Globally Optimal Image Partitioning by Multicuts

Jörg Hendrik Kappes, Markus Speth, Björn Andres, Gerhard Reinelt, and Christoph Schnörr

Image & Pattern Analysis Group; Discrete and Combinatorial Optimization Group; Multidimensional Image Processing Group – University of Heidelberg, Germany

Abstract. We introduce an approach to both image labeling and unsupervised image partitioning as different instances of the multicut problem, together with an algorithm returning globally optimal solutions. For image labeling, the approach provides a valid alternative. For unsupervised image partitioning, the approach outperforms state-of-the-art labeling methods with respect to both optimality and runtime, and additionally returns competitive performance measures for the Berkeley Segmentation Dataset as reported in the literature.

Key words: image segmentation, partitioning, labeling, multicuts, multiway cuts, multicut polytope, cutting planes, integer programming

1 Introduction

Partitioning an image into a number of segments is a key problem in computer vision. We distinguish (i) *image labeling* where segments are associated with a finite number of classes (e.g., street, sky, person, etc.) based on pre-defined features, and (ii) *unsupervised partitioning* where no prototypical features as class representatives are available, but only pairwise distances between features.

Concerning (i), partitions are determined by inference with respect to variables that take values in a finite set of labels and are assigned to the *nodes* of the underlying graph [1]. Accordingly, the marginal polytope has become a focal point of research on relaxations and approximate inference for image labeling [2–4].

In this paper, we focus on the image partitioning problem as a *multicut problem* which appears natural for unsupervised partitioning (ii) and includes image labeling (i) as a special case. Here, random variables are assigned to the *edges* of the underlying graph. This is appealing because in order to form a partition, edges have to adjoin in order to separate nodes properly and thus *explicitly represent local shape*, which can only indirectly be achieved through labelled nodes by taking differences. Clearly, edge indicator vectors have to be constrained in order to form valid partitions [5–7], and the resulting combinatorial problem is NP-hard. We demonstrate below, however, that especially for unsupervised scenarios (ii), our multicut approach enables us to compute efficiently globally optimal image partitions – see Fig. 1.

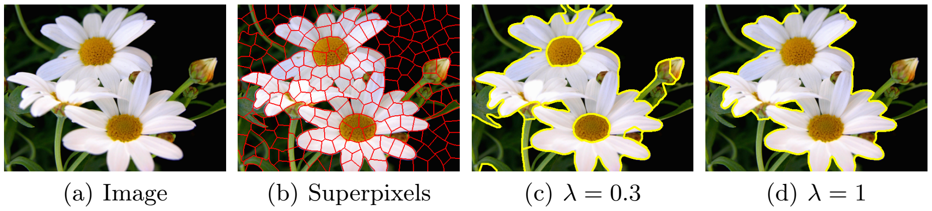


Fig. 1. (a-b) An image preprocessed into a set of superpixels [11]. **(c-d)** Globally optimal partition minimizing the objective function (11) for two different values of λ in an unsupervised setting, computed in less than 0.1 seconds excluding the time for computing superpixels.

A *subclass* of the multicut problem, the *multiway cut problem*, has been introduced to computer vision in [8] as a generalization of the basic min-cut/max-flow approach. For specific problems in this subclass, efficient approximative methods exist [9], and for few special cases, e.g., for planar graphs, exact polynomial time algorithms are known [10].

Our approach applies to both scenarios (i) and (ii) sketched above and can deal with generalized Potts terms that might have negative signs.

Our contributions are the following:

1. We reformulate *image labeling* in terms of generalized Potts models as a multicut problem (Sec. 2).
2. We provide an economical multicut formulation of the *unsupervised partitioning* problem. It can be based on arbitrary features and pairwise distances, and generates a hierarchy of partitions by varying a single parameter (Sec. 2).
3. We devise a three-phase iterative procedure for computing globally optimal partitions in both scenarios (i) and (ii) based on LP-relaxation, integer programming, and the cutting plane method (Sec. 3). For our LP-relaxation, a deterministic rounding procedure suggested by [12] returns a $\left(\frac{3}{2} - \frac{1}{k}\right)$-approximate integer solution of the k-multiway cut problem and improves the performance bound $2 - \frac{2}{k}$ known in computer vision so far [13].
4. We compare our approach for *image labeling* problems (Sec. 4). State-of-the-art methods [4, 13] are competitive concerning both optimality (though they do not provide any guarantee) and runtime. Thus, our approach only provides a competitive alternative using a different problem formulation.
5. We compare our approach for *unsupervised partitioning* (Sec. 4). Our optimization method clearly outperforms competing methods concerning both optimality and runtime. Concerning the *Berkeley Segmentation Dataset and Benchmark (BSD)*, our approach even is on par with approaches that rely on edge *detection* (privileging them), rather than on image *partitioning*.

In contrast to [14], the present paper discusses a 3-phase algorithm together with a hierarchy of inequality constraints (Sec. 3) and examines experimentally optimization methods for the multicut/image partitioning problem.

