Int J Comput Vis (2015) 115:155–184
DOI 10.1007/s11263-015-0809-x

A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems

Jörg H. Kappes · Bjoern Andres · Fred A. Hamprecht · Christoph Schnörr · Sebastian Nowozin · Dhruv Batra · Sungwoong Kim · Bernhard X. Kausler · Thorben Kröger · Jan Lellmann · Nikos Komodakis · Bogdan Savchynskyy · Carsten Rother

Received: 2 April 2014 / Accepted: 9 February 2015 / Published online: 14 March 2015

Abstract Szeliski et al. published an influential study in 2006 on energy minimization methods for Markov random fields. This study provided valuable insights in choosing the best optimization technique for certain classes of problems. While these insights remain generally useful today, the phenomenal success of random field models means that the *kinds* of inference problems that have to be solved changed significantly. Specifically, the models today often include higher order interactions, flexible connectivity structures, large label-spaces of different cardinalities, or learned energy tables. To reflect these changes, we provide a modernized and enlarged study. We present an empirical comparison of more than 27 state-of-the-art optimization techniques on a corpus of 2453 energy minimization instances from diverse applications in computer vision. To ensure reproducibility, we evaluate all methods in the OpenGM 2 framework and report extensive results regarding runtime and solution quality. Key insights from our study agree with the results of Szeliski et al. for the types of models they studied. However, on new and challenging types of models our findings disagree and suggest that polyhedral methods and integer programming solvers are competitive in terms of runtime and solution quality over a large range of model types.

Keywords Discrete graphical models · Combinatorial optimization · Benchmark

Communicated by K. Ikeuchi.

Electronic supplementary material The online version of this article (doi:10.1007/s11263-015-0809-x) contains supplementary material, which is available to authorized users.

J. H. Kappes (✉) · F. A. Hamprecht · C. Schnörr · B. X. Kausler · T. Kröger
Heidelberg University, Speyerer Strasse 6, 69115 Heidelberg, Germany
e-mail: kappes@math.uni-heidelberg.de

B. Andres
Combinatorial Image Analysis, Max Planck Institute for Informatics, Campus E1.4, 66123 Saarbrücken, Germany

S. Nowozin
Machine Learning and Perception, Microsoft Research, 21 Station Road, Cambridge CB12FB, United Kingdom

D. Batra
Virginia Tech, 302 Whittemore Hall, Blacksburg, VA 24061, USA

S. Kim
Qualcomm Research Korea, 15th FL., POBA Gangnam Tower, 343, Hakdong-ro, Gangnam-gu, 135-820 Seoul, Republic of Korea

1 Introduction

Discrete energy minimization problems, in the form of factor graphs, Markov or conditional random field models (MRF/CRF) are a mainstay of computer vision research. Their applications are diverse and range from image denoising, segmentation, motion estimation, and stereo, to object recognition and image editing. To give researchers some guidance as to which optimization method is best suited for

J. Lellmann
DAMTP, University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, United Kingdom

N. Komodakis
Universite Paris-Est, Ecole des Ponts ParisTech, Cité Descartes, 6-8 Avenue Blaise Pascal, 77455 Champs-sur-Marne, France

B. Savchynskyy · C. Rother
Dresden University of Technology, 01062 Dresden, Germany

their models, Szeliski et al. (2008) conducted a comparative study on 4-connected grid models. Along with the study, they provided a unifying software framework that facilitated a fair comparison of optimization techniques. The study was well-received in computer vision community and has till date been cited more than 700 times.

Since 2006 when the study was published, the field has made rapid progress. Modern vision problems involve more complex models, larger datasets and use machine learning techniques to train model parameters.

To summarize, these changes gave rise to challenging energy minimization problems that fundamentally differ from those considered by Szeliski et al. In particular, in Szeliski et al. (2008) the models were restricted to 4-connected grid graphs with unary and pairwise factors only, whereas modern ones include arbitrary structured graphs and higher order potentials.

It is time to revisit the study (Szeliski et al. 2008). We provide a modernized comparison, updating both the problem instances and the inference techniques. Our models are different in the following four aspects:

1. Higher order models, e.g. factors of order up to 300;
2. Models on “regular” graphs with a denser connectivity structure, e.g. 27-pixel neighborhood, or models on “irregular” graphs with spatially non-uniform connectivity structure;
3. Models based on superpixels with smaller number of variables;
4. Matching and image partitioning models without unary terms. Later also with unknown number of classes.

Inference methods have changed since 2006 as well, often as a response to the development of challenging models. The study of Szeliski et al. (2008) compared the performance of the state of the art at that time, represented by primal move-making methods, loopy belief propagation, a tree-reweighted message passing, and a set of more traditional local optimization heuristics like iterated conditional modes (ICM).

We augment this set with recent updates of the move-making and local optimization methods, methods addressing higher order models, and polyhedral methods considering the energy minimization as an (Integer) Linear Program.

1.1 Contributions

We provide a modernized experimental study of energy minimization methods. Our study includes the cases and algorithms studied by Szeliski et al. (2008), but significantly expand it in the scope of both used inference methods and considered models. Both the methods and the considered models are implemented and stored within a single uniform multi-platform software framework, OpenGM 2 (Andres et al. 2014). Together with results of our evaluation they are available on-line on the project web-page (Andres et al. 2014).

Such a unification provides researchers with an easy access to the considered wide spectrum of modern inference techniques. Our study suggests which techniques are suited for which models. The unification boosts development of novel inference methods by providing a set of models for their comprehensive evaluation.

1.2 Related Inference Studies

Apart from the study of Szeliski et al. (2008), a number of recent articles in CV have compared inference techniques for a small specialized class of models, such as Alahari et al. (2010), Komodakis and Paragios (2008), Kolmogorov and Rother (2006). Unfortunately, the models and/or inference techniques are often not publicly available. Even if they were available, the lack of a flexible software-framework which includes these models and optimization techniques makes a fair comparison difficult. Closely related is the smaller study of Andres et al. (2010) that uses the previous and now deprecated version of OpenGM. It compares several variants of message passing and move making algorithms for higher order models on mainly synthetic models. In contrast to Andres et al. (2010), we consider a broader range of inference methods, including polyhedral ones, and a bigger number of non-synthetic models.

Outside computer vision, the Probabilistic Inference Challenge (PIC) by Elidan and Globerson (2011) covers a broad class of models used in machine learning. We include the leading optimization techniques and some challenging problems of PIC in our study.

Furthermore, a previous and shorter version of the our study was published at a conference as Kappes et al. (2013).

1.3 Key Insights and Suggested Future Research

In comparison with Szeliski et al. (2008), perhaps the most important new insight is that recent, advanced polyhedral LP and ILP solvers are competitive for a wide range of problems in computer vision. For a considerable number of instances, they are able to achieve global optimality. For some problems they are even superior or on a par with approximative methods in terms of overall runtime. This is true for problems with a small number of labels, variables and factors of low order that have a simple form. But even for some problems with a large number of variables or complex factor form, specialized ILP and LP solvers can be applied successfully and provide globally optimal solutions. For problems with many variables and cases in which the LP relaxation is not tight, polyhedral methods are often not competitive. In this regime, primal move-making methods typically achieve the best results, which is consistent with the findings of Szeliski et al. (2008).

Our new insights suggest two major areas for future research. Firstly, in order to capitalize on existing ILP solvers, small but expressive models, e.g. superpixels, coarse-to-fine approaches, or reduction of the model size by partial optimality, should be explored and employed. Secondly, our findings suggest that improving the efficiency and applicability of ILP and LP solvers should and will remain an ongoing active area of research.

A more detailed synopsis and discussion of our insights will be given in Sect. 7.

The present study is solely devoted to structured energy minimization, that is to MAP estimation from a Bayesian viewpoint. From a statistical viewpoint, inference methods that explore posterior distributions beyond mere point estimation would be preferable, but are too expensive for most large-scale applications of current research in computer vision. Recent works (Papandreou and Yuille 2011; Orabona et al. 2014) exploits *multiple MAP inference* in order to get closer to this objective in a computationally feasible way. This development underlines too the importance of research on energy minimization as assessed in this paper.

2 Graphical Models

We assume that our discrete energy minimization problem is specified on a factor graph $G = (V, F, E)$, a bipartite graph with finite sets of variable nodes V and factors F, and a set of edges $E \subset V \times F$ that defines the relation between those (Koller and Friedman 2009; Nowozin and Lampert 2011). The variable x_a assigned to the variable node $a \in V$ lives in a discrete label-space X_a and notation X_A, $A \subset V$, stands for a Cartesian product $\otimes_{a\in A} X_a$. Each factor $f \in F$ has an associated function $\varphi_f : X_{ne(f)} \to \mathbb{R}$, where $ne(f) := \{v \in V : (v, f) \in E\}$ defines the variable nodes connected to the factor f. The functions φ_f will also be called *potentials*.

We define the order of a factor by its degree $|ne(f)|$, e.g. pairwise factors have order 2, and the order of a model by the maximal degree among all factors.

The energy function of the discrete labeling problem is then given as

$$J(\mathbf{x}) = \sum_{f\in F} \varphi_f(\mathbf{x}_{ne(f)}), \tag{1}$$

where the assignment of the variable $\mathbf{x}$ is also known as the labeling. For many applications the aim is to find a labeling with minimal energy, i.e. $\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}} J(\mathbf{x})$. This labeling is a maximum-a-posteriori (MAP) solution of a Gibbs distribution $p(\mathbf{x}) = \exp\{-J(\mathbf{x})\}/Z$ defined by the energy $J(\mathbf{x})$. Here, Z normalizes the distribution.

It is worth to note that we use factor graph models instead of Markov random field models (MRFs), also known as undirected graphical models, or conditional random fields (CRF). The reason is that factor graphs represent the structure of the underlying problem in a more precise and explicit way than MRFs, cf. Koller and Friedman (2009). MRFs and CRFs express the conditional independence assumptions, which implies a factorization in maximum cliques only under some technical conditions (Koller and Friedman 2009; Lauritzen 1996). Unfortunately, these terms widely have been used a bit sloppy in computer vision. Often (hyper-) graphs have been used to describe a factorization over (hyper-) edges but models have been called MRF/CRF - even those cannot express this fine factorization. Examples of such models in the present study are among others *scene-decomp*, *matching* or *color-seg*.

2.1 Categorization of Models

One of the main attributes we use for our categorization is the *meaning* of a variable, i.e. if the variable is associated with a pixel, superpixel or something else. The *number* of variables is typically related to this categorization.

Another modeling aspect is the *number* of labels the variable can take. Note that the size of the label-space restricts the number of methods that are applicable, e.g. QPBO or MCBC can be used when each variable takes no more than two values. We also classify models by *properties* of the factor graph, e.g. average number of factors per node, mean degree of factors, or structure of the graph, e.g. grid structure. Finally, the *properties/type* of the functions embodied by the factors are of interest, since for some subclasses specialized optimization methods exists, e.g. metric energies (Szeliski et al. 2008) or Potts functions (Kappes et al. 2011).

2.2 OpenGM 2: A General Modeling Framework

For this comparison we use OpenGM 2 (Andres et al. 2014) a C++ library for discrete graphical models. It provides support for models of arbitrary order, allows the use of arbitrary functions and sharing of functions between factors. OpenGM decouples optimization from the modeling process and allows to store models in the scientific HDF5 format. That is, all model instances stored into a single file and no application specific code has to be released/used to make models available or do evaluation on those. Within OpenGM 2 we provide several own implementations of inference methods in native C++ code as well as wrappers for several existing implementations, and making those available in a unified framework.

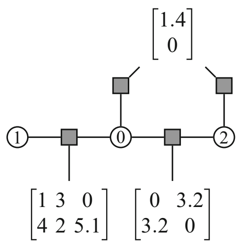
$\begin{bmatrix}1.4\\0\end{bmatrix}$
1
0
2
$\begin{bmatrix}1&3&0\\4&2&5.1\end{bmatrix}$
$\begin{bmatrix}0&3.2\\3.2&0\end{bmatrix}$

Fig. 1 Toy-model constructed by Algorithm 1. The model include 3 variables with 2, 3, and 2 labels. The unary factors of variable 0 and 2 share the same function

Making new model instances or optimization methods available within this framework is rather simple: Fig. 1 illustrates a toy problem with 3 variables having 2, 3, and 2 states respectively. The two first order factors represent the same function mapped to different variables. Algorithm 1 shows pseudo code for its construction in OpenGM 2. In the first step (lines 1–2) a graphical model with the variables and corresponding label space is set up. When constructing a model we also fix the operation, in this case addition (+), which couples the single terms to a global objective. Then (line 3–5) the three functions are added. Note that OpenGM 2 allows special implementations for functions, e.g. for Potts functions. In the last step (line 6–9) factors are added to the model and connected to variables (first parameter) and functions (second parameter). Finally the model is stored to file (line 10). OpenGM 2 allows to reuse functions for different factors, which saves a lot of memory if e.g. the same regularizers are used everywhere. We call this concept *extended factor graphs*.

Algorithm 1 Creating a model in OpenGM 2

```
1: statespace = [ 2 3 2 ]
2: gm = createModel<+>(statespace)
3: fid1 = gm.addFunction( [2], [1.4 0] )
4: fid2 = gm.addFunction( [2 3], [1 3 0; 4 2 5.1] )
5: fid3 = gm.addFunction( [2 2], Potts(0,3.2) )
6: gm.addFactor( [0] , fid1 )
7: gm.addFactor( [2] , fid1 )
8: gm.addFactor( [0,1] , fid2 )
9: gm.addFactor( [0,2] , fid3 )
10: storeModel( gm ,"model.h5")
```

Given a problem defined on an (extended) factor graph one can find the labeling with the (approximately) lowest energy. Algorithm 2 illustrates how to approach this within OpenGM 2. After loading a model (line 1), one initializes an inference object (lines 2–3). Inference is always done with respect to *an accumulative* operation such as minimum (min) or maximum. Optionally, one can set up a visitor object (line 4), which will take care of logging useful information during optimization. The core inference is done within the method "infer" (line 5). After that one can get the inferred labeling (line 6). Additionally, the visitor can give informations about the progress the method has made over time.

Algorithm 2 Inference with OpenGM 2

```
1: gm = loadModel("model.h5")
2: InferenceMethod<min,+>::Parameter para
3: InferenceMethod<min,+> inf( gm , para )
4: Visitor vis
5: inf.infer(vis)
6: x = inf.arg()
7: vis.journalize("log.h5")
```

For a new inference method, one needs to implement only constructor and methods infer() and arg(). Tutorials for the supported languages can be found on the project website (Andres et al. 2014).

3 Benchmark Models

Table 1 gives an overview of the models summarized in this study. Note that, some models have a single instance, while others have a larger set of instances which allows to derive some statistics. We now give a brief overview of all models. Further specifics in connection with inference will be discussed in Sect. 6. A detailed description of all models is available online.

3.1 Pixel-Based Models

For many low-level vision problems it is desirable to make each pixel a variable in the model. A typical property of such models is that the number of variables is large. For 2D images, where variables are associated to pixels in a 2D lattice, a simple form of a factor graph model connects each pixel with its four nearest neighbors (Fig. 2a) using a pairwise energy. This simple form is popular and was the sole subject of the study (Szeliski et al. 2008). In our study we incorporated the models **mrf-stereo**, **mrf-inpainting**, and **mrf-photomontage** from Szeliski et al. (2008) with three, two and two instances, respectively. The pairwise terms of these models are truncated convex functions on the label space for mrf-stereo and mrf-inpainting and a general pairwise term for mrf-photomontage.

Additionally, we used three models which have the same 4-connected structure. For inpainting problems (Lellmann and Schnörr 2011) **inpainting-N4** and color segmentation problems (Lellmann and Schnörr 2011) **color-seg-N4**[1] the

[1] The inpainting-N4/8 and color-seg-N4/8-models were originally used in variational approaches together with total variation regularizers (Lellmann and Schnörr 2011). A comparison with variational models is beyond the scope of this study.

