UNIVERSITY OF OSLO
Department of Informatics

Large-scale integer programs in image analysis

G. Dahl, G. Storvik
and A. Fadnes

Report 262,
ISBN 82-7368-193-9

May 1998

Large-scale integer programs in image analysis

Geir Dahl* Geir Storvik† Alice Fadnes‡

May 1998

Abstract

An important problem in image analysis is to segment an image into regions with different class-labels. This is releveant in applications in medicine and cartography. In a proper statistical framework this problem may be viewed as a discrete optimization problem. We present two integer linear programming formulations of the problem and study some properties of these models and associated polytopes. Different algorithms for solving these problems are suggested and compared on some realistic data. In particular, a Lagrangian algorithm is shown to have a very promising performance. The algorithm is based on the technique of cost splitting and uses the fact that certain relaxed problems may be solved as shortest path problems.

Keywords: Integer programming, image analysis, Lagrangian relaxation.

1 Introduction

Digitized images are nowadays routinely recorded in a huge number of applications. Examples are in remote sensing, medical image analysis and industrial inspection. In most cases the observed image consists of a matrix of pixels (picture elements), where a (possible vector-valued) observation is given at each pixel. One aim in the analysis of such images is *segmentation* which is a method for labeling each pixel by a class-label describing the content in the image (e.g. tissue types in medical images).

In most situations, the observed image is quite noisy, making segmentation based on the observed image difficult. In such cases external information on how the class-labels typically appear needs to be incorporated. One popular way for doing so is to assume the class-labels are distributed according to a stochastic model [2]. A class of stochastic models frequently used in image analysis is the Markov Random Field (MRF). Such models mainly incorporate smoothness in the images and are popular because of their flexibility.

*University of Oslo, Dept. of Informatics, P.O.Box 1080, Blindern, 0316 Oslo, Norway. (Email:geird@ifi.uio.no)

†University of Oslo, Department of Mathematics, P.O.Box 1053, Blindern, 0316 Oslo, Norway.

‡University of Oslo, Dept. of Informatics, P.O.Box 1080, Blindern, 0316 Oslo, Norway.

When using stochastic models for the class-labels, the aim is to find the most probable configuration of labels based on the observed image. Since the number of pixels is large (typically 256×256 or larger) and the number of classes can be from 2 and up to 40-50, this results in a huge combinatorial problem. In most situations this problem is solved through stochastic simulation and simulated annealing [5] or by using some sort of heuristic [1].

In this paper we will consider the use of integer linear programming (ILP) methods for solving the image segmentation problem. Such an approach has been considered in the special case of two classes [6]. We will however consider the general case for which little, to our knowledge, has been done. The approach is to transform the image segmentation problem into an integer linear programming problem for which different algorithms will be constructed. In [9], the main approach was presented and applied on some real images. In this paper we will consider the approach in a more theoretical setting, describing different ILP models, discussing their theoretical properties and constructing different algorithms for these problems. By applying the different algorithms on a large number of test cases, evaluation on their properties will be performed.

The paper is organized as follows. The image segmentation problem is described in mathematical terms in section 2. This also includes the statistical model which reflects how errors are introduced in the observed image. Section 3 presents two integer linear programming models for the problem and discusses some basic relations between these models. Further properties of the models and associated polytopes are presented in section 4. For certain special cases (e.g. two classes) the image segmentation problem may be solved as basic combinatorial optimization problem as described in section 5. Section 6 presents different methods for finding optimal or near-optimal solutions of the problem while the last section presents some numerical results and experiences with these methods on some realistic problems.

2 The image segmentation problem

For our purposes a (two-dimensional) image consists of a set $V = \{1, \ldots, n\}$ of pixels where each pixel belongs to a set K of classes $K = \{1, ..., \kappa\}$. The pixels are organized as a matrix where entries correspond to pixels. Pixels are denoted by $u, v, \ldots$. We say that two pixels u and v are *neighbors*, or *adjacent*, whenever the two pixels lie in the same row but in adjacent columns or vise versa. Let E denote the set of (unordered) neighbor pairs of pixels. We may view V and E as the node and edge set of a graph $G = (V, E)$ and we call this graph the *image graph*. Thus, nodes correspond to pixels and edges to adjacent pixels. Note that our image graph G is a "grid graph", but other graphs (e.g. with additional edges for diagonals) may also be of interest in certain applications. Let $n := |V|$ denote the number of pixels (nodes) and $m := |E|$ the number of edges (neighbor pairs). An edge between nodes u and v is denoted by $[u, v]$.

Formally, an image may be viewed as a function $\mathbf{c}$ which associates a class $c_v \in K$ to each pixel $v \in V$. It will be convenient to view $\mathbf{c}$ as an n-dimensional

vector $\mathbf{c} = (c_1, \ldots, c_n)$ where c_i is the class of the ith pixel.

The image $\mathbf{c} = (c_1, \ldots, c_n)$ is not directly observed, but is observed through a degraded image $\mathbf{z} = (z_1, \ldots, z_n)$. The connection between $\mathbf{z}$ and $\mathbf{c}$ is given through a statistical model which contains the conditional (probability) distribution $f(\mathbf{z}|\mathbf{c})$ of $\mathbf{z}$ given $\mathbf{c}$. We assume conditional independence between pixels meaning that

$$f(\mathbf{z}|\mathbf{c}) = \prod_{i \in V} f(z_i|c_i). \tag{1}$$

Often some prior information about the true image $\mathbf{c}$ is available. A simple, and popular, class of models is the Markov random field (MRF), which builds in smoothness properties of the image. In its simplest form it is given by

$$\pi(\mathbf{c}) = \frac{1}{s} \exp\{\sum_{v \in V} \alpha_{c_v} + \beta \sum_{[u,v] \in E} I(c_u = c_v)\} \tag{2}$$

where α_k defines the prior probabilities for the different classes k, $\beta > 0$ is a parameter giving the degree of smoothness, s is a normalization constant making $\pi(\cdot)$ a proper distribution (i.e. total probability is 1) and the indicator function $I(c_u = c_v)$ equals 1 if $c_u = c_v$ and 0 otherwise. A generalization would be to let β be dependent on the classes c_u and c_v, or even on u and v. Both these extensions are directly applicable for the methods to follow. However, to keep the notation simple, we assume that (2) holds. The *posterior* distribution for $\mathbf{c}$ given $\mathbf{z}$ may be found through Bayes formulae to be

$$\pi(\mathbf{c}|\mathbf{z}) = \phi(\mathbf{z})\pi(\mathbf{c})f(\mathbf{z}|\mathbf{c}) \tag{3}$$

where ϕ is a suitable normalization function. Consider now $\mathbf{z}$ as fixed; it is the observed image. Thus $\phi(\mathbf{z})$ is a constant. The Bayesian paradigm is to base all inference on this posterior distribution. In particular, one estimate of $\mathbf{c}$ is the maximum a posteriori (MAP) solution $\hat{\mathbf{c}}$ which maximizes $\pi(\mathbf{c}|\mathbf{z})$, or equivalently, is an optimal solution of

$$\max\{\pi(\mathbf{c})f(\mathbf{z}|\mathbf{c}) : \mathbf{c} \in K^n\}. \tag{4}$$

In this paper we discuss models and methods for finding such a MAP solution based on the optimization problem (4). This is a discrete optimization problem as each variable c_v is restricted to lie in the finite set K of classes. For the application we have in mind the number κ of classes is low (e.g., due to unordered classes). We call (4) the *image segmentation problem*, or (IMS) for short.

3 Integer linear programming models for (IMS)

We shall formulate the (IMS) problem (4) as an integer linear programming problem. Actually, two possible formulations will be presented and some basic relations between these are discussed.

First, we observe that it is more convenient to maximize the logarithm of the posterior distribution (3), or equivalently, after removing a constant (depending only on $\mathbf{z}$), the following function

$$\begin{aligned} U(c) &= \sum_{v\in V} \alpha_{c_v} + \beta \sum_{[u,v]\in E} I(c_u = c_v) + \sum_{v\in V} \log f(z_v|c_v) \\ &= \sum_{v\in V}\sum_{k\in K} (\log f(z_v|k) + \alpha_k) I(c_v = k) + \beta \sum_{[u,v]\in E} I(c_u = c_v). \end{aligned}$$

In the image analysis literature (minus) U is usually referred to as the energy function. Define $d_{v,k} = \log f(z_v|k) + \alpha_k$ for $v \in V$ and $k \in K$. Consider the following integer linear programming model which we denote by (ILP1):

$$\begin{array}{lll} \max & \sum_{v\in V}\sum_{k\in K} d_{v,k}x_{v,k} + \beta\sum_{e\in E}\sum_{k\in K} y_{e,k} & \\ \text{s.t.} & & \\ \text{(i)} & y_{e,k} \le x_{v,k} & \text{for } k \in K,\ e \in E \text{ and } v \in e; \\ \text{(ii)} & \sum_{k\in K} x_{v,k} = 1 & \text{for } v \in V; \\ \text{(iii)} & 0 \le x_{v,k}, y_{e,k} \le 1 & \text{for } v \in V,\ e \in E,\ k \in K; \\ \text{(iv)} & x_{v,k}, y_{e,k} \text{ are integral} & \text{for } v \in V,\ k \in K. \end{array} \tag{5}$$

We write $v \in e$ to mean that v is an endnode of the edge e. In this model we have *class variables* $x_{v,k}$ and *neighbor variables* $y_{e,k}$. All these are $(0,1)$-variables and $x_{v,k}$ equals 1 if pixel v is given class k. If two pixels u and v are adjacent, i.e., $e = [u,v] \in E$, the neighbor variable $y_{e,k}$ may be set to 1 if both u and v are given the same class k, and $y_{e,k} = 0$ otherwise. The objective function to be maximized is the sum of two terms: the *class term* and *the neighbor term*. The class term involves the class variables and reflects how well the estimated image fits the observed image. The neighbor term is β times the number of neighbors that are given the same class. Thus one seeks a balance between fitness to the observed image and shoothness. The parameter β reflects a weighting of these two conflicting goals. Consider $k \in K$ and $e = [u,v] \in E$. The corresponding constraint (5)(i) says that in order to have the neighbor variable $y_{e,k}$ equal to 1 both the class variables $x_{u,k}$ and $x_{v,k}$ must be 1, i.e., both pixels u and v are given class k. The constraints (5)(ii) just says that each pixel must be given precisely one class. Hereafter we assume that all the class variables $x_{v,k}$ resp. the neighbor variables are organized into a vector $\mathbf{x}$ (of dimension $n\kappa$) resp. $\mathbf{y}$ (of dimension $m\kappa$).