2 Problem Description

2.1 Image Labeling

Given a graph $G = (V, E)$, we assign labels from a label set $L = \{1, \dots, k\}$ to all nodes $v \in V$ by using node variables $x_v \in L$. A labeling $x = (x_v)_{v \in V} \in L^{|V|}$ defines a partition of V into subsets of nodes S_l assigned to class l, i.e., $\bigcup_{l \in L} S_l = V$. Furthermore, for a logical expression φ, we define an indicator function $\mathbb{I}(\varphi)$ which is 1 if the expression is true and 0 otherwise.

The cost of a labeling is the sum of the label assignment costs for each node, plus the sum of weighted edges connecting different classes which may be considered as an approximation of the weighted length of the separating boundaries:

$$J(x) = \sum_{v \in V} f_v(x_v) + \sum_{uv \in E} \beta_{uv}\, \mathbb{I}(x_u \neq x_v). \tag{1}$$

Note that the weight of the boundary $\beta_{uv} \in \mathbb{R}$ depends only on the edge uv and not on the labels assigned to u and v. In the simplest case, all edges are treated equally, i.e., $\beta_{uv} = \hat{\beta}$ for all $uv \in E$.

The function $f_v(l)$ encodes the similarity of data observed at location v to class l. Since the number of labels is finite, we can represent the function by a vector $\theta_v \in \mathbb{R}^k$:

$$\theta_{v,l} = f_v(l), \qquad f_v(x_v) = \sum_{l \in L} \theta_{v,l} \mathbb{I}(x_v = l). \tag{2}$$

A common approach to determine a labeling is to consider the combinatorial optimization problem

$$\min_{x \in L^{|V|}} \sum_{v \in V} f_v(x_v) + \sum_{uv \in E} \beta_{uv}\, \mathbb{I}(x_u \neq x_v). \tag{P1}$$

Instead of optimizing over the set of all node labelings (node domain), we optimize over the set of all separating boundaries related to valid partitions (edge domain), which is known as the multicut problem, see Sec. 2.3.

2.2 Unsupervised Pairwise Image Partitioning

We will also study the following important variant of problem (P1):

$$\min_{x \in L^{|V|}} \sum_{uv \in E} \beta_{uv} \mathbb{I}(x_u \neq x_v), \qquad\qquad L = \{1, \dots, |V|\}. \tag{P2}$$

Here, in comparison to (P1), we have $f_v \equiv 0$ for all $v \in V$. Coefficients β_{uv} may depend on data but are assumed *not* to depend on prototypical prior information about a fixed number of classes L, so that the maximum number of labels is $|V|$. Rather, only pairwise distances between data (or features) are used. To obtain a well-posed problem, the sign of β_{uv} is not restricted.

As for the image labeling problem in the previous section, we will also study solving problem (P2) by multicuts which turns out to offer an economical representation – cf. Fig. 1.

2.3 The Multicut Problem

Let $G = (V, E)$ and $\bigcup_{i=1}^{k} S_i = V$ be a partition of V. Then we call the edge set

$$\delta(S_1, \ldots, S_k) := \{uv \in E \mid \exists i \neq j : \ u \in S_i \text{ and } v \in S_j\} \tag{3}$$

a *multicut* and the sets $S_1, \ldots, S_k$ the *shores* of the multicut.

To obtain a polyhedral representation of multicuts, we define *incidence vectors* $\chi(F) \in \mathbb{R}^{|E|}$ for each subset $F \subseteq E$:

$$\chi_e(F) = \begin{cases} 1, & \text{if } e \in F, \\ 0, & \text{if } e \in E \setminus F. \end{cases}$$

The *multicut polytope* is given by

$$\mathrm{MC}(G) := \mathrm{conv}\,\{\chi(\delta(S_1, \ldots, S_k)) \mid \delta(S_1, \ldots, S_k) \text{ is a multicut of } G\}\,. \tag{4}$$

For an overview and further details on the geometry of this and related polytopes, we refer to [5].

For given edge weights $w(e) \in \mathbb{R}$, $e \in E$, the *multicut problem* is to find a multicut for which the sum of the weights of cut edges is minimal. Since all vertices of the multicut polytope correspond to multicuts, this amounts to solving the linear program

$$\min_{y \in \mathrm{MC}(G)} \sum_{e \in E} w(e)\, y_e. \tag{P3}$$

In order to apply linear programming techniques, we have to represent $\mathrm{MC}(G)$ as intersection of half-spaces given by a system of affine inequalities. Since the multicut problem is NP-hard [15], we cannot expect to find a system of polynomial size. But, as we will see later, partial systems may be very helpful to solve the multicut problem.

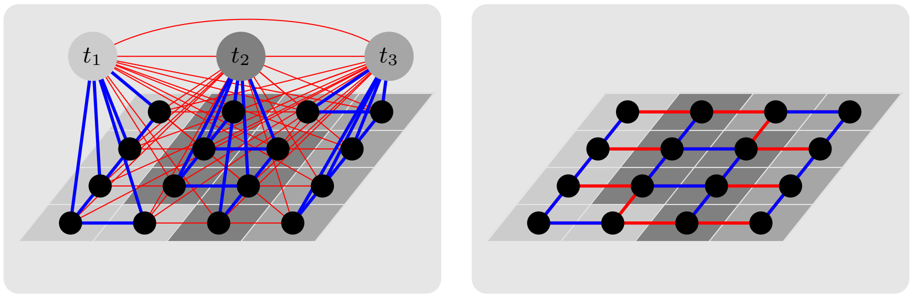
Before discussing how problem (P3) can be solved efficiently, we will show how the problems (P1) and (P2) can be transformed into problem (P3).

