NETWORK FLOW OPTIMIZATION FOR RESTORATION OF IMAGES

BORIS A. ZALESKY

Received 5 October 2001

The network flow optimization approach is offered for restoration of gray-scale and color images corrupted by noise. The Ising models are used as a statistical background of the proposed method. We present the new multiresolution network flow minimum cut algorithm, which is especially efficient in identification of the maximum a posteriori (MAP) estimates of corrupted images. The algorithm is able to compute the MAP estimates of large-size images and can be used in a concurrent mode. We also consider the problem of integer minimization of two functions, $U_1(\mathbf{x}) = \lambda \sum_i |y_i - x_i| + \sum_{i,j} \beta_{i,j}|x_i - x_j|$ and $U_2(\mathbf{x}) = \sum_i \lambda_i (y_i - x_i)^2 + \sum_{i,j} \beta_{i,j}(x_i - x_j)^2$, with parameters $\lambda, \lambda_i, \beta_{i,j} > 0$ and vectors $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n) \in \{0, \ldots, L-1\}^n$. Those functions constitute the energy ones for the Ising model of color and gray-scale images. In the case $L = 2$, they coincide, determining the energy function of the Ising model of binary images, and their minimization becomes equivalent to the network flow minimum cut problem. The efficient integer minimization of $U_1(\mathbf{x})$, $U_2(\mathbf{x})$ by the network flow algorithms is described.

1. Introduction

We present a new multiresolution algorithm for finding the minimum network flow cut and methods of efficient integer minimization of the function $U_1(\mathbf{x}) = \lambda \sum_i |y_i - x_i| + \sum_{i,j} \beta_{i,j}|x_i - x_j|$ and $U_2(\mathbf{x}) = \sum_i \lambda_i (y_i - x_i)^2 + \sum_{i,j} \beta_{i,j}(x_i - x_j)^2$. The problem is posed in terms of Bayesian approach to image restoration to have unified canvas of presentation of the results, and since the results developed were tested and turned out efficient for the processing of corrupted images. Nevertheless, the minimization of

Journal of Applied Mathematics 2:4 (2002) 199–218
2000 Mathematics Subject Classification: 62M40, 90C10, 90C35, 68U10
URL: http://dx.doi.org/10.1155/S1110757X02110035

the functions $U_1(\mathbf{x})$ and $U_2(\mathbf{x})$ concerns not only the maximum a posteriori (MAP) estimation, it can be understood, for instance, as l_1- and l_2-regression and so on.

The restoration of degraded images is a branch of image processing that is now extensively studied for its evident practical importance as well as theoretical interest. There are many approaches to solution of the problem.

We consider here the MAP Bayesian estimation [5, 6]. As usual, in this statistical method we suppose an unknown image to be corrupted by a random noise, and that a corrupted version of the image is only observable. The unknown original image is presented as a random realization of some Markov random field (MRF), which is independent of the random noise. The distributions of the MRF and the random noise are assumed to be known. The MAP estimate is determined as the maximum likelihood evaluation of the conditional distribution of the original image given the observed one.

Because of the use of an appropriate prior information, the Bayesian MAP estimators are among those which give the best quality of restored images. The theory of evaluation of the MAP estimates is also significantly developed [5, 6, 7, 8]. But one of the disadvantages of almost all known algorithms is their NP-hardness (their execution time is exponent in a degree of image size).

The first efficient method for evaluation of the Ising MAP estimate of binary images, which requires polynomial execution time in number of image pixels, was proposed by Greig et al. [8]. The authors used the network flow optimization approach to reduce the problem to the identification of the maximum network flow. The finding of the exact Ising MAP estimate of binary image required $O(n^3)$ operations, where n was an image size. But known maximum network flow algorithms still did not allow computation of the real binary images having size $n = 256 \times 256$ or more. Therefore, in [8] a heuristic modification of the algorithm has been described. That algorithm was implemented not for entire images but for pieces of images with fixed values at their frontiers.

The similar idea was used for the proposed *multiresolution network flow minimum cut algorithm* (MNFC) (actually, to find the MAP of a corrupted image, we need only the minimum network flow cut). Some additional theoretical reasons allowed describing a new algorithm, which identifies the exact minimum cut using special modifications of subnetworks of the partitioned original network. This algorithm can be exploited for an arbitrary network. At worst, it takes $O(n^3)$ operations for computing the minimum network flow cut, but for networks that correspond to real corrupted images, it usually requires only $O(n)$ operations, which would be implemented in a concurrent mode.

The rest of the paper is organized as follows. In Section 2, we present the description, theoretical ground, and use of the multiresolution network flow minimum cut algorithm. The brief entry to two Ising models of color and gray-scale images is done in Section 3. Those models are widely used for reconstruction of corrupted images, they coincide when images are binary [5, 6, 7]. In the same section the problem of finding the exact MAP estimates is formulated in terms of the integer programming. In Section 4, we describe methods of the integer minimization of the functions (which are the energy functionals of the considered Ising models) $U_1(\mathbf{x}) = \lambda \sum_i |y_i - x_i| + \sum_{i,j} \beta_{i,j} |x_i - x_j|$ and $U_2(\mathbf{x}) = \sum_i \lambda_i (y_i - x_i)^2 + \sum_{i,j} \beta_{i,j} (x_i - x_j)^2$ with parameters $\lambda, \beta_{i,j} > 0$ and vectors $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n) \in \{0, \ldots, L-1\}^n$ by the network flow algorithms (including the MNFC). Applications of the developed methods is given in Section 5.

2. Multiresolution network flow minimum cut algorithm

2.1. *Preliminaries and notations*

We now describe the *multiresolution network flow minimum cut algorithm*. For some types of networks, this highly parallel method turns out more speedy and more efficient in comparison with known maximum network flow algorithms. It was successfully used for identification of minimum cuts of large networks (for instance, for determination of the MAP of binary images [8]), while classical methods were not able to solve the problem.

The essence of the MNFC is the following. A network $\mathfrak{G}$ is partitioned into several subnetworks of appropriate sizes. Every subnetwork is modified in a special way and evaluated two times. The first time, all boundary arcs going from the outside into the subnetwork are considered as going from the source (to the same nodes), and all boundary arcs going from within the subnetwork outside are ruled out of the consideration. The minimum cut of the modified subnetwork is determined by known algorithm. It is proven that all nodes, which are connected with the sink by a directed path, will be connected with the sink in the solution of the original network by the same path. The second time, all boundary arcs going from the outside into the subnetwork are excluded from the consideration, and all boundary arcs going from within the subnetwork outside are supposed to be connected with the sink. The minimum cut of the modified subnetwork is identified once more. This time all directed paths going from the source are determined. The paths identified are excluded from the further consideration. The reduced network $\mathfrak{G}_1$ is divided into several subnetworks again. The procedure of

identification of arcs belonging to $\mathfrak{G}_1$ but not belonging to the minimal cut is repeated now with one difference: we take into account arcs connected with excluded nodes. The arcs not belonging to the minimal cut of $\mathfrak{G}_1$ are found and removed anew. We go to higher levels until we obtain the network $\mathfrak{G}_k$ that can be solved by a usual maximum network flow algorithm.

In more details, let the network $\mathfrak{G}$ consist of $n+2$ numbered *nodes* $\tilde{S} = \{0,1,2,\dots,n,n+1\}$, where $s=0$ is the *source*, $t=n+1$ is the *sink*, and $S = \{1,2,\dots,n\}$ are usual nodes. The set of directed *arcs* is $A = \{(i,j) : i,j \in \tilde{S}\}$. Capacities of arcs are denoted by $d_{i,j} > 0$. Suppose that the network $\mathfrak{G}$ satisfies the following condition:

(G) For every usual node $i \in S$ there is either an arc $(s,i) \in A$ connecting the source s with i, or an arc $(i,t) \in A$ connecting i with the sink t, but not both of these arcs.