Table 1 List of datasets used in the benchmark

Model name	#	Variables	Labels	Order	Structure	Function type	Loss-function	References
Pixel								
Mrf-stereo	3	∼100,000	16–60	2	Grid-N4	TL1, TL2	PA2	Szeliski et al. (2008)
Mrf-inpainting	2	∼50,000	256	2	Grid-N4	TL2	CE	Szeliski et al. (2008)
Mrf-photomontage	2	∼500,000	5, 7	2	Grid-N4	Explicit	–	Szeliski et al. (2008)
Color-seg-N4/N8	2 × 9	76,800	3, 12	2	Grid-N4/N8	Potts$^+$	–	Lellmann and Schnörr (2011)
Inpainting-N4/N8	2 × 2	14,400	4	2	Grid-N4/N8	Potts$^+$	–	Lellmann and Schnörr (2011)
Object-seg	5	68,160	4–8	2	Grid-N4	Potts$^+$	PA	Alahari et al. (2010)
Color-seg	3	21,000, 424,720	3, 4	2	Grid-N8	Potts$^+$	–	Alahari et al. (2010)
Dtf-chinese-char	100	∼ 8000	2	2	Sparse	Explicit	PA	Nowozin et al. (2011)
Brain-3/5/9mm	3 × 4	400000–2000000	5	2	Grid-3D-N6	Potts$^+$	–	Cocosco et al. (1997)
Inclusion	1	1024	4	**4**	Grid-N4 + X	g-potts	PA	Kappes et al. (2013)
Superpixel								
Scene-decomp	715	∼ 300	8	2	Sparse	Explicit	PA	Gould et al. (2009)
Geo-surf-seg-3	300	∼ 1000	3	**3**	Sparse	Explicit	PA	Gallagher et al. (2011), Hoiem et al. (2011)
Geo-surf-seg-7	300	∼ 1000	7	**3**	Sparse	Explicit	PA	Gallagher et al. (2011), Hoiem et al. (2011)
Partition								
Correlation-clustering	715	∼ 300	∼300	∼**300**	Sparse	g-potts*	VOI	Kim et al. (2011)
Image-seg	100	500–3000	500–3000	2	Sparse	Potts*	VOI	Andres et al. (2011)
Image-seg-o3	100	500–3000	500–3000	**3**	Sparse	g-potts*	VOI	Andres et al. (2011)
Knott-3d-seg-150/300/450	3x8	∼ 800, 5000, 16000	∼800–16000	2	Sparse	Potts*	VOI	Andres et al. (2012b)
Modularity-clustering	6	34-115	34-115	2	Full	Potts*	–	Brandes et al. (2008)
Other								
Matching	4	∼ 20	∼20	2	Full or sparse	Explicit	MPE	Komodakis and Paragios (2008)
Cell-tracking	1	41134	2	**9**	Sparse	Explicit	–	Kausler et al. (2012)
Protein-folding	21	33-1972	81-503	2	Full	Explicit	–	Yanover et al. (2008), Elidan and Globerson (2011)
Protein-prediction	8	14258-14441	2	**3**	Sparse	Explicit	–	Jaimovich et al. (2006), Elidan and Globerson (2011)

Listed properties are number of instances (#), variables and labels, the order and the underlying structure. Furthermore, special properties of the functions are listed; truncated linear and squared (TL1/2), Potts function with positive (potts$^+$) and arbitrary (potts*) coupling strength, its generalization to higher order (g-potts) and functions without special properties (explicit). For some models we have an additional loss function commonly used for this application, namely: 2pixel-accuracy (PA2), color-error (CE), pixel-accuracy (PA), variation of information (VOI), and geometric error (MPE). For the other models no ground truth or loss function was available
Higher order models are in bold

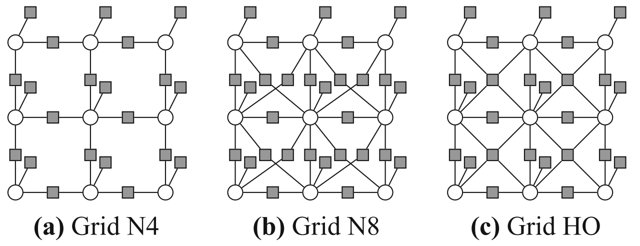

Fig. 2 Common **pixel based models** are grid structured with respect to a four (**a**) or eight (**b**) neighborhood-structure. Some models also use couplings to remoter variables. Also higher order structures (**c**) have been successfully used for modeling

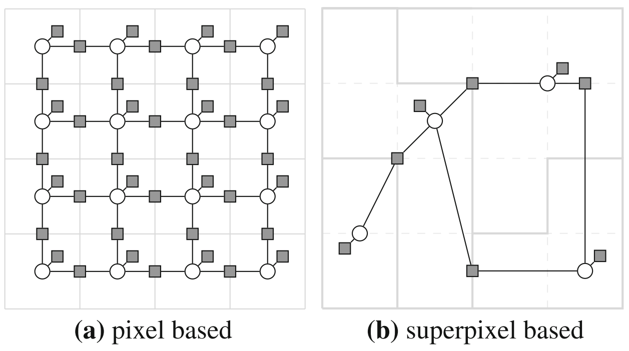

Fig. 3 While in pixel based models (**a**) each variable/node is assigned to one pixel, in superpixel based models (**b**) each variable/node is assigned to a set of pixels forming the superpixel. As a consequence the number of variables becomes smaller and the graph-structure is usually no longer regular

task is to assign each pixel one color out of a preselected finite set. For the object segmentation problems (Alahari et al. 2010) **object-seg** labels correspond to predefined object classes. Each single instance has the same small set of labels for all its variables and Potts terms are used to penalize the boundary length between different classes. In inpainting-N4 and color-seg-N4 this regularizer is the same for all factors. In object-seg, it depends on the image-gradient. The unary terms measure the similarity to predefined class-specific color models.

From a modeling point the 4-neighborhood is quite restricted, important relations cannot be modeled by a simple grid structure in many applications. Therefore, models with denser structures (Fig. 2b) as well as higher order models (Fig. 2c) have been introduced in the last decade. For instance, better approximations of the boundary regularization were obtained by increasing the neighborhood (Boykov 2003). The datasets **inpainting-N8** and **color-seg-N8** (Lellmann and Schnörr 2011) include the same data-term as inpainting-N4 and color-seg-N4 but approximate the boundary length using an 8-neighborhood (Fig. 2b). Another dataset with an 8-neighborhood and Potts terms depending on the image-gradient is **color-seg** by Alahari et al. (2010).

We also use a model with a 6-neighborhood connectivity structure in a 3D-grid. It is based on simulated 3D MRI-brain data (Cocosco et al. 1997), where each of the 5 labels represent color modes of the underlying histogram and boundary length regularization (Boykov 2003). We let the simulator generate 4 scans for 3 different slice-thickness. These models are denoted by **brain-9 mm**, **brain-5 mm**, and **brain-3 mm**. These replace the brain dataset used in Kappes et al. (2013).

We also consider the task of inpainting in binary images of Chinese characters, **dtf-chinesechar** (Nowozin et al. 2011). Potentials, related to the factors of these models, are learned from a decision tree field. Although each variable has only two labels, it is connected via pairwise factors to 27 other variables selected during learning from a 17 × 17 window. Such an increased connectivity and discriminative learned potential make the resulting inference problem highly non-sub-modular and therefore challenging.

Pixel based models that include higher-order terms are really rare. The main reason for this is that pixel based models usually have a large number of variables, such that a systematical enrichment of the model with higher-order terms often becomes intractable. To cope with this, higher order models in computer vision often include terms that can be reduced linearly to a second order model by including a small number of auxiliary variables and additional factors, e.g. labeling costs (Delong et al. 2012) and P^n-Potts (Kohli et al. 2009) terms. Our benchmark includes one model called **inclusion** that has a fourth-order term (Fig. 2c) for each junction of four pixels that penalizes a splitting of the boundary, as suggested in Kappes et al. (2013).

3.2 Superpixel-Based Models

In superpixel-based models, all pixels belonging to the same superpixel are constrained to have the same label, as shown in Fig. 3. This reduces the number of variables in the model and allows for efficient inference even with more complex, higher order factors. However, the grouping of pixels is an irreversible decision and it is hard to treat grouped pixels differently later on.

The models we consider in this regime are used for semantic image segmentation. The number of superpixels vary between 29 and 1133 between the instances. In the **scene-decomposition**-dataset (Gould et al. 2009) every superpixel has to be assigned to one of 8 scene classes. Pairwise factors between neighboring superpixels penalize unlikely label-pairs. The datasets **geo-surf-3** and **geo-surf-7** (Gallagher et al. 2011; Hoiem et al. 2011) are similar but have additional third-order factors that enforce consistency of labels for three vertically neighboring superpixels.

3.3 Partition Models

Beyond classical superpixel models, this study also considers a recent class of superpixel models which aim at partitioning an image without any class-specific knowledge (Kim et al. 2011; Andres et al. 2011, 2012a, b). These use only similarity measures between neighbored regions encoded by (generalized) Potts functions with positive and negative coupling strength. Since the partition into isolated superpixels is a feasible solution, the label space of each variable is equal to the number of variables of the model, and therefore typically very large, cf. Table 1. That is why many solvers switch to a binary edge-label problem together with huge system of constraints, cf. Kappes et al. (2013).

For unsupervised image segmentation we consider the probabilistic image partition dataset **image-seg** (Andres et al. 2011), which contains factors between pairs of superpixels, and its extension **image-seg-o3** (Andres et al. 2011) that also uses a learned third-order prior on junctions. The 3D neuron segmentation model 3d-neuron-seg used in Kappes et al. (2013) are replaced by 3 datasets **knott-3d-seg-150**, **knott-3d-seg-300**, and **knott-3d-seg-450** with 8 instances each. The number denotes the edge length of the 3D volume. These datasets allow a better evaluation of the scalability of methods and are generated from sub-volumes of the model described in Andres et al. (2012a, b).

The hyper-graph image segmentation dataset **correlation-clustering** (Kim et al. 2011) includes higher order terms that favor equal labels for sets of superpixels in their scope if those are visually similar. These sets are preselected and incorporate higher level proposals in the objective. The partition models are completed by some network clustering problems **modularity-clustering** (Brandes et al. 2008) from outside the computer vision community. Contrary to the previous ones, these instances include a fully connected structure.

3.4 Other Models

We also consider computer vision applications that assign variables to keypoints in the image-data.

The first model deals with the non-rigid point matching problem (Komodakis and Paragios 2008) **matching**. Given two sets of keypoints the task is to match these such that the geometric relations are kept. The model instances include no unary terms, whereas the pairwise terms penalize the geometric distortion between pairs of points in both point-sets.

The second application is **cell-tracking** (Kausler et al. 2012). Variables correspond to the assignments of cells in a video sequence, which need to be consistent over time. Since a track can either be active or dormant, the variables are binary. Higher-order factors are used to model the likelihood of a "splitting" and "dying" event of a cell.

Finally, we include models from outside computer vision, taken from the Probabilistic Inference Challenge (PIC) (Elidan and Globerson 2011) into the corpus. The **protein-folding** instances (Yanover et al. 2008) have a moderate number of variables, but are fully connected and have for some variables huge label spaces. The **protein-prediction** instances (Jaimovich et al. 2006) include sparse third-order binary models. For both dataset we include only the hardest instances, which were used also in Elidan and Globerson (2011).

4 Inference Methods

We evaluate a large number of different inference methods. The selection of methods is representative of the state of the art in the field. As shown in Fig. 4, we can cluster the methods into four groups: (i) global optimal methods that provide optimal solution if no time- and memory-constraints are given, (ii) methods solving linear programming relaxations, providing a lower bound, (iii) move-making methods that iteratively improve a feasible labeling, and (iv) message passing methods that perform local calculations between nodes. Furthermore, subsets of methods leverage max-flow problems or dual representations; we call these methods max-flow- and DD-based methods, respectively. In this study we do not consider inference algorithms based on Monte Carlo simulation; only in one data set (dtf-chinesechar) we report results obtained from simulated annealing. In general these methods converge slowly and may not be practical for most computer vision applications. In addition they often require careful tuning of parameters such as the temperature schedule. As Monte Carlo methods are not very popular in computer vision, we are not able to give a fair comparison and skip them in our current study. We now give a brief overview of the considered methods.

4.1 Polyhedral Methods

A large class of algorithms solves a linear programming relaxation (LP) of the discrete energy minimization problem. An advantage of these methods is that they also provide a lower bound for the optimum, but on the other hand can converge to non-integer solutions, which need to be rounded to an integer solution.

Perhaps the most commonly used relaxation is the LP relaxation over the *local polytope* (Wainwright et al. 2005; Schlesinger 1976; Werner 2007). We can solve small instances using off-the-shelf LP-solvers e.g. CPLEX (IBM, ILOG CPLEX 2013) as used in **LP-LP** (Andres et al. 2012). For large problems this is no longer possible and special solvers have been proposed that optimize a dual formulation of the problem. A famous example is the block-coordinate-

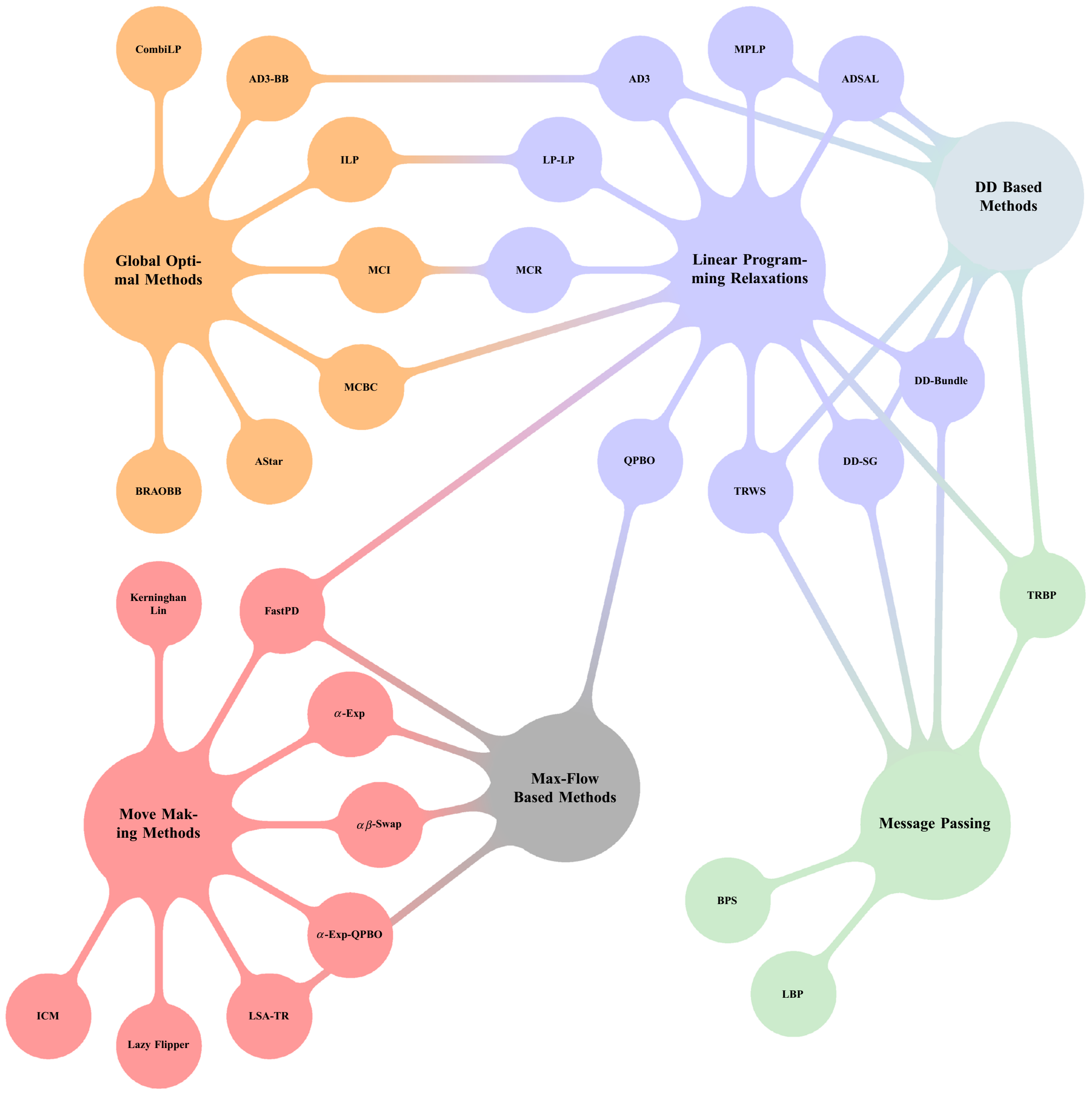

Fig. 4 The inference methods used in this benchmark can be roughly grouped into four classes. These are (1) methods based on **linear programming**, (2) methods providing **global optimal** solutions, which are often strongly related to linear programming, (3) methods based on **move-making** procedures which iteratively improves the labeling, and (4) methods based on **message passing**—often motivated by linear programming and variational optimization. Some methods make use of max-flow methods for fast optimization of binary (sub-)problems or based on the dual decomposition framework, which is also sketched in the diagram. A fifth class of methods are methods based on **sampling**, which are not covered in this study since they are rarely used in computer vision. For hard models they might perform reasonable, with a certain amount of tuning of involved hyper-parameters and sampling procedures, as shown for the dtf-chinesechar model. For some of the inference algorithms we use different implementations. Even when algorithmically identical, they often vary in speed because of implementation differences and specialized algorithms. We always try to use the fastest one and use the prefix ogm- and mrf- to state that the used implementation was Andres et al. (2012) or Szeliski et al. (2008), respectively. For other methods the core of the implementation has been provided by the original authors of the methods and we wrapped them within OpenGM 2

ascent method **TRWS** (Kolmogorov 2006), which, however, can get stuck in suboptimal fix points.