We remark that we may assume without loss of generality that all the numbers $d_{v,k}$ in (5) are nonnegative. This follows from the observation that if we increase each $d_{v,k}$ by a number p then the optimal value is increased by np due to (5)(ii).

A simple observation concerning the form of the optimal solutions in the problem (ILP1) can be made. Assume that $(\mathbf{x}, \mathbf{y})$ is an optimal solution of this problem. Then the variable $\mathbf{y}$ may be expressed in terms of $\mathbf{x}$ as follows:

$$y_{e,k} = \min\{x_{u,k}, x_{v,k}\} \tag{6}$$

for each $e = [u, v] \in E$ and $k \in K$.

In order to count the number of variables in this model assume that the image (matrix) has n_1 rows and n_2 columns. From the adjacency structure we see that $n = |V| = n_1 n_2$ and $m = |E| = 2n_1 n_2 - n_1 - n_2$. Therefore in (ILP1) there are $N_1 := n\kappa + m\kappa = (3n_1 n_2 - n_1 - n_2)\kappa$ variables and $M_1 := (4\kappa + 1)n_1 n_2 - 2(n_1 + n_2)\kappa$ constraints (apart from the simple bounds). We should point out that for some interesting application we might have e.g. $n_1 = n_2 = 256$ and $\kappa = 10$ which gives $N_1 = 1960960$ and $M_1 = 2676736$. Thus we are clearly confronted with large-scale integer programming problems; even the corresponding linear programs are large! Based on this observation it is natural to seek other models with possibly fewer variables and, when it comes to algorithms, study decomposition-based methods. We shall consider both issues and start with a second model.

Consider the integer linear programming model (ILP2)

$$\begin{array}{lll} \max & \sum_{k\in K}\sum_{v\in V} d_{v,k}x_{v,k} + \beta\sum_{e\in E} y'_e & \\ \text{s.t.} & & \\ (\text{i}_1) & x_{u,k} - x_{v,k} + y'_e \le 1 & \text{for } k \in K,\ e = [u,v] \in E; \\ (\text{i}_2) & -x_{u,k} + x_{v,k} + y'_e \le 1 & \text{for } k \in K,\ e = [u,v] \in E; \\ (\text{ii}) & \sum_{k\in K} x_{v,k} = 1 & \text{for } v \in V; \\ (\text{iii}) & 0 \le x_{v,k}, y'_e \le 1 & \text{for } v \in V,\ k \in K; \\ (\text{iv}) & x_{v,k}, y'_e \text{ are integral} & \text{for } v \in V,\ k \in K. \end{array} \tag{7}$$

Comparing this model to (ILP1) we see that, for each $e = [u, v] \in E$, the neighbor variables $y_{e,k}$ for $k \in K$ are replaced by the single neighbor variable y'_e. The new constraints (7)(i) says that $|x_{u,k} - x_{v,k}| + y'_e \le 1$: in order to set the neighbor variable y'_e to 1 (which gives contribution β in the objective) we must have $x_{u,k} = x_{v,k}$ for all k, i.e., the two pixels are given the same class. Let $\mathbf{y}'$ be the vector of the variables y'_e for $e \in E$. Similar to what we saw for model (ILP1) the variable $\mathbf{y}'$ may be expressed in terms of $\mathbf{x}$ for an optimal solution $(\mathbf{x}, \mathbf{y}')$ in (ILP2):

$$y'_e = 1 - \max_{k\in K}|x_{u,k} - x_{v,k}| \tag{8}$$

for each $e = [u, v] \in E$.

The number of variables and constraints in (ILP2) are given by $N_2 = n\kappa + m = (\kappa + 2)n_1 n_2 - n_1 - n_2 = N_1 - (\kappa - 1)m$ and $M_2 = M_1$. This means that we have reduced the number of variables by $(\kappa - 1)m$ while the number of constraints is unaltered. In our example with $n_1 = n_2 = 256$ and $\kappa = 10$ we get $N_2 = 785920$ which is less than half the number of variables in (ILP1). A further comparison of the two models is given in the next section.

4 Properties of the models

In this section we discuss some theoretical properties of the two models introduced in section 3. In particular we give some conditions under which the optimal value of the integer program coincides with the optimal value of the

corresponding LP relaxation. In such situations the (IMS) problem reduces to solving a linear programming problem.

4.1 Comparing the LP relaxations

The following result relates different optimal values and solutions. We let $v(R)$ denote the optimal value of an optimization problem (R). Moreover, the LP relaxations of (ILP1) and (ILP2) are denoted by (LP1) and (LP2), respectively.

Theorem 4.1 *The following relations hold*

$$v(ILP1) = v(ILP2) \leq v(LP1) \leq v(LP2).$$

Moreover, $(\mathbf{x}, \mathbf{y})$ *is an optimal solution of (ILP1) if and only if* $(\mathbf{x}, \mathbf{y}')$ *is an optimal solution of (ILP2) where* $\mathbf{y}$ *and* $\mathbf{y}'$ *are determined by* $\mathbf{x}$ *according to (6) and (8).*

Proof. We only prove that $v(LP1) \leq v(LP2)$ as the remaining statements are easy to verify. Let X denote the set of vectors $\mathbf{x}$ that satisfy (5) (ii) and (iii). Based on the relation (6) we may write

$$v(LP1) = \max\{\sum_{v,k} d_{v,k}x_{v,k} + \beta \sum_{e=[u,v],k} \min\{x_{u,k}, x_{v,k}\} : \mathbf{x} \in X\}$$

and similarly from (8) we get

$$v(LP2) = \max\{\sum_{v,k} d_{v,k}x_{v,k} + \beta \sum_{e=[u,v]} (1 - \max_k |x_{u,k} - x_{v,k}|) : \mathbf{x} \in X\}.$$

Thus, we only need to prove that for every $\mathbf{x} \in X$ and $e = [u, v]$ the inequality $\sum_k \min\{x_{u,k}, x_{v,k}\} \leq 1 - \max_k |x_{u,k} - x_{v,k}|$ holds. To this end, let $\mathbf{x} \in X$ and $e = [u, v]$ and define $K' = \{k \in K : x_{u,k} \leq x_{v,k}\}$ and $K'' = K \setminus K'$. Then $\sum_{k\in K} \min\{x_{u,k}, x_{v,k}\} = \sum_{k\in K'} x_{u,k} + \sum_{k\in K''} x_{v,k} = \sum_{k\in K'} x_{u,k} + 1 - \sum_{k\in K'} x_{v,k} = 1 - \sum_{k\in K'}(x_{v,k} - x_{u,k}) \leq 1 - \max_{k\in K'}(x_{v,k} - x_{u,k})$ where the inequality is due to the nonnegativity of the elements in the summation. Similarly we derive $\sum_{k\in K} \min\{x_{u,k}, x_{v,k}\} \leq 1 - \max_{k\in K''}(x_{u,k} - x_{v,k})$. But then we have two upper bounds for $\sum_k \min\{x_{u,k}, x_{v,k}\}$ and by taking the minimum of these we arrive at the desired inequality. □

4.2 Model 1

Let P_1 be the solution set of the constraints (5)(i)–(iii), i.e. P_1 is the feasible set of the LP relaxation of model (ILP1). We may also write $P_1(G)$ to indicate the dependence of P_1 of the image graph G. Then P_1 is a bounded polyhedron, or polytope, in $\mathbb{R}^{N_1}$. Observe that all the constraints $\mathbf{x} \geq \mathbf{0}$, $\mathbf{x} \leq \mathbf{1}$ and $\mathbf{y} \leq \mathbf{1}$ are redundant and may be omitted from the description of P_1.

We recall that a matrix is totally unimodular (TU) if each subdeterminant is -1, 0 or 1. A general result (see [7]) is that a matrix $\mathbf{M}$ is TU if and only if

the matrix $\mathbf{M}'$ obtained from $\mathbf{M}$ by a (simplex) pivot operation is TU. Assume that $\mathbf{M} = (m_{i,j})$ is a $(-1, 0, 1)$-matrix and that the element in position (i, j) is nonzero. A rowwise pivot in position (i, j) is to (a) multiply the ith row by -1 if $m_{i,j} = -1$, and the (b) add $-m_{k,j}$ times the (new) ith row to the kth row for $k \neq i$. As a result one obtains a unit vector in column j with the one in row i. Columnwise pivots are defined similarly. Another useful fact is that the TU-property of a matrix is insensitive to row and column permutations on the matrix. Recall also that if $\mathbf{M} \in \mathbb{R}^{m,n}$ is TU, then for each *integral* $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^m$ with $\mathbf{b}_1 \leq \mathbf{b}_2$ the polytope $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{b}_1 \leq \mathbf{M}\mathbf{x} \leq \mathbf{b}_2, \ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$ has vertices with all components being 0 or 1, see [7].

We next present some important results concerning the LP relaxations of (ILP1). Here we no longer restrict the attention to the specific image graph G introduced in the beginning of section 2 (the grid graph). The first result deals with arbitrary image graph G but only two classes and the second result treats the situation where G is a forest (a graph without cycles).

Theorem 4.2 *Assume that one of the two conditions (i) and (ii) holds where*

(i) G is arbitrary and $|K| = 2$;

(ii) G is a tree and K is arbitrary.

Then the coefficient matrix $\mathbf{M}$ given by the constraints (5)(i) and (ii) is totally unimodular. In particular, the polytope $P_1(G)$ is integral.

Proof. We show that $\mathbf{M}$ is TU, and then $P_1(G)$ is integral due to the general result mentioned above. Assume first that condition (i) holds and let $K = \{1, 2\}$. The columns of $\mathbf{M}$ may be divided into two classes. In class 1 we have all the variables $x_{v,1}$ and $y_{e,1}$ and in class 2 we have the remaining variables ($x_{v,2}$ and $y_{e,2}$). Then $\mathbf{M}$ has the following four properties: (a) $\mathbf{M}$ is a (-1,0,1)-matrix, (b) each row in $\mathbf{M}$ has two nonzeros, (c) each row in $\mathbf{M}$ corresponding to a constraint $y_{e,k} \leq x_{v,k}$, or rather $y_{e,k} - x_{v,k} \leq 0$, has its two nonzeros (1 and -1) in the same class, and, finally, (d) each row in $\mathbf{M}$ corresponding to a constraint $x_{v,1} + x_{v,2} = 1$ has its two nonzeros (both 1) in different classes. These properties imply that $\mathbf{M}$ is TU (for multiplying each column in class 2 by -1 produces a network matrix, see e.g. [8]).

Next, we assume that condition (ii) holds. Suppose first that G is a tree. Since G is a tree it is well known that one may order its nodes $v_1, \ldots, v_n$ and edges $e_1, \ldots, e_{n-1}$ such that for $i = 1, \ldots, n-1$ the edge e_i has endnodes v_i and $v_{k(i)}$ for some $k(i) > i$. This is due to the fact that each tree has a leaf, i.e., a node of degree one.