Remark 2.1. Condition (G) does not restrict the use of the MNFC. It has been done to simplify writing down notation and proofs. Any network $\mathfrak{G}$ can be easily modified to satisfy (G) for $O(n)$ operations (one look through usual network nodes S). The modified network will have the same minimum cut.

Let the sets $W, B \subset S$, such that $W \cap B = \emptyset$ and $W \cup B = S$, be a partition of S. The vector $\mathbf{x} : S \to \{0,1\}^{|S|}$ with coordinates $x_i = 1$ if $i \in W$, and $x_i = 0$ otherwise, is the indicator vector of the set W. The set of all such indicator vectors $\mathbf{x}$ is denoted by $\mathbf{X}$. The *capacity of the s-t cut* $C(\mathbf{x})$ is defined as the sum of capacities of all forward arcs going from the set $W \cup \{s\}$ to the set $B \cup \{t\}$ [9], that is,

$$C(\mathbf{x}) = \sum_{i \in W \cup \{s\}} \sum_{j \in B \cup \{t\}} d_{i,j}. \tag{2.1}$$

Let $\mathbf{y}$ be the vector with coordinates $y_i = 1$, if $(s,i) \in A$, or $y_i = 0$, if $(i,t) \in A$ (remember that condition (G) is valid). Set $\lambda_i = d_{s,i}$ if $(s,i) \in A$, or $\lambda_i = d_{i,t}$ if $(i,t) \in A$. The capacities of usual arcs (i,j), $i,j \in S$ will be denoted by $\beta_{i,j} = d_{i,j}$. Then in new notation,

$$C(\mathbf{x}) = \sum_{\substack{i \in W, \\ y_i = 0}} \lambda_i + \sum_{\substack{i \in B, \\ y_i = 1}} \lambda_i + \sum_{(i,j) \in S \times S} \beta_{i,j} (x_i - x_j) x_i. \tag{2.2}$$

Now introduce the function

$$U(\mathbf{x}) = \sum_{i \in S} \lambda_i (1 - 2y_i) x_i + \sum_{(i,j) \in S \times S} \beta_{i,j} (x_i - x_j) x_i. \tag{2.3}$$

It is easy to see that $C(\mathbf{x}) = U(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i y_i$. Since the term $\sum_{i=1}^{n} \lambda_i y_i$ does not depend on $\mathbf{x}$, the functions $C(\mathbf{x})$ and $U(\mathbf{x})$ have minima in the same points, $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in X} C(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x}\in X} U(\mathbf{x})$. Therefore, the solutions $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in X} U(\mathbf{x})$ entirely identify minimum network flow cuts [4, 10].

For an arbitrary subset of usual nodes $E \subset S$, we define two functions,

$$\begin{aligned} U_E(\mathbf{x}) &= \sum_{i\in E} \lambda_i (1-2y_i) x_i + \sum_{(i,j)\in (E\times E)\cup(E\times E^c)\cup(E^c\times E)} \beta_{i,j}(x_i - x_j)x_i, \\ V_E(\mathbf{x}) &= \sum_{i\in E^c} \lambda_i (1-2y_i) x_i + \sum_{(i,j)\in (E^c\times E^c)} \beta_{i,j}(x_i - x_j)x_i, \end{aligned} \tag{2.4}$$

whose sum is equal to $U(\mathbf{x}) = U_E(\mathbf{x}) + V_E(\mathbf{x})$. Also, we define the restriction $\mathbf{x}_E$ of $\mathbf{x}$ onto the set E such that $\mathbf{x} = (\mathbf{x}_E, \mathbf{x}_{E^c})$. The function $V_E(\mathbf{x})$ depends only on $\mathbf{x}_E$ (i.e., $V_E(\mathbf{x}) = V_E(\mathbf{x}_E)$), therefore the following simple proposition is valid.

Proposition 2.2. *If $\mathbf{x}'_E$ minimizes the function $\phi(\mathbf{x}_E) = U(\mathbf{x}_E, \mathring{\mathbf{x}}_{E^c})$ for some fixed $\mathring{\mathbf{x}}_{E^c}$, then for any set $D \subset E$, the restriction $\mathbf{x}'_D$ of $\mathbf{x}'_E$ minimizes the function $\psi(\mathbf{x}_D) = U_D(\mathbf{x}_D, \mathbf{x}'_{E\setminus D}, \mathring{\mathbf{x}}_{E^c})$ and vice versa.*

For vectors $\mathbf{x}_E$, $\mathbf{z}_E$ we write

$$\begin{aligned} &\mathbf{x}_E \le \mathbf{z}_E, \quad \text{if } x_i \le z_i,\ i \in E, \\ &\mathbf{x}_E < \mathbf{z}_E, \quad \text{if } x_i \le z_i,\ i \in E; \end{aligned} \tag{2.5}$$

and there is at least one node $j \in E$ such that $x_j < z_j$; $\mathbf{x}_E \not\le \mathbf{z}_E$, if there are nodes $i, j \in E$ such that $x_i < z_i$ and $x_j > z_j$.

Our method is based on the monotone (in some sense) dependence of restrictions $\mathbf{x}^*_E$ of solutions $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in X} U(\mathbf{x})$ on values $\mathbf{x}^*_{E^c}$.

2.2. Properties of minimum cuts

Consider the property of the monotony of $\mathbf{x}^*_E$ in details. For fixed $\mathring{\mathbf{x}}_{E^c}$ and $\mathring{\mathbf{z}}_{E^c}$, set $\mathbf{x}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \mathring{\mathbf{x}}_{E^c})$ and $\mathbf{z}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \mathring{\mathbf{z}}_{E^c})$. It is clear that every set $\{\mathbf{x}'_E\}_{\mathring{\mathbf{x}}_{E^c}}$, $\{\mathbf{z}'_E\}_{\mathring{\mathbf{z}}_{E^c}}$ can consist of more than one element. In general, in the case $\mathring{\mathbf{z}}_{E^c} \le \mathring{\mathbf{z}}_{E^c}$ the inequality $\mathbf{x}'_E \le \mathbf{z}'_E$ is not satisfied, but without fail there exist solutions $\mathbf{x}'_E$ and $\mathbf{z}'_E$ such that $\mathbf{x}'_E \le \mathbf{z}'_E$.

Theorem 2.3. *Suppose that $\mathring{\mathbf{x}}_{E^c} \le \mathring{\mathbf{z}}_{E^c}$ are fixed and let $\mathbf{x}'_E \not\le \mathbf{z}'_E$ (i.e., solutions $\mathbf{x}'_E$ and $\mathbf{z}'_E$ are not comparable). Then for nonempty set $D = \{i \in E : x'_i = 1,\ z'_i = 0\}$, the vector $(\mathbf{z}'_D, \mathbf{x}'_{E\setminus D}) = (\mathbf{0}_D, \mathbf{x}'_{E\setminus D})$ minimizes the function U under*

condition $\overset{\circ}{\mathbf{x}}_{E^c}$, *and the vector* $(\mathbf{x}'_D, \mathbf{z}'_{E\setminus D}) = (\mathbf{1}_D, \mathbf{z}'_{E\setminus D})$ *minimizes the function* U *under condition* $\overset{\circ}{\mathbf{z}}_{E^c}$, *that is,* $(\mathbf{z}'_D, \mathbf{x}'_{E\setminus D}) \in \{\mathbf{x}'_E\}_{\overset{\circ}{\mathbf{x}}_{E^c}}$, $(\mathbf{x}'_D, \mathbf{z}'_{E\setminus D}) \in \{\mathbf{z}'_E\}_{\overset{\circ}{\mathbf{z}}_{E^c}}$, *and* $(\mathbf{z}'_D, \mathbf{x}'_{E\setminus D}) \le (\mathbf{x}'_D, \mathbf{z}'_{E\setminus D})$.