2.4 Image Labeling as Multicut Problem

To write problem (P1) as a multicut problem, we use its defining graph G and introduce k additional terminal nodes $T = \{t_1, \ldots, t_k\}$. Then we define the graph $G' = (V', E')$ by

$$V' = V \cup T, \quad E' = E \cup \{(t, v) \mid t \in T, v \in V\} \cup \{(t_i, t_j) \mid 1 \leq i < j \leq k\}.$$

Each node $v \in V$ is connected to all terminal nodes $t \in T$. The terminal nodes represent the k labels, and label l is assigned to variable x_v, $v \in V$, if edge $t_l v$ is not part of the multicut. Since we want to assign only a single label to each variable, $k - 1$ edges joining node v and the terminal nodes have to be part of



(a) Multicut graph for (P1) (b) Multicut graph for (P2)

Fig. 2. Construction of $G' = (V', E')$ for 4×4-grid for the supervised case with $L = \{1, 2, 3\}$ **(a)** and the unsupervised case **(b)**. Red edges are part of the multicut, i.e., they separate shores. Blue edges join nodes of the same shore of the partition.

the multicut. Let $\mathcal{E}$ be the matrix of all ones and $\mathcal{I}$ be the identity matrix, both of size $k \times k$. Then the weights $w(t_l v)$, $l \in L$, $v \in V$, are given by

$$\begin{pmatrix} f_v(1) \\ \vdots \\ f_v(k) \end{pmatrix} = (\mathcal{E} - \mathcal{I}) \begin{pmatrix} w(t_1 v) \\ \vdots \\ w(t_k v) \end{pmatrix}, \qquad (\mathcal{E} - \mathcal{I})^{-1} = \frac{1}{k-1}\mathcal{E} - \mathcal{I}, \tag{5}$$

so as to represent problem (P1). For edges $uv \in E$, we use $w(uv) = \beta_{uv}$. Edges between terminal nodes have the weight $-\infty$ to enforce that all terminal nodes belong to different shores. Note that this can also be considered as a multiway cut problem [8].

For the unsupervised partitioning problem (P2), we would have to add $|V|$ terminal nodes and $|V|^2 + \frac{|V| \cdot (|V|-1)}{2}$ edges. As shown in [6], we can remove the terminal nodes from our graph without changing the optimal partition if the maximal number of possible shores is the number of nodes. This observation is crucial since it reduces the number of variables in (P3) to $|E|$. Thus, to represent (P2) as a multicut problem (P3), we just use the graph G defining (P2), i.e., $T = \emptyset$. As before, we use $w(uv) = \beta_{uv}$ for $uv \in E$.

3 Finding an Optimal Multicut

3.1 Linear Programming Formulations

Finding a minimal cost multicut is NP-hard in general [15]. However, since images induce a certain structure, there is some hope that the problems are easier to solve in practice than problems without any structure.

We use a cutting plane approach to iteratively tighten an outer relaxation of the multicut polytope. In each step we solve a problem relaxation in terms of a linear program, detect violated constraints from a pre-specified finite list, and augment the constraint system accordingly. This procedure is repeated until no more violated constraints are found.

After each iteration we obtain a lower bound as the solution of the LP and an upper bound by mapping the obtained solution to the multicut polytope (rounding, see Sec. 3.2). Finally, if the relaxed solution is not integral, we use integer linear programming and again add violated constraints after each round of optimization. Overall, we optimize the following integer linear program:

$$\min_{y\in[0,1]^{|E'|}} \sum_{e\in E'} w(e)y_e \tag{6a}$$

$$\text{s.t.} \quad \sum\nolimits_{t\in T} y_{(t,v)} = (k-1)\cdot \mathbb{I}(T\neq\emptyset) \qquad \forall v\in V \tag{6b}$$

$$y_{tu}+y_{tv} \geq y_{uv} \qquad \forall uv\in E,\ t\in T \tag{6c}$$

$$y_{tu}+y_{uv} \geq y_{tv} \qquad \forall uv\in E,\ t\in T \tag{6d}$$

$$y_{tv}+y_{uv} \geq y_{tu} \qquad \forall uv\in E,\ t\in T \tag{6e}$$

$$y_{uv} \geq \sum\nolimits_{t\in S} (y_{tu}-y_{tv}) \qquad \forall uv\in E, S\subseteq T \tag{6f}$$

$$\sum\nolimits_{e\in C\setminus\{e'\}} y_e \geq y_{e'} \qquad \forall \text{ cycles } C\subseteq E, e'\in C \tag{6g}$$

$$y_e \in \{0,1\} \qquad \forall e\in E' \tag{6h}$$

Note that not every $y\in\{0,1\}^{|E'|}$ lies inside the multicut polytope. As shown in [7], Lemma 2.2, y is a vertex of the multicut polytope if and only if

$$\sum_{e\in C} y_e \neq 1 \qquad \forall \text{ cycles } C\subseteq E', \tag{7}$$

i.e., there exist no active edges inside a shore. If T is not empty, (6b)–(6e) implies (7) [7] and if T is empty, (7) is equivalent to (6g). Therefore, any y that satisfies (6b)–(6h) is a vertex of the multicut polytope. Later, we will also consider the linear programming relaxation (6a)–(6f) introduced in [12].