In contrast, subgradient methods (Komodakis et al. 2011; Kappes et al. 2012) based on dual decomposition (DD) (Komodakis et al. 2011; Guignard and Kim 1987) with adaptive stepsize **DD-SG-A** and bundle methods (Kappes et al. 2012) with adaptive **DD-Bundle-A** or heuristic **DD-Bundle-H** stepsize are guaranteed to converge to the optimum of the relaxed dual[2]. In both cases primal integer solutions are reconstructed from the subgradients. As dual decomposition methods can be sensitive to the stepsize, we evaluate different stepsize-rules. While a more detailed evaluation is beyond the scope of this work, we consider beside our base line stepsize-rule also stepsizes scaled by 10 and 10^{-1}, denoted with post-fix $+$ and $-$, respectively.

Other methods based on dual decomposition are Alternating Directions Dual Decomposition **AD3** (Martins et al. 2011), the Adaptive Diminishing Smoothing ALgorithm **ADSAL** (Savchynskyy et al. 2012), which smooth the dual problem to avoid local suboptimal fix-points, and Max-Product Linear Programming **MPLP** (Globerson and Jaakkola 2007). For MPLP an extension **MPLP-C** (Sontag et al. 2012) exists that iteratively adds violated constraints over cycles of length 3 and 4 to the problem. This leads to a tighter relaxations than the local polytope relaxation.

Algorithm 3 Dual-Decomposition

```
1: Decompose problem: $J(\mathbf{x}|\theta) = \sum_i J_i(\mathbf{x^i}|\theta^i)\ s.t. \mathbf{x^i} \equiv \mathbf{x}$
2: repeat
3:    Solve subproblems: $\forall i:\ \mathbf{x^{i*}} = \arg\min_{x^i} J_i(\mathbf{x^i}|\theta^i + \lambda^i)$
4:    Update dual variables: $\forall i:\ \lambda^i = \lambda^i + \xi^i(\mathbf{x^{1*}}, \ldots, \mathbf{x^{n*}})$
5:    Ensure by projection that: $\sum_i \lambda^i \equiv 0$
6: until Stopping condition is fulfilled
```

The idea of most dual methods is sketched in Algorithm 3. Starting with a decomposition of the original into several tractable subproblems, the equality constraints $\mathbf{x^i} \equiv \mathbf{x}$ are dualized by means of Lagrange multipliers. The subgradients, which can be obtained by solving the subproblems, are used to update the dual variables λ. If the update has moved λ to a point outside the feasible set an additional projection onto the feasible set is required. This step can also be understood as a re-parameterization of the problem, with the goal that the subproblems will agree in their solutions, i.e. $\mathbf{x^i} \equiv \mathbf{x^j}$. The different dual methods differ mainly by the choice of decomposition (Algorithm 3 line 1) and the used update strategy ξ^i for dual variables (Algorithm 3 line 4).

For binary second order problems the **QPBO** method (Rother et al. 2007) can be used to find the solution of the local polytope relaxation in low order polynomial time. It reformulates the LP as a network flow problem, which is then solved efficiently. For Potts models we also compare to a cutting-plane algorithm, **MCR** (Kappes et al. 2011), that deals with a polynomially tractable relaxation of the multiway cut polytope and the multicut polytope. For Potts models the former polytope is equivalent to the local polytope relaxation (Osokin et al. 2011; Nieuwenhuis et al. 2013; Kappes et al. 2013). We compare different types of relaxations and separation procedures as described in Kappes et al. (2013)[3].

4.2 Global Optimal Methods

Integer Linear Programs (ILPs) are related to polyhedral methods. These include additional integer constraints and guarantee global optimality, contrary to the methods based on LP-relaxations which may achieve optimality in some cases only. Solutions of ILPs are found by solving a sequence of LPs and either adding additional constraints to the polytope (cutting plane techniques) as sketched in Algorithm 4, or branching the polytope or discrete candidate-set into several polytopes or candidate-sets (Branch and Bound techniques), sketched in Algorithm 5. Inside Branch and Bound methods the cutting plane methods can be applied to get better bounds which allow to exclude subtrees of the branch-tree earlier.

Algorithm 4 Cutting-Plane

```
1: Initial Relaxation: $\min_{\mu \in P} \langle \theta, \mu \rangle$
2: repeat
3:    Solve current relaxation: $\mu^* = \arg\min_{\mu \in P} \langle \theta, \mu \rangle$
4:    Add constraints violated by $\mu^*$ to P
5: until No violated constraints found
```

Algorithm 5 Branch and Bound

```
1: Initial Problem: $\min_{\mathbf{x} \in X} J(\mathbf{x})$, $S = \{X\}$
2: repeat
3:    Select branch node: $\tilde{X}$ with $\tilde{X} \in S$
4:    Split selected node: $\tilde{X}^1, \ldots, \tilde{X}^n$ with $\tilde{X} = \tilde{X}^1 \cup \ldots \cup \tilde{X}^n$
5:    Branch: $S = (S \setminus \tilde{X}) \cup \tilde{X}^1 \cup \ldots \cup \tilde{X}^n$
6:    Bound: $\forall i = 1, \ldots, n : B^{\tilde{X}^i} \le \min_{\mathbf{x} \in \tilde{X}^i} J(\mathbf{x})$
7:    Solution (if possible): $\forall i = 1, \ldots, n : V^{\tilde{X}^i} = \min_{\mathbf{x} \in \tilde{X}^i} J(\mathbf{x})$
8: until $\exists s \in S : V^s \le \min_{s \in S} B^s$
```

We evaluate four state-of-the-art general combinatorial solvers. We wrapped the off-the-shelf ILP solver from IBM CPLEX (IBM, ILOG CPLEX 2013) by OpenGM 2 (Andres

[2] Here we consider spanning trees as subproblems such that the relaxation is equivalent to the local polytope relaxation.

[3] This includes terminal constraints *TC*, multi-terminal constraints *MTC*, cycle inequalities *CC* and facet defining cycle inequalities *CCFDB* as well as odd-wheel constraints *OWC*.

et al. 2012) and denoted by **ILP**. For the best performing method in the PIC 2011, called breadth-rotating and/or branch-and-bound (Otten and Dechter 2011) (**BRAOBB**), we observed for several instances that the optimality certificate, returned by the algorithm, was not correct. We reported that to the authors, who confirmed our observations; we chose to not report the invalid bounds returned by BRAOBB. We evaluate three variants: BRAOBB-1 uses simple and fast preprocessing, BRAOBB-2 uses stochastic local search (Hutter et al. 2005) to quickly find a initial solution, and BRAOBB-3 is given more memory and running time to analyse the instances during pre-processing. Bergtholdt et al. (2010) suggested a tree-based bounding heuristic for use within A-star search; we call this method **A-Star**. This Branch & Bound method does not scale to large problems. The same is true for the Branch & Bound extension of AD3, which uses upper and lower bounds of AD3 for bounding; we denote this extended method by **AD3-BB** (Martins et al. 2011).

The recently proposed **CombiLP** solver (Savchynskyy et al. 2013) utilizes the observation that the relaxed LP solution is integral almost everywhere in many practical computer vision problems. It confines application of a combinatorial solver to the typically small part of the graphical model corresponding to the non-integer coordinates of the LP solution. If consistence between the LP and ILP solution cannot be verified the non-integral subparts grow and the procedure repeats. This allows to solve many big problems exactly. If the combinatorial subproblem becomes too large, we return bounds obtained by the LP solver.

To reduce the large memory requirements, we also consider the integer multicut-representation introduced by Kappes et al. (2011). This multicut solver with integer constraints (**MCI**) can only be applied for functions which include terms that are either invariant under label permutations or of the first-order. As for MCR similar separation-procedures are available. Additionally, we can take advantage from integer solutions and use more efficient shortest-path methods, noted by an I within the separation acronym.

We also consider a max-cut branch and cut solver **MCBC** (Bonato et al. 2014; Kappes et al. 2013) for pairwise binary problems, which could not be made publicly available due to license restrictions.

4.3 Message-Passing Methods

Message passing methods are simple to implement and can be easily parallelized, making them a popular choice in practice. The basic idea is sketched in Algorithm 6. In the simplest case messages are defined on the edges of the factor graph, better approximations can be obtained by using messages between regions. Messages can be understood as a re-parameterization of the model, such that local optimization becomes globally consistent. Polyhedral methods can often be reformulated as message passing where the messages represent the re-parameterization of the models, as in **TRWS** and **MPLP**. Its non-sequential pendant **TRBP** (Wainwright et al. 2005) is written as a message passing algorithm. TRBP can be applied to higher order models but has no convergence guarantees. Practically it works well if sufficient message damping (Wainwright et al. 2005) is used. Maybe the most popular message passing algorithm is loopy belief propagation (LBP). While LBP converges to the global optimum for acyclic models, it is only a heuristic for general graphs, which turns out to perform reasonably well in practice (Yedidia et al. 2004). We evaluate the parallel (**LBP**) and sequential (**BPS**) versions from Szeliski et al. (2008), as well the general higher order implementation using parallel updates from Andres et al. (2012). For non-sequential methods we use message damping.

Algorithm 6 Message Passing

```
1: Setup: $\forall e \in E$ : initialize a message for each direction.
2: repeat
3:   Update: $\forall e \in E$ : update the message given the other messages.
4: until no significant change of messages
5: Decoding: Re-parameterize the original problem by the messages
   and decode the state locally or greedy.
```

Another advantage of message passing methods is that they can be parallelized easily and speeded up further for the calculation of the message updates in special cases, e.g. when distance transform (Felzenszwalb and Huttenlocher 2006) is available.

4.4 Move-Making Methods

Another class of common greedy methods applies a sequence of minimizations over subsets of the label space, iteratively improving the current labeling. The corresponding subproblems have to be tractable and the current label has to be included into the label-subset over which the optimization is performed, cf. Algorithm 7.

Algorithm 7 Move Making Methods

```
1: Setup: Select an initial labeling $x^* \in X$
2: repeat
3:   Select Move: $\tilde{X} \subset X$ s.t. $x^* \in \tilde{X}$
4:   Move/Update: $x^* = \arg\min_{x \in \tilde{X}} E(x)$
5: until No more progress is possible
```

The α–β-Swap-Algorithm (Boykov et al. 2001; Kolmogorov and Zabih 2002) (α–β-**Swap**) selects two labels, α and β, and considers moves such that all variables currently labelled α or β are allowed to either remain with their current label or change it to the other possible label within the set $\{\alpha, \beta\}$. In each round all possible pairs of labels are processed. For each label pair the corresponding auxiliary problems are binary and under some technical conditions submodular, hence they can be solved efficiently by max-flow algorithms, see Verma and Batra (2012) for a recent review.

Alternatively, the same authors (Boykov et al. 2001; Kolmogorov and Zabih 2002) suggested the α-Expansion algorithm (α**-Exp**). This move making algorithm selects a label α and considers moves which allows all variables to either remain with their current label or to change their label to α. In each round α is sequentially assigned to all possible labels. As for α-β-Swap the auxiliary problem in 7 line 3 can be reduced to a max-flow problem under some technical conditions.

Because α-Expansion and $\alpha\beta$-swap require submodular subproblems (Boykov et al. 2001; Kolmogorov and Zabih 2002), which cannot be guaranteed for all benchmark problems, several extensions have been proposed in the literature. Boykov et al. (2001) and Rother et al. (2005) suggest to use submodular approximations of the move energy function that still guarantee to not increase the original energy of the labeling in each move. The "truncation" trick of Rother et al. (2005) is used in the MRF-library (Szeliski et al. 2008)—in our study denoted by **mrf-**α-Exp-trunc and **mrf-**$\alpha\beta$**-Swap-trunc**—in order to make α-expansion and $\alpha\beta$-swap applicable to more models.

Alternatively one may apply other solvers to the non-submodular subproblems. The solvers need not be optimal but have to guarantee to not increase the energy of the labelling (Kappes et al. 2014). As suggested in Woodford et al. (2009), Lempitsky et al. (2010), Fix et al. (2011) we use QPBO to solve the subproblems. In order to deal with higher-order models we apply the reduction techniques of Fix et al. (2011) to the auxiliary problems. We name this method α-Expansion-QPBO (α**-Exp-QPBO**) to indicate that we use QPBO as sub-solver. However, it is worth to note that neither the truncated nor the QPBO version can guarantee that the best possible move is performed in each step.

Furthermore, other proposal-oracles, e.g. "non-constant", may perform better for some models, but finding such cases depends on the application and hence is beyond the scope of this work. We refer to Lempitsky et al. (2010), Kappes et al. (2014) for a more detailed discussion.

The **FastPD** algorithm (Komodakis and Tziritas 2007) is similar to α-Expansion in the way moves are selected and evaluated. Additionally, the dual solution of the max-flow solver is used to re-parameterize the objective function. This leads to significant speed up and allows as a by-product to calculate a lower bound by using the re-parameterized objective. However, FastPD might get stuck in suboptimal fix-points and will not reach the optimal dual objective.

While these three methods are based on solvable max-flow subproblems, there are other algorithms which perform local moves. The most prominent algorithm in this line is the iterated conditional modes algorithm (Besag 1986) (**ICM**), which iteratively improves the labeling of a single variable by keeping the other fixed. Andres et al. (2012) extended this idea to an exhaustive search over all sets of variables of a given size k. While this would be intractable in a naive implementation, they introduced several data-structures to keep track of the changes and ended up with the **Lazy Flipper**. As with ICM this method also guarantees to converge to a local fix-point that can only be improved by changing more than 1 or k variables for ICM and Lazy Flipper, respectively.

For second order binary models Gorelick et al. recently suggested a method based on local submodular approximations with trust region terms (Gorelick et al. 2014) called **LSA-TR**. This method iteratively approximated the non submodular part of the objective by a linear approximation around the current labeling. For the trust region term we evaluate Euclidean and Hamming distance as suggested in Gorelick et al. (2014).

Another method specialized for partition problems is the Kernighan–Lin algorithm (Kernighan and Lin 1970) (**KL**), which iteratively merges and splits regions in order to improve the energy.

4.5 Rounding

Linear programming relaxations and message passing methods typically do not provide a feasible integer solution. They provide either a fractional indicator vector or pseudo-min-marginals. The procedure to transform those into a feasible integer solution is called rounding. The final quality of the integer solution does also depend on the used rounding procedure.