Let $\mathbf{M}_1$ be the $(2n-2) \times (2n-1)$ block matrix where each element is a $\kappa \times \kappa$ matrix. The columns of $\mathbf{M}_1$ correspond to $v_1, e_1, v_2, e_2, \ldots, v_{n-1}, e_{n-1}, v_n$ in that order. For $i = 1, \ldots, n-1$ the $(2i-1)$th row of $\mathbf{M}_1$ has $\mathbf{I}$ (the identity matrix of order κ) in column $2i-1$ and $-\mathbf{I}$ in column $2i$. Moreover, for $i = 1, \ldots, n-1$ the $(2i)$th row of $\mathbf{M}_1$ has $-\mathbf{I}$ in column $2i$ and $\mathbf{I}$ in column $2k(i)-1$. All other elements of $\mathbf{M}_1$ are equal to $\mathbf{0}$, the $\kappa \times \kappa$ matrix with all zeros. Next, let $\mathbf{M}_2$

be the $n \times (2n-1)$ block matrix which, for $i = 1, \ldots, n$, has the element $\mathbf{1}$, the $(1 \times \kappa)$-dimensional matrix with all ones, in row i and column $2i-1$. All other elements in $\mathbf{M}_2$ are zero. Finally, we define the matrix $\mathbf{M}$ by

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \end{bmatrix}.$$

For a small example, see fig. 1. Then $\mathbf{M}$ is the coefficient matrix associated with the LP problem (5) constraints (i) and (ii) with suitable ordering of variables and constraints ($\mathbf{M}_1$ and $\mathbf{M}_2$ corresponds to constraints (i) and (ii), respectively).

We apply the following procedure to $\mathbf{M}$: for $j = 1, \ldots, n-1$ add column j to column $j+1$ if j is odd and to column $2k(j)-1$ if j is even. Let $\mathbf{M}'$ be the resulting matrix, see again fig. 1 for our example. On the element level these block operations on $\mathbf{M}$ are columnwise pivots on the diagonal elements (except that we allow the pivot elements to be -1). Thus, as remarked above, we only need to prove that the matrix $\mathbf{M}'$ is TU. We have that $\mathbf{M}' = \begin{bmatrix} \mathbf{M}'_1 \\ \mathbf{M}'_2 \end{bmatrix}$ where $\mathbf{M}'_1$ consists of (i) a diagonal matrix with diagonal elements being $\mathbf{I}$ and $-\mathbf{I}$ (alternating) and (ii) a final column of zeros. In the matrix $\mathbf{M}'_2$ each element is either $\mathbf{0}$ or $\mathbf{1}$ and one can easily show (by induction) that the ith row of $\mathbf{M}'_2$ (which corresponds to the node v_i) has an element $\mathbf{1}$ precisely for those nodes and edges lying in the unique $v_i v_n$-path in G.

Consider the $((2n-2)\kappa+n) \times ((2n-1)\kappa)$-dimensional matrix obtained from $\mathbf{M}'$ by replacing each element (block) by the corresponding matrix; we also denote this matrix by $\mathbf{M}'$. We apply to $\mathbf{M}'$ a number of rowwise pivots; this is done in positions (i,i) for each row i in the $\mathbf{M}_1$ part (where the pivot elements are 1 or -1). Again, we know that the resulting matrix $\mathbf{M}''$ is TU iff $\mathbf{M}'$ is. Moreover, $\mathbf{M}''$ has a simple structure: it is the direct product of a diagonal matrix $\mathbf{E}_1$ having 1 and -1 on the diagonal and a matrix $\mathbf{E}_2$ with all ones. But each subdeterminant of $\mathbf{E}_2$ is zero (as all rows are equal), so both $\mathbf{E}_1$ and $\mathbf{E}_2$ are trivially TU. From this it follows that $\mathbf{M}''$ and therefore also $\mathbf{M}$ is TU as desired.

Thus, when G is a tree the matrix $\mathbf{M}$ is TU. This implies that the similar result holds for any forest as the the matrix $\mathbf{M}$ then may be written as a direct product of TU matrices (associated with each of the trees in the forest).

□

The following result is a consequence of the theorem.

Corollary 4.3 *Assume that one of the conditions (i) and (ii) in Theorem 4.2 holds. Then the optimal values of (ILP1) and its LP relaxation coincide, and therefore the integer program (ILP1) may be solved in polynomial time using linear programming. The problem (ILP2) is also solvable in polynomial time via (ILP1).*

Proof. The results follows from Theorem 4.2 and the fact that one can find an optimal vertex solution of a linear programming problem in polynomial time.

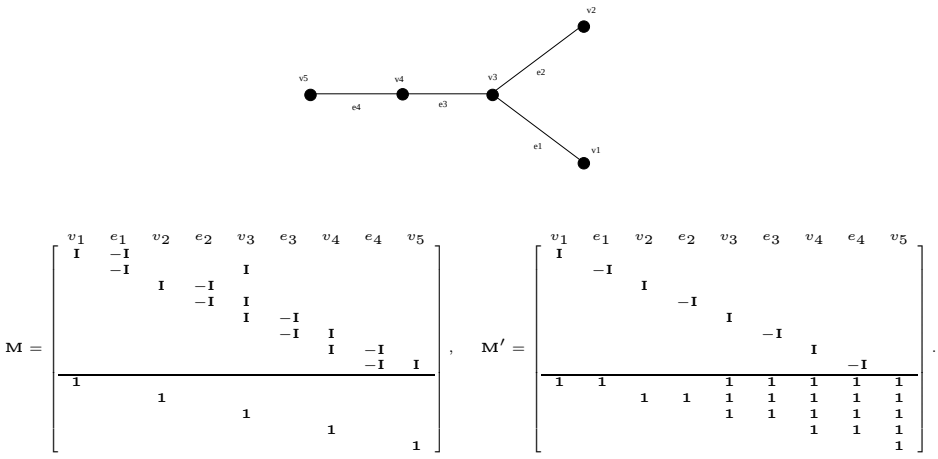

Figure 1: *An example to the proof of Theorem 4.2: a tree T and the matrices* $\mathbf{M}$ *and* $\mathbf{M}'$.

The last part of the corollary now follows from Proposition 4.1 which says that the two problems are computationally equivalent. □

Remark. It also follows from Theorem 4.2 that all the conclusions of Corollary 4.3 hold in the more general situation where the objective function coefficient of the variables $y_{e,k}$ may not all be equal, but depend on (e,k) in any manner. As mentioned before this general problem is also of interest.

The grid graphs defined in section 3 are not forests (except when m or n is 1) and the polytope $P_1(G)$ has (many) fractional vertices when $|K| \geq 3$. For instance, using a cycle of four nodes in G and three classes one may construct a vertex with components being 0 or 1/2. However, it is an interesting *empirical* fact in our computational results that the LP relaxations are tight for the objective functions of interest. We return to this discussion (which involves Theorem 4.2) at the end of section 7. Moreover, different algorithms discussed later are based on solving subproblems corresponding to trees (even paths).

4.3 Model 2

We consider the polytope $P_2(G)$ of feasible solutions in the LP relaxation of (ILP2), i.e. $P_2(G)$ is the solution set of (7)(i')–(iii). Here, as in the previous subsection we consider an arbitrary image graph G.

Theorem 4.4 *Let G be an arbitrary graph and assume that $|K| = 2$. Then $P_2(G)$ is integral.*

Proof. We use induction on the number of edges in G. The result is trivial if G has no edges. Let $K = \{1,2\}$. We may eliminate half of the $\mathbf{x}$-variables by

the equations $x_{v,2} = 1 - x_{v,1}$ for $v \in V$ and then the linear system (7) turns into

$$\begin{array}{lll} \text{(i')} & x_{u,1} - x_{v,1} \leq 1 - y'_e & \text{for } e = [u,v] \in E; \\ \text{(i")} & -x_{u,1} + x_{v,1} \leq 1 - y'_e & \text{for } e = [u,v] \in E; \\ \text{(iii)} & 0 \leq x_{v,1}, y'_e \leq 1 & \text{for } v \in V,\ e \in E. \end{array} \tag{9}$$

Clearly, it suffices to show that all the vertices in the polytope P defined by the new system (9) are integral. Let $\mathbf{w} = (\mathbf{x}, \mathbf{y}')$ be a vertex of P. Then this point is determined by a certain subset of the inequalities (9) set to equality; these are the active inequalities in $(\mathbf{x}, \mathbf{y}')$. Assume that, for some $e = [u,v] \in E$, both $x_{u,1} - x_{v,1} + y'_e \leq 1$ and $-x_{u,1} + x_{v,1} + y'_e \leq 1$ are active. This implies that $x_{u,1} = x_{v,1}$. Let G' be the graph obtained from G by shrinking the edge e (delete the edge and identify the nodes u and v). It is easy to check that the point $\mathbf{w}_1$ with components given by the remaining variables must be a vertex of the polytope $P_2(G')$ (more precisely: $\mathbf{w}_1$ is the vector where y'_e is omitted and the variables $x_{u,1}$ and $x_{v,1}$ are replaced by a single variable). Since G' has fewer edges than G we conclude, by induction, that $\mathbf{w}_1$ is integral and this implies that $\mathbf{w}$ is integral (as y'_e is either equal to 1 or $|x_{u,1} - x_{v,1}|$; otherwise $\mathbf{w}$ would not be a vertex).