3.2 Rounding Fractional Solutions

As pointed out by Călinescu et al. [12], the integrality ratio of the relaxed LP (6a)–(6f) is $\frac{3}{2}-\frac{1}{k}$. This is superior to the α-expansion algorithm and the work of Dahlhaus et al. [10], that guarantees only a ratio of $2-\frac{2}{k}$. However, while derandomized rounding procedures as suggested in [12] provide optimality bounds, they may perform worse than simple heuristics.

We therefore proceed as follows. For problem (P1), we obtain a partition by assigning each node to the terminal with minimal edge costs:

$$x_a = \arg\min\nolimits_{l\in L} y_{t_l a}, \qquad \forall a\in V. \tag{8}$$

For problems of type (P2), we determine the connected components by a union-set structure in $O(|V|+|E|)$ and assign a single label to each connected component. In short, both mappings transform a vector $y\in[0,1]^{|E'|}$ into a partition that in turn implies a valid multicut vector $y'\in \mathrm{MC}(G)$.

3.3 Finding Violated Constraints

Starting with the linear program (6a)–(6b), we add violated constraints and re-optimize the new LP. If its solution still violates constraints, we repeat the current phase, otherwise we continue with the next one.

Phase 1: Given the optimal solution for the current LP, we check (6c)–(6e) for violated constraints to be added. This requires $3 \cdot |E| \cdot |T|$ checks per iteration. If no violated constraint is found, we have the optimal solution for the LP-relaxation (6a)–(6e)[1] and continue with phase 2.

Phase 2: We search for violated constraints of the form (6f). The number of these constraints is exponential in $|T|$, but can be represented in polynomial size using slack variables [12]. To avoid additional slack variables, we include in each round for each $uv \in E$ only the subset S corresponding to the most violated constraint. If no violated constraint is found, we have determined the optimal solution for the LP-relaxation (6a)–(6f)[1] and continue with phase 3.

Phase 3: We switch to the integer program by including (6h) and check (6g) and (6c)–(6e) for violated constraints. If none exist, we have found the integer solution of (6). Otherwise, the current solution is outside the multicut polytope. In this case, we calculate a mapping to a vertex of the multicut polytope as described in Sec. 3.2 to obtain a partition of V'. When checking for (6g), we consider without loss of generality only edges $uv \in E$ for which $y_{uv} = 1$ and check if this edge is consistent with the partition. If not, this is an active edge inside a shore. We then compute the shortest path from u to v in the shore by breadth-first search and add the corresponding constraint to our ILP.

It is well known that if the cycle is chordless, the constraint is facet-defining. If there is a chord, the constraint is not facet-defining with respect to the multicut polytope but still a valid and maybe useful and facet-defining constraint with respect to the polytope relaxation. Consequently, the constraints (6c)–(6e) are facet-defining by construction and the constraints (6g) can be facet-defining.

In cases where no terminal nodes are included (cf. Sec. 2.2), the constraint set (6b)–(6f) is empty and we can start directly with phase 3. Of course, it is also possible to start with a relaxation and add constraints of the form (6g) to the relaxed problem, but then (i) shortest paths have to be computed for all edges e with $y_e > 0$, i.e., usually for all, and (ii) the shortest path search can no longer be performed by breadth-first search so that more time consuming methods have to be used.

4 Experiments

We propose two algorithms: The multicut algorithm **MCA** that optimizes (6) and **MCA-LP** (MCA until phase 2) that solves the LP-relaxation (6a)–(6f). We compare them with three state-of-the-art algorithms:

ILP-N: The commercial integer linear program solver *CPLEX 12.1* is used to solve the integer problem in the node domain, i.e., the LP-relaxation over

[1] If the solution is integral, this is the solution of the complete ILP (6).

the local polytope [2] with integer constraints. This method guarantees global optimality, and due to the progress of ILP solvers in the last years, it is applicable for small problems but does not scale. We will refer to this method in the following as ILP-N (ILP in node domain).

TRW-S: For models with grid structure, we use the tree-reweighted message passing code from the Middlebury benchmark [16], for other structures an alternative code provided by the author of TRW-S [4]. Since this code provides no stopping criteria, we run the algorithm a sufficiently large but fixed number of iterations. When we measure the runtime of TRW-S, we consider the iteration in which the best lower bound was obtained the first time.

α-expansion: The α-expansion algorithm [13] is used if the model includes no negative Potts terms, i.e., if $\beta_{uv} \geq 0$. Again, we use the implementation available in the Middlebury benchmark [16] provided by the corresponding authors.