The simplest rounding procedure is to round per variable based on the first order (pseudo) min-marginal given by $b_v(x_v) \cong \min_{x' \in X, x'_a = x_a} J(x)$

$$x_v^* = \arg\min_{x_v \in X_v} b_v(x_v) \qquad \forall v \in V \tag{2}$$

While this is simple and can be performed efficiently it is very brittle and might fail even for tree-structured models by mixing two modes. Ideally, decisions should not be made independently for each variable. One popular option is to condition eq. (2) on already rounded variables. This can be done on the re-parameterized objective (Kolmogorov 2006) or on the marginals by

$$x_v^* = \arg\min_{x_v \in X_v} b_v(x_v) + \sum_{\substack{f \in ne(v) \\ ne(f)\setminus\{v\} \text{are fixed}}} b_f(x_v, x^*_{ne(f)\setminus v}) \quad \forall v \in V$$

Both methods are not guaranteed to do an optimal rounding on a cyclic graph, but at least try to track modes. Similar rounding schemes can be applied for indicator functions, too.

In some special cases rounding methods exists that can guarantee a k-approximation error, i.e. the energy of rounded solution is less than a factor k-times larger than the relaxed one. A well known class of such models, are models with metric regularizers (Călinescu et al. 2000; Kleinberg and Tardos 1999; Chekuri et al. 2004). For the relaxed multiway cut solver MCR we use such a de-randomized rounding procedure (Călinescu et al. 2000; Kappes et al. 2013). For methods based on dual decomposition such rounding methods are not feasible directly, because dual methods do not provide a feasible primal solution of the LP relaxation which is essential for these rounding procedures.

However, since the rounding problem can be as hard as the original problem, which is NP-hard, there will always be cases in which one rounding method performs worse than another. It is important to understand that when we compare the objective value of the rounded integer solution of a method we implicitly also evaluate the rounding method used.

4.6 Post- and Pre-Processing

By combining mentioned solvers or augmenting them with other procedures, the further ”meta-solvers” can be generated. The **CombiLP** is one example for this. It combines TRWS with ILP.

Alahari et al. (2008) suggested a pre-processing procedure that first computes the optimal states for a subset of variables in polynomial time and then continues with the remaining smaller problem with more involved solvers. In Alahari et al. (2008) this was proposed in order to improve the runtime, and Kappes et al. (2013) showed that using this preprocessing can make combinatorial methods as efficient as approximate methods. In Kappes et al. (2013) the authors also suggest to treat acyclic substructures and connected components separately. We combine different methods with this preprocessing denoted by the postfix **-ptc**. For Potts models we make use of Kovtun's method (Kovtun 2003) to efficiently obtain partially optimal solutions.

An alternative approach to reduce the problem size is to group variables based on the energy function (Kim et al. 2011). However, this has two major drawbacks; first it is not invariant to model reparameterizations, and second the reduced problem is an approximation of the original problem.

Another meta-solver we considered is InfAndFlip, which first runs a solver to obtain a good starting point for the Lazy Flipper. Lazy Flipping never degrades the energy, but relies heavily on its initialization. We denote lazy flipping post-processing by **-LFk**, where k specifies the search depth.

4.7 Comparability of Algorithms and their Implementations

When we compare algorithms empirically, we really compare specific *implementations* of said algorithms, which are affected by domain-specific implementation choices. It is important to keep in mind that a general implementation will usually be much slower than a specialized one. The reason is that specialized efficient data structures can be used, more efficient calculations for subproblems become feasible, interfaces can be simplified, compiler optimization (e.g. loop unrolling) becomes applicable and many more or less obvious side-effects can show up.

Our study includes implementations that follow two different paradigms; very fast, specialized but less flexible code, e.g. TRWS and BPS of Vladimir Kolmogorov or FastPD of Nikos Komodakis, and very general and flexible but less fast code, e.g. the OpenGM 2 implementations of Dual Decomposition or LBP.

While both paradigms have merit, it becomes challenging to quantify their relative performance in a fair manner. Due to the algorithmic complexity we expect that for some methods a speedup of a factor of ≈ 100 for specialized implementations may be possible (Table 2).

Specialized solvers are very fast for problem classes they support and were designed for, but often are not easily generalized to other classes and at that point restrict the degrees of freedom of modeling. On the other hand more general solvers are able to solve a large class of models, but are often orders of magnitudes slower. Specifically, some solvers are specialized implementations for a certain class of problems (e.g. grids with Potts functions), while others make no assumptions about the problem and tackle the general case (i.e. arbitrary functions and order).

As one of several possible examples Table 3 shows six different implementations of TRWS. The original TRWS implementation for general graphs (TRWS) and grid graphs (mrf-TRWS) by Vladimir Kolmogorov and the TRWS implementation provided in OpenGM 2 (ogm-TRWS). Each is implemented for general functions and for Potts functions using distance transform (Felzenszwalb and Huttenlocher 2006).

4.8 Algorithmic Scheduling

A number of algorithms depend on an explicit or implicit ordering of factors, messages, labels, etc. While the detailed evaluation of such implementation details or implicit parameters is beyond the scope of this work, we describe the choices in our implementations and possible alternatives for the sake of completeness.

For α-Expansion and $\alpha\beta$-Swap we choose the default order of the moves. Recently, Batra and Kohli (2011) have shown, that an optimized ordering can often improve the results. However, there are no theoretical guarantees that this

Table 2 List of methods used in this study

Method	Abbreviation	Restrictions on		Comment	Reference
		Topology	Functions and labels		
Kernighan Lin	ogm-KL	–	Potts, $\|L\| = \|V\|$		Kernighan and Lin (1970)
Fast Primal Dual	FastPD	–	2nd order	Implementation has further restrictions	Komodakis and Tziritas (2007)
α-Expansion	α-Exp	–	2nd order, subm. moves		Boykov et al. (2001)
$*$	mrf-α-Exp-trunc	grid	2nd order		Szeliski et al. (2008)
$*$	α-Exp-QPBO	–	2nd order		Kappes et al. (2014)
$\alpha\beta$-Swap	$\alpha\beta$-Swap	–	2nd order, subm. moves		Boykov et al. (2001)
$*$	mrf-$\alpha\beta$-Swap-trunc	grid	2nd order		Szeliski et al. (2008)
Lazy Flipper	ogm-LF	–	–		Andres et al. (2012b)
Iterated Conditional Modes	ogm-ICM	–	–		Besag (1986)
Local Submodular Approximation	ogm-LSA-TR	–	2nd order , $\|L\| = 2$	Using Trust Region	Gorelick et al. (2014)
Tree Reweighted Belief Propagation	ogm-TRBP	–	–		Wainwright et al. (2005)
Sequential Belief Propagation	BPS		2nd order		Pearl (1988)
$*$	mrf-BPS	grid	2nd order		Szeliski et al. (2008)
$*$	ogm-BPS	–	–		Kolmogorov (2006)
Loopy Belief Propagation	ogm-LBP	–	–		Pearl (1988)
$*$	mrf-LBP	grid	2nd order		Szeliski et al. (2008)
Dual Decomposition	ogm-DD-BUNDLE-H	–	–	Tree-decomposition, heuristic stepsize	Kappes et al. (2012)
$*$	ogm-DD-BUNDLE-A	–	–	Tree-decomposition, adaptive stepsize	Kappes et al. (2012)
$*$	ogm-DD-SG-A	–	–	Tree-decomposition, adaptive stepsize	Komodakis et al. (2011), Kappes et al. (2012)
Tree Reweighted Message Passing	TRWS	–	2nd order		Kolmogorov (2006)
$*$	mrf-TRWS	grid	2nd order		Szeliski et al. (2008)
Quadratic Pseudo Boolean Optimization	QPBO	–	2nd order , $\|L\| = 2$		Rother et al. (2007)
Relaxed Multicut	ogm-MCR-[workflow]	–	gen. Potts , $\|L\| = \|V$		Kappes et al. (2013)
Relaxed Multiwaycut	ogm-MCR-[workflow]	–	1st order and gen. Potts		Kappes et al. (2013)
Linear Program over Local Polytope	ogm-LP-LP	–	–	Holds exhaustive model copy	
Alternating Directions Dual Decomposition	AD3	–	–	Holds exhaustive model copy	Martins et al. (2011)
Max-Product Linear Programming	MPLP	–	–	Holds exhaustive model copy	Globerson and Jaakkola (2007)
Max-Product Linear Programming with cycle constraints	MPLP-C	–	–	holds exhaustive model copy	Globerson and Jaakkola (2007)
Adaptive Diminishing Smoothing Algorithm	ogm-ADSAL	–	2nd order		Savchynskyy et al. (2012)

Table 2 continued

Method	Abbreviation	Restrictions on		Comment	Reference
		Topology	Functions and labels		
CombiLP	ogm-CombiLP	–	–	Implementation supports only 2nd order	Savchynskyy et al. (2013)
A^*	ogm-A-Star	–	–		Bergtholdt et al. (2010)
Integer Linear Program	ogm-ILP	–	–	holds Exhaustive model copy	
Integer Multicut	ogm-MCI-[workflow]	–	gen. Potts , $\|L\| = \|V\|$		Kappes et al. (2013)
Integer Multiway Cut	ogm-MCI-[workflow]	–	1st order and gen. Potts		Kappes et al. (2013)
Alternating Directions Dual Decomposition with B&B	AD3-BB	–	–		Martins et al. (2011)
MaxCut Branch & Cut	MCBC	–	2nd order , $\|L\| = 2$	Code not public available	Bonato et al. (2014); Kappes et al. (2013)
Breadth Rotating And Or B&B	BRAOBB	–	–	Holds exhaustive model copy, unfixed bug	Otten and Dechter (2011)

pre-/postfix	Meaning	Reference
-VIEW	Inject OpenGM as data-structure into external code	Szeliski et al. (2008)
-TABLE	Uses tables (flat memory) as data-structure in external code	Szeliski et al. (2008)
-TL1	Uses truncated linear label distance as data-structure in external code	Szeliski et al. (2008)
-TL2	Uses truncated squared label distance as data-structure in external code	Szeliski et al. (2008)
-[workflow]	Workflow for seperation used for multicut algorithms	Kappes et al. (2013)
ogm-	Uses OpenGM data-structures which allows the use of different and specialized functions	Andres et al. (2014)
mrf-	Uses MRF-Lib-data-structures which are optimized for second order grid models	Szeliski et al. (2008)
-pct	Use partial optimality and tentacle elimination to reduce problem size, each connected component is then solved independently	Kappes et al. (2013)
-LF[K]	Use lazy flipping with maximal search depth K for post-processing	Andres et al. (2012b)

For some methods we have different implementations, which than have different abbreviations. Furthermore the table shows restrictions of the methods and implementations, respectively, as well as additional comments and corresponding references. The four groups are move-making methods, message passing methods, methods based on linear programming relaxations and finally methods that guarantees global optimal solutions when no time and memory restrictions are given. Furthermore, some pre- and postfixes indicate the used data-structures, seperation procedures or some pre- and post-processing-method which can be used together with several other methods

algorithm will produce an ordering strictly better than the default one. Similarly, the order of moves has a strong impact on the quality of solutions and runtime in the case of other move-making algorithms such as ICM and Lazy-Flipper.

Sequential message passing methods depend on the ordering of the messages. This ordering can be predefined or selected dynamically during optimization, as suggested by Tarlow et al. (2011). Parallel message passing methods do not require an ordering, but typically underperform compared to asynchronous algorithms and often require damping to achieve convergence. While empirically sequential methods perform often better, they cannot be parallelized that easy.

For methods using cutting plane techniques and branch and bound, the cutting plane management/scheduling (Wesselmann and Stuhl 2012) and branching strategy (Achterberg et al. 2005) define an additional degree of freedom, respectively. We always use the default option of the methods which provide in average good choices.

4.9 Stopping Condition

Another critical point which has to be taken into account is the stopping condition used. Not all methods have a unique stopping condition. For example move making methods stop if no move from the considered set of moves gives an improvement. This is practical only if the set of possible moves is manageable, for example if it can be optimized over exactly.

Table 3 Runtime (in seconds) for a single iteration for six different implementations of TRWS for the same instance

	mrf-TRWS (s)	TRWS (s)	ogm-TRWS (s)
Potts	0.04	0.09	0.18
General	0.13	0.21	1.51

Using distance transform for Potts models always leads to a speed up compared to standard updates. The mrf-TRWS which is specialized for grids is fastest, followed by the original implementation TRWS which requires the same regularizer type everywhere, and ogm-TRWS which only restricts the model to be of second order

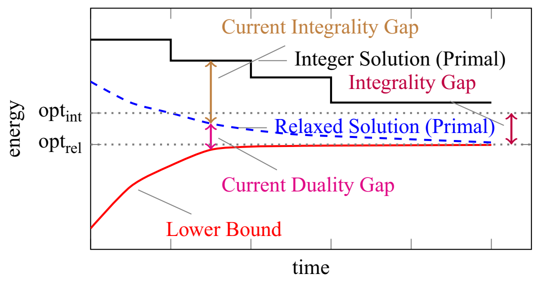
Current Integrality Gap
Integer Solution (Primal)
Integrality Gap
opt_{int}
opt_{rel}
energy
Relaxed Solution (Primal)
Current Duality Gap
Lower Bound
time

Fig. 5 Illustration of temporal changes of the objectives within linear programming frameworks. LP solves either the relaxed problem (*blue*) or its dual problem (*red*). Under mild technical condition their optimal value is identical, i.e. the duality gap vanishes. Extracting a primal solution from a dual is non-trivial and typically the solution is non-integral. Rounding to a feasible integer solution causes an additional integrality gap

In order to deal with the accuracy of floating-point arithmetic (Goldberg 1991), linear programming approaches often solve the problem up to a certain precision, here set to 10^{-7}. Solving the LP to a higher precision requires more runtime, but may not improve (or even change) the final integer solution since LPs typically involve a rounding procedure.

For LP solvers that work on the dual problem like TRWS or MPLP it is non-trivial to evaluate the duality gap. To overcome this problem Savchynskyy and Schmidt (2013, 2014) proposed a method to generate feasible primal estimates from duals and so get an estimate on the primal-dual-gap.

Unfortunately, in theory even small changes in the primal or dual solution can have large impact on the rounded integer solution. Moreover, the method of Savchynskyy and Schmidt is not available for all algorithms so far, so we use the total gap (current integrality + duality gap) as our stopping condition, cf. Fig. 5.

Another option is to check if the changes made by the method are numerically relevant. If for example the largest change of a message of LBP is smaller than 10^{-5} it is more likely to run into numerical problems than to make further improvements. The same holds true if the improvements of the dual within TRWS become too small.

We will use the following additional stopping conditions for all algorithms: **(1)** A method is stopped after 1 h. **(2)** A method is stopped if the gap between the energy of the current integer solution and the lower bound is smaller than 10^{-5}. **(3)** A method is stopped if the numerical changes within its data is smaller than 10^{-7}.

With this additional stopping condition, we obtain better numerical runtimes for some methods, e.g. TRWS and LBP, as reported in Kappes et al. (2013) without worsening the other results. Such methods suffer when a large number of iterations is used as the only stopping-condition.

5 Experimental Setup

The hardware platform for this study is the Intel Core i5-4570 CPU with 3.20 GHz, equipped with 32 GB of RAM[4]. In order to minimize the effect of operating system activity on runtime measurements, experiments were conducted on only three of the four cores of a CPU. No multi-threading and no hyper-threading was used. An evaluation of the parallelizability of optimization algorithms or the runtime of parallel implementations is beyond the scope of this study.

The software platform for this study is Version 2.3.3 of the C++ template library OpenGM (Andres et al. 2014, 2012). Compiling was done with GCC 4.8.2 and O3 options. The operating system was Ubuntu 14.04. OpenGM imposes no restrictions on the graph or functions of a graphical model and provides state-of-the-art optimization algorithms, custom implementations as well as wrappers around publicly available code. In the tables below, prefixes indicate the origin of each particular implementation, *ogm* (Andres et al. 2012) and *mrf* (Szeliski et al. 2008). The lack of a prefix indicates that code was provided by the corresponding author and wrapped for use in OpenGM. All graphical models considered in this study are available from Andres et al. (2014) in the OpenGM file format, a platform independent binary file format built on top of the scientific computing standard HDF5.

To make runtime measurements comparable, we exclude from these measurements the time for copying a graphical model from the OpenGM data structure into a data structures used in wrapped code, the time spent on memory allocation in the initialization phase of algorithms, as well as the overhead we introduce in order to keep track, during optimization, of the current best integer solution and, where available, the current best lower bound. To keep resources in check, every algorithm was stopped after 1 h if it had not converged.

