hat{c}_{t_1t_2}(\mathbf{Y})\big)$ are obtained from different objective functions, and it is quite likely that the assumed vertical identity or mutual similarity of pattern sequences will not be provided. The way of improving this defect consists in re-evaluating the pattern variables independently in each column t_1 with respect to the fixed values of references $\hat{k}_{t_1t_2}(\mathbf{Y})$:

$$\big(\hat{c}_{t_11}(\mathbf{Y}), \ldots, \hat{c}_{t_1N_2}(\mathbf{Y})\big) = \arg \min_{c_{t_1t_2} \in \mathcal{C}} J''_{t_1}\big(c_{t_11}, \ldots, c_{t_1N_2} | \hat{k}_{t_11}(\mathbf{Y}), \ldots, \hat{k}_{t_1N_2}(\mathbf{Y})\big),$$

$$J''_{t_1}\big(c_{t_11}, \ldots, c_{t_1N_2} | \hat{k}_{t_11}(\mathbf{Y}), \ldots, \hat{k}_{t_1N_2}(\mathbf{Y})\big) = \sum_{t_2=1}^{N_2} \psi\left(c_{t_1t_2}, \hat{k}_{t_1t_2}(\mathbf{Y}) | \mathbf{y}_{t_2}\right) + \sum_{t_2=2}^{N_2} \gamma(c_{t_1t_2-1}, c_{t_1t_2}).$$

In this case, the dynamic programming procedure (3.13)-(3.17) is applied once again to each column with $c_{t_1t_2} \in \mathcal{C}$, $t_2 = 1, \ldots, N_2$, $t_1 = \text{const}$, instead of $x_t \in \mathcal{X}$, $t = 1, \ldots, N$.

5 Conclusions: About what we have done and what questions remain to be answered

We have considered three, seemingly, absolutely different groups of data analysis problems. Nevertheless, it turned out that there exists such an interpretation of these problems which allows for a formal treatment of all of them at once as variations of a standard optimization problem. We considered the structure of the data array subject to an analysis as collection of elementary information units y_t at the nodes $t \in T$ of an appropriate undirected graph G meant to express the natural neighborhood $(t', t'') \in G$ of data array elements, and represented the original data array as a function $\mathbf{Y} = (y_t,\, t \in T)$ that is defined on the set of nodes T and takes values $y_t \in \mathcal{Y}$ from an appropriate range $\mathcal{Y}$. Within the bounds of this work, our concern has been restricted to the variety of practical data analysis problems which can be represented as problems of transforming the original data array into another function $\mathbf{X} = (x_t,\, t \in T)$ on the same set T with a range of values $x_t \in \mathcal{X}$ specific for each particular data analysis problem.

We have shown that the ultimate intent of data analysis in such problems consists, as a rule, in co-ordinating two kinds of the available information on the values of secondary variables x_t, namely, the data-dependent information usually carried by respective fragments of the original data array, on the one hand, and the model-based one that imposes some a priori constraints upon the pairs of adjacent hidden values $(x_{t'}, x_{t''})$, $(t', t'') \in G$, on the other.

In accordance with this principle, we reformulated the generalized data analysis problem and preliminarily set it as that of finding the global minimum point $\hat{\mathbf{X}}(\mathbf{Y}) = (\hat{x}_t(\mathbf{Y}), t \in T)$ of a separable objective function

$$J(\mathbf{X}|\mathbf{Y}) = \sum_{t \in T} \psi_t(x_t|\mathbf{Y}) + \sum_{(t',t'') \in G} \gamma_{t',t''}(x_{t'}, x_{t''}),$$

where elementary functions $\psi_t(x_t|\mathbf{Y})$ and $\gamma_{t',t''}(x_{t'}, x_{t''})$, respectively, data-dependent and model-based, received the names of node and edge functions.

In this concluding Section, we briefly review the results obtained within the bounds of the reported investigation, and pose some questions which naturally arise from the principal idea of the approach proposed here but remain unanswered as yet.