We may therefore assume that for all $e = [u,v] \in E$ at most one of the inequalities $x_{u,1} - x_{v,1} + y'_e \leq 1$ and $-x_{u,1} + x_{v,1} + y'_e \leq 1$ are active. The edges for which one of these constraints is active will be called *active* in the following. Let $\mathbf{A}$ be a matrix with one row for each active edge and a column for each variable $x_{v,1}$. Let $e = [u,v]$ be an active edge. If $x_{u,1} - x_{v,1} + y'_e = 1$ holds, let $\mathbf{A}$ have a corresponding row with a 1 in the column defined by $x_{u,1}$, a -1 in the column defined by $x_{v,1}$ and all other elements are zero. Alternatively, if $-x_{u,1} + x_{v,1} + y'_e = 1$ holds, define a row in a similar way except that the 1 and -1 are switched. This means that the active constraints from (9)(i')–(i") may be written

$$\mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{y}' = \mathbf{1}.$$

But $\mathbf{A}$ is the (edge-node) oriented incidence matrix of G, so it is TU, and this implies that the matrix $[\,\mathbf{A} \quad \mathbf{I}\,]$ is TU (see [7]). From this it is clear that the vertex $\mathbf{w}$ is integral as desired. □

When $|K| \geq 2$, the polytope $P_2(G)$ typically has many fractional vertices, even for the simplest graphs. Indeed, let G' be the graph with nodes u and v and the single edge $[u,v]$. We shall discuss some properties of $P_2(G')$; for proofs and further results, see [4]. In this case there is only one neighbor variable y'_e. Define S as the set of vectors $(\mathbf{x}_u, \mathbf{x}_v, z)$ where $0 \leq z \leq 1$ and $\mathbf{x}_u, \mathbf{x}_v \in \mathbb{R}^\kappa_+$ satisfy $\sum_k x_{u,k} = \sum_k x_{v,k} = 1$. Using a change of variables $z = 1 - y'_e$ we see that the polytope $P_2(G')$ is isomorphic to the polytope given by $M = \{(\mathbf{x}_u, \mathbf{x}_v, z) \in S : \|\mathbf{x}_u - \mathbf{x}_v\|_\infty \leq z\}$. Thus the constraints (7)(i'), (i") are expressed in terms of the l_∞-norm. M is referred to as l_∞-*distance polytope* in [4]. The integer points in M are $(\mathbf{e}_i, \mathbf{e}_j, 1)$ for $i, j \leq n$, $i \neq j$ and $(\mathbf{e}_i, \mathbf{e}_i, z)$ for $i \leq n$ and $z \in \{0, 1\}$. These are vertices of M, but M also has many fractional vertices as described next. Let S_1 and S_2 be disjoint subsets of $K = \{1, \ldots, \kappa\}$ (where

$\kappa \geq 2$) such that $s := |S_1| = |S_2|$ and define $\mathbf{v}^{S_1,S_2} = \frac{1}{s}(\chi^{S_1}, \chi^{S_2}, 1) \in \mathbb{R}^{2\kappa+1}$. It can be show that the vertices of M are the points (i) $\mathbf{v}^{S_1,S_2}$ for S_1 and S_2 disjoint subsets of K of the same cardinality, and (ii) $(\mathbf{e}_i, \mathbf{e}_i, z)$ where $i \leq n$ and $z \in \{0,1\}$. As an example, let $K = \{1,\ldots,6\}$, $\mathbf{x}_u = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0)$ and $\mathbf{x}_v = (0,0,0,\frac{1}{3},\frac{1}{3},\frac{1}{3})$. Then $(\mathbf{x}_u, \mathbf{x}_v, \frac{1}{3})$ is a vertex of M and therefore $(\mathbf{x}_u, \mathbf{x}_v, \frac{2}{3})$ is a vertex of $P_2(G')$. It is clear that $P_2(G)$, for a general image graph G, has a huge number of fractional vertices which arise as above for a pair of adjacent pixels. For instance, let u and v be adjacent pixels and choose sets S_1 and S_2 as above. Then each point $(\mathbf{x}, \mathbf{y}') \in P_2(G)$ with $\mathbf{x}_u = (1/s)\chi^{K_1}$, $\mathbf{x}_v = (1/s)\chi^{K_2}$, $y'_{[u,v]} = (s-1)/s$ and all other variables (0,1) must be a vertex of $P_2(G)$.

A natural question is how to strengthen the formulation in (ILP2), that is, to find additional linear inequalities such that one obtains a tighter approximation of the convex hull of feasible points in this model. To this end, we again consider G' (the graph with two nodes). Let M^I denote the integer hull of the polytope M defined above; this is the convex hull of the integer points in M (see above). Let T be a strict subset of the class set K and consider the so-called *set difference inequality* $\mathbf{x}_u(T) - \mathbf{x}_v(T) \leq z$. It is easy to see that each set size inequality is valid for M^I. Note that for $T = \{k\}$ the set difference inequality reduces to $x_{u,k} - x_{v,k} \leq z$ (see (7)i'). It was shown in [4] (by direct methods) that a complete linear description of M^I consists of the (trivial) inequalities defining S and the set difference inequalities $\mathbf{x}_u(T) - \mathbf{x}_v(T) \leq z$ for each $T \subset K$. Note that the set difference inequalities are equivalent to the condition $\|\mathbf{x}_u - \mathbf{x}_v\| \leq z$ where $\|\cdot\|$ is the vector norm given by $\|\mathbf{w}\| = \max\{|\mathbf{w}(T)| : T \subseteq K\}$. Thus, one obtains the integer hull of M by just changing the norm in the condition on $\mathbf{x}_u - \mathbf{x}_v$! As a consequence we obtain a complete linear description of the integer hull of $P_2(G')$ for the case of two pixels when we replace z by $1 - y'_e$. Finally, we return to the case with general image graph G. For each $e = [u,v] \in E$ and $T \subset K$ the set difference inequality $\mathbf{x}_u(T) - \mathbf{x}_v(T) \leq 1 - y'_e$ is valid for the integer points in (ILP2). Thus one obtains a stronger formulation of (ILP2) by adding all such inequalities to (7)(i')–(iii). The separation problem for the set difference inequalities is easy (although the number of such inequalities grows exponentially in κ). This is seen from the expression $\max_{T \subset K}(\mathbf{x}_u(T) - \mathbf{x}_v(T)) = \mathbf{x}_u(S^+) - \mathbf{x}_v(S^+)$ where the set $S^+ = \{k \in K : x_{u,k} \geq x_{v,k}\}$ is found by $|K|$ comparisons.

5 Two important special cases

There are two special cases of the (IMS) problem that lead to well-known combinatorial optimization problems; these are the case of (i) a single row (or column) in the image, and (ii) two classes, i.e., $|K| = 2$. Both these facts are useful in the case of general images as discussed in the next section. Note that in both these situations Corollary 4.3 tells us that the problem (ILP1) may be solved using linear programming. In this section we give combinatorial algorithms for these two special cases.

5.1 A single row

Assume that the image consists of a single row (a single column is similarly treated). We denote the nodes by $i = 1, \ldots, n$ (rather than $v_1, \ldots, v_n$). The neighbor edges are $[i, i+1]$ for $i = 1, \ldots, n-1$. Consider again the model (ILP1) in (5). The relation (6) says that we may remove the variable $\mathbf{y}$ from the model as it may be expressed as a function of $\mathbf{x}$. We explain how this leads to a maximum weight path problem in an acyclic graph.

We let H denote the directed graph with node set $\{s,t\} \cup \{(i,k) : i \leq n,\ k \in K\}$. Here s is a source node and t a sink node and the remaining nodes are associated with the variables $x_{i,k}$. H contains the following arcs: (i) an arc from (i,k) to $(i+1,k')$ for $i = 1, \ldots, n-1$ and $k, k' \in K$, and (ii) an arc from s to $(1,k)$ and an arc from (n,k) to t for $k \in K$. The graph H therefore has $n+2$ layers and arcs go from each node in a layer to each node in the next layer. We define the weight of the arcs in the following way. If $e = ((i,k),(i+1,k'))$, where $i \leq n-1$, then we let $l_e = d_{i+1,k'} + \beta I(k = k')$. For $k \in K$ we define the weight of $(s,(1,k))$ to be $d_{1,k}$ and the weight of each edge incident to t is zero. Now, there is a one-to-one correspondence between the feasible solutions of (ILP1) that satisfy (6) and the directed st-paths in H. Moreover, the objective function value of the solution $\mathbf{x}$ and the weight of the corresponding path are equal. Thus, one finds an optimal solution of the problem (ILP1) by finding a maximum weight st-path in H. Since H is *acyclic*, this may be viewed as a shortest path problem. The special structure of H and the weights make it possible to solve the maximum weight path problem faster than in the general case. This is due to the fact that a maximum weight path from s to a node $w' = (i+1,k')$ must consist of a maximum weight path from s to a some node $w = (i,k)$ plus the arc from w to w'. Thus, if z_w denotes the maximum weight of a path from s to node w we get from the Bellmann-Ford equations that

$$\begin{aligned} z_{(i+1,k')} = \quad & \max_w(z_w + l_{(w,(i+1,k'))}) = \\ & \max_{k\in K}(z_{(i,k)} + l_{((i,k),(i+1,k'))}) = \\ & \max_{k\in K}(z_{(i,k)} + d_{i+1,k'} + \beta I(k = k')) = \\ & d_{i+1,k'} + \max\{z_{(i,k')} + \beta, \max_{k\neq k'} z_{(i,k)}\}. \end{aligned} \tag{10}$$

Thus the algorithm may calculate the maximum weight path lengths layer by layer as follows: *Step 1:* Initialize z by $z_{(1,k)} = d_{1,k}$ for $k \in K$ and let $i = 1$; *Step 2:* Determine the largest number s_1 and the second largest number s_2 in the set $\{z_{(i,k)} : k \in K\}$, and also determine the "maximizers" $M := \{k \in K : z_{i,k} = s_1\}$; *Step 3:* For each $k' \in K$ do the following : if $M = \{k'\}$ let $z_{(i+1,k')} = d_{i+1,k'} + \max\{z_{(i,k')} + \beta, s_2\}$, otherwise let $z_{(i+1,k')} = d_{i+1,k'} + \max\{z_{(i,k')} + \beta, s_1\}$. Finally, let $i := i+1$ and return to Step 2. The algorithm is terminated, of course, when the node t is reached (in iteration $i = n+1$, in which just finding the smallest of the numbers $z_{n,k}$ is required). The complexity of this specialized Bellman-Ford algorithm is $3\kappa n$. We shall refer to this algorithm as a shortest path algorithm later (instead of "maximum weight path in an acyclic graph").

The so-called *Viterbi algorithm* is known in the image analysis literature for solving the (IMS) problem with a single row. It is easy to see that the Viterbi

algorithm is equivalent to the algorithm above.

5.2 Two classes

We consider the special case when $K = \{1, 2\}$, i.e., there are only two classes. This is of interest for black-white images arising e.g. when one wants to identify objects, their number and geometrical shape.