Obviously, not every implementation of every algorithm is applicable to all graphical models. We made our best effort

[4] Due to the increased workload compared to the experiments in Kappes et al. (2013), we switch to a homogeneous cluster and no longer use the Intel Xeon W3550 3.07GHz CPU equipped with 12 GB RAM.

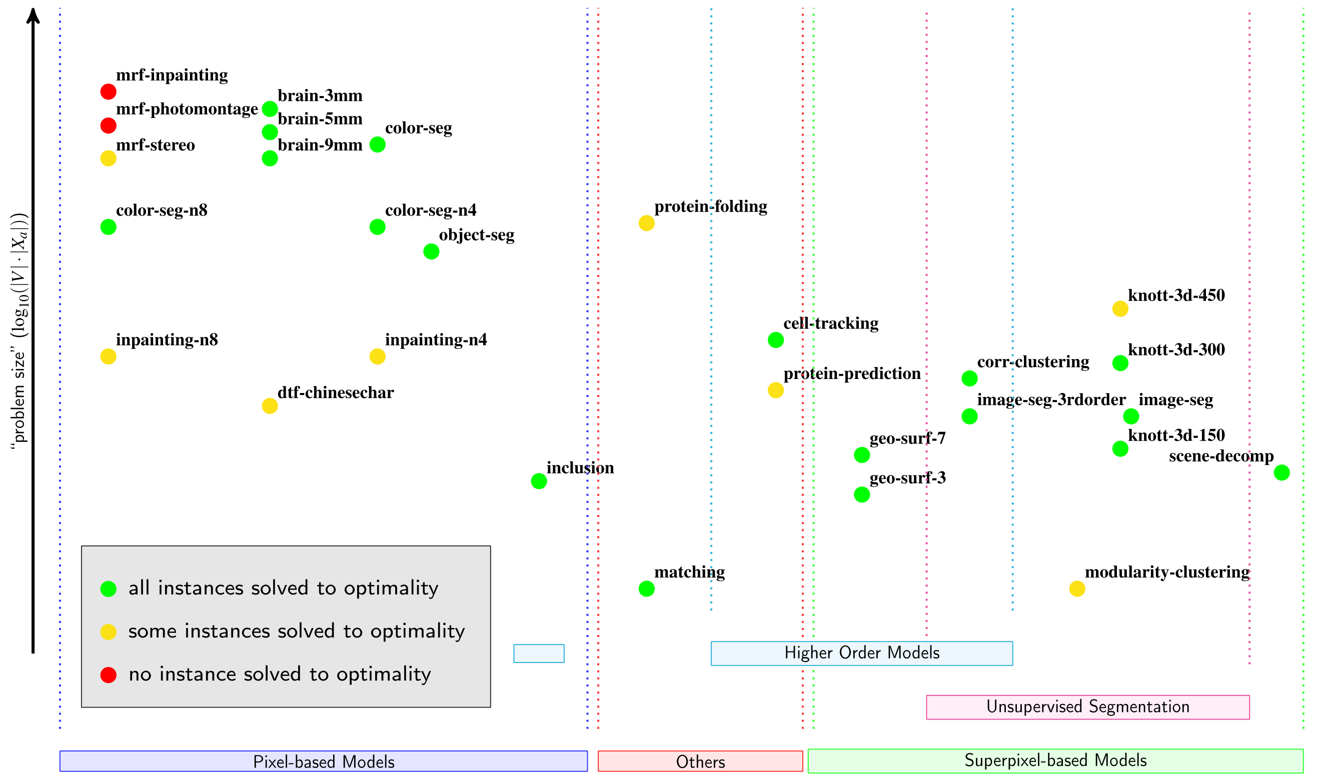

Fig. 6 List of models used in the benchmark. The x-axis groups the model into specific classes and the y-axis reflects the size of the models. Datasets for which we can calculate the optimal solution within 1 h for each instances are marked *green*, for which some solved to optimality within 1 h in yellow, and those which are unsolved within 1 h so far *red*. If we gave combinatorial methods more time for optimization, say 10 h, we would be able to solve some more models to optimality (Savchynskyy et al. 2013). We were able to find better solutions than reported in Szeliski et al. (2008) for the models there considered, even we were not able to solve a single instance in mrf-photomontage or mrf-inpainting within 1 h. Non surprisingly, larger models are usually harder than smaller ones. If special properties can be used, as for Potts models, solvers scale better. Also small models can be hard if large interactions exist, as for dtf-chinesechar. While our study gives some indications how the "hardness" of an instance could be estimated, a principle measure is so far not known

to apply as many algorithms as possible to every model. As a consequence of this effort, the study compares implementations of algorithms which are highly optimized for and restricted to certain classes of graphical models with less optimized research code applicable to a wide range of graphical models. As discussed in Sect. 4.7, this aspect needs to be taken into account when comparing runtimes (Fig. 6).

6 Evaluation

This section summarizes the experimental results. More detailed evaluation in different forms can be found in the supplementary material and on the project website (Andres et al. 2014). For selected graphical models and instances, proper numbers are reported in Tables 4–24.

All tables in this section as well as all tables online show the *runtime*, the objective *value* of the final integer solution as well as the lower *bound*, averaged over all instances of a particular model. Note that bad results of a method on a single instance can have large impact on the mean measurements, even when the method performs well on all other instances. That is why we report, in addition, the number of instances for which an algorithm returned the *best* (not necessary optimal) integer solution among all algorithms and the number of instances for which the algorithm verified optimality by its own, denoted by *opt*. As we deal with floating-point numbers and terminate algorithms if the gap between the current best integer solution and the lower bound is less than 10^{-5}, we need to take the precision of floating-point operations into account. An output is taken to be the best and verified optimal if the difference is less than 10^{-5} in terms of its absolute value or less than 10^{-8} in terms of its relative value.

For the first time and contrary to Kappes et al. (2013) we also measure the memory requirements. The tables in this paper show the maximal memory requirements by a method for all instances of a model given in GB, denoted by *mem*. Memory requirements per instance are shown online and in the supplementary material.

Table 4 *mrf-stereo* (3 instances): on average, TRWS-LF2 and CombiLP afford the best solutions. FastPD is the fastest algorithm. Solutions obtained by BPS are better in terms of the two-pixel accuracy (PA2) than solutions with lower objective value. Storing the functions of a graphical model explicitly, as value tables, instead of as implicit functions, slows algorithms down, as can be seen for TRWS

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	PA2
FastPD	**3.11**	1614255.00	301059.33	2.55	0	0	0.6828
mrf-α-Exp-trunc	11.36	1615349.00	$-\infty$	**0.26**	0	0	0.6835
mrf-$\alpha\beta$-Swap-trunc	13.12	1927265.67	$-\infty$	**0.26**	0	0	0.6781
ogm-FastPD-LF2	156.87	1611484.33	−33495282.00	2.70	0	0	0.6828
ogm-TRWS-LF2	365.90	**1587043.67**	**1584746.53**	0.59	0	0	0.6803
mrf-LBP-TL	242.10	1633343.00	$-\infty$	0.45	0	0	0.6804
mrf-BPS-TL	224.42	1738696.00	$-\infty$	0.56	0	0	**0.7051**
mrf-TRWS-TAB	1518.01	1587932.00	1584745.90	19.62	0	0	0.6803
mrf-TRWS-TL	216.41	1587928.67	**1584746.53**	0.56	0	0	0.6803
ogm-ADSAL	3163.13	1589318.00	1584664.58	0.98	1	1	0.6814
ogm-BUNDLE-H	2152.15	1645250.33	1584466.49	1.94	1	0	0.6802
ogm-CombiLP	835.92	1587560.67	1584724.04	7.17	**2**	**2**	0.6809

Table 5 *mrf-inpainting* (2 instances): the best results are obtained by TRWS-LF2. α-Expansion is a faster but worse alternative. The smallest color error (CE) was obtained by BPS

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	CE
FastPD	7.72	32939430.00	0.00	4.10	0	0	14.70
mrf-α-Exp-trunc	**42.07**	27266168.50	$-\infty$	**0.30**	0	0	11.57
mrf-$\alpha\beta$-Swap-trunc	92.06	27055552.00	$-\infty$	0.33	0	0	11.60
ogm-FastPD-LF1	174.04	27509437.00	−891985522.00	4.14	0	0	13.07
ogm-TRWS-LF1	679.63	26464015.00	**26462450.59**	0.99	0	0	10.99
ogm-TRWS-LF2	2404.03	**26463829.00**	**26462450.59**	1.01	**1**	0	10.99
mrf-LBP-TL	573.91	26597364.50	$-\infty$	0.63	0	0	**10.59**
mrf-BPS-TL	593.35	26612532.50	$-\infty$	0.82	0	0	12.01
mrf-TRWS-TL	563.15	26464865.00	**26462450.59**	0.82	0	0	10.99
ogm-ADSAL	3892.14	26487768.50	26445564.61	2.46	0	0	10.96
ogm-CombiLP	48723.23	26467926.00	26461874.39	2.09	**1**	0	10.99

Table 6 *mrf-photomontage* (2 instances): for these instances, α-Expansion is clearly the first choice. Due to the lack of unary data-terms, the LP relaxation is weak and rounding is hard

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
mrf-α-Exp-trunc-TAB	**7.37**	**168457.00**	$-\infty$	**0.89**	**2**	0
mrf-$\alpha\beta$-Swap-trunc-TAB	9.77	170858.50	$-\infty$	**0.89**	0	0
ogm-TRWS-LF2	323.48	735193.00	166827.12	1.46	0	0
mrf-LBP-TAB	458.73	438611.00	$-\infty$	0.93	0	0
mrf-BPS-TAB	188.37	2217579.50	$-\infty$	1.36	0	0
mrf-TRWS-TAB	203.79	1243144.00	166827.07	1.36	0	0
ogm-ADSAL	3605.24	185560.00	**167274.34**	1.09	0	0

For some models, we are able to evaluate the output also with respect to an application specific measurement, cf. Table 1. This addresses the question whether the absolute differences between algorithms are relevant for a particular application.

6.1 Pixel/Voxel-Based Models

Stereo Matching (mrf-stereo) We now consider three instances of a graphical model for the stereo matching problem in vision. Results are shown in Table 4. It can be seen

Table 7 *color-seg-n4* (9 instances): TRWS gives optimal or nearly optimal results on all instances. CombiLP also solves all problems. MCI solves all but one instance

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
FastPD	**0.29**	20034.80	13644.72	0.31	0	0
mrf-α-Exp-trunc-TL	1.11	20033.56	$-\infty$	**0.07**	0	0
mrf-$\alpha\beta$-Swap-trunc-TL	0.64	20060.91	$-\infty$	**0.07**	0	0
ogm-FastPD-LF2	6.15	20033.21	12543.39	0.38	0	0
ogm-TRWS-LF1	7.58	20012.17	20012.14	0.15	7	7
mrf-LBP-TL	39.87	20053.25	$-\infty$	0.09	0	0
mrf-BPS-TL	23.35	20094.03	$-\infty$	0.09	0	0
MCR-TC-MTC	440.57	20450.12	19807.10	1.31	7	7
mrf-TRWS-TL	23.52	20012.18	**20012.14**	0.09	8	7
ogm-ADSAL	311.96	20012.15	**20012.14**	0.21	8	7
ogm-BUNDLE-A	224.07	20024.78	20012.01	0.38	7	7
MCI-TC-MTC-TCI	442.89	20450.11	19807.10	2.66	8	8
MCI-pct	429.54	20889.89	$-\infty$	1.67	8	8
ogm-CombiLP	36.68	**20012.14**	**20012.14**	0.38	**9**	**9**

Best values are in bold

Table 8 *color-seg* (3 instances): for all instances, the local polytope relaxation is tight. Nevertheless, the fixed point of TRWS is suboptimal and for one instance, ADSAL does not converge within 1 h. MCI-pct provides verified solutions as fast as α-Expansion and FastPD which do not provide optimality certificates

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
α-Exp-pct	**0.82**	**308472274.33**	$-\infty$	0.56	**3**	0
α-Exp-VIEW	5.98	308472275.67	$-\infty$	0.70	2	0
FastPD	**0.31**	308472275.00	308420090.33	1.00	2	0
ogm-LF-2	11.82	309850181.00	$-\infty$	0.74	0	0
$\alpha\beta$-Swap-VIEW	6.25	308472292.33	$-\infty$	0.70	2	0
BPS-TL	68.24	308733349.67	$-\infty$	**0.40**	0	0
ogm-BPS	106.57	308494459.00	$-\infty$	2.76	0	0
ogm-LBP-0.95	117.09	308494213.33	$-\infty$	2.76	0	0
MCR-TC-MTC	89.46	**308472274.33**	**308472274.33**	3.76	**3**	**3**
MCR-pct	0.82	**308472274.33**	**308472274.33**	0.56	**3**	**3**
ogm-ADSAL	2156.82	308472289.00	308472273.99	1.19	2	2
TRWS-TL	90.76	308472310.67	308472270.43	**0.40**	2	1
TRWS-pct	1.07	308472290.67	**308472274.33**	0.56	2	2
MCI-TC-MTC-TCI	80.19	**308472274.33**	**308472274.33**	3.76	**3**	**3**
MCI-pct	**0.98**	**308472274.33**	**308472274.33**	0.56	**3**	**3**
ogm-CombiLP	483.64	**308472274.33**	**308472274.33**	1.84	**3**	**3**

Best values are in bold

from these results that two instances were solved to optimality. Only for the instance *teddy* in which variables have 60 labels, no integer solution could be verified as optimal. For the two instances for which optimal solutions were obtained, suboptimal approximations that were obtained significantly faster are not significantly worse in terms of the two pixel accuracy (PA2), i.e. the number of pixels whose disparity error is less than or equal to two. On average, the solution obtained by BPS is 2% better in terms of the two-pixel accuracy (PA2) than solutions with smaller objective value.

Inpainting (mrf-inpainting) We now consider two instances of a graphical model for image inpainting. In these instances, every variable can attain 256 labels. Thus, efficient implementation is essential and only some implementations of approximative algorithms could be applied. Results are shown in Table 5 and Fig. 7. It can be seen from these results that TRWS outperforms move making methods. The best result is obtained by taking the solution provided by TRWS as the starting point for a local search by lazy flipping. While FastPD and α-expansion converge faster than TRWS, their solution is significantly worse in terms of the objective value and also in terms of the mean color error (CE).

Photomontage (mrf-photomontage) We now consider two instances of graphical models for photomontage. Results are

Table 9 *object-seg* (5 instances): for instances like these which are large and for which the local polytope relaxation is tight, combinatorial algorithms offer no advantages in practice

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
FastPD	**0.11**	31317.60	29611.23	0.17	4	0
mrf-α-Exp-trunc-TL	0.40	31317.60	$-\infty$	**0.05**	4	0
mrf-$\alpha\beta$-Swap-trunc-TL	0.23	31323.23	$-\infty$	**0.05**	2	0
ogm-TRWS-LF1	3.27	31317.23	31317.23	0.09	5	5
mrf-LBP-TL	29.43	32400.01	$-\infty$	**0.05**	0	0
mrf-BPS-TL	11.19	35775.27	$-\infty$	**0.05**	0	0
ogm-LBP-0.95	61.38	32673.75	$-\infty$	0.31	0	0
MCR-TC-MTC	421.26	32376.56	**31317.23**	0.86	4	4
MCR-pct	62.36	31674.41	**31317.23**	0.20	2	2
mrf-TRWS-TL	2.21	**31317.23**	**31317.23**	**0.05**	**5**	**5**
ogm-ADSAL	99.50	**31317.23**	**31317.23**	0.14	**5**	**5**
TRWS-pct	0.96	**31317.23**	**31317.23**	0.08	**5**	**5**
MCI-TC-MTC-TCI	428.32	**31317.23**	**31317.23**	1.81	**5**	**5**
MCI-pct	69.84	**31317.23**	**31317.23**	0.39	**5**	**5**
ogm-CombiLP	32.61	**31317.23**	**31317.23**	0.28	**5**	**5**

Table 10 **inpainting-n4** (2 instances): one instance is easy because the LP relaxations is tight. The other instance is designed to be hard. On average, α-Expansion and BPS perform best. Subgradient algorithms with small step-size give the best primal solutions, but dual convergence is slow. Global optimal algorithms cannot solve the hard instance within 1 h

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
FastPD	**0.01**	454.75	294.89	0.03	1	0
mrf-α-Exp-trunc-TL	**0.01**	454.75	$-\infty$	0.01	1	0
mrf-$\alpha\beta$-Swap-trunc-TL	**0.01**	**454.35**	$-\infty$	0.01	2	0
ogm-TRWS-LF2	1.45	489.30	448.09	0.03	1	1
mrf-LBP-TL	4.25	475.56	$-\infty$	0.01	1	0
mrf-BPS-TL	1.69	**454.35**	$-\infty$	0.01	**2**	0
MCR-TC-MTC	1386.81	645.89	448.27	0.20	1	**1**
MCR-pct	1248.88	1179.00	448.27	0.18	0	0
mrf-TRWS-TL	0.97	490.48	448.09	0.01	1	**1**
ogm-ADSAL	59.91	454.75	448.27	0.03	1	**1**
ogm-BUNDLE-A	39.98	455.25	448.23	0.04	1	0
ogm-BUNDLE-H	19.13	455.25	448.22	0.04	1	**1**
TRWS-pct	2.77	489.30	448.10	0.02	1	**1**
MCI-TC-MTC-TCI	1812.16	462.60	448.86	0.44	1	**1**
MCI-pct	1807.10	270479.80	$-\infty$	0.41	1	**1**
ogm-CombiLP	129.04	461.81	446.66	0.49	1	**1**

shown in Table 6. It can be seen from these results that move making algorithms outperform algorithms based on linear programming relaxations. This observation is explained by the fact that the second-order factors are more discriminative in this problem than the first-order factors. Therefore, the LP relaxation is loose and thus, finding good primal solutions (rounding) is hard.