Proof. Suppose that $\mathbf{x}'_E \in \{\mathbf{x}'_E\}_{\overset{\circ}{\mathbf{x}}_{E^c}}$ and $\mathbf{z}'_E \in \{\mathbf{z}'_E\}_{\overset{\circ}{\mathbf{z}}_{E^c}}$ are two incomparable solutions $\mathbf{x}'_E \not\le \mathbf{z}'_E$ for frontier conditions $\overset{\circ}{\mathbf{x}}_{E^c} \le \overset{\circ}{\mathbf{z}}_{E^c}$. It follows from Proposition 2.2, equalities $\mathbf{x}'_D = \mathbf{1}_D$, and $\mathbf{z}'_D = \mathbf{0}_D$, and the inequality $(\mathbf{x}'_{E\setminus D}, \overset{\circ}{\mathbf{x}}_{E^c}) \le (\mathbf{z}'_{E\setminus D}, \overset{\circ}{\mathbf{z}}_{E^c})$ that

$$U_D\left(\mathbf{z}'_D, \mathbf{z}'_{E\setminus D}, \overset{\circ}{\mathbf{z}}_{E^c}\right) \le U_D\left(\mathbf{x}'_D, \mathbf{z}'_{E\setminus D}, \overset{\circ}{\mathbf{z}}_{E^c}\right) \le U_D\left(\mathbf{x}'_D, \mathbf{x}'_{E\setminus D}, \overset{\circ}{\mathbf{x}}_{E^c}\right). \tag{2.6}$$

And by analogy,

$$U_D\left(\mathbf{x}'_D, \mathbf{x}'_{E\setminus D}, \overset{\circ}{\mathbf{x}}_{E^c}\right) \le U_D\left(\mathbf{z}'_D, \mathbf{x}'_{E\setminus D}, \overset{\circ}{\mathbf{x}}_{E^c}\right) \le U_D\left(\mathbf{z}'_D, \mathbf{z}'_{E\setminus D}, \overset{\circ}{\mathbf{z}}_{E^c}\right). \tag{2.7}$$

Gathering estimates (2.6) and (2.7) in one chain, we can see that all inequalities in (2.6) and (2.7) are actually equalities. This fact and Proposition 2.2 finish the proof. □

The following corollary, which characterizes some structural properties of the set of solutions $\{\mathbf{x}^*\}$, is deduced from Theorem 2.3.

Corollary 2.4. (i) *If fixed frontier vectors satisfy the condition* $\overset{\circ}{\mathbf{x}}_{E^c} \le \overset{\circ}{\mathbf{z}}_{E^c}$, *then for any solution* $\mathbf{x}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \overset{\circ}{\mathbf{x}}_{E^c})$, *there exists a solution* $\mathbf{z}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \overset{\circ}{\mathbf{z}}_{E^c})$ *such that* $\mathbf{x}'_E \le \mathbf{z}'_E$.

(ii) *If fixed frontier functions satisfy the condition* $\overset{\circ}{\mathbf{x}}_{E^c} \le \overset{\circ}{\mathbf{z}}_{E^c}$, *then for any solution* $\mathbf{z}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \overset{\circ}{\mathbf{z}}_{E^c})$, *there exists a solution* $\mathbf{x}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \overset{\circ}{\mathbf{x}}_{E^c})$ *such that* $\mathbf{x}'_E \le \mathbf{z}'_E$.

(iii) *For any frontier condition* $\overset{\circ}{\mathbf{x}}_{E^c}$, *the set* $\{\mathbf{x}'_E\}_{\overset{\circ}{\mathbf{x}}_{E^c}}$ *has the minimal (the maximal) element* $\underline{\mathbf{x}}'_E$ ($\overline{\mathbf{x}}'_E$).

(iv) *If* $\overset{\circ}{\mathbf{x}}_{E^c} \le \overset{\circ}{\mathbf{z}}_{E^c}$, *then* $\underline{\mathbf{x}}'_E \le \underline{\mathbf{z}}'_E$ *and* $\overline{\mathbf{x}}'_E \le \overline{\mathbf{z}}'_E$.

Sentences (i) and (ii) follow immediately from Theorem 2.3. Sentence (iii) is deduced from Theorem 2.3 for $\overset{\circ}{\mathbf{x}}_{E^c} = \overset{\circ}{\mathbf{z}}_{E^c}$. And (iv) follows from (i), (ii), and the definitions of minimal and maximal elements.

To identify a local solution $\mathbf{x}'_E = \operatorname{argmin}_{\mathbf{x}_E} U(\mathbf{x}_E, \overset{\circ}{\mathbf{x}}_{E^c})$, we use the network $\mathfrak{G}'_E$ with nodes $\tilde{S}_E = \{E, \{s\}, \{t\}\}$, and arcs (i,j), $i,j \in \tilde{S}$, having

capacities

$$d_{i,j} = \beta_{i,j}, \quad i,j \in E; \qquad d_{s,i} = \lambda_i y_i + \sum_{\substack{j \in E^c \\ x_j = 1}} \beta_{j,i},$$
$$d_{i,t} = \lambda_i (1 - y_i) + \sum_{\substack{j \in E^c \\ x_j = 0}} \beta_{i,j}, \tag{2.8}$$

since the following proposition can be easily proved.

Proposition 2.5. *The local solution $\mathbf{x}'_E$ sets the minimum cut of the network $\mathfrak{G}'_E$ and vice versa.*

The subset $E \in S$ is chosen so that the maximum flow in the network $\mathfrak{G}'_E$ can be computed by usual maximum network flow algorithms.

2.3. The MNFC algorithm

The main idea of the MNFC is to estimate at least parts of the restrictions $\mathbf{x}^*_{E_i}$ of the solutions $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} U(\mathbf{x})$ for a suitable partition $\cap E_i = S$, $E_i \cap E_j = \emptyset$. For this purpose, the monotone dependence of local solutions $\mathbf{x}^*_{E_i}$ on the frontier values of $\mathbf{x}^*_{E_i^c}$ (in the sense of Corollary 2.4) is exploited. The parts of $\mathbf{x}^*_{E_i}$ estimated by the special local solutions $\mathbf{x}'_{E_i}$ are ruled out of the further consideration. It significantly reduces computational expenses.

More precisely, let the sets $E_1(1), \dots, E_{k_1}(1)$ partition the set S so that $\bigcap_{i=1}^{k_1} E_i(1) = \emptyset$ and $\bigcup_{i=1}^{k_1} E_i(1) = S$. It follows from Proposition 2.2 that $\mathbf{x}'_{E_i(1)} = \operatorname{argmin}_{\mathbf{x}_{E_i(1)}} U(\mathbf{x}_{E_i(1)}, \mathbf{x}^*_{E_i^c(1)}) = \mathbf{x}^*_{E_i(1)}$, and the solution $\mathbf{x}'_{E_i(1)}$ minimizes the function $U_{E_i(1)}(\mathbf{x}_{E_i(1)}, \mathbf{x}^*_{E_i^c(1)})$ for the frontier condition $\mathbf{x}^*_{E_i^c(1)}$. Corollary 2.4 guarantees the existence of two solutions (that can be found by known maximum network flow algorithms),

$$\mathbf{x}_{0,E_i(1)} = \operatorname{argmin}_{\mathbf{x}_{E_i(1)}} U\left(\mathbf{x}_{E_i(1)}, \mathbf{0}_{E_i^c(1)}\right),$$
$$\mathbf{x}_{1,E_i(1)} = \operatorname{argmin}_{\mathbf{x}_{E_i(1)}} U\left(\mathbf{x}_{E_i(1)}, \mathbf{1}_{E_i^c(1)}\right), \tag{2.9}$$

satisfying the inequality

$$\mathbf{x}_{0,E_i(1)} \le \mathbf{x}'_{E_i(1)} \le \mathbf{x}_{1,E_i(1)}. \tag{2.10}$$

Denote the sets on nodes by $B_i(1) = \{k \in E_i(1) \mid x_{1,E_i(1),k} = 0\}$ and $W_i(1) = \{k \in E_i(1) \mid x_{0,E_i(1),k} = 1\}$. The equalities $\mathbf{0}_{B_i(1)} = \mathbf{x}_{1,B_i(1)} = \mathbf{x}^*_{B_i(1)}$ and $\mathbf{1}_{W_i(1)} = \mathbf{x}_{0,W_i(1)} = \mathbf{x}^*_{W_i(1)}$ are inferred from (2.10).