We explain how this problem may be transformed into a minimum st-cut problem in a directed graph derived from the image graph G. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the directed graph obtained from G by (i) adding the two nodes s and t, (ii) replacing each edge $[u, v] \in E$ by the two directed arcs (u, v) and (v, u), and (iii) adding the arcs (s, v) and (v, t) for each $v \in V$. Define arc weights by $c_{s,v} = (d_{v,2} - d_{v,1})^+$, $c_{v,t} = (d_{v,1} - d_{v,2})^-$ and for all other arcs the weights are equal to β. The minimum st-cut problem in $\tilde{G}$ is the problem

$$\min\{\sum_{u \in U_1, v \notin U_1} c_{u,v} : U_1 \subset \tilde{V}, s \in U_1, t \notin U_1\}. \tag{11}$$

Consider again model (ILP1) and note that $K = \{1, 2\}$ means that $x_{v,1} + x_{v,2} = 1$. Thus we may eliminate variable $x_{v,2}$ by $x_{v,2} = 1 - x_{v,1}$. This, combined with (6), may be used to show that (ILP1) and (11) are equivalent. This equivalence may also be explained more directly as follows. Let $U_1 \subset \tilde{V}$ and define $U_2 = V \setminus U_1$ and view U_i for $i = 1, 2$ as the set of nodes given class i. Let $s(U_1)$ denote the number of edges in E having one end node in U_1 and the other in U_2. The weight of the cut in $\tilde{G}$ induced by the set $\{s\} \cup U_1$ is equal to

$$\begin{array}{l} \beta s(U_1) + \sum_{v \in U_1} c_{v,t} + \sum_{v \in U_2} c_{s,v} = \\ \beta s(U_1) + \sum_{v \in U_1} (d_{v,1} - d_{v,2})^- + \sum_{v \in U_2} (d_{v,2} - d_{v,1})^+. \end{array} \tag{12}$$

The term $\beta s(U_1)$ is the loss of classifying adjacent nodes differently. Since $(d_{v,1} - d_{v,2})^+ = \max\{d_{v,1}, d_{v,2}\} - d_{v,1}$ the term $\sum_{v \in U_1} (d_{v,1} - d_{v,2})^+$ is the "loss" of classifying the nodes in U_1 to class 1 compared to the best solution $\max\{d_{v,1}, d_{v,2}\}$. The final term is interpreted similarly for U_2. Note that a trivial upper bound on the optimal value $v(ILP1)$ in (ILP1) is $\hat{z} = \beta m + \sum_{v \in V} (\max\{d_{v,1}, d_{v,2}\})$. We now observe that the weight of the cut equals the difference between $\hat{z}$ and the value of the objective function evaluated in (the feasible point) corresponding to U_1. This shows that the minimum cut problem and the problem (ILP1) are equivalent and, moreover, that the minimum cut value equals $\hat{z} - v(ILP1)$.

6 Methods for the general problem

In this section we present two algorithms for solving the (IMS) problem based on (ILP1) and (ILP2), respectively. Both algorithms use decomposition in terms of Lagrangian relaxation. We first discuss the subproblems obtained in each

of the two algorithms (subsections 6.1 and 6.2) while in subsection 6.3 we describe the subgradient procedure used for solving the Lagrangian dual problems. Numerical results are found in section 7.

6.1 Lagrangian relaxation in (ILP1)

The (IMS) problems arising in practice are large-scale so one needs special-purpose algorithms for solving the problem exploiting its structural properties. This may be done in many ways, but a common theme would be to decompose the problem somehow. A natural idea is to split the image into several smaller subimages by relaxing the constraints connecting adjacent subimages. We have done this for the model (ILP2) (see subsection 6.1) and similar ideas may be used on (ILP1), but we do not discuss this here. Another decomposition idea is to split the problem into several similar problems, one for each class k. The remaining part of this subsection is devoted to such an algorithm where the decomposition is achieved through Lagrangian relaxation.

If we in (ILP1) relax constraints (5)(ii) using Lagrangian multipliers λ_v for each $v \in V$ we obtain the following Lagrangian subproblem (LR(λ)):

$$\begin{array}{lll} \max & \sum_{v\in V}\sum_{k\in K} d_{v,k}x_{v,k} + \beta\sum_{e\in E}\sum_k y_{e,k} + \sum_{v\in V}\lambda_v(\sum_{k\in K} x_{v,k} - 1) & \\ \text{s.t.} & & \\ \text{(i)} & y_{e,k} \le x_{v,k} & \text{for } k \in K,\ e \in E \text{ and } v \in e; \\ \text{(ii)} & x_{v,k} \text{ and } y_{e,k} \text{ are } (0,1) & \text{for } v \in V,\ e \in E,\ k \in K. \end{array}$$

This problem is separable in k, and for each $k \in K$ we need to solve a problem of the form

$$\begin{array}{lll} \max & \sum_{v\in V} d_v(\lambda)x_v + \beta\sum_{e\in E} y_e & \\ \text{subject to} & & \\ \text{(i)} & y_e \le x_v & \text{for } e \in E \text{ and } v \in e; \\ \text{(ii)} & 0 \le x_v, y_e \le 1 & \text{for } e \in E \text{ and } v \in V; \\ \text{(iii)} & x_{v,k} \text{ and } y_{e,k} \text{ are integral} & \text{for } e \in E \text{ and } v \in V. \end{array} \qquad (13)$$

We have here removed a constant from the objective function and omitted the index k to simplify the discussion below. Moreover, we have defined $d_v(\lambda) = d_{v,k} + \lambda_v$ and this number may be negative as λ_v is unrestricted in sign. The LP relaxation of (13) is (essentially) the dual of a transportation problem so it is possible to use specialized network (simplex) algorithms for solving this problem. Moreover, the coefficient matrix is TU so one then actually solves the integer program (13). Another approach, which performed better in our computational tests, is to transform (13) to a certain min-cut problem, and this is discussed next. Note first that an optimal solution of (13) satisfies $y_e = x_u x_v$ for each $e = [u, v]$ and therefore we may rewrite the problem to the following nonlinear optimization problem

$$\begin{array}{lll} \max & \sum_{v\in V} d_v(\lambda)x_v + \beta\sum_{[u,v]\in E} x_u x_v & \\ \text{subject to} & & \\ & 0 \le x_v \le 1 & \text{for } v \in V; \\ & x_v \text{ are integral} & \text{for } v \in V. \end{array} \qquad (14)$$

Now consider a reformulation of the objective function:

$$\begin{aligned}
&\sum_{v\in V} d_v(\lambda)x_v + \beta \sum_{[u,v]\in E} x_u x_v = \sum_{v\in V} d_v(\lambda)x_v + \beta \sum_{[u,v]\in E} x_u x_v \\
&-\frac{\beta}{2} \sum_{[u,v]\in E} (1-x_u)(1-x_v) + \frac{\beta}{2} \sum_{[u,v]\in E} (1-x_u)(1-x_v) \\
= \quad & C + \sum_{v\in V} (d_v(\lambda) + \beta D_v)x_v + \frac{\beta}{2} \sum_{[u,v]\in E} x_u x_v + \frac{\beta}{2} \sum_{[u,v]\in E} (1-x_u)(1-x_v)
\end{aligned}$$

where C is a constant not depending on the x's and where D_v is the degree of node v in G (i.e., the number of neighbors). The coefficients $\tilde{r}_v = d_v(\lambda) + \beta D_v$ may be negative. In order to avoid this feature, write $\tilde{r}_v = \tilde{r}_v^+ - \tilde{r}_v^-$ where both terms on the right hand side are positive. Then

$$\tilde{r}_v x_v = (\tilde{r}_v^+ - \tilde{r}_v^-)x_v = \tilde{r}_v^+ x_v + \tilde{r}_v^-(1-x_v) - \tilde{r}_v^-$$

where the last term may be put into the constant C. Now identify $\tilde{x}_{v,0} = x_v$, $\tilde{x}_{v,1} = 1 - x_v$, $\tilde{d}_{v,0} = \tilde{r}_v^+$, $\tilde{d}_{v,1} = \tilde{r}_v^-$ and $\tilde{y}_{e,0} = x_u x_v, \tilde{y}_{e,1} = (1-x_u)(1-x_v)$ for $e = [u,v]$. Then we may rewrite (14) as

$$\begin{array}{lll}
\text{max} & \sum_{v\in V} \sum_{k\in\{0,1\}} \tilde{d}_{v,k}\tilde{x}_{v,k} + \frac{\beta}{2} \sum_{[u,v]\in E} \sum_{k\in\{0,1\}} \tilde{y}_{e,k} & \\
\text{subject to} & & \\
& \tilde{y}_{e,k} = \tilde{x}_{u,k}\tilde{x}_{v,k} & \text{for } e\in E,\ u,v\in e,\ k\in\{0,1\}; \\
& 0 \le \tilde{x}_{v,k}, \tilde{y}_{e,k} \le 1 & \text{for } v\in V,\ e\in E,\ k\in\{0,1\}; \\
& \tilde{x}_{v,k}, \tilde{y}_{e,k} \text{ are integral} & \text{for } v\in V,\ e\in E,\ k\in\{0,1\}.
\end{array} \qquad (15)$$

Finally, by noting that we may replace the nonlinear constraint $\tilde{y}_{e,k} = \tilde{x}_{u,k}\tilde{x}_{v,k}$ by the linear constraints $\tilde{y}_{e,k} \le \tilde{x}_{u,k}$ and $\tilde{y}_{e,k} \le \tilde{x}_{v,k}$, we see that the problem in (15) is of the form (ILP1) given in (5) where we have two classes. Thus, as explained in subsection 5.2, the problem may be transformed into a min-cut problem in a certain graph.

The Lagrangian subproblem (LR(λ)) has the integrality property, i.e., the feasible polytope of its LP relaxation has only integral vertices (as remarked above, the coefficient matrix in (13) is TU). Based on a well-known result for Lagrangian duality (see e.g. [7]) we obtain the following result.

Proposition 6.1 *The optimal value of the Lagrangian dual based on relaxing constraints (5)(ii) in (ILP1) coincides with the optimal value $v(LP1)$ of the LP relaxation of (ILP1).*

6.2 Lagrangian relaxation in (ILP2)

Consider now the integer linear programming problem (ILP2) given in (7) where the underlying image graph is as defined in section 3. We discuss how Lagrangian relaxation techniques may be developed for this problem. Here the relaxations are motivated by the fact that finding an optimal solution when the

image has a single row (or column) may be done using a specialized shortest path algorithm, see section 5.