Color Segmentation (col-seg-n4/-n8) We now consider nine instances of a graphical model for color segmentation. These are Potts models with 76.800 variables and few labels and a 4-connected or 8-connected grid graph. Results are shown in Table 7 and Fig. 8. It can be seen form these results that all instances could be solved by the multicut algorithm and CombiLP, both of which verified optimality of the respective solutions. LP relaxations over the local polytope are tight for 7 instances and are overall better than other approximations. FastPD, α-expansion and $\alpha\beta$-swap converged to somewhat worse but still reasonable solutions very quickly. For the hardest instance, algorithms based on multiway cuts did not find a solution within 1 h. It can be seen from Fig. 8 that approximate solu-

Table 11 *brain-5mm* (4 instances): although the local polytope relaxation is tight for these instances, rounding is not trivial. Only MCI-pct is able to solve all instances within 1 h. FastPD is 10 times faster but solutions are worse in terms of the objective value. For real world applications, these solutions might, however, be sufficient

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
α-Exp-pct	7.02	19088999.75	$-\infty$	1.53	0	0
α-Exp-VIEW	100.66	19089080.00	$-\infty$	1.82	0	0
FastPD	**1.32**	19089484.75	17052089.25	3.11	0	0
ogm-FastPD-LF2	48.51	19088812.00	17052089.25	4.59	0	0
ogm-TRWS-LF2	184.22	19087628.00	19087612.50	2.62	0	0
$\alpha\beta$-Swap-VIEW	91.84	19089768.00	$-\infty$	1.83	0	0
BPS-TL	450.43	19090723.25	$-\infty$	**1.08**	0	0
ogm-BPS	3601.41	19099086.75	$-\infty$	7.29	0	0
ogm-LBP-0.95	1574.35	19091228.75	$-\infty$	7.28	0	0
ogm-ADSAL	3610.13	19087679.25	19087612.49	3.37	0	0
TRWS-TL	120.29	19087730.25	**19087612.50**	**1.08**	0	0
TRWS-pct	21.93	19087728.50	**19087612.50**	1.53	0	0
MCI-pct	25.63	**19087612.50**	**19087612.50**	1.99	**4**	**4**
ogm-CombiLP	2022.87	19087626.75	**19087612.50**	4.84	3	3

Table 12 *dtf-chinesechar* (100 instances): best results are obtained by combinatorial methods after model reduction. MCBC use special cutting plane methods which lead to tighter relaxations and better lower bounds. When a shorter running time is needed the Lazy Flipper and BPS are alternatives

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	PA
LSA-TR (euc.)	**0.05**	−49548.10	$-\infty$	**0.10**	30	0	0.6712
LSA-TR (ham.)	0.06	−49536.76	$-\infty$	**0.10**	1	0	0.6433
ogm-LF-1	0.23	−49516.08	$-\infty$	0.16	1	0	0.5725
ogm-LF-2	7.34	−49531.11	$-\infty$	0.25	7	0	0.6003
ogm-LF-3	637.92	−49535.37	$-\infty$	2.29	16	0	0.6119
ogm-TRWS-LF2	83.78	−49519.42	−50119.41	0.31	10	0	0.5945
BPS-TAB	62.69	−49537.08	$-\infty$	0.12	30	0	**0.6715**
ADDD	9.74	−48656.71	−50119.38	0.77	0	0	0.5079
MPLP	516.36	−49040.57	−50119.46	0.94	0	0	0.6064
ogm-ADSAL	730.78	−49524.30	−50119.39	0.31	1	0	0.6445
QPBO	0.17	−49501.95	−50119.38	0.13	0	0	0.5520
TRWS-TAB	78.84	−49497.01	−50119.41	0.13	3	0	0.5649
TRWS-pct	4.43	−49496.76	−50119.38	0.13	2	0	0.5636
ogm-ILP-pct	3553.71	−49547.41	−50061.15	1.06	63	0	0.6556
MCBC-pct	2053.89	**−49550.10**	**−49612.38**	–	**80**	**56**	0.6624
ogm-ILP	3569.52	−49536.00	−50092.16	7.53	8	0	0.6444
SA	–	−49533.02	$-\infty$	–	13	0	0.6541

tions differ, especially at the yellow feathers around the neck.

Color Segmentation (color-seg) We now consider three instances of a graphical model for color segmentation provided by Alahari et al. (2010). Results are shown in Table 8. It can be seen from these results that the local polytope relaxation is tight. Approximate algorithms find the optimal solution for two instances and a near optimal solution for one instance of the problem. When Kovtun's method is used to reduce the problem size—which works well for this problem—the reduced problem can be solved easily and overall faster than with approximative algorithms alone.

Object Segmentation (object-seg) We now consider five instances of a graphical model for object segmentation provided by Alahari et al. (2010). Results are shown in Table 9. As for the color segmentation instances, the local polytope relaxation is tight. Compared to TRWS, FastPD is 10 times faster and worse in terms of the objective value only for 1 instance and only marginally. Furthermore, the pixel accuracy (PA) for results of FastPD and α-expansion is slightly

Table 13 *inclusion* (10 instances): only the commercial ILP software solves all instances of this dataset. While the local polytope relaxation is quite tight, rounding is still hard. Lazy flipping can help to correct some rounding errors

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	PA
α-Exp-QPBO	**0.04**	1587.13	$-\infty$	**0.01**	0	0	0.6771
ogm-LBP-LF1	19.33	1400.66	$-\infty$	**0.01**	4	0	0.9490
ogm-LBP-LF2	19.37	1400.61	$-\infty$	0.02	7	0	0.9495
ogm-LF-3	1.14	1461.23	$-\infty$	**0.01**	0	0	0.8011
ogm-BPS	21.42	2200.68	$-\infty$	**0.01**	6	0	0.9489
ogm-LBP-0.5	19.77	2100.61	$-\infty$	**0.01**	7	0	0.9487
ogm-TRBP-0.5	21.42	1900.84	$-\infty$	**0.01**	5	0	0.9491
ADDD	6.23	3400.81	1400.31	0.03	1	1	0.9479
MPLP	5.94	4000.44	1400.30	0.02	2	1	0.9479
MPLP-C	3579.25	4200.37	1400.35	0.08	2	1	0.9470
ogm-BUNDLE-H	73.24	1400.76	1400.32	0.02	3	1	0.9496
ogm-LP-LP	18.27	4100.60	1400.33	0.30	1	1	0.9482
MCI-TC-MTC-TCI	61.46	**1400.57**	**1400.57**	0.56	**10**	**10**	**0.9496**
BRAOBB-3	3600.01	1401.64	$-\infty$	1.79	0	0	0.9467
ogm-ILP	**6.21**	**1400.57**	**1400.57**	0.74	**10**	**10**	**0.9496**

Best values are in bold

Table 14 *Scene-decomposition* (715 instances): almost all algorithms provide optimal or nearly optimal solutions. Also in terms of pixel accuracy (PA), the difference between these solutions is marginal. The PA of optimal solutions is 0.2algorithm, followed by CombiLP

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	accuracy
α-Exp-QPBO	**0.01**	−866.85	$-\infty$	**0.01**	587	0	0.7694
ogm-LBP-LF2	0.06	−866.76	$-\infty$	**0.01**	576	0	0.7699
ogm-LF-3	0.45	−866.27	$-\infty$	**0.01**	420	0	0.7699
ogm-TRWS-LF2	**0.01**	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
BPS-TAB	0.10	−866.73	$-\infty$	**0.01**	566	0	0.7701
ogm-BPS	0.02	−866.77	$-\infty$	**0.01**	585	0	0.7694
ogm-LBP-0.95	0.02 sec	−866.76	$-\infty$	**0.01**	580	0	0.7696
ogm-TRBP-0.95	0.11	−866.84	$-\infty$	**0.01**	644	0	**0.7708**
ogm-TRBPS	0.13	−866.79	$-\infty$	**0.01**	644	0	0.7705
ADDD	0.06	−866.92	**−866.93**	**0.01**	701	697	0.7693
MPLP	0.04	−866.91	**−866.93**	**0.01**	700	561	0.7693
MPLP-C	0.04	−866.92	**−866.93**	**0.01**	710	567	0.7693
ogm-ADSAL	0.04	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
ogm-BUNDLE-H	0.26	**−866.93**	**−866.93**	**0.01**	**715**	673	0.7693
ogm-BUNDLE-A+	0.07	**−866.93**	**−866.93**	**0.01**	**715**	712	0.7693
ogm-LP-LP	0.23	−866.92	**−866.93**	0.05	712	712	0.7693
TRWS-TAB	**0.01**	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
BRAOBB-1	17.61	−866.90	$-\infty$	0.27	670	0	0.7688
ADDD-BB	0.11	**−866.93**	**−866.93**	**0.01**	**715**	**715**	0.7693
ogm-CombiLP	0.02	**−866.93**	**−866.93**	0.03	**715**	**715**	0.7693
ogm-ILP	0.17	**−866.93**	**−866.93**	0.09	**715**	**715**	0.7693

Best values are in bold

better than for optimal solutions. For instances like these which are large and easy to solve, combinatorial algorithms offer no advantages in practice.

Inpainting (inpainting-n4/n8) We now consider four synthetic instances of a graphical model for image inpainting, two instances with a 4-connected grid graph and two instances with an 8-connected grid graph. For each graph structure, one instance has strong first-order factors and the other instance (with postfix ’inverse’) is constructed such that, for every variable, a first-order factors assigns to same objective value to two distinct labels. Results are shown in Table 10 and Fig. 9. It can be seen from these results that even Potts models can give rise to hard optimization problems, in particular if the first-order factors do not discriminate well between labels. Moreover, it can be seen from Fig. 9

Table 15 *geo-surf-7* (300 instances): the commercial ILP solver performs best for these small models. The local polytope relaxation is tight for almost all instances. While α-Exp-QPBO is very fast, the solutions it provides are significant worse than the optimal solution, for some instances, both in terms of the objective value and in terms of pixel accuracy (PA). The suboptimal solutions provided by LBP have a better PA than optimal solutions

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	PA
α-Exp-QPBO	**0.02**	477.83	$-\infty$	**0.01**	257	0	0.6474
ogm-LBP-LF1	0.60	498.45	$-\infty$	**0.01**	66	0	0.6988
ogm-LBP-LF2	0.65	498.44	$-\infty$	**0.01**	66	0	0.6988
ogm-BPS	0.37	498.34	$-\infty$	**0.01**	69	0	**0.7035**
ogm-LBP-0.5	0.60	498.45	$-\infty$	**0.01**	67	0	0.6988
ogm-TRBP-0.5	8.07	486.42	$-\infty$	**0.01**	128	0	0.6768
ADDD	0.55	476.95	476.94	0.03	296	293	0.6531
MPLP	1.31	477.56	476.94	0.02	278	195	0.6529
MPLP-C	1.43	477.34	**476.95**	0.05	282	198	0.6529
ogm-BUNDLE-H	41.45	476.95	476.86	0.07	299	180	0.6529
ogm-BUNDLE-A+	32.29	476.95	476.91	0.06	298	238	0.6529
ogm-LP-LP	2.74	476.95	476.94	0.42	299	299	0.6530
BRAOBB-3	685.84	477.11	$-\infty$	5.69	269	0	0.6531
ogm-ILP	0.95	**476.95**	**476.95**	0.74	**300**	**300**	0.6529

Best values are in bold

Table 16 *modularity-clustering* (6 instances): the largest instance cannot be solved by any variant of MCI within 1 h. KL is robust, fast and better on large instances, leading to a better mean objective value

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
ogm-ICM	0.09	0.0000	$-\infty$	**0.01**	0	0
ogm-KL	**0.01**	**−0.4860**	$-\infty$	**0.01**	3	0
ogm-LF-1	0.03	0.0000	$-\infty$	**0.01**	0	0
MCR-CC	100.37	−0.4543	−0.5094	0.14	2	1
MCR-CCFDB	2.15	−0.4543	−0.5094	0.03	1	1
MCR-CCFDB-OWC	602.75	−0.4652	**−0.4962**	0.03	**5**	**5**
MCI-CCFDB-CCIFD	601.38	−0.4400	−0.5021	1.58	**5**	**5**
MCI-CCI	1207.07	−0.4312	−0.5158	2.69	4	4
MCI-CCIFD	1204.03	−0.4399	−0.5176	3.02	4	4

Best values are in bold

that increasing the neighborhood-system helps to avoid discretization artefacts. However, even with an 8-neighborhood, the model favors 135° over 120° angles, as can be seen in Fig. 9c.

Brain Segmentation (brain-3/5/7mm) We now consider twelve instances of a graphical model for segmenting MRI scans of human brains defined by four simulated scans at three different resolutions. These instances have 10^5–10^6 variables. For such large instances, the efficient data structures are helpful. Results are shown in Table 11. It can be seen from these results that TRWS provides tight lower bounds but suboptimal approximate solutions which shows that the rounding problem remains hard. Multiway cut cannot be applied directly because the instances are too large. However, with model reduction as pre-processing, MCI-ptc is the only algorithm that could solve all instances within 1 h. FastPD terminates in 1/10 of the runtime, providing approximations which are worse in terms of the objective value but reasonable in the application domain.

DTF Chinese Characters Inpainting (dtf-chinesechar) We now consider 100 instances of a graphical model for Chinese character inpaining. In contrast to all models considered so far, these instances are decision tree fields (DTF) which are learned in a discriminative fashion. This gives rise to frustrated cycles which render the inference problem hard. Results are shown in Fig. 10 and Table 12. It can be seen from these results that the local polytope relaxation is loose. Moreover, it can be seen that instead of applying a combinatorial algorithm directly, it is beneficial to first reduce the problem. Here, MCBC-pct performs best, verifying optimality of 56 of 100 instances. In shorter time, sequential belief propagation (BPS) and lazy flipping give good results. With respect to the pixel accuracy (PA) in the inpainting-region, BPS is with 67.15 % correct in-painted pixels, better than MCBC-pct which has PA of 66.24 %.