All code is written in C/C++ and compiled with the same compiler and flags, experiments are performed on a standard desktop computer with a Pentium Dual processor (2.00 GHz) without multi-threading. The subproblems in each iteration of MCA and MCA-LP are solved by the commercial solver *CPLEX 12.1* using warm-start.

Synthetic Problems: For an evaluation of the influence of different parameters, we generate synthetic $N \times N$-grid models and vary the width of the grid (N), the number of labels (k), and the coupling strength (λ). The corresponding energy function has the form

$$J(x) = (1-\lambda)\left(\sum_{v\in V}\sum_{l\in L}\theta_{v,l}\mathbb{I}(x_v = l)\right) + \lambda\left(\sum_{uv\in E}\beta_{uv}\mathbb{I}(x_u \neq x_v)\right) \qquad (9)$$

where $\theta_{v,l}$ for all $v \in V$ and $l \in L$, and β_{uv} for all $uv \in E$ are sampled uniformly from $[-1, 1]$. The coupling strength λ adjusts the influence of the pairwise terms relative to the unary ones and is selected from $[0, 1]$. Note that since β_{uv} can be negative, common approximations for the multiway cut problem [13] can not be applied.

Fig. 3 shows the influence of changing the parameters on the mean relative optimality gap of the rounded integer solution and bound of TRW-S and MCA-LP as well as the mean runtimes for the compared methods. Reported numbers are averaged over 10 sampled models per setting. The maximal number of iterations for TRW-S was set to 5000.

Our method is faster and requires less memory than ILP-N. The objective of the ILP-N has $|V|\cdot k + |E|\cdot k^2$ variables, while MCA has only $|V|\cdot k + |E| + \frac{k(k-1)}{2}$. Furthermore, we keep the number of required constraints low by using the cutting plane scheme. In contrast to TRW-S, our method is able to compute the global optimum in all cases. However, with increasing number of variables the runtime of MCA increases faster than for TRW-S.

Image Labeling: We use the four-color images that were introduced in [17]. They contain segment boundaries in all directions and points in which three

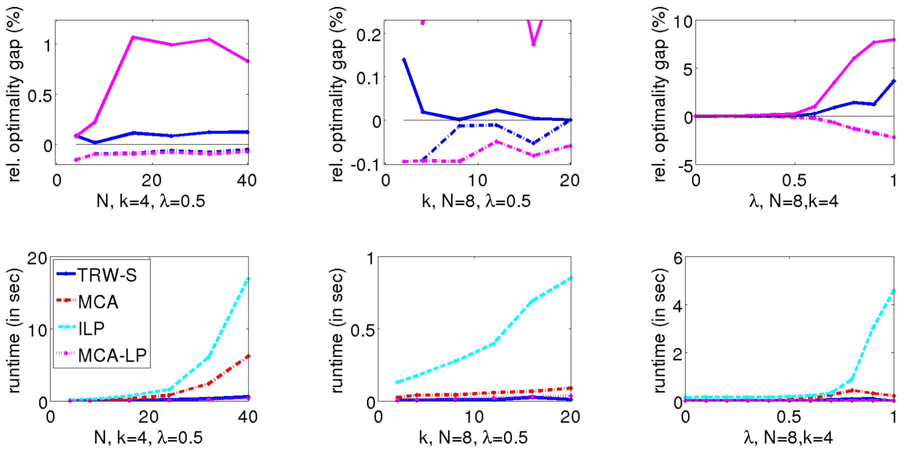


Fig. 3. Synthetic Problems (P1) We generate synthetic data according to (9). We vary the image width N, the number of labels k, and the coupling strength λ and set the other two values to the default parameters $N = 8$, $k = 4$, and $\lambda = 0.5$. The top row shows the relative optimality gap $(J(x) - J(x^{\text{opt}}))/J(x^{\text{opt}})$ for integer solutions (solid) and lower bounds (dashed) for MCA-LP and TRW-S. The bottom row shows runtimes in seconds for MCA, MCA-LP, ILP-N, and TRW-S. MCA and ILP-N always return a globally optimal solution, TRW-S and MCA-LP return a rounded integer solution and a lower bound. While the runtime for ILP-N increases in all cases, MCA scales quite good. However, with increasing number of variables the runtime of MCA grows faster than that of TRW-S. MCA-LP gives similar results as TRW-S, but produces slightly worse integer solutions, since TRW-S uses more advanced rounding methods.

classes meet. We add Gaussian noise with variance 1 to each of the three color channels independently and use the ℓ_1-norm of the difference between pixel-color (I_v) and class-color (C_l) as unary data term. As regularizer, we use a Potts prior which for indicator functions provides an anisotropic approximation of the total variation (TV) measure:

$$J(x) = \left(\sum_{v \in V} \sum_{l \in L} \|I_v - C_l\|_1 \cdot \mathbb{I}(x_v = l) \right) + \lambda \sum_{uv \in E} \mathbb{I}(x_u \neq x_v). \qquad (10)$$

We generate 50 noisy images as illustrated in Fig. 4 for different image sizes shown in Tab. 1. For a reconstruction, we minimize the energy function (10) with $\lambda = 0.5$ by MCA, ILP-N, MCA-LP, TRW-S, and α-expansion. The number of globally optimal integer solutions, the mean integrality gap[2], and the runtime for these algorithms are reported in Tab. 1. While ILP-N and MCA always find the globally optimal solution, MCA does this much faster. TRW-S finds better solutions than MCA-LP and α-expansion but all fail to find the optimal integer solution for larger problems. However, from the practical point of view,

[2] The integrality gap is the gap between the calculated and optimal integer solution.