If the set $R(1) = \bigcup_{i=1}^{k_1}(W_i(1) \cup B_i(1))$ is not empty, then we identified the part of the solution $\mathbf{x}^*_{R(1)}$. There is a sense to continue the MNFC. Assign $S(2) = S \backslash R(1)$ and consider now the reduced problem of identification $\mathbf{x}^*_{S(2)} = \operatorname{argmin}_{\mathbf{x}_{S(2)}} U(\mathbf{x}_{S(2)}, \mathbf{x}^*_{R(1)}) = \operatorname{argmin}_{\mathbf{x}_{S(2)}} U_{S(2)}(\mathbf{x}_{S(2)}, \mathbf{x}^*_{R(1)})$ for the frontier condition $\mathbf{x}^*_{R(1)}$. Partition $S(2) = \bigcup_{i=1}^{k_2} E_i(2)$, $E_i(2) \cap E_j(2) = \emptyset$, and estimate

$$\mathbf{x}'_{E_i(2)} = \mathbf{x}^*_{E_i(2)} = \operatorname{argmin}_{\mathbf{x}_{E_i(2)}} U_{E_i(2)}\left(\mathbf{x}_{E_i(2)}, \mathbf{x}^*_{S(2)\backslash E_i(2)}, \mathbf{x}^*_{R(1)}\right), \tag{2.11}$$

by the solutions

$$\begin{aligned} \mathbf{x}_{1,E_i(2)} &= \operatorname{argmin}_{\mathbf{x}_{E_i(2)}} U_{E_i(2)}\left(\mathbf{x}_{E_i(2)}, \mathbf{1}_{S(2)\backslash E_i(2)}, \mathbf{x}^*_{R(1)}\right), \\ \mathbf{x}_{0,E_i(2)} &= \operatorname{argmin}_{\mathbf{x}_{E_i(2)}} U_{E_i(2)}\left(\mathbf{x}_{E_i(2)} \mathbf{0}_{S(2)\backslash E_i(2)}, \mathbf{x}^*_{R(1)}\right), \end{aligned} \tag{2.12}$$

satisfying the inequality $\mathbf{x}_{0,E_i(2)} \le \mathbf{x}'_{E_i(2)} \le \mathbf{x}_{1,E_i(2)}$. Then, consider the sets $B_i(2) = \{k \in E_i(2) \mid x_{1,E_i(2),k} = 0\}$ and $W_i(2) = \{k \in E_i(2) \mid x_{0,E_i(2),k} = 1\}$, which identify $\mathbf{x}^*_{B_i(2)} = \mathbf{0}_{B_i(2)}$ and $\mathbf{x}^*_{W_i(2)} = \mathbf{1}_{W_i(2)}$ correspondingly. If the set $R(2) = \bigcup_{i=1}^{k_2}(W_i(2) \cup B_i(2))$ is nonempty, the algorithm is employed at the third level. The MNFC is iterated at higher levels until the problem is completely solved or until $R(l) \neq \emptyset$. At the uppermost level an appropriate known *maximum network flow algorithm* (MNF) is applied.

We describe the MNFC algorithm in brief.

Step 1 (Initialization). Assign the level number $l = 1$, the set of not estimated nodes $S(l) = S$, the set of estimated nodes $R = \emptyset$, and the set of arcs $A(l) = A$.

Step 2 (Partition of the network). If the maximum flow in the network $\{S(l), \{s\}, \{t\}, A(l)\}$ can be efficiently identified by usual MNF, we do not need its partition. Assign the number of partition elements $k_l = 1$, and $E_1(l) = S(l)$. Identify the maximum flow in $\{S(l), \{s\}, \{t\}, A(l)\}$, *stop*. Otherwise, partition the set $S(l)$ by not intersected sets $E_1(l), E_2(l), \ldots, E_{k_l}(l)$, $\bigcap_{i=1}^{k_l} E_i(l) = \emptyset$, $\bigcup_{i=1}^{k_l} E_i(l) = S(l)$, so that every network $\{E_i(l), \{s\}, \{t\}, A_i(l)\}$ with arcs $A_i(l) = A_{E_i}(l) = \{(\mu, \nu) : \mu \in E_i(l) \text{ or } \nu \in E_i(l)\}$ can be

evaluated by an appropriate MNF algorithm (e.g., by the labeling algorithm or other ones).

Step 3 (Computation of the local estimates $\mathbf{x}_{0,E_i(l)}$, $\mathbf{x}_{1,E_i(l)}$). Compute the vectors

$$\begin{aligned}\mathbf{x}_{0,E_i(l)} &= \operatorname{argmin}_{\mathbf{x}_{E_i(l)}} U\left(\mathbf{x}_{E_i(l)}, \mathbf{0}_{S(l)\setminus E_i(1)}, \mathbf{x}^*_R\right) \le \mathbf{x}_{1,E_i(l)} \\ &= \operatorname{argmin}_{\mathbf{x}_{E_i(l)}} U\left(\mathbf{x}_{E_i(l)}, \mathbf{1}_{S(l)\setminus E_i(l)}, \mathbf{x}^*_R\right)\end{aligned} \tag{2.13}$$

(see Corollary 2.4 and Proposition 2.5) by an appropriate MNF. These two vectors correspond to the minimum cut of the networks $\{E_i(l), \{s\}, \{t\}, A_i(l)\}$ for the frontier conditions $(\mathbf{0}_{S(l)\setminus E_i(1)}, \mathbf{x}^*_R)$ and $(\mathbf{1}_{S(l)\setminus E_i(1)}, \mathbf{x}^*_R)$, respectively.

Step 4 (Identification of the set $R(l)$ of estimated nodes). Find the sets of nodes $B_i(l) = \{i \in E_i(l) \mid x_{1,E_i(l),i} = 0\}$ and $W_i(l) = \{i \in E_i(l) \mid x_{0,E_i(l),i} = 1\}$ keeping their values in $\mathbf{x}^*$. Assign the set $R(l) = \bigcup_{i=1}^{k_l} (W_i(l) \cup B_i(l))$.

Step 5 (Check whether the multiresolution approach can be continued). If $R(l) = \emptyset$, interrupt execution of the MNFC and try to identify $\mathbf{x}^*_{S(l)} = \operatorname{argmin}_{\mathbf{x}_{S(l)}} U(\mathbf{x}_{S(l)}, \mathbf{x}^*_R)$ by an appropriate MNF, *stop*.

Step 6 (Jump to the higher level). Assign $R = R \cup R(l)$ and $S(l+1) = S(l)\setminus R$. If $S(l+1) = \emptyset$, the problem is solved—*stop*. Otherwise, go to the higher level, that is, assign $l = l+1$, $A(l) = \{(\mu, \nu) : \mu \in S(l) \text{ or } \nu \in S(l)\}$ and go to Step 2.

Step 3 of the MMC can be efficiently executed in a concurrent mode.

2.4. The number of operations and execution time

The number of operations required to execute the MNFC essentially depends on properties of the initial network $\mathfrak{G}$. In the case $R(1) = \emptyset$ the MNFC is reduced to the usual MNF algorithm. But there are a lot of networks admitting efficient application of the MNFC. Applications of the algorithm to the decision of real problems are described in Section 5. In those applications, identification of the network minimum cuts has required only $O(n)$ operations.