For a separable function of general kind, the problem of global optimization does not lend itself to an effective algorithmical solution. Therefore, we reformulated the generalized data analysis problem once again by heuristic splitting it into a series of partial optimization problems with tree-like separability of the particular objective functions for appropriately chosen nonoverlapping subsets of goal variables. Each of the particular objective functions is function of all the variables, but only the optimal values of the variables in the respective subset are meant to participate in forming the entire result of processing $\hat{\mathbf{X}}(\mathbf{Y})$.

What is the relation between the hypothetical strictly optimal result delivering the minimum value to the original separable objective function and the combined solution assembled of partial decisions for subsets of variables after the variable adjacency graph of general kind is decomposed into a series of tree-like ones?

We considered here only one class of data analysis problems which allow for treating them as formal problems of optimization of a separable objective function, namely, the class for which the natural kind of variable adjacency graph is rectangular lattice.

What other kinds of variable adjacency are adequate to practical applications?

For the lattice-like adjacency graphs, we proposed only one heuristic way of replacing the original objective function by a series of particular ones with tree-like separability of variables, namely, that in which all the particular trees have the same horizontal branches and differ from each other only in the variables forming the stem.

What are disadvantages of such a tree-like decomposition of the variable adjacency lattice in each of the particular data analysis problems considered in this work?

What other ways of decomposition can be proposed for the lattice-like adjacency graph in each of these problems?

We have shown that optimization of a separable function $J(x_1, \ldots, x_N)$ with tree-like adjacency of $N = |T|$ variables can be easily fulfilled by an effective computing procedure based on reducing the original problem into a series of N partial optimization problems each for an intervening function of only one variable $\tilde{J}_t(x)$. In case the tree of variable adjacency is chain, such a procedure boils down to the classical dynamic programming procedure, and the intervening functions to be optimized at each step become nothing else than Bellman functions, therefore, we call them here the extended Bellman functions.

What other particular kinds of variable adjacency in a separable objective function provide a substantial computational simplification of the optimization problem as compared to the general case, if to a lesser extent than the tree-like one?

The numerical realization of the generalized dynamic programming procedure presupposes that there exists a finitely parametrized function family $\tilde{J}(x; a)$ concordant with the node functions $\psi_t(x_t)$ and edge functions $\gamma_{t',t''}(x_{t'}, x_{t''})$ in the sense that all the extended Bellman functions belong to this family $\tilde{J}_t(x) = \tilde{J}(x, a_t)$. In addition to the trivial discrete case when the arguments of the objective function take values from a finite set $x_t \in \mathcal{X} = \{1, \ldots, m\}$, we studied only one example of such a parametric family, namely, that of quadratic functions for the case when the range of variables is n-dimensional real space $x_t \in \mathsf{R}^n$ and the node and edge functions are quadratic.

What other parametric families of extended Bellman functions conjugate with practically suitable classes of node and edge functions exist for discrete as well as for continuous ranges of goal variables?

For the class of data analysis problems for which the rectangular lattice-like adjacency of goal variables is the natural basis of treating them as problems of optimization of a separable objective function, with the proviso that there are no contra-indications against the heuristic replacement of the original lattice by a series of trees with common horizontal branches, we created a generalized data analysis procedure which performs the processing of the data array $\mathbf{Y} = (y_t, t \in T)$ in twice as many runs through all of its elements than the spatial dimensionality of the original lattice. The result of each of the runs is stored in the memory in the form an intervening array $\mathbf{A} = (a_t, t \in T)$ in which the original data elements are replaced by parameter values representing the Bellman functions at each of the lattice nodes in terms of the chosen parametric family. The result of processing $\hat{\mathbf{X}}(\mathbf{Y}) = (\hat{x}_t(\mathbf{Y}), t \in T)$ is array of minimum points of the Bellman functions obtained at the last run. The essence of the particular data analysis problem affects only the class of node and edge functions and, respectively, the specificity of the parametric family containing the intervening Bellman functions.

Finally, we considered the specific traits of each of the particular data analysis problems that had served, throughout the report, as basis for the generalizations. As regards visual and seismic explorative data, the generalized data analysis procedure depicted in this report has already been successfully used in the engineering practice, whereas the new approach to the problem of multi-alignment of molecular sequences is reported here as a project that needs some further improvements and experimental testing.