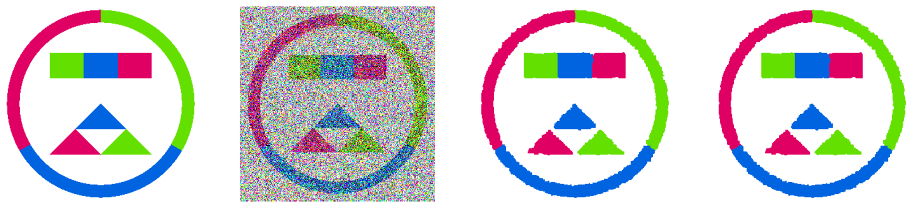


Fig. 4. Image Labeling (P1) Denoising of the noisy data is done by minimizing an energy function. Here we show the optimal result found by our multicut method and the result of α-expansion. While both look similarly good, α-expansion has not found the global optimum. The reconstructions differ in 110 pixel, e.g., border between blue and red rectangle.

the quality is similar, see Fig. 4. In Fig. 5, we illustrate the 3-phase optimization of MCA. For phase one, it requires 4.76 seconds, for phase two 0.47 seconds, and for phase three 5.42 seconds, for the particular image of width 128.

Table 1. Image Labeling (P1) Results for the labeling problem with synthetic four-color images of size $N \times N$. # denotes the number of optimal solutions found and i-gap is the average integrality gap of 50 runs. MCA and ILP-N guarantee to find the optimal integer solution. MCA makes use of the problem structure, is significantly faster than ILP-N, and requires less memory. MCA-LP, TRW-S, and α-expansion are approximative methods and do not guarantee optimal solutions. However, they are much faster and, after rounding if needed, return integer solutions close to optimality.

	MCA		ILP-N		MCA-LP			TRW-S			α-expansion		
N	#	time	#	time	#	i-gap	time	#	i-gap	time	#	i-gap	time
16	50	0.05	50	0.25	46	0.03	0.03	47	0.01	0.01	13	0.26	0.01
32	50	0.25	50	0.83	38	0.12	0.16	38	0.05	0.05	0	1.01	0.01
64	50	1.25	50	3.59	40	0.07	0.82	44	0.02	0.16	0	2.28	0.05
128	50	13.50	50	27.59	10	0.80	5.28	22	0.14	1.51	0	6.15	0.27
192	50	35.72	50	89.39	5	1.20	15.33	9	0.27	4.44	0	12.11	0.67
256	50	86.20	50	209.04	2	1.67	33.45	1	0.40	9.76	0	19.80	1.23
320	50	156.59	50	587.18	0	2.14	61.92	1	0.53	15.76	0	28.95	1.92

Unsupervised Image Partitioning: Finally, we consider the case when the number of parts in which the image should be segmented is unknown and no data term for a single pixel label is given. In this case, distances between local features codetermine the edge weights, and large distances vote for including the corresponding edges into the multicut. As a counterpart to this term, we force the total length of the boundary between segments to be small by adding a total

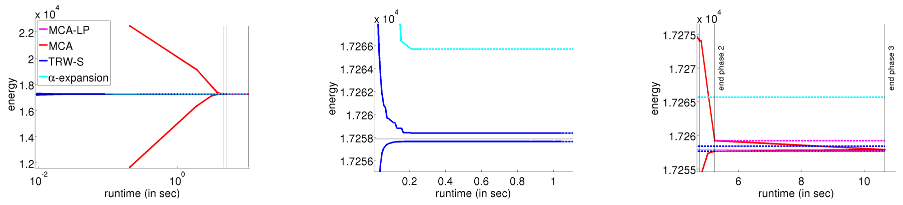


Fig. 5. Image Labeling (P1) Exemplarily, for the behavior of our multicut algorithm, we show the bounds for a four-color image of width 128. MCA-LP already leads to useful results. If after simple rounding an integrality gap remains, we enforce a boolean solution by integer constraints (MCA). This leads to better results than TRW-S and α-expansion but requires more runtime (dotted lines show objectives after termination).

variation term. Instead of working on pixel-level, we suggest to work on superpixels. This has several advantages, firstly, it makes the model robust to pixel noise, and secondly, it prunes the search space. On the other hand, it is somehow critical to use superpixels, since decisions made by a superpixel segmentation are irreversible. Therefore it is important to avoid an under-segmentation, i.e., each edge between segments should be an edge between superpixels.