In the following proposition we formulate the simple necessary and sufficient condition in order that the MNFC does not degenerate to the usual maximum network flow algorithm.

Proposition 2.6. *The set $W_i(l) \neq \emptyset$ (resp., $B_i(l) \neq \emptyset$) if and only if there exists such a set $D \subset E_i(l)$ for which the inequality*

$$\begin{aligned} &\sum_i \lambda_i(1-2y_i) + \sum_{(i,j)\in(D\times D^c)} \beta_{i,j} \leq 0, \quad \textit{respectively}, \\ &\sum_i \lambda_i(1-2y_i) + \sum_{(i,j)\in(D^c\times D)} \beta_{i,j} \geq 0, \end{aligned} \tag{2.14}$$

is valid.

We will write down the estimates of the operations number N_{op} and the time of parallel execution T_{par} in general form and consider several special cases. Let $n_i(l)$ be the number of nodes in a subnetwork, $m_i(l)$ the number of arcs in a modified subnetwork, and $a_i(l)$ the number of boundary arcs such that $a_i(l) = (\mu, \nu)$, $\mu \in E_i(l)$, $\nu \in E_i^c(l)$, or $\nu \in E_i(l)$, $\mu \in E_i^c(l)$. It is supposed that the network satisfies condition (G) (recall that the modification of the network to satisfy this condition needs $O(n)$ operations). The amount of operations N_{op} has the same order as the number of operations required to construct all the local estimates $\mathbf{x}_{0,E_i(l)}$, $\mathbf{x}_{1,E_i(l)}$. But the computation of each local estimate requires $O(a_i(l)) + O(n_i^3(l))$ operations, where $O(a_i(l))$ operations are needed to modify the subnetwork capacities (see, (2.8)), and $O(n_i^3(l))$ operations are required directly to identify the local estimates (the second term $O(n_i(l)^3)$ depends on the used maximum network flow algorithm. It can be replaced by $O(n_i(l)m_i^2(l))$, etc.). If the number of levels required to compute the minimum cut is equal to L, then the number of operations is $N_{\mathrm{op}} = O(\sum_{l=1}^{L}\sum_{i=1}^{k_l}(a_i(l) + n_i^3(l)))$ and the parallel execution time is $T_{\mathrm{par}} = O(\sum_{l=1}^{L}\max_i(a_i(l) + n_i^3(l)))$.

First, we consider the worst case. Suppose that all usual nodes S are completely connected, that is, every two usual nodes are connected by two directed arcs. And suppose that a prior information about a strategy of partitions is not known. Then, the pyramidal approach allows to avoid excessive computational expenses. At level $1 \leq l \leq \log_2 n = L$, partition the original network into $2^{k(l)}$, $k(l) = \log_2 n - l$ subnetworks containing the same number of nodes. Therefore, at worst, the number of operations is $N_{\mathrm{op}} = O(n^3)$ and the parallel execution time is $T_{\mathrm{par}} = O(n^3)$. The amount of operations will be $O(n^3)$ if at each level only the small number of nodes is identified, and the others are determined at the uppermost level. Even in this case, computational expenses of the MNFC are not excessive in comparison with the traditional MNF algorithms.

The situation changes if there are large subnetworks that satisfy condition (2.14). The MNFC becomes very efficient. The number of operation can amount to $O(n)$. For instance, consider networks that arise from the Ising MAP estimation. Suppose that usual nodes S form the finite 2D lattice, $S = \{1,\dots,N\} \times \{1,\dots,N\}$, and that every usual node has 2D index, $i = (i_1, i_2)$. The nearest-neighbor nodes are connected by two differently directed arcs (i,j) and (j,i) with the same capacity $\beta_{i,j} = \beta$. Besides, suppose that each node is connected either with the source s or with the sink t by directed arcs having capacity λ. Obviously, in this case condition (2.14) will be satisfied for the subnetworks with nodes having constant values in $\mathbf{y}$ such that the set of these nodes possesses small enough isoperimetric ratio. Networks that correspond to corrupted binary images usually contain large (or even, as so-called, gigantic) subnetworks satisfying the property mentioned above. Those subnetworks are often estimated by the MNFC and ruled out the further consideration at the first level of execution of the algorithm. So that the evaluation of the MAP estimate takes fractions of a second, while the classical maximum network flow algorithms cannot identify the solution at all. To justify this arguments, we formulate the following sentence.

Proposition 2.7. (i) *Let* $B_{\mathbf{y}} = \{i \in S \mid y_i = 0\}$ *or* $W_{\mathbf{y}} = \{i \in S \mid y_i = 1\}$ *be the connected component of usual nodes of the constant value in* $\mathbf{y}$. *If* $\beta \le \lambda/2$ *and* $B_{\mathbf{y}}$ *(resp.,* $W_{\mathbf{y}}$*) has a contour,* $B_{\mathbf{y}}$ *(resp.,* $W_{\mathbf{y}}$*) maintains the same value in the minimum cut* $\mathbf{x}^*$.

(ii) *If for some* $M > 0$ *the capacities satisfy the inequality* $\beta \le (M/2(M+1))\lambda$, *then every connected component* $B_{\mathbf{y}}$ *(resp.,* $W_{\mathbf{y}}$*) of constant values in* $\mathbf{y}$, *consisting of more than* M *nodes, maintains the same value in the minimum cut* $\mathbf{x}^*$.

In the case $\beta > \lambda/2$, the MNFC also easily finds the MAP estimates of real binary images (i.e., equivalent to the identification of the minimum cut of the large network). The results of the practical restoration of corrupted images by the MNFC are placed in Section 5.

3. Two Ising models of color and gray-scale images

We describe the two simplest Ising models of color and gray-scale images with energy functions $U_1(\mathbf{x}) = \lambda \sum_i |y_i - x_i| + \sum_{i,j} \beta_{i,j} |x_i - x_j|$ and $U_2(\mathbf{x}) = \sum_i \lambda_i (y_i - x_i)^2 + \sum_{i,j} \beta_{i,j} (x_i - x_j)^2$ with parameters $\lambda, \lambda_i, \beta_{i,j} > 0$. Those models are well investigated [3, 5, 6, 7] and quite often applied to practical image restoration. Our main goal is developing efficient algorithms of their integer minimization.

Let now $S = \{1,\dots,n\} \times \{1,\dots,n\}$ denote the finite square lattice of real numbers; then $i = (i_1, i_2) \in S$ denotes points or pixels of our images and later on, nodes of appropriate networks. The *gray-scale image* is a matrix $\mathbf{x}_{\text{gray}}$ with nonnegative integer coordinates $x_i \in \{0,\dots,L-1\}$, $L > 2$. The term *binary image* is used if the coordinates of the matrix $\mathbf{x}_{\text{bin}}$ take two values $x_i \in \{0,1\}$. The *color images* $\mathbf{x}_{\text{col}}$ are specified by three matrices $(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{g}}, \mathbf{x}_{\text{b}})$.

First, we recall briefly of the Ising model of corrupted binary images [3, 5, 6, 7]. This Bayesian approach specifies an *a priori distribution*,

$$p(\mathbf{x}) = c \cdot \exp\left\{ -\sum_{i,j\in S} \beta_{i,j} (x_i - x_j)^2 \right\} \tag{3.1}$$

(with the normalizing constant c and $\beta_{i,j} \geq 0$) over all binary images $\mathbf{x} = \mathbf{x}_{\text{bin}}$. The conditional probability of the original binary image $\mathbf{x}_{\text{bin}}$ given the corrupted version $\mathbf{y} = \mathbf{y}_{\text{bin}}$ is represented in the form

$$p(\mathbf{x} \mid \mathbf{y}) = d(\mathbf{y}) \cdot \exp\left\{ -\sum_{i\in S} \lambda_i (y_i - x_i)^2 - \sum_{i,j\in S} \beta_{i,j} (x_i - x_j)^2 \right\}, \tag{3.2}$$

where the function $d(\mathbf{y})$ is independent of $\mathbf{x}$. The MAP estimate, which is defined as a mode of the conditional distribution $\mathbf{x}^* = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x} \mid \mathbf{y})$, is equal to

$$\mathbf{x}^* = \operatorname{argmin}_x \left\{ \sum_{i\in S} \lambda_i (y_i - x_i)^2 + \sum_{i,j\in S} \beta_{i,j} (x_i - x_j)^2 \right\}. \tag{3.3}$$

Remark 3.1. Often, to recover real binary images it is supposed that $\lambda_i = \lambda$ and $\beta_{i,j} > 0$ only for i, j belonging to local neighborhoods (under the last condition the a priori distribution $p(\mathbf{x})$ becomes the Markov random field). We knowingly consider more complicated model, since even in this case there is an efficient solution of the problem.