The edge set E of the image graph G may be partitioned by $E = E_r \cup E_c$ where E_r are all the "horizontal edges" joining two adjacent pixels in the same row and E_c are all the "vertical edges" joining two adjacent pixels in the same column. A first idea is to relax all those constraints (7)(i') and (i") corresponding to edges in E_c. This gives the following subproblem depending on the Lagrangian multiplier vector λ:

$$\begin{array}{lll} \max & \sum_{k\in K}\sum_{v\in V} d_{v,k}(\lambda)x_{v,k} + \beta\sum_{e\in E}\mu_e(\lambda)y'_e & \\ \text{s.t.} & & \\ \text{(i')} & x_{u,k} - x_{v,k} + y'_e \le 1 & \text{for } k\in K,\ e=[u,v]\in E_r; \\ \text{(i")} & -x_{u,k} + x_{v,k} + y'_e \le 1 & \text{for } k\in K,\ e=[u,v]\in E_r; \\ \text{(ii)} & \sum_{k\in K} x_{v,k} = 1 & \text{for } v\in V; \\ \text{(iii)} & 0 \le x_{v,k}, y'_e \le 1 & \text{for } v\in V,\ k\in K,\ e\in E; \\ \text{(iv)} & x_{v,k}, y'_e \text{ are integral} & \text{for } v\in V,\ k\in K,\ e\in E. \end{array} \qquad (16)$$

Here the coeffients in the objective function depend on λ and we have omitted a term which is independent of $\mathbf{x}$ and $\mathbf{y}$. The key point is that (16) decomposes into independent subproblems, one for each row in the image where only class and neighbor variables for that row are involved. These problems may be solved efficiently using the algorithm in section 5. Finally, each of the variables y'_e for $e \in E_c$ are set to 0 or 1 depending on the sign of $\mu_e(\lambda)$ (note that the only constraints on these variables are the simple bounds).

Thus, when solving the subproblem (16), one essentially solves n_1 (the number of rows in the image) shortest path problems. The updating of the multiplier λ may be done according to the subgradient procedure, see the next subsection. Let the Lagrangian dual problem in this approach be denoted by (LDr). An alternative and similar approach is to relax the constraints (7)(i')(i") corresponding to $e \in E_r$ instead. One then solves subproblems for each column in the image. Let (LDc) denote the corresponding Lagrangian dual problem. Simple heuristics may be used for finding a feasible solution of (ILP2) based on the possibly nonfeasible optimal solution of a Lagrangian subproblem. This can be done by fixing the $\mathbf{x}$ from the subproblem and setting $\mathbf{y}$ according to (8).

Since the optimal value of each of the Lagrangian subproblems represents an upper bound on the desired value $v(ILP2)$ we have that

$$v(LDr), v(LDc) \ge v(ILP2).$$

However, our experience is that these bounds may not be very good (see [9]) so we next discuss an improved algorithm where the idea is to combine the strengths of the two approaches (LDr) and (LDc) using the technique of *cost splitting* (see [7]) which is a way of strengthening the Lagrangian dual. Consider again (ILP2). We now replace the variable $\mathbf{x}$ by the "row class variable" $\mathbf{x}^r$ and the "column class variable" $\mathbf{x}^c$ and get the following integer linear programming model.

$$
\begin{array}{lll}
\max & \frac{1}{2}\sum_{k\in K}\sum_{v\in V} d_{v,k}x^r_{v,k} + \beta\sum_{e\in E_r} y'_e + & \\
& \frac{1}{2}\sum_{k\in K}\sum_{v\in V} d_{v,k}x^c_{v,k} + \beta\sum_{e\in E_c} y'_e & \\
\text{s.t.} & & \\
\text{(i)} & x^r_{u,k} - x^r_{v,k} + y'_e \le 1 & \text{for } k\in K,\ e=[u,v]\in E_r; \\
\text{(ii)} & -x^r_{u,k} + x^r_{v,k} + y'_e \le 1 & \text{for } k\in K,\ e=[u,v]\in E_r; \\
\text{(iii)} & \sum_{k\in K} x^r_{v,k} = 1 & \text{for } v\in V; \\
\text{(i')} & x^c_{u,k} - x^c_{v,k} + y'_e \le 1 & \text{for } k\in K,\ e=[u,v]\in E_c; \\
\text{(ii')} & -x^c_{u,k} + x^c_{v,k} + y'_e \le 1 & \text{for } k\in K,\ e=[u,v]\in E_c; \\
\text{(iii')} & \sum_{k\in K} x^c_{v,k} = 1 & \text{for } v\in V; \\
\text{(iv)} & x^r_{v,k} = x^c_{v,k} & \text{for } v\in V,\ k\in K; \\
\text{(v)} & \mathbf{x}^h, \mathbf{x}^v, \mathbf{y} \text{ are (0,1).} &
\end{array}
\tag{17}
$$

The model is equivalent (in terms of feasible solutions) to (ILP2) since the constraints (17)(iv) assures that $\mathbf{x}^r = \mathbf{x}^c$. We now relax these equality constraints and add a term $\lambda_{v,k}(x^r_{v,k} - x^c_{v,k})$ to the objective function. The resulting Lagrangian subproblem decomposes into a "row problem" involving only the $\cdot^r$-variables and a "column problem" involving only the $\cdot^c$-variables. Furthermore, in the row problem there are no constraints between variables from different rows, making it possible to solve for each row separately using the shortest path procedure. The column problem may be treated similarly. Let (LDrc) denote the Lagrangian dual problem with this approach.

We may now summarize the relation between the bounds of the relaxations of (ILP2) that be have discussed. Let (LP2) denote the LP relaxations of (ILP2). The result may all be derived from Lagrangian duality theory, in particular from the fact that cost-splitting represents a strengthening of the bound.

Theorem 6.2

$$v(ILP2) \le v(LDrc) \le \min\{v(LDr), v(LDc)\} \le v(LP2).$$

6.3 The subgradient procedure

In the previous two subsections we presented different Lagrangian relaxations in connection with (ILP1) and (ILP2). These subproblems give, for each Lagrangian multiplier vector λ, an upper bound $z(\lambda)$ on the optimal value of the integer program. The Lagrangian dual problem of finding the best (i.e. smallest) upper bound by varying λ may be solved using the subgradient algorithm, see [7] for a general description. We briefly describe the principle as applied to the problem (LDrc). Consider the Lagrangian subproblem of (17) when $\mathbf{x}^r = \mathbf{x}^c$ is relaxed using the multiplier vector λ^s; this is in iteration s. Let $\bar{x}^r_{v,k}$ and $\bar{x}^c_{v,k}$ denote the x^r and x^c variables in an optimal solution of the subproblem corresponding to λ^s. Then $\mathbf{g} := \bar{\mathbf{x}}^r - \bar{\mathbf{x}}^c$ is a subgradient of (the piecewise linear and) convex function $\lambda \to z(\lambda)$ at the point λ^s. The new multiplier λ^{s+1} is obtained

by taking a step from λ^s in the direction of the subgradient, that is

$$\lambda_{v,k}^{s+1} = \lambda_{v,k}^{s} - \theta_s(\bar{x}_{v,k}^{r} - \bar{x}_{v,k}^{c})$$

where $\theta_s > 0$ is the steplength. We decrease the steplengths according to $\theta_s = (1/2)^s(z(\lambda_s) - z_L)/\|\mathbf{g}\|^2$ where z_L is a lower bound on the optimal value $\min_\lambda z(\lambda)$. In theory there are better choices, but this usually works in practice. A problem in connection with (LRrc) (as for other problems when equalities are relaxed) is that the solutions "flip around" too much and then the subgradients may change a lot. A good idea is then to stabilize the process by moving λ in a direction given by some convex combination of the new and the previous subgradients (with the larger weight on the new subgradient). This was done in the algorithms reported in the next section. The final point we mention is that we used simple heuristics for turning the nonfeasible solutions in the subproblems into feasible ones. In the case of (LRrc) this may be done by letting $\mathbf{x}^c := \mathbf{x}^r$, and defining the variable $\mathbf{y}'$ accordingly (see (8)). This also means that our lower bound z_L may be updated when a better integer solution has been found and the new bound is used in the steplength calculation.

7 Computational results

In this section we report computational results and experiences with two of the algorithms discussed in section 6. *Algorithm 1* is based on decomposing (ILP1), see (5), into κ minimum cut subproblems, as described in subsection 6.1. The subproblems could also be solved as transportation problems, but some preliminary experiments showed that this approach (using general routines from the LP solver CPLEX, see [3]) was significantly slower than using the min-cut approach. *Algorithm 2* is based on the cost-splitting technique for (ILP2), see (7) (and the Lagrangian dual (LDrc)). In [9], this algorithm performed much better than the others discussed in subsection 6.2. For both algorithms we have used the subgradient method as described in subsection 6.3 in order to solve the Lagrangian dual problem.

In order to evaluate the algorithms, we generated a large number of different instances of the (IMS) problem by varying the different parameters involved. In order to analyze the running times of the algorithms, a *regression analysis* was used. Compared to the more common approach of presenting a large number of tables, regression analysis gives direct estimates for the influence of the different parameters.

The input images to our algorithms were simulated from the statistical models described in section 2. The numerical experiments consisted of the following stages

1. simulating the underlying *true image* $\mathbf{c}$ according to the prior model 2;
2. conditioned on $\mathbf{c}$, simulating the *observed image* $\mathbf{z}$ according to the observation model 1;

3. running the different algorithms on the obtained (IMS) problem.

The performance of the algorithms may be evaluated in terms of the *quality* of the integer solutions (measured via an optimality gap) and the *computational time*, the CPU time until the subgradient procedure converged. For *all the test problems that were generated our algorithms found an optimal solution*; this was verified by the coincidence of lower and upper bounds for the optimal value. This was true for both algorithms. This, of course, was amazing, having in mind that these were all large-scale integer programs. Thus, in this section, we may concentrate on a comparison of the computational times for the two algorithms. At the end of this section we comment on the fact that optimal solutions were found consistently.

The parameters in the models that may influence computational time are: the size of the images, the amount of smoothness (as defined through β in (2), the class probabilities (defined through $\alpha_c$in (2), and the shape of the distributions for $\mathbf{z}$ ($f(z_i|c_i)$ in (1)). In addition, the models are *stochastic* so different results may occur even if the parameters are fixed. We generated a wide range of test instances by varying each of the parameters as described in the following.

Consider first the size of the images. In practice the size of images are typically 256×256 or 512×512. However, in order to be able to process a large number of images, smaller sizes were considered. In particular, we have used the sizes 20×20, 40×40 and 60×60.

Next, consider the parameters κ, $\alpha_1, ..., \alpha_\kappa$ and β in the prior model (2). κ, the number of classes, was chosen to be either 2, 4 or 6. Although the methods may depend on the different class probabilities, the important aspect of the prior model in this context is the smoothing parameter β. We therefore chose each class to be equally probable, corresponding to $\alpha_c = 0$ for all c. The smoothing parameter β was given the values 0.5, 0.7 or 0.9 (covering the range of values commonly used in practice).