Bibliography

[1] Besag, J.E. On the statistical analysis of dirty pictures (with discussion). *Journ. Royal Statist. Soc.*, B 48, 1986, pp. 259-302.

[2] Kittler, J., and Föglein, J. Contextual classification of multispectral pixel data. *Image Vision Comput.*, 1984, 2, 13-29.

[3] Ripley, B.D. Statistics, images and pattern recognition (with discussion). *Canad. J. Statist.*, 1986, 14, pp. 83-111.

[4] Geman, S., and Geman, D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on PAMI*, Vol. 6, November 1984, pp. 721-741.

[5] Smith, A.F.M., and Roberts, G.O. Bayesian computation via the Gibbs sampler and related Markov Chain Monte Carlo methods. *J. R. Statist. Soc.*, 1993, B55, No.1, pp. 3-23.

[6] Tierney, L. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 1994, Vol. 22, No. 4, pp. 1701-1762.

[7] Pincus, M. A closed form solution of certain programming problems. *Oper. Res.*, 1968, 16, pp. 690-694.

[8] Pincus, M. A Monte-Carlo method for the approximate solution of certain types of constrained optimization problems. *Oper. Res.*, 1970, 18, 1225-1228.

[9] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. Optimization by simulated annealing. *Science*, 1983, 220, 671-680.

[10] Bellman, R. *Dynamic Programming.* Princeton University Press, Princeton, N.J., 1957.

[11] Waterman, M.S., Smith, T.F., and Beyer, W.A. Some biological sequence metrics. *Advanced Mathematics*, 20, 1976, pp. 367-387.

[12] Taylor, W.R. Identification of protein sequence homology by consensus template alignment. *Journal of Molecular Biology*, 188, 1986, pp. 233-258.

[13] Taylor, W.R. A flexible method to align large numbers of biological sequences. *Journal of Molecular Evolution*, 28, 1988, pp. 161-169.

[14] Mirkin, B., and Roberts, F.S. Consensus functions and patterns in molecular sequences. *Technical Report* 92-37. DIMACS, Rutgers University, August, 1992.

[15] Gribskov, M., McLachlan A.D., and Eisenberg, D. Profile analysis: detection of distantly related proteins. *Proc. Nat. Acad. Sci. USA*, 84, 1987, pp. 4355-4358.

[16] Smith, R.F., and Smith, T.F. Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modeling. *Protein Engineering*, 5, 1992, pp. 35-41.

[17] Bashford, D., Chothia, C., and Lesk, A.M. Determinants of a protein fold. *Journal of Molecular Biology*, 196, 1987, pp. 199-216.

[18] Mottl, V.V., Kopylov, A.V., Blinov, A.B., and Zheltov, S.Yu. Quasi-statistical approach to the problem of stereo image matching. *Proceedings SPIE*, 2363, 1994, pp. 50-61.

[19] Mottl, V.V., Muchnik, I.B., Ivanova, T.O., and Blinov, A.B. Automatic classification of local peculiarities in data fields with the use of hidden Markov models. In: *Mathematical Problems of Classification*, A.P. Kovalenko, and B.G. Mirkin (Ed.). *Survey of Applied and Industrial Mathematics*, 3, No. 1. Moscow, Publishing House "Probability Theory and its Applications", 1996 (in Russian).

[20] Mottl, V.V., Muchnik, I.B., Blinov, A.B., and Kopylov, A.V. Hidden tree-like quasi-Markov model and generalized technique for a class of image processing problems. *13th International Conference on Pattern Recognition*. Vienna, Austria, August 25-29, 1996. Track B, pp. 715-719.

[21] Mottl, V.V., Blinov, A.B., Kopylov, A.V., and Kostin, A.A. Optimization techniques on pixel neighborhood graphs for image processing. *Proceedings of the International Workshop on Graph-Based Representations*. Lyon, France, April 17-18, 1997 (to appear).

[22] Box, G.E.P., and Jenkins, G.M. *Time Series Analysis. Forecasting and Control.* Holden-Day, 1970.