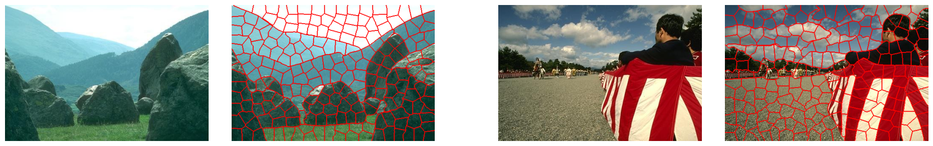
For our simple model, we use the publicly available code of Mori [11] to generate a superpixel segmentation of the image. We apply the default parameter values and omit any further data specific tuning. As similarity measure between superpixels, we use the ℓ_2-distance of the mean RGB-colors. We denote the mean color of the superpixel v by I_v and the length of the boundary between two superpixels u and v by l_{uv}. Our objective function is

$$J(x) = \sum_{uv\in E} - \left(\|I_u - I_v\|_2 \cdot l_{uv} \cdot \mathbb{I}(x_u \neq x_v)\right) + \lambda \cdot \sum_{uv\in E} \left(l_{uv} \cdot \mathbb{I}(x_u \neq x_v)\right). \quad (11)$$

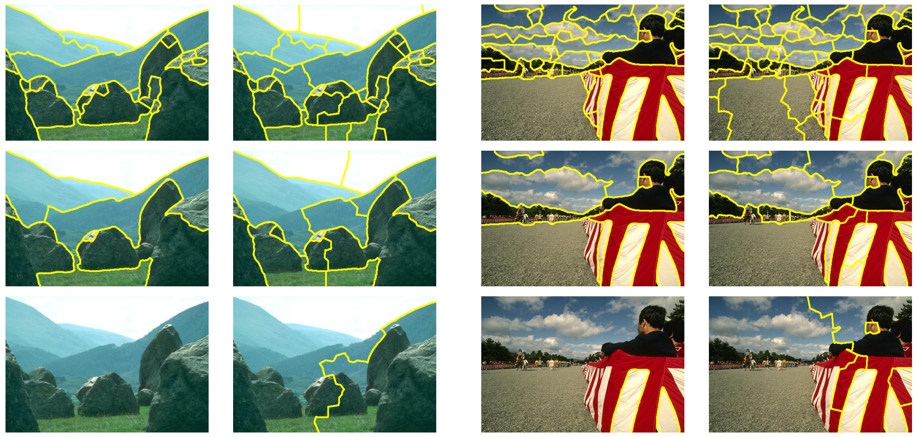
We illustrate this approach in Fig. 6. Starting from the superpixel representation, we show segmentations for three different values of λ. While MCA can deal with the maximal number of labels $k = |V|$, we set the number of labels for TRW-S sufficiently high, here $k = 100$. For the plots in the bottom row, we set k to the optimal number of segments calculated by MCA and run TRW-S. Even in this case, where the number of segments is already given, TRW-S does not find the optimal solution. Note that α-expansion can not be used since some edge weights are negative.

Segmentation results on the Berkley Segmentation Dataset: We apply the proposed method on the Berkley Segmentation Dataset (BSD) [18]. Instead of the simple model above, more complex features are used and edge weights are calculated by a random forest, see [14] for details. From the optimization point of view, this does not make any difference.

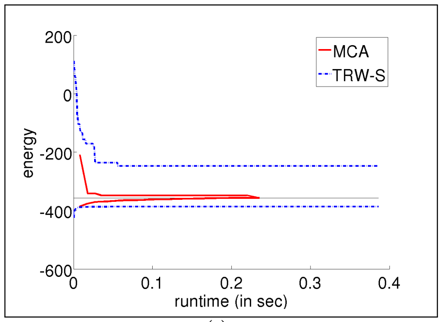
At the time of writing, the quality of the partitioning as measured by the F-score [18] in the setting where the same (optimal) parameterization of algorithms is used for all images, our method [14] ($F = 0.67$, $Pre = 0.64$, $Rec = 0.74$)



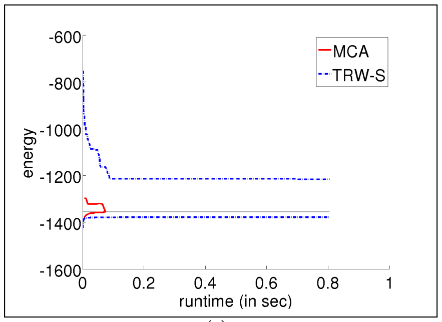
(a) Image (b) Superpixels (c) Image (d) Superpixels



(e) MCA (f) TRW-S (g) MCA (h) TRW-S



(i)



(j)

Fig. 6. Unsupervised Partitioning (P2) To deal with pixel-noise and scale to large images, we use a superpixel representation **(b)** and **(d)** of the images and apply a simple model to join superpixels based on their similarity and border length, cf. (11). We compare our method MCA with TRW-S and restrict the number of labels for TRW-S to a sufficiently large number, here $k = 100$. MCA can deal with the maximal number of superpixels $k = |V|$ and selects the optimal number implicitly in the optimization process. The images show the resulting segmentation from low (top) to high (bottom) values of λ. TRW-S never finds the global optimum and tends to include additional segments. If we set the number of labels for TRW-S for a fixed λ (corresponding to the middle segmentation) to the optimal number of segments found by MCA, TRW-S is still not able to solve this problem and converges to a non-optimal fixpoint as shown in **(i)** and **(j)**. Note that if we increase the number of labels, TRW-S becomes significantly slower.