Color Segmentation with Inclusion (inclusion) We now consider ten instances of a higher-order graphical models for color segmentation with inclusion. Due to the higher-order

Table 17 *image-seg* (100 instances): variants of MCI solve all instances to optimality and are as fast as approximative algorithms

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	VI
ogm-ICM	3.98	4705.07	$-\infty$	**0.01**	0	0	2.8580
ogm-KL	**1.46**	4608.49	$-\infty$	**0.01**	0	0	2.6432
ogm-LF-1	1.35	4705.01	$-\infty$	**0.01**	0	0	2.8583
MCR-CC	8.54	4447.14	4442.34	0.15	35	35	2.5471
MCR-CCFDB	4.34	4447.14	4442.34	0.06	35	35	2.5469
MCR-CCFDB-OWC	4.34	4447.09	4442.34	0.06	35	35	2.5468
MCI-CCFDB-CCIFD	4.89	**4442.64**	**4442.64**	0.17	**100**	**100**	**2.5365**
MCI-CCI	2.43	**4442.64**	**4442.64**	0.14	**100**	**100**	**2.5365**
MCI-CCIFD	**2.24**	**4442.64**	**4442.64**	0.09	**100**	**100**	**2.5367**

Best values are in bold

Table 18 *image-seg-3rdorder* (100 instances): variants of MCI provide optimal solutions for all instances. Numerical problems arise only for some instances. Approximative methods are faster but results are worse

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	VI
ogm-ICM	6.30	6030.49	$-\infty$	**0.01**	0	0	2.7089
ogm-LF-1	**2.23**	6030.29	$-\infty$	**0.01**	0	0	2.7095
MCR-CC	32.56	5822.31	5465.15	0.28	0	0	2.7722
MCR-CCFDB	20.85	5823.09	5465.15	0.27	0	0	2.7705
MCR-CCFDB-OWC	21.63	5823.59	5465.29	0.27	0	0	2.7705
MCI-CCFDB-CCIFD	46.68	**5627.52**	**5627.52**	0.68	**100**	**100**	**2.6586**
MCI-CCI	70.54	5628.39	5627.49	0.73	99	98	2.6589
MCI-CCIFD	50.78	5627.52	**5627.52**	0.72	99	**100**	**2.6586**

Best values are in bold

Table 19 *hierarchical-image-seg* (715 instances): for these instances, the standard LP relaxation (MCR-CC) is quite tight. For all 715 instances, the feasible solutions output by MCR are close to the optimum, both in terms of their objective as well as in terms of the VI

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	VI
ogm-ICM	1.28	−585.60	$-\infty$	**0.01**	0	0	2.6245
ogm-LF-1	0.63	−585.60	$-\infty$	0.02	0	0	2.6245
MCR-CC	0.12	−626.76	−628.89	0.04	166	98	2.0463
MCR-CCFDB	**0.08**	−626.75	−628.90	0.03	164	98	2.0463
MCR-CCFDB-OWC	**0.08**	−626.77	−628.89	0.03	166	100	2.0460
MCI-CCFDB-CCIFD	0.78	**−628.16**	**−628.16**	0.08	**715**	**713**	**2.0406**
MCI-CCI	1.28	**−628.16**	−628.17	0.08	**715**	707	**2.0406**
MCI-CCIFD	1.25	**−628.16**	**−628.16**	0.07	**715**	**713**	**2.0406**

Best values are in bold

Table 20 *knott-3d-300* (8 instances): MCI-CCIFD affords the best solutions. MCR-CCFDB-OWC affords good solutions as well but suffers form the fact that the separation procedure is more complex and time consuming for fractional solutions

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	VI
ogm-ICM	84.37	−25196.51	$-\infty$	**0.01**	0	0	4.1365
ogm-KL	**13.16**	−25556.93	$-\infty$	**0.01**	0	0	4.1318
ogm-LF-1	29.08	−25243.76	$-\infty$	0.02	0	0	4.1297
MCR-CC	3423.65	−26161.81	−27434.30	0.57	1	1	1.7995
MCR-CCFDB	1338.99	−27276.12	−27307.22	0.15	1	1	**1.6336**
MCR-CCFDB-OWC	1367.03	−27287.23	−27309.62	0.15	6	6	1.6342
MCI-CCFDB-CCIFD	1261.99	−26826.57	−27308.19	0.37	6	6	1.7010
MCI-CCI	220.30	**−27302.78**	−27305.02	0.28	**8**	7	1.6352
MCI-CCIFD	104.55	**−27302.78**	**−27302.78**	0.16	**8**	**8**	1.6352

Best values are in bold

Table 21 *matching* (4 instances): pure branch & bound methods such as BRAOBB and AStar afford optimal solutions fast. Cycle constraints as used by MPLP-C tighten the relaxation sufficiently. For optimal solutions, the objective value correlates with the mean position error (MPE)

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt	MPE
ogm-LBP-LF2	0.21	38.07	$-\infty$	**0.01**	1	0	5.4469
ogm-LF-1	**0.01**	95.73	$-\infty$	**0.01**	0	0	6.3151
ogm-LF-2	0.29	40.79	$-\infty$	**0.01**	0	0	5.7689
ogm-LF-3	12.35	39.81	$-\infty$	**0.01**	0	0	5.6346
ogm-TRWS-LF2	0.32	33.31	15.22	**0.01**	0	0	3.1763
BPS-TAB	**0.11**	40.26	$-\infty$	**0.01**	0	0	4.9692
MPLP-C	3.51	**21.22**	**21.22**	0.03	**4**	**4**	**0.0907**
ogm-ADSAL	1380.55	32.47	15.62	**0.01**	0	0	2.9236
TRWS-TAB	0.03	64.19	15.22	**0.01**	0	0	3.8159
BRAOBB-1	2.05	**21.22**	$-\infty$	0.06	**4**	0	**0.0907**
ogm-ASTAR	**0.80**	**21.22**	**21.22**	0.05	**4**	**4**	**0.0907**
ogm-CombiLP	314.52	**21.22**	**21.22**	0.17	**4**	**4**	**0.0907**
ogm-ILP	402.09	**21.22**	**21.22**	0.17	**4**	**4**	**0.0907**

Best values are in bold

Table 22 *cell-tracking* (1 instance): the commercial ILP software solves this instance fast. The commercial LP solver affords good results three times faster. In contrast, dual LP solvers such as ADDD and MPLP do not find good integer solutions

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
ogm-LBP-LF2	62.12	7515575.61	$-\infty$	0.12	0	0
ogm-LF-2	0.43	14075743.46	$-\infty$	**0.06**	0	0
ogm-LF-3	**1.55**	8461693.24	$-\infty$	**0.07**	0	0
ogm-BPS	60.07	207520418.28	$-\infty$	0.08	0	0
ogm-LBP-0.95	61.97	307513873.84	$-\infty$	0.08	0	0
ogm-TRBP-0.95	65.97	107517017.88	$-\infty$	0.09	0	0
MPLP	459.93	107514359.61	7513851.52	0.08	0	0
ogm-BUNDLE-A	532.38	7696631.53	7501985.37	0.18	0	0
ogm-BUNDLE-H	522.32	7748583.42	7501948.96	0.18	0	0
ogm-LP-LP	**4.21**	7516359.61	7513851.52	0.51	0	0
ogm-ILP-pct	12.36	**7514421.21**	**7514421.21**	1.05	**1**	**1**
ogm-ILP	11.99	**7514421.21**	**7514421.21**	1.20	**1**	**1**

Best values are in bold

factors, some methods, e.g. TRWS, are no longer applicable. Results are shown in Table 13. It can be seen from these results that the local polytope relaxation is quite tight. However, standard rounding procedures do not yield good integer solutions. Overall, the ILP solver performs best. For larger problems, ILPs might not scale well. In such a case, LBP followed by lazy flipping is a good alternative. The best pixel accuracy (PA) is obtained by Bundle-methods, which is 0.1 % better than optimal results that have a pixel accuracy of 94.96 %.

6.2 Superpixel-Based Models

Graphical models defined with respect to an adjacency graph of superpixels have fewer variables than graphical models defined with respect to the pixel grid graph. The relative difference in the number of variables can be several orders of magnitude. Thus, combinatorial algorithms are more attractive for superpixel-based models.

Scene Decomposition (scene-decomposition) We now consider 715 instances of a graphical model for scene-decomposition. These instances have between 150 and 208 variables. Results are shown in Table 14. It can be seen from these results that the differences between the solutions of the different methods are marginal, both in terms of objective value and in terms of pixel accuracy (PA), i.e. the percentage of correctly labeled pixels. While TRWS is the fastest method, CombiLP is the fastest method that solves all problems to optimality. The best results in terms of pixel accuracy are given by TRBP. The PA of TRBP is with 77.08 % slightly better that the PA for optimal solutions.

Geometric Surface Labeling (geo-surf-3/7) We now consider 2×300 instances of higher-order graphical models for geometric surface labeling. Results are shown in Table 15. It can be seen from these results that the local polytope relaxation is very tight and the commercial ILP solver performs best. With increasing number of labels (from three

Table 23 *protein-folding* (21 instances): due to the large number of labels, combinatorial methods are not applicable. A notable exception is CombiLP, which manages to solve 18 instances to optimality. MPLP-C gives the best results in terms of the lower bound, but is not the best in generating labelings. The best integer solutions are obtained by BPS and LBP-LF2

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
ogm-LBP-LF2	134.88	−5923.01	$-\infty$	**0.25**	12	0
ogm-LF-2	54.54	−5747.56	$-\infty$	**0.25**	0	0
ogm-TRWS-LF2	51.89	−5897.06	−6041.38	**0.25**	6	1
BPS-TAB	**24.48**	−5917.15	$-\infty$	**0.25**	11	0
ogm-LBP-0.5	106.99	−5846.70	$-\infty$	**0.25**	13	0
ogm-TRBP-0.5	145.44	−5810.68	$-\infty$	**0.25**	9	0
MPLP	510.90	−5611.60	−6033.98	0.62	1	1
MPLP-C	1639.52	−5765.28	−5984.52	11.14	12	9
ogm-ADSAL	1014.89	−5881.47	−6128.90	0.51	4	1
TRWS-TAB	**22.18**	−5771.50	−6041.38	**0.25**	2	1
ogm-CombiLP	700.10	**−5955.77**	**−5955.77**	24.26	**21**	**21**

Table 24 *protein-prediction* (8 instances): except for one instance, the commercial ILP software solves all instances. LBP followed by lazy flipping is a good approximation

Best values are in bold

Algorithm	Runtime (s)	Value	Bound	Mem	Best	Opt
ogm-ICM	**0.03**	60414.84	$-\infty$	**0.01**	0	0
ogm-LBP-LF2	25.03	**52942.95**	$-\infty$	0.06	1	0
ogm-LF-1	**0.03**	60427.60	$-\infty$	0.02	0	0
ogm-LF-2	0.70	58682.74	$-\infty$	0.02	0	0
ogm-LF-3	19.08	57944.06	$-\infty$	0.09	0	0
ogm-BPS	27.64	75286.37	$-\infty$	0.05	0	0
ogm-LBP-0.5	24.79	53798.89	$-\infty$	0.05	0	0
ogm-TRBP-0.5	35.60	61386.17	$-\infty$	0.05	0	0
ADDD	10.70	106216.86	41124.16	0.05	0	0
MPLP	69.09	101531.75	43123.68	0.05	0	0
ogm-BUNDLE-A	1287.27	81035.49	44090.57	0.28	0	0
ogm-BUNDLE-H	1301.94	81039.93	44092.42	0.28	0	0
ogm-LP-LP	169.61	102829.40	44347.16	0.24	0	0
ogm-ILP	2263.46	57477.07	**44674.02**	0.65	**7**	**3**

to seven), non-commercial combinatorial algorithms suffer more than the commercial solver which performs well across the entire range, finding optimal solutions faster than approximative algorithms take to converge. For *geo-surf-7*, only α-Exp-QPBO is significant faster, but the approximate solutions are also worse. In terms of pixel accuracy, suboptimal approximations are better than optimal solutions. LBP-LF2 has a 0.07 and 4.59 % higher pixel accuracy (PA) than the optimal labeling for *geo-surf-3* and *geo-surf-7*, respectively. This indicates that, at least for *geo-surf-7*, the model does not reflect the pixel accuracy loss and can potentially be improved.

6.3 Partition Models

The properties of graphical models for unsupervised image segmentation, namely (1) absence of unary terms, (2) invariance to label-permutation, and (3) huge label spaces, exclude most common inference methods for graphical models. Most important, the invariance to label-permutation causes that the widely used local polytope relaxation is more or less useless. That is why we will compare here solvers that are designed to make use of or can deal with this additional properties.

Modularity Clustering (modularity-clustering) We now consider six instances of graphical models for finding a clustering of a network that maximize the modularity. Results are shown in Table 16. The Kerninghan-Lin algorithm is an established, fast and useful heuristic for clustering networks with respect to their modularity. It can be seen from the table that local search by means of ICM or LF-1 does not find better feasible solution than the initial labeling (a single cluster). While multicut methods work well for small instances, they do not scale so well because the graph is fully connected. It can also be seen that odd-wheel constraints tighten the standard LP-relaxation (with only cycle constrains) significantly. Combined LP/ILP cutting-plane methods (MCI-CCFDB-CCIFD) is the overall fastest exact method for networks of moderate

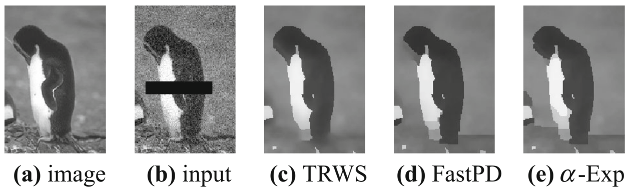

Fig. 7 *mrf-inpainting*: depicted above is one example of image inpainting. From the original image (**a**), a box is removed and noise is added to the remaining part (**b**). The result of inpainting and denoising by means of TRWS (**c**) is better than that of FastPD (**d**) and α-expansion (**e**) which show artifacts

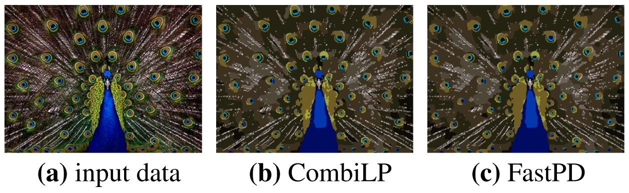

Fig. 8 *color-seg-n4*: for the hardest instance of the color segmentations problems, the differences between the optimal solution (**b**) and the approximate ones, here exemplary for FastPD (**c**), are small but noticeable

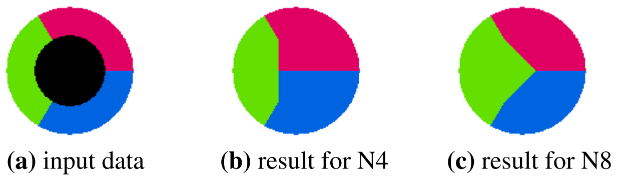

Fig. 9 *inpainting*. Depicted above are solutions (**b**, **c**) of synthetic instances of the inpainting problem (**a**). It can be seen that discretization artifacts due to the 4-neighborhood (**b**) are smaller than discretization artifacts due to the 8-neighborhood (**c**)

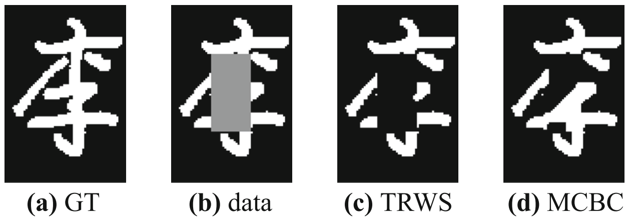

Fig. 10 *dtf-chinesechar*: depicted above is one example of the Chinese character inpainting problem. The purpose of the model is to reconstruct the original image (**a**), more precisely, the mask (**b**), from the rest of the image. It can be seen in **c** that TRWS labels the complete inpainting area as background. MCBC finds the optimal solution which reflects the full potential of decision tree fields for this application

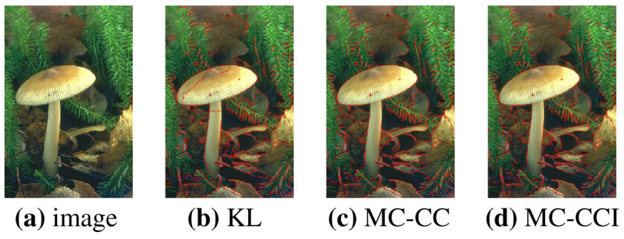

Fig. 11 *image-seg*: depicted above is one example of the image segmentation problem. KL produces a segmentation which do not separate plants from background and constains incorrect boundaries. The ILP and LP-based multicut algorithms lead to reasonable results with differ only in the lower right part of the image

size. However, for the largest instances, even MCI-CCFDB-CCIFD does not converge within 1 h.