For binary images, the functions $U_1(\mathbf{x})$ and $U_2(\mathbf{x})$ coincide and the sum in (3.2), (3.3) is equal to both of them. But in the case of gray-scale images, $U_1(\mathbf{x}) \neq U_2(\mathbf{x})$. Therefore, for $\mathbf{x} = \mathbf{x}_{\text{gray}}$, $\mathbf{y} = \mathbf{y}_{\text{gray}}$, we consider two different Bayesian models with the a posteriori probabilities,

$$p_1(\mathbf{x} \mid \mathbf{y}) = d_1(\mathbf{y}) \cdot \exp\{-U_1(\mathbf{x})\}, \tag{3.4}$$

$$p_2(\mathbf{x} \mid \mathbf{y}) = d_2(\mathbf{y}) \cdot \exp\{-U_2(\mathbf{x})\}. \tag{3.5}$$

Both of these models are interesting from the theoretical and applied points of view. The first one is less investigated theoretically but it gives

better quality of restored images. Identification of the MAP for the first model is equivalent to finding $\mathbf{x}_1^* = \operatorname{argmin}_{\mathbf{x}} U_1(\mathbf{x})$, and for the second, $\mathbf{x}_2^* = \operatorname{argmin}_{\mathbf{x}} U_2(\mathbf{x})$, where *argmin* is taken over all gray-scale images.

For color images $\mathbf{x}_{\mathrm{col}} = (\mathbf{x}_{\mathrm{r}}, \mathbf{x}_{\mathrm{g}}, \mathbf{x}_{\mathrm{b}})$, we consider the Ising models (3.4), (3.5) for each component separately (with $\mathbf{x} = \mathbf{x}_{\mathrm{r}}$, $\mathbf{x} = \mathbf{x}_{\mathrm{g}}$, $\mathbf{x} = \mathbf{x}_{\mathrm{b}}$, and, respectively, $\mathbf{y} = \mathbf{y}_{\mathrm{r}}$, $\mathbf{y} = \mathbf{y}_{\mathrm{g}}$, $\mathbf{y} = \mathbf{y}_{\mathrm{b}}$). Thus, the problem is the same: to find $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ for every component of a color image.

4. Integer minimization of $U_1(\mathbf{x})$ and $U_2(\mathbf{x})$ by network flow algorithms

The network flow methods can be successfully applied to the efficient integer minimization of $U_1(\mathbf{x})$ and $U_2(\mathbf{x})$. In particular, the MNFC, which has been described in Section 2, turns out very efficient for finding integer solutions $\mathbf{x}_1^* = \operatorname{argmin}_{\mathbf{x}_{\mathrm{gray}}} U_1(\mathbf{x})$ and $\mathbf{x}_2^* = \operatorname{argmin}_{\mathbf{x}_{\mathrm{gray}}} U_2(\mathbf{x})$ in gray-scale (and color) image restoration problems. The same idea is used in both cases; it is to represent $\mathbf{x}$ with integer coordinates $x_i \in \{0, \dots, L-1\}$, $L > 2$ by $L-1$ matrices (by vectors, in the general minimization problem) having 0,1-coordinates.

4.1. Efficient minimization of $U_1(\mathbf{x})$

Let for integer $0 < l \le L-1$ the indicator functions $\mathbf{1}_{(x_i \ge l)}$ equal to 1, if $x_i \ge l$, and equal to 0 otherwise. Then any gray-scale image $\mathbf{x}$ is represented as the sum $\mathbf{x} = \sum_{l=1}^{L-1} \mathbf{x}(l)$ of binary images-layers $\mathbf{x}(l)$ with coordinates $x_i(l) = \mathbf{1}_{(x_i \ge l)}$. Those binary images-layers specify *the monotone decreasing sequence* $\mathbf{x}(1) \ge \mathbf{x}(2) \ge \cdots \ge \mathbf{x}(L-1)$, and vice versa, any sequence of binary images $\mathbf{x}(1) \ge \mathbf{x}(2) \ge \cdots \ge \mathbf{x}(L-1)$ determines the gray-scale image $\mathbf{x} = \sum_{l=1}^{L-1} \mathbf{x}(l)$. We will use the following obvious proposition.

Proposition 4.1. *For any integer $a, b \in \{0, \dots, L-1\}$, the equality*

$$|a-b| = \sum_{l=1}^{L-1} \left| \mathbf{1}_{(a \ge l)} - \mathbf{1}_{(b \ge l)} \right| \tag{4.1}$$

is valid. Therefore for any gray-scale images $\mathbf{x}$, $\mathbf{y}$, the function $U_1(\mathbf{x})$ can be written in the form

$$U_1(\mathbf{x}) = \sum_{l=1}^{L-1} u_l\big(x(l)\big), \tag{4.2}$$

where

$$u_l(x(l)) = \lambda \sum_{i\in S} |y_i(l) - x_i(l)| + \sum_{(i,j)\in S} \beta_{i,j} |x_i(l) - x_j(l)|. \tag{4.3}$$

Let

$$\check{x}(l) = \operatorname{argmin}_{x_{\text{bin}}} u_l(x_{\text{bin}}), \quad l \in \{1,\dots,L-1\} \tag{4.4}$$

denote the binary solutions that minimize the functions $u_l(x_{\text{bin}})$. Since $\mathbf{y}(1) \ge \mathbf{y}(2) \ge \cdots \ge \mathbf{y}(L-1)$, the following sentence is valid.

Proposition 4.2. *There is a monotone decreasing sequence* $\check{x}_M(1) \ge \check{x}_M(2) \ge \cdots \ge \check{x}_M(L-1)$ *of solutions of (4.4).*

The proof of Proposition 4.2 is similar to the proof of Theorem 2.3. To show the existence of solutions $\mathbf{x}_1^*$ of the form $\mathbf{x}_1^* = \mathbf{x}_{1,M}^* = \sum_{l=1}^{L-1} \check{x}_M(l)$, it is enough to use Propositions 4.1 and 4.2. Indeed, for any $\mathbf{x}$,

$$U_1(\mathbf{x}) = \sum_{l=1}^{L-1} u_l(x(l)) \ge \sum_{l=1}^{L-1} u_l(\check{x}_M(l)) = U_1(\mathbf{x}_{1,M}^*). \tag{4.5}$$

Each binary solution $\check{x}_M(l)$ can be identified by the maximum network flow algorithms, or by the MNFC for polynomial number of operations. At worst, it takes $O(Ln^3)$ operations but quite often, the use of MNFC allows reducing operations up to $O(Ln)$ operations that are held in a concurrent mode (see Section 5).