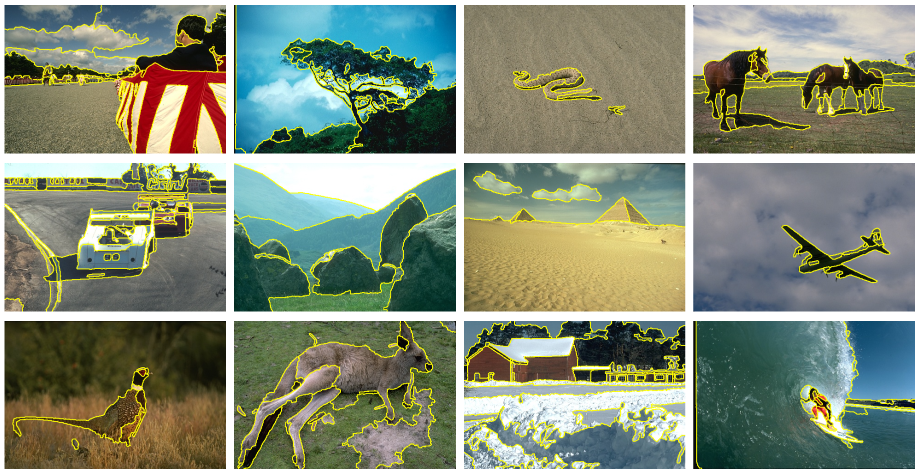


Fig. 7. Unsupervised Partitioning (P2) Exemplary results on the BSD

is on a par with [19] ($F = 0.67$, $Pre = 0.66$, $Rec = 0.69$) with a higher recall but lower precision. Note that no other algorithm that produces *closed* contours has a better F-score. Pure boundary detectors that need not produce closed contours [20, 21] still have a higher F-score. From the viewpoint of polyhedral theory, these solutions lie outside the multicut polytope and do not correspond to any partition.

5 Conclusions

We present an image partitioning framework for supervised and unsupervised scenarios together with a novel optimization algorithm (MCA) that solves these problems to optimality. We show that this framework is appealing and that MCA outperforms state-of-the-art optimization methods in the unsupervised case, i.e., when no unary data term is included.

In general, it provides a more compact linear program than methods working in the node domain, i.e., it has less variables. MCA calculates an optimal solution by using cutting plane and integer programming techniques, and in its variant MCA-LP an approximative solution by solving a polynomial size LP.

Even without any post-processing, our results on the Berkley Segmentation Dataset and Benchmark (BSD) are on par with the best-performing methods that ensure closed contours.

Acknowledgement. This work has been supported by the German Research Foundation (DFG) within the programme "Spatio-/Temporal Graphical Models and Applications in Image Analysis", grant GRK 1653.

References

1. Kleinberg, J., Tardos, É.: Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In: FOCS. (1999)
2. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. FTML **1** (2008) 1–305
3. Sontag, D., Jaakkola, T.: New outer bounds on the marginal polytope. In: NIPS. (2007)
4. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. TPAMI **28** (2006) 1568–1583
5. Deza, M., Grötschel, M., Laurent, M.: Complete descriptions of small multicut polytopes. In: Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift. American Mathematical Society (1991)
6. Chopra, S., Rao, M.R.: On the multiway cut polyhedron. Networks **21** (1991) 51–89
7. Chopra, S., Rao, M.R.: The partition problem. Mathematical Programming **59** (1993) 87–115
8. Boykov, Y., Veksler, O., Zabih, R.: Markov random fields with efficient approximations. In: CVPR. (1998)
9. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? TPAMI **26** (2004) 147–159
10. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiway cuts (extended abstract). In: STOC. (1992)
11. Mori, G. (`http://www.cs.sfu.ca/~mori/research/superpixels/`)
12. Călinescu, G., Karloff, H., Rabani, Y.: An improved approximation algorithm for multiway cut. JCSS **60** (2000) 564–574
13. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. TPAMI **23** (2001) 1222–1239
14. Andres, B., Kappes, J.H., Beier, T., Köthe, U., Hamprecht, F.: Probabilistic image segmentation with closedness constraints. (submitted to ICCV 2011)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1979)
16. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. TPAMI **30** (2008) 1068–1080
17. Lellmann, J., Kappes, J., Yuan, J., Becker, F., Schnörr, C.: Convex multi-class image labeling by simplex-constrained total variation. In: SSVM. (2009)
18. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV. (2001)
19. Arbeláez, P.: Boundary extraction in natural images using ultrametric contour maps. In: CVPRW. (2006)
20. Maire, M., Arbeláez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: CVPR. (2008)
21. Ren, X.: Multi-scale improves boundary detection in natural images. In: ECCV. (2008)