Consider finally the observation model $f(z_i|c_i)$. Now there is a whole range of possibilities, the choice heavily depends on the application. The performance of a method for reconstructing the true image will mainly depend on the so-called signal to noise ratio in the observation model and not too much on the shape of the observation model. We will therefore only consider one (commonly assumed) model for the observation, the Gaussian one:

$$f(z_v|c_v) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{1}{2\sigma^2}(z_v - \mu_{c_v})^2\}.$$

Here μ_{c_v} is the expected value of z_v given that the class in pixel i is c_v, while σ^2 is the variance of the observations (here assumed to be equal for all classes). For simplicity, we chose $\mu_c = 100(c-1)$. The signal to noise ratio is defined by

$$SR = \frac{\sqrt{\sum_{c=1}^{\kappa}(\mu_c - \bar{\mu})^2/\kappa}}{\sigma}.$$

Such a measure gives information about the difficulty in restoring $\mathbf{c}$ from $\mathbf{z}$. For each κ, the values of σ^2 were chosen such that the signal to noise ratios were 0.5, 1.0 and 2.0, corresponding to low, medium and high signal to noise ratios.

In conclusion, all these parameter choices gave $3\times3\times3\times3 = 81$ combinations. In addition, each combination was repeated 5 times, giving a total of 405 test instances.

In fig. 2, the CPU time (in seconds) for Algorithm 1 and Algorithm 2 are plotted against each other. (For points on the linear curve $y = x$, the times were equal). The computations were done on a SPARC 20 computer. Note first, as expected, that increase in size or the number of classes typically result in larger computational time for both algorithms. We see that Algorithm 2 consistently is faster than the other. This difference may either be because the number of subgradient iterations needed is much larger for Algorithm 1 *or* because the amount of computation needed inside each iteration differ (or both). In order to (partly) evaluate this, the number of iterations needed for convergence in the two algorithms are plotted in fig. 3. In this case, the difference between the algorithms is much less clear, indicating that the main difference between the two algorithms is caused by the computation time inside each iteration.

In order to explain how the computational time depends on the different parameters involved, a deeper analysis is required. We therefore analyzed the number of subgradient iterations and the time per iteration separately.

For Algorithm 1, a min-cut algorithm is run for each class in each iteration. We used an implementation of the Goldberg-Tarjan push-relabel method which (essentially) has worst-time complexity $O(nm)$ in a graph with n nodes and m arcs. We used regression analysis to analyze how the different parameters influenced the computational time in our experiments. Typically, the factors (parameters) involved will influence multiplicatively (up to some power). A reasonable regression model for the time per iteration, t say, is therefore

$$t = C \cdot (n_1 n_2)^{\gamma_1} \cdot \kappa^{\gamma_2} \cdot \beta^{\gamma_3} \cdot (\sigma^2)^{\gamma_4} \cdot \varepsilon \tag{18}$$

where γ_i defines the order of complexity while ε is a multiplicative noise factor (relating to that the actual image observed is stochastic). Our aim is to estimate the unknown coefficients γ_i based on the observed times per iteration. It is advantageous to transform the regression model to a linear one by taking the logarithm on both sides of (18):

$$\log(t) = \log(C) + \gamma_1 \log(n_1 n_2) + \gamma 2 \log(\kappa) + \gamma_3 \beta + \gamma 4 \log(\sigma^2) + \tilde{\varepsilon} \tag{19}$$

where now $\tilde{\varepsilon} = \log(\varepsilon)$. The coefficients can now be estimated by ordinary linear regression. Table 7 shows the result from the linear regression. 94% of the variation in computational times were explained by the model, indicating that the regression model is realistic. Since we are solving one min-cut problem for each class in one iteration, the time per iteration should be approximately linear with respect to the number of classes. This means to that the coefficient corresponding the $log(\kappa)$ should be equal to 1.0. We see that the estimated value was 1.15 which indicated linearity. The size of the image seems to influence by a power higher than 1, although lower than the theoretical worst case corresponding to a power of 2. Note that also β and σ^2 seems to influence on the time per iteration, an increase in either parameter results in increased time per iteration.

	$\hat{\gamma}_i$	Std. Error	p-value
(Intercept)	-12.390	0.158	0.000
$\log(n_1 n_2)$	1.368	0.020	0.000
$\log(\kappa)$	1.145	0.040	0.000
$\log(\beta)$	0.379	0.040	0.000
$\log(\sigma^2)$	1.230	0.040	0.000

Table 1: *Results of performing a regression of the time per iteration for Algorithm 1. 94% of the variation in time per iteration was explained by this model. The estimated value (corresponding to the power for which the corresponding parameter influence on the time per iteration) is given in column 2. The uncertainty involved in this estimate, measured through standard deviation, is given in column 3. The p-value corresponds to testing if the true coefficient is equal to zero (no influence on time per iteration) is given in the last column. A small p-value indicate significant influence.*

	$\hat{\gamma}_i$	Std. Error	p-value
(Intercept)	-10.227	0.063	0.000
$\log(n_1 n_2)$	0.984	0.016	0.000
$\log(\kappa)$	0.785	0.016	0.000
$\log(\beta)$	0.012	0.016	0.471
$\log(\sigma^2)$	-0.073	0.016	0.000

Table 2: *Results of performing a regression of the time per iteration for Algorithm 2. Almost 98% of the variation in time per iteration is explained by the regression model. The different columns are as explained for Table 7.*

A similar analysis was done for Algorithm 2. In this case, almost 98% of the variation was explained by the regression model. Efficient implementation of the Viterbi algorithm (or shortest path algorithm) involved in Algorithm 2 results in a theoretical complexity of $O(n_1 n_2 \kappa)$ per iteration. This indicates that the coefficients corresponding to size $n_1 n_2$ and to the number of classes κ should be equal to 1. The estimated coefficient corresponding to size is almost equal to 1, confirming the theory. The estimated coefficient corresponding to κ is somewhat lower than 1. The coefficients corresponding to β and σ^2 are both small, so these parameters do not influence much on the time per iteration, again confirming the theory.

It is more difficult to explain how the number of iterations depend on the parameters involved. There are many reasons for this: (*i*) few theoretical results are available, (*ii*) the experiments do not give any clear indications on how this dependence is, (*iii*) in some cases the algorithm did not converge during the maximum number of iterations (1000). Regarding the last aspect, some of the cases for which convergence (i.e., upper bound equal to lower bound) were not obtained in 1000 iterations (the chosen maximum number of iterations

	$\hat{\gamma}_i$	Std. Error	p-value
(Intercept)	-0.475	0.532	0.372
$\log(n_1 n_2)$	0.345	0.050	0.000
$\log(\bar{k})$	1.679	0.193	0.000
$\log(\beta)$	0.065	0.101	0.520
$\log(\sigma^2)$	1.133	0.135	0.000
log(error rate)	0.300	0.077	0.000

Table 3: *Results of performing a regression on the (logarithm of) number of iterations for Algorithm 1. 68% of the variation in time per iteration is explained by the regression model. The different columns are as explained for Table 7.*

	$\hat{\gamma}_i$	Std. Error	$\Pr(>\|t\|)$
(Intercept)	-1.592	0.406	0.000
$\log(n_1 n_2)$	0.458	0.038	0.000
$\log(\bar{k})$	1.880	0.147	0.000
$\log(\beta)$	0.307	0.077	0.000
$\log(\sigma^2)$	1.848	0.103	0.000
log(error rate)	0.362	0.059	0.000

Table 4: *Results of performing a regression on the (logarithm of) number of iterations for Algorithm 2. 85% of the variation in time per iteration is explained by the regression model. The different columns are as explained for Table 7.*

during the experiments) were investigated further. In all cases convergence was obtained either by increasing the maximum number of iterations or by changing some tuning parameters in the subgradient algorithm.

Regression analysis were again performed for the two algorithms, but now with (the logarithm of) the number of iterations as response. In this case, one might expect that the complexity would depend on the "quality" of the MAP estimate. Here quality refers to the error rate which is the percentage of the pixels that are classified wrong. The results from the regression analysis are given in tables 3 and 4 for Algorithms 1 and 2, respectively. Note that both β, σ and the error rate seem to influence the number of iterations. Smoother images, more observation noise or higher error rates (corresponding to poorer quality of the MAP estimate) all result in a higher number of iterations. Regarding the size, the order is less than a half for both algorithms, while a near quadratic order is obtained for the number of classes.

Fig. 4 (left panel) shows a real image Magnetic Resonance (MR) image of the brain. Such images are usually recorded in several bands, making the observation in pixel v, z_v, a vector. The image displayed is the so-called T2-weighted band, but in the analysis below one additional channel was used. The classes of interest were in this case *fat, air/bone, connective tissue, cerebrospinal fluid* and *brain parenchyma*, giving a total of $\kappa = 5$ classes. The size of the image was

256×256 pixels. For model (ILP1), this gives 980480 variables while model (ILP2) contains 458240 variables. Both models contain 715776 constraints. Both algorithms converged also in this case, so an optimal solution was found. Algorithm 1 needed 25 subgradient iterations while Algorithm 2 needed 63. The corresponding CPU times were 1668.3 and 201.9 seconds, showing that although Algorithm 1 converged much faster in terms of number of iterations while Algorithm 2 was superior in actual CPU time. Fig. 5 displays the upper and lower bounds obtained as function of CPU time. Again the superiority of Algorithm 2 is clearly seen. In the right panel of fig. 4, the actual optimal solution is displayed, giving a very reasonable estimate of the true distribution of classes.

Finally, we discuss the fact that the algorithms found an optimal solution in all our test cases. In section 4 we showed a number of situations where the polytopes $P_1(G)$ and $P_2(G)$ are integral and this directly explains that we find integral optimal solutions whenever $\kappa = 2$. In all other test instances these polytopes have lots of fractional vertices; for $P_2(G)$ this was discussed in section 4 while for $P_1(G)$ it is easy to see that there are many fractional solutions corresponding to every cycle in the image graph G. This means that the success of our algorithms must be (partially) due to the particular class of objective functions involved. We have so far not been able to show any theoretical result in this direction, but we suspect that there may exist such results saying that for particular choices of the distribution functions $f(z_i|x_i)$ an integral optimal solution of the LP relaxation exists. One such situation might be when there are some "stochastic ordering" between the classes involved, which is the case for the test instances considered here (obtained by the ordering of the expectations $\mu_1, ..., \mu_\kappa$). Another, still vague, explanation relates to Theorem 4.2. Assume that the true image may be divided into "homogeneous regions" such that the borders between adjacent regions constitute a tree T. Within each region all the nodes have the same class and different regions have different classes. Provided that the observed image is not too blurred, all nodes inside a region will be given the correct class in the solution of the LP relaxation. This seems reasonable since they have the same class as the "best one" and one gets the maximum contribution from the neighbor term when all pixels are given the same class. Then the remaining variables correspond to the tree T and they are essentially found by optimizing over the polytope $P_1(T)$ which makes all these variables integral according to Theorem 4.2. The values of variables in all the regions just influence the objective function in the optimization problem over $P_1(T)$.