4.2. Efficient minimization of $U_2(\mathbf{x})$

The main idea of the efficient integer minimization of the function $U_2(\mathbf{x}) = \sum_i \lambda_i (y_i - x_i)^2 + \sum_{i,j} \beta_{i,j} (x_i - x_j)^2$ lies in the replacement of the variable $\mathbf{x}$ by the sum of *unordered Boolean variables* $\mathbf{x}(l)$, and then considering another polynomial $\tilde{Q}(\mathbf{x}(1),\dots,\mathbf{x}(L-1)) \ge U_2(\sum_{l=1}^{L-1} \mathbf{x}(l))$ of many Boolean variables with points of minimum $\mathbf{q}(1),\dots,\mathbf{q}(L-1)$ such that $\mathbf{x}_2^* = \sum_{l=1}^{L-1} \mathbf{q}(l)$ minimizes $U_2(\mathbf{x})$. The new polynomial Q is chosen so that it admits the efficient minimization by the network flow optimization algorithms.

In more details, we represent $\mathbf{x}$ as the sum of arbitrary Boolean variables $\mathbf{x} = \sum_{l=1}^{L-1} \mathbf{x}(l)$. In new variables, the polynomial $U_2(\mathbf{x})$ will take the form

$$U_2(\mathbf{x}) = \sum_i \lambda_i \left(y_i - \sum_{l=1}^{L-1} x_i(l) \right)^2 + \sum_{i,j} \beta_{i,j} \left(\sum_{l=1}^{L-1} (x_i(l) - x_j(l)) \right)^2. \tag{4.6}$$

Assign for simplicity $b_{i,j} = 0$ and use the equality $x_i^2(l) = x_i(l)$ for Boolean variables $x_i(l)$ to write $U_2(\mathbf{x})$ in the form

$$U_2(\mathbf{x}) = \sum_{i\in S} \lambda_i y_i^2 + P\big(\mathbf{x}(1),\dots,\mathbf{x}(L-1)\big), \tag{4.7}$$

where the polynomial of many Boolean variables

$$\begin{aligned} P(\mathbf{x}) &= P\big(\mathbf{x}(1),\dots,\mathbf{x}(L-1)\big) \\ &= \sum_{i\in S}\Big[\lambda_i - 2\lambda_i y_i - (L-2)\sum_{j\in S}\big(b_{i,j}+b_{j,i}\big)\Big]\sum_{l=1}^{L-1} x_i(l) \\ &\quad + 2\sum_{i\in S}\Big[\lambda_i + \sum_{j\in S}\big(b_{i,j}+b_{j,i}\big)\Big]\sum_{1\le l<m\le L-1} x_i(l)x_i(m) \\ &\quad + \sum_{i,j\in S} b_{i,j}\sum_{l=1}^{L-1}\big(x_i(l)-x_j(l)\big)^2 \\ &\quad + \sum_{i,j\in S} b_{i,j}\sum_{1\le l<m\le L-1}\Big[\big(x_i(m)-x_j(l)\big)^2 + \big(x_i(l)-x_j(m)\big)^2\Big]. \end{aligned} \tag{4.8}$$

Introduce another polynomial of many Boolean variables,

$$\begin{aligned} Q\big(\mathbf{x}(1),\dots,\mathbf{x}(L-1)\big) &= \sum_{i\in S}\Big[\lambda_i - 2\lambda_i y_i - (L-2)\sum_{j\in S}\big(b_{i,j}+b_{j,i}\big)\Big]\sum_{l=1}^{L-1} x_i(l) \\ &\quad + 2\sum_{i\in S}\Big[\lambda_i + \sum_{j\in S}\big(b_{i,j}+b_{j,i}\big)\Big]\sum_{1\le l<m\le L-1} x_i(m) \\ &\quad + \sum_{i,j\in S} b_{i,j}\sum_{l=1}^{L-1}\big(x_i(l)-x_j(l)\big)^2 \\ &\quad + \sum_{i,j\in S} b_{i,j}\sum_{1\le l<m\le L-1}\Big[\big(x_i(m)-x_j(l)\big)^2 + \big(x_i(l)-x_j(m)\big)^2\Big] \end{aligned} \tag{4.9}$$

such that $Q(\mathbf{x}(1),\dots,\mathbf{x}(L-1)) \ge P(\mathbf{x})$ and which differs from $P(\mathbf{x})$ by the term $\sum_{1\le l<m\le L-1} x_i(m)$ in the second summand.

Denote by

$$\big(\mathbf{q}^*(1),\mathbf{q}^*(2),\dots,\mathbf{q}^*(L-1)\big) = \operatorname{argmin}_{x(1),x(2),\dots,x(L-1)} Q\big(\mathbf{x}(1),\dots,\mathbf{x}(L-1)\big) \tag{4.10}$$

any collection of Boolean vectors that minimize $Q(\mathbf{x}(1),\dots,\mathbf{x}(L-1))$. We

will prove that without fail $\mathbf{q}^*(1) \geq \mathbf{q}^*(2) \geq \cdots \geq \mathbf{q}^*(L-1)$. This feature will allow expressing solutions of the initial problem as $\mathbf{x}_2^* = \sum_{l=1}^{L-1} \mathbf{q}^*(l)$.

THEOREM 4.3. *Any collection* $(\mathbf{q}^*(1), \mathbf{q}^*(2), \ldots, \mathbf{q}^*(L-1))$ *that minimizes* Q *forms the decreasing sequence. It identifies the solution of the original problem by the formula* $\mathbf{x}_2^* = \sum_{l=1}^{L-1} \mathbf{q}^*(l)$, *and vice versa, each solution* $\mathbf{x}_2^*$ *specifies the solution* $\mathbf{q}^*(l) = \mathbf{x}^*(l)$.

Proof. Represent Q as

$$Q\big(\mathbf{x}(1), \ldots, \mathbf{x}(L-1)\big) = P(\mathbf{x}) + \sum_{i \in S} \left[\lambda_i + \sum_{j \in S} \big(b_{i,j} + b_{j,i}\big)\right] \sum_{1 \leq l < m \leq L-1} \big(1 - x_i(l)\big) x_i(m), \tag{4.11}$$

where $\mathbf{x} = \sum_{l=1}^{L-1} \mathbf{x}(l)$. Then suppose that there exists an unordered collection $(\mathbf{q}^*(1), \mathbf{q}^*(2), \ldots, \mathbf{q}^*(L-1))$ minimizing Q. For this collection,

$$\sum_{1 \leq l < m \leq L-1} \big(1 - q_i^*(l)\big) q_i^*(m) > 0, \tag{4.12}$$

and, therefore, for $\bar{\mathbf{x}} = \sum_{l=1}^{L-1} \mathbf{q}^*(l)$ the inequality $Q(\mathbf{q}^*(1), \ldots, \mathbf{q}^*(L-1)) > P(\bar{\mathbf{x}})$ holds (remember that all $\lambda_i, \beta_{i,j} > 0$). But, evidently, for ordered Boolean matrices-layers, $\bar{\mathbf{x}}(1) \geq \cdots \geq \bar{\mathbf{x}}(L-1)$, with coordinates $\bar{x}_i(l) = \mathbf{1}_{(\bar{x}_i \geq l)}$ we have

$$Q\big(\bar{\mathbf{x}}(1), \ldots, \bar{\mathbf{x}}(L-1)\big) = P(\bar{\mathbf{x}}). \tag{4.13}$$

Hence, unordered sequence $(\mathbf{q}^*(1), \mathbf{q}^*(2), \ldots, \mathbf{q}^*(L-1))$ cannot minimize Q. Since for any ordered collection $(\mathbf{x}(1) \geq \cdots \geq \mathbf{x}(L-1))$, the equality $Q(\mathbf{x}(1), \ldots, \mathbf{x}(L-1)) = P(\mathbf{x})$ is valid, actually, $\mathbf{x}_2^* = \sum_{l=1}^{L-1} \mathbf{q}^*(l)$, and vice versa, each solution $\mathbf{x}_2^*$ specifies the solution $\mathbf{q}^*(l) = \mathbf{x}^*(l)$. □

It follows from Theorem 4.3 that P and Q have the same ordered set of Boolean matrices $(\mathbf{x}^*(1), \mathbf{x}^*(2), \ldots, \mathbf{x}^*(L-1))$ that minimize these polynomials. But the problem of minimization of Q is known in discrete optimization [1, 2, 10]. It is equivalent to identification of the minimum network flow cut. To describe the corresponding network, rearrange the

polynomial to the form

$$\begin{aligned} &Q(\mathbf{x}(1),\dots,\mathbf{x}(L-1)) \\ &\quad= \sum_{i\in S}\sum_{l=1}^{L-1}\Big[(2l-1)\lambda_i - 2\lambda_i y_i + (2l-L)\sum_{j\in S}(b_{i,j}+b_{j,i})\Big]x_i(l) \\ &\quad+ \sum_{i,j\in S} b_{i,j}\sum_{l=1}^{L-1}\big(x_i(l)-x_j(l)\big)^2 \\ &\quad+ \sum_{i,j\in S} b_{i,j}\sum_{1\le l<m\le L-1}\Big[\big(x_i(m)-x_j(l)\big)^2+\big(x_i(l)-x_j(m)\big)^2\Big]. \end{aligned} \tag{4.14}$$

Then define for brevity

$$d_{i,l} = \Big[(2l-1)\lambda_i - 2\lambda_i y_i + (2l-L)\sum_{j\in S}(b_{i,j}+b_{j,i})\Big] \tag{4.15}$$

and introduce the network $N(s,t,V,A)$ with the source s, the sink t, and the usual nodes that are labeled by $i_l \in V$, where the multi-index $i \in S$ and l shows a layer number (there are $|V| = (L-1)|S|$ usual nodes in the $N(s,t,V,A)$). For instance, i_3 means the node i at the third layer. The set of arcs with corresponding capacities is specified as

$$A = \begin{cases} (s,i_l) & -d_{i,l} & \text{if } d_{i,l}<0, \\ (i_l,t) & d_{i,l} & \text{if } d_{i,l}>0, \\ (i_l,j_l) & b_{i,j}+b_{j,i} & \text{if } i\ne j, \\ (i_l,j_m) & b_{i,j}+b_{j,i} & \text{if } i\ne j,\ l\ne m, \end{cases} \tag{4.16}$$

for $l \in \{1,\dots,L-1\}$. In general, it takes $O(Ln^3)$ operations to find a solution $\mathbf{x}_2^*$, but the use of the MNFC or other decomposition methods can reduce the operation number.

5. Applications

It was mentioned in the introduction that the MNFC turned out to be very efficient in recovering of images (both binary and gray-scale) corrupted by random noise. Figure 5.1 shows the binary 512×512 image corrupted by rather strong Bernoullian noise with the parameter $p = 0.3$. The Ising model with the energy function $U_1(\mathbf{x}_{\text{bin}}) = U_2(\mathbf{x}_{\text{bin}})$ was used for image restoration.

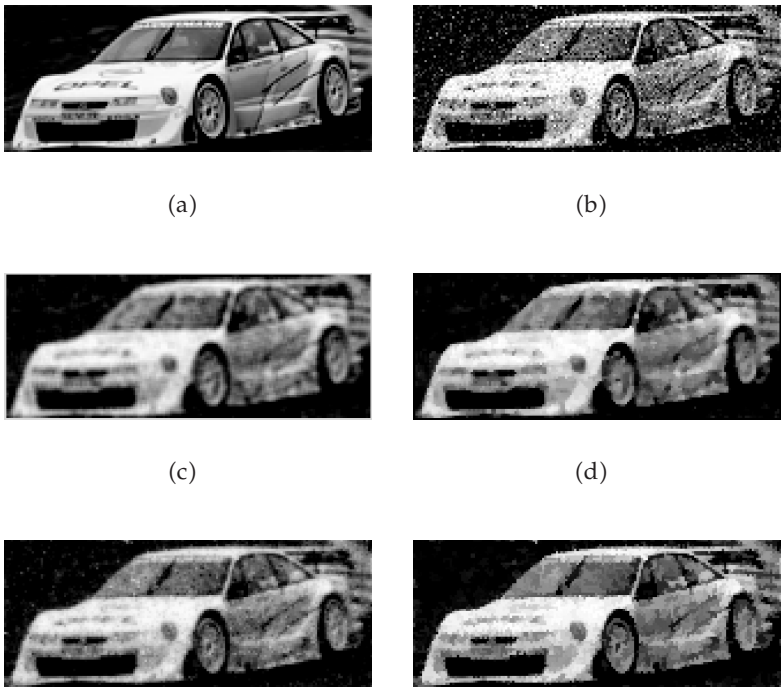
The result of implementing the MNFC at the first level is drawn in Figure 5.2. The pixels that were not estimated at the first level are depicted in gray color. They form small sparse sets at the boundaries of

FIGURE 5.1

FIGURE 5.2

the partition's squares. The identification of $\mathbf{x}^*_{\text{bin}}$ of the corrupted image takes fractions of a second while the known maximum network flow algorithms was not able to evaluate the estimate at all.

Now, consider the application of the proposed method of the integer minimization to the restoration of corrupted gray-scale images. The Ising model with the energy function $U_1(\mathbf{x})$ was used to construct the MAP estimates. This model was preferred since, first, the function $U_1(\mathbf{x})$ admits more efficient minimization, compared to $U_2(\mathbf{x})$ and, second, because of the better visual quality of the estimate. Beside the MAP estimate $\mathbf{x}^*_1$, the moving average, the moving median estimates, and the gradient estimate of the $U_2(\mathbf{x})$ were computed to compare estimators.



(a) (b)

(c) (d)

(e) (f)

FIGURE 5.3

The original gray-scale image in Figure 5.3a was corrupted by the exponential noise in Figure 5.3b. The results of the 3×3 moving average and 3×3 moving median filtration are depicted in Figures 5.3c and 5.3d, respectively. The continuous gradient estimate for $\mathbf{x}_2^*$ is placed in Figure 5.3e. Figure 5.3f is the exact $\mathbf{x}_1^*$ estimate of the image in Figure 5.3b. Recall that it is determined separately for every binary layer of a gray-scale image. Therefore, the use of MNFC allows computation of $\mathbf{x}_1^*$ in a highly concurrent mode.

References

[1] A. A. Ageev, *Complexity of problems of minimization of polynomials in Boolean variables*, Upravlyaemye Sistemy (1983), no. 23, 3–11 (Russian).

[2] ———, *Minimization of quadratic polynomials of Boolean variables*, Upravlyaemye Sistemy (1984), no. 25, 3–16 (Russian).

[3] P. A. Ferrari, M. D. Gubitoso, and E. J. Neves, *Reconstruction of gray-scale images*, Methodol. Comput. Appl. Probab. **3** (2001), no. 3, 255–270.

[4] L. R. Ford Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, New Jersey, 1962.

[5] D. Geman, *Random fields and inverse problems in imaging*, École d'été de Probabilités de Saint-Flour XVIII—1988, Lecture Notes in Math., vol. 1427, Springer, Berlin, 1990, pp. 113–193.

[6] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intell. **6** (1984), 721–741.

[7] B. Gidas, *Metropolis-type Monte Carlo simulation algorithms and simulated annealing*, Topics in Contemporary Probability and Its Applications, Probab. Stochastics Ser., CRC, Florida, 1995, pp. 159–232.

[8] D. M. Greig, B. T. Porteous, and A. H. Seheult, *Exact maximum a posteriori estimation for binary images*, J. Roy. Statist. Soc. Ser. B **51** (1989), no. 2, 271–279.

[9] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, New Jersey, 1982.

[10] J. C. Picard and H. D. Ratliff, *Minimum cuts and related problems*, Networks **5** (1975), no. 4, 357–370.

Boris A. Zalesky: Institute of Engineering Cybernetics NAN, Surganov Street 6, Minsk, 220012, Belarus

E-mail address: zalesky@mpen.bas-net.by