8 Concluding remarks

The Lagrangian algorithms presented seem promising for solving interesting image segmentation problems. All the test problems considered in this study were solved to optimality. The fastest algorithm (Algorithm 2) was based on decomposing the image into separate row and column problems and using cost-splitting techniques. Moreover, we have investigated some theoretical properties of the integer programming models involved, in particular, proving integrality

results for certain subclasses of problems.

Further work could, for instance, be to design a specialized algorithm for solving the min-cut (or transportation) subproblems in Algorithm 1.

Finally, we would like to point out that the area of image analysis is a source for interesting large-scale computational optimization problems.

Acknowledgement. The authors thanks Kaj Holmberg for interesting and useful discussions.

References

[1] J. Besag. On the statistical analysis of dirty pictures. *Journal of Royal Statistical Society, Series B*, 48(3):259–302, 1986.

[2] J. Besag. Towards Bayesian image analysis. *Journal of Applied Statistics*, 16(3):395–407, 1989.

[3] CPLEX. Using the CPLEX callable library. Technical report, CPLEX Optimization, Inc., 1994.

[4] G. Dahl. Polytopes related to some polyhedral norms. Technical Report 226, University of Oslo, Institute of Informatics, Oslo, Norway, November 1996. To appear in *Oper. Res. Letters.*

[5] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images. *IEEE Tran. on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[6] D.M. Greig, B.T. Porteous, and A.H. Seheult. Exact maximum a posterior estimation for binary images. *Journal of Royal Statistical Society, Series B*, 51:271–279, 1989.

[7] G. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization.* Wiley, 1988.

[8] A. Schrijver. *Theory of linear and integer programming.* Wiley, Chichester, 1986.

[9] G. Storvik and G. Dahl. Lagrangian based methods for finding MAP solutions for MRF models. Technical Report 17, University of Oslo, Institute of Mathematics, Oslo, Norway, 1996.

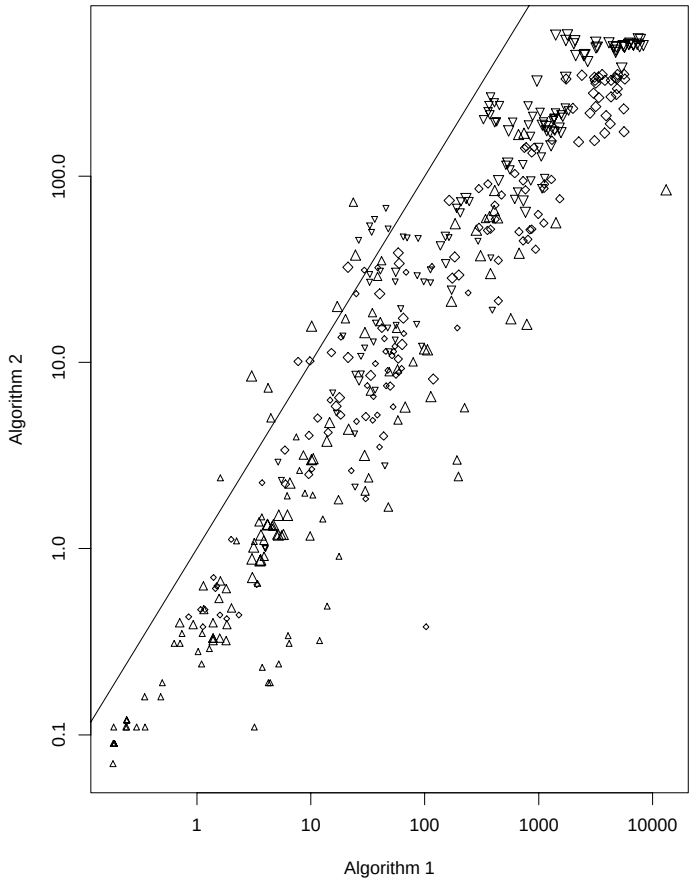

Figure 2: *Plot of time in seconds with Algorithm 1 on the x-axis and Algorithm 2 on the y-axis. Symbols $\triangle$, $\diamond$ and $\bigtriangledown$ are used for $\kappa = 2, 4$ and 6, respectively. The sizes of the symbols are proportional to the sizes of the images. Both the x- and y-axes are on logarithmic scales. The linear curves in the plots are the lines $y = x$.*

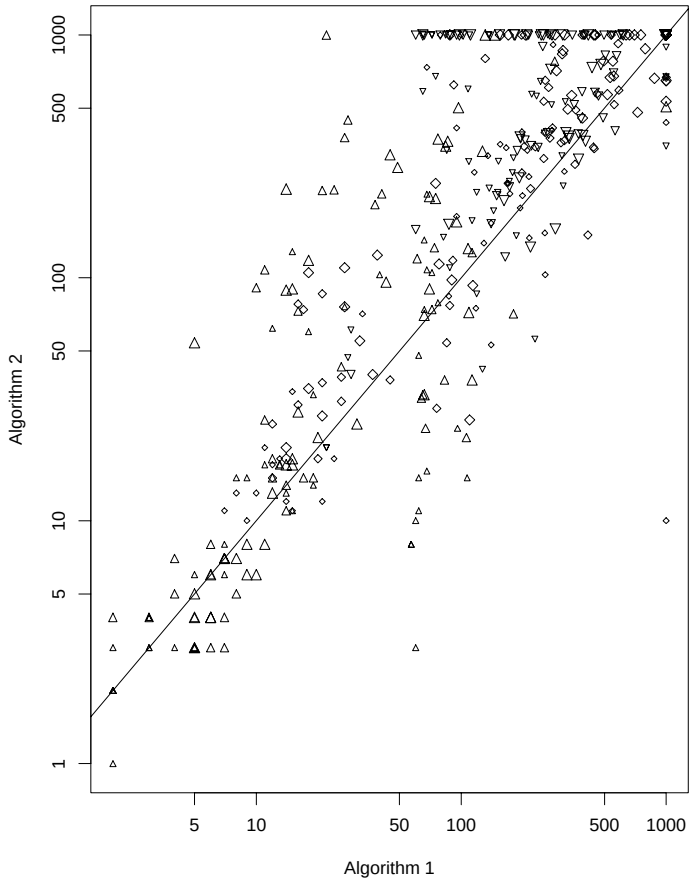

Figure 3: *Plot of number of iterations with Algorithm 1 on the x-axis and Algorithm 2 on the y-axis. Symbols $\triangle$, $\diamond$ and $\bigtriangledown$ are used for $\kappa = 2, 4$ and 6, respectively. The sizes of the symbols are proportional to the sizes of the images. Both the x- and y-axes are on logarithmic scales. The linear curves in the plots are the lines $y = x$.*

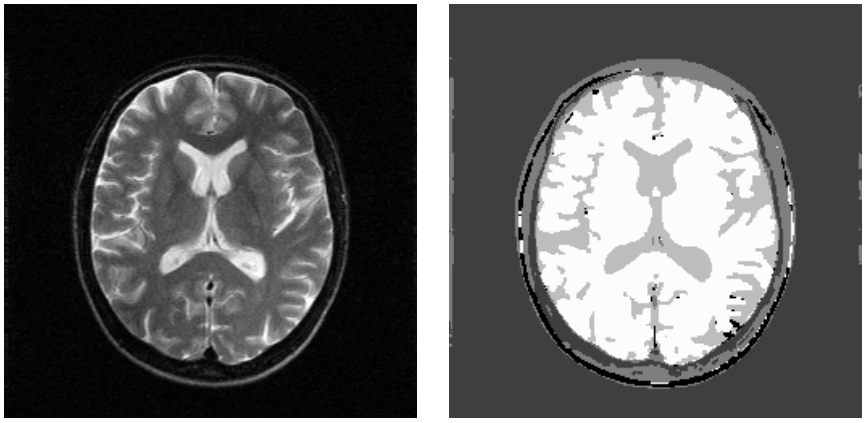

Figure 4: *To the left, T2-weighted Magnetic Resonance image of the brain. The gray values correspond to values in the range from 0 (black) to 255 (white). To the right, the MAP solution obtained by Algorithm 2. The classes are* fat, air/bone, connective tissue, cerebrospinal fluid *and* brain parenchyma*, displayed as grey-levels with the first class being white and the last one being black.*

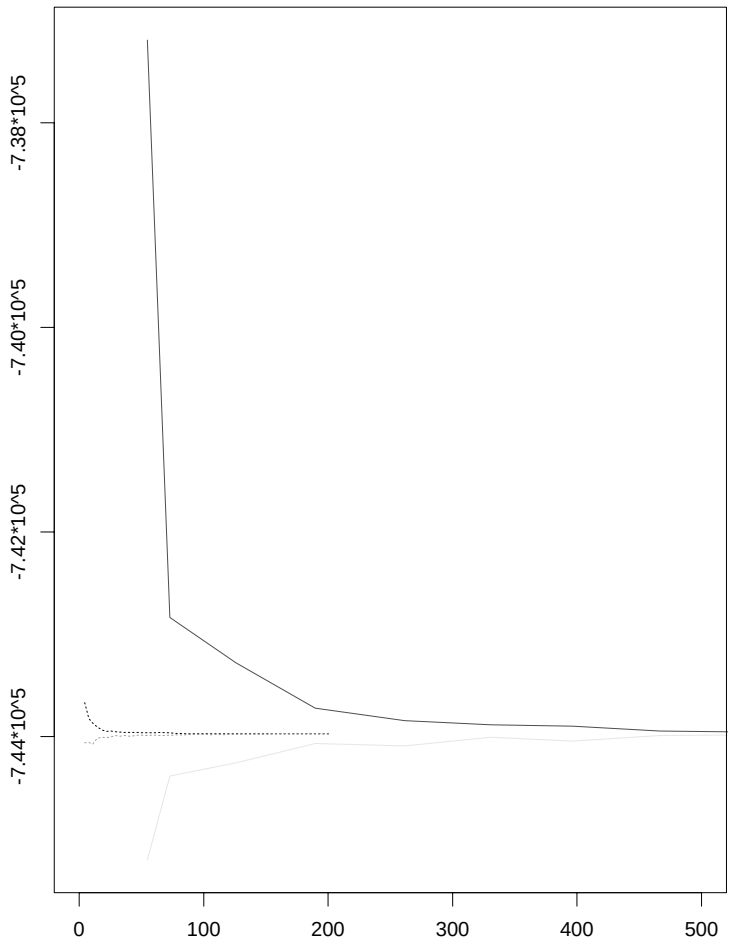

Figure 5: *Upper and lower bounds obtained by using Algorithms 1 and 2 on the brain image given in fig. 4. The bounds are plotted as function of CPU time. Solid lines are for Algorithm 1 and dashed lines are for Algorithm 2.*