Image Segmentation (image-seg) We now consider 100 instances of graphical models for image segmentation. These models differ from models for network-analysis in that the former are sparse. Results are shown in Table 17. It can be seen from these results that standard LP-relaxations with only cycle constraints (MCR-CC and MCR-CCFDB) work well and significantly better than KL, both in terms of objective value and variation of information (VI). Adding odd-wheel constraints (MCR-CCFDB-OWC) gives almost no improvement. Pure integer cutting plane methods (MCI-CCI and MCI-CCIFD) provide optimal solutions faster than LP-based methods and KL. Using only facet defining constraints reduces the number of added constraints and give better runtimes for MCR and MCI. Visually, the results are similar and vary only locally, cf. Fig. 11, indicating that fast and scalable approximative algorithms might be useful in practice.

Higher-Order Image Segmentation (image-seg-3rd order) We now consider 100 instances of higher-order graphical models for image segmentation. Here, factors of order three are defined w.r.t. the angles between the tangents of contours at points in which contours meet. Results are shown in Table 18. It can be seen from these results that the standard LP relaxation (MCR-CCFDB) is no longer as tight as for the second-order models *image-seg*, cf. Table 17. Odd-wheel constraints (MCR-CCFDB-OWC) tighten this relaxation only marginally. Integer cutting plane methods (MCI) suffer from the weaker relaxations and need longer for optimization, but provide optimal solutions for all 100 instances. Consistent with the results reported in Andres et al. (2011), we find that the VI of segmentations defined by optimal solutions of the third-order models is higher than the VI of segmentations defined by optimal solutions of the second-order models, indicating that either the hyper-parameter of the model needs to be estimated differently or the third-order terms are uninformative in this case.

Hierarchical Image Segmentation (hierarchical-image-seg) Next, we consider 715 instances of a graphical model for hierarchical image segmentation which include factors of orders up to 651. Results are shown in Table 19. For these instances, the standard LP relaxation (MCR-CC) is quite tight. Without odd-wheel constraints, 98 instances can be solved by the MCR. With odd-wheel constraints, 100 instances can be solved by the MCR, all with zero gap. For all 715 instances, the feasible solutions output by MCR are close to the optimum, both in terms of their objective as well as in terms of the VI. While MCR is 10 times faster than exact MCI algorithms, we emphasize that MCR was used to learn these instances which might be an advantage.

Due to well-known numerical issues, e.g. slackness introduced to improve numerical stability, bounds are not always tight for MCI. Parameters can be adjusted to overcome this problem for these particular instances, but such adjustments can render the solution of other models less numerically stable. Thus, we use the same parameters in all experiments.

3D Neuron Segmentation (knott-3d-150/300/450) We now consider 3×8 instances of graphical models for the segmentation of supervoxel adjacency graphs, i.e. for the segmentation of volume images. Results for instances on volume images of 300^3 voxels are shown in Table 20. It can be seen from these results that MCR and MCI become slower. For the instances based on volume images of 300^3 voxels, MCI-CCIFD solved instances within reasonable time. Without the restriction to facet defining constraints only, the system of inequalities grows too fast inside the cutting-plane procedure.

6.4 Other Models

Matching (matching) We now consider four instances of a graphical model for the matching problem. These instances have at most 21 variables. Results are shown in Table 21. It can be seen from these results that pure branch-and-bound algorithms, such as BRAOBB-1 or AStar, can solve these instances fast. In contrast, algorithms based on LP relaxations converge slower. The local polytope relaxation, used e.g. in MPLP, is loose because of the second-order soft-constraints used in this graphical model formulation of the matching problem. Thus, the rounding problem is hard. Adding additional cycle constraints, e.g. in MPLP-C, is sufficient to close the duality gap and obtain exact integer solutions. Another way to improve the objective value of poor integer solutions caused by violated soft-constraint is local search by lazy flipping.

Cell Tracking (cell-tracking) We now consider one instance of a graphical model for tracking biological cells in sequences of volume images. Results are shown in Table 22. It can be seen from these results that the ILP solver clearly outperforms all alternatives. Only the off-the-shelf LP-solver (LP-LP) manages to find a solution that satisfies all soft-constraints. Algorithms which solve the same relaxation, e.g. ADDD and MPLP, are slower and their rounded solutions violate soft-constraints. Lazy Flipping as a post-processing step can overcome this problem, as shown for LBP.

Side-Chain Prediction in Protein Folding (protein-folding) We now consider 21 instances of graphical models for side-chain prediction in protein folding. These instances have many variables and are highly connected. Results are shown in Table 23. It can be seen from these results that MPLP, with additional cycle-constraints, obtain the best lower bounds and CombiLP verified optimality for 21 instances, within 1 h. The best results are obtained by BPS and LBP followed by lazy flipping with search-depth 2. The rounding techniques used in the algorithms based on linear programming are insufficient for these instances.

Prediction Protein–Protein Interactions (protein-prediction) We now consider eight instances of a higher-order graphical model for prediction of protein-protein interactions. Results are shown in Table 24. The range of algorithms applicable to these instances is limited. ILP solvers found and verified solutions for 3 instances and performed best for 7 instances, within 1 h. For one remaining instances, the ILP solver affords a solution far from the optimum. Thus, LBP with lazy flipping gives the best results on average.

7 Discussion and Conclusions

Our comparative study has shown that there is no single technique which works best for all cases. The reason for this is that, firstly, the models are rather diverse and, secondly, there are several objectives, e.g. running time, best energy or loss functions. We would also like to point out, again, that not all compared solvers are on the same level of generality and modularity, and some implementations are highly optimized compared to others. Consequently, we do not advise to overemphasize the relative runtimes reported for the different methods which have different implementation paradigms. Moreover, sometimes a single instance can decrease the average performance of a method, which is reported in Tables 4–24. Consequently, methods that are best for all but one instance are not leading in the average score, e.g. MCI for color-seg-n4 or ILP for protein-prediction.

For most models, we have found that approximative methods provide nearly optimal solutions very fast. According to application specific measurements, these are often not worse than those with optimal energy. With the suggested (heuristic) stopping conditions, TRWS performs well for all models to which it can be applied. This is also due to the fact that for many models the local polytope relaxation is nearly tight. FastPD and α-expansion are slightly worse, but have a deterministic stopping condition and do not suffer on problems where the local polytope relaxation is loose, e.g.

mrf-photomontage. While methods that can deal with higher order-terms are inferior in terms of runtimes for second order models, they can be directly applied to higher-order models, where best performing methods for second-order models are no longer applicable.

For difficult models like dtf-chinesechar and matching, the exact solutions are significantly better than approximate ones, both in terms of energy and quality. Figure 6 gives an overview of models where all (green), some (yellow) or none (red) instances have been solved to optimality within 1 h. As reported in Savchynskyy et al. (2013) more instances, including those from mrf-inpainting and mrf-photomontage, can be solved to optimality with a time limit of less than a day.

For some models combinatorial optimization methods are *faster* than currently reported state-of-the-art methods. While for small problems combinatorial methods can often be applied directly, for larger problems the reducing of the problem size by partial optimality is required to make them tractable. Solutions from these exact methods are used for evaluating the sub-optimality of the approximate solutions.

Furthermore we observe that the energy-optimal labeling is not always best in terms of application specific loss function, cf. Table 15. While methods that find global optimal solutions select the global optima—regardless of similar solutions that have a considerable higher energy—, approximative methods often tend to avoid such "isolated" solutions and prefer more "consistent" modes.

While this shows that full evaluation of models is only possible in presence of optimal solvers, it also raises the question if approximative methods are preferable when they are not so sensitive to optimal "outliers" or if the models itself need to be improved. The answer to this question might vary for different applications and models and as we have shown, for many models the energy correlates quite well with the loss functions.

Methods based on LP-relaxations over the local polytope often lead to empirically good solutions and are in general not restricted to special subclasses but are also not the fastest ones. Recently, Prua and Werner (2013) showed that solving LPs over the local polytope is in general as hard as solving general LPs, which are of high polynomial complexity. When the local polytope relaxation is too weak, tightening the polytope can help a lot, multicuts for Potts models and MPLP-C for the matching instances are examples of that kind.

While our study covers a broad range of models used nowadays (without being all-embracing), the models used in the last decade might have been biased by solvers that are available and work well. Consequently, second order Potts or truncated convex regularized models, as considered in Szeliski et al. (2008), were in the focus of research. In this study we show alternative methods that can deal with more complex models, including higher order and more densely structured models, cf. dtf-chinesechar, matching or protein-prediction.

With availability of more general optimization methods we hope to stimulate the use of complex and powerful discrete models. This may then inspire the development of new, efficient approximative methods that can meet hard time-constraints in real world applications.

Acknowledgments We thank Rick Szeliski and Pushmeet Kohli for inspiring discussions. This work has been supported by the German Research Foundation (DFG) within the program "Spatio- / Temporal Graphical Models and Applications in Image Analysis", Grant GRK 1653.

References

Achterberg, T., Koch, T., & Martin, A. (2005). Branching rules revisited. *Operations Research Letters*, *33*(1), 42–54.

Alahari, K., Kohli, P., & Torr, P. H. S. (2008). Reduce, reuse and recycle: Efficiently solving multi-label MRFs. In: *CVPR*.

Alahari, K., Kohli, P., & Torr, P. H. S. (2010). Dynamic hybrid algorithms for MAP inference in discrete MRFs. *IEEE PAMI*, *32*(10), 1846–1857.

Andres, B., Beier, T., & Kappes, J. H. (2014). OpenGM2. http://hci.iwr.uni-heidelberg.de/opengm2/.

Andres, B., Beier, T., & Kappes, J. H. (2012). OpenGM: A C++ library for discrete graphical models. ArXiv e-prints. http://arxiv.org/abs/1206.0111.

Andres, B., Kappes, J. H., Beier, T., Köthe, U., & Hamprecht, F. A. (2011). Probabilistic image segmentation with closedness constraints. In *ICCV*.

Andres, B., Kappes, J. H., Beier, T., Köthe, U., & Hamprecht, F. A. (2012). The lazy flipper: Efficient depth-limited exhaustive search in discrete graphical models. In *ECCV*.

Andres, B., Kappes, J. H., Köthe, U., Schnörr, C., & Hamprecht, F. A. (2010). An empirical comparison of inference algorithms for graphical models with higher order factors using OpenGM. In *DAGM*.

Andres, B., Köthe, U., Kroeger, T., Helmstaedter, M., Briggman, K. L., Denk, W., & Hamprecht, F. A. (2012). 3D segmentation of SBFSEM images of neuropil by a graphical model over supervoxel boundaries. *Medical Image Analysis*, *16*(4), 796–805. doi:10.1016/j.media.2011.11.004. http://www.sciencedirect.com/science/article/pii/S1361841511001666.

Andres, B., Kröger, T., Briggman, K. L., Denk, W., Korogod, N., Knott, G., Köthe, U., & Hamprecht, F. A. (2012). Globally optimal closed-surface segmentation for connectomics. In *ECCV*.

Batra, D., & Kohli, P. (2011). Making the right moves: Guiding alpha-expansion using local primal-dual gaps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011* (pp. 1865–1872). IEEE.

Bergtholdt, M., Kappes, J. H., Schmidt, S., & Schnörr, C. (2010). A study of parts-based object class detection using complete graphs. *IJCV*, *87*(1–2), 93–117.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, *48*(3), 259–302. doi:10.2307/2345426.

Bonato, T., Jünger, M., Reinelt, G., & Rinaldi, G. (2014). Lifting and separation procedures for the cut polytope. *Mathematical Programming A*, *146*(1–2), 351–378. doi:10.1007/s10107-013-0688-2.

Boykov, Y. (2003). Computing geodesics and minimal surfaces via graph cuts. In *ICCV*.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE PAMI*, *23*(11), 1222–1239. doi:10.1109/34.969114.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., et al. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, *20*(2), 172–188.

Călinescu, G., Karloff, H., & Rabani, Y. (2000). An improved approximation algorithm for multiway cut. *Journal of Computer and System Sciences*, *60*(3), 564–574.

Chekuri, C., Khanna, S., Naor, J., & Zosin, L. (2004). A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal of Discrete Mathematics*, *18*(3), 608–625.

Cocosco, C. A., Kollokian, V., Kwan, R. S., & Evans, A. C. (1997). Brainweb: Online interface to a 3d MRI simulated brain database. *NeuroImage*, *5*(4), S425.

IBM, ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (2013).

Delong, A., Osokin, A., Isack, H., & Boykov, Y. (2012). Fast approximate energy minimization with label costs. *International Journal of Computer Vision, 96*, 1–27. http://www.csd.uwo.ca/~yuri/Abstracts/ijcv10_lc-abs.shtml.

Elidan, G., & Globerson, A. (2011) The probabilistic inference challenge (PIC2011). http://www.cs.huji.ac.il/project/PASCAL/.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision*, *70*(1), 41–54.

Fix, A., Gruber, A., Boros, E., & Zabih, R. (2011). A graph cut algorithm for higher-order Markov random fields. In *ICCV*. doi:10.1109/ICCV.2011.6126347.

Gallagher, A. C., Batra, D., & Parikh, D. (2011). Inference for order reduction in Markov random fields. In *CVPR*.

Globerson, A., & Jaakkola, T. (2007). Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*.

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, *23*(1), 5–48. doi:10.1145/103162.103163.

Gorelick, L., Veksler, O., Boykov, Y., Ben Ayed, I., & Delong, A. (2014). Local submodular approximations for binary pairwise energies. In *Computer Vision and Pattern Recognition*.

Gould, S., Fulton, R., & Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *ICCV*.

Guignard, M., & Kim, S. (1987). Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, *39*(2), 215–228.

Hoiem, D., Efros, A. A., & Hebert, M. (2011). Recovering occlusion boundaries from an image. *IJCV*, *91*(3), 328–346.

Hutter, F., Hoos, H. H., & Stützle, T. (2005). Efficient stochastic local search for MPE solving. In L. P. Kaelbling & A. Saffiotti (Eds.), *IJCAI* (pp. 169–174).

Jaimovich, A., Elidan, G., Margalit, H., & Friedman, N. (2006). Towards an integrated protein–protein interaction network: A relational Markov network approach. *Journal of Computational Biology*, *13*(2), 145–164.

Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Lellmann, J., Komodakis, N., & Rother, C. (2013). A comparative study of modern inference techniques for discrete energy minimization problem. In *CVPR*.

Kappes, J. H., Beier, T., & Schnörr, C. (2014). MAP-inference on large scale higher-order discrete graphical models by fusion moves. In *ECCV—International Workshop on Graphical Models in Computer Vision*.

Kappes, J. H., Savchynskyy, B., & Schnörr, C. (2012). A bundle approach to efficient MAP-inference by Lagrangian relaxation. In *CVPR*.

Kappes, J. H., Speth, M., Andres, B., Reinelt, G., & Schnörr, C. (2011). Globally optimal image partitioning by multicuts. In *EMMCVPR*.

Kappes, J. H., Speth, M., Reinelt, G., & Schnörr, C. (2013). Higher-order segmentation via multicuts. ArXiv e-prints. http://arxiv.org/abs/1305.6387.

Kappes, J. H., Speth, M., Reinelt, G., & Schnörr, C. (2013). Towards efficient and exact MAP-inference for large scale discrete computer vision problems via combinatorial optimization. In *CVPR*.

Kausler, B. X., Schiegg, M., Andres, B., Lindner, M., Leitte, H., Hufnagel, L., Koethe, U., & Hamprecht, F. A. (2012). A discrete chain graph model for 3d+t cell tracking with high misdetection robustness. In *ECCV*.

Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell Systems Technical Journal*, *49*(2), 291–307.

Kim, S., Nowozin, S., Kohli, P., & Yoo, C. D. (2011). Higher-order correlation clustering for image segmentation. In *NIPS* (pp. 1530–1538).

Kim, T., Nowozin, S., Kohli, P., & Yoo, C. D. (2011). Variable grouping for energy minimization. In *CVPR* (pp. 1913–1920).

Kleinberg, J., & Tardos, É. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

Kohli, P., Ladicky, L., & Torr, P. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, *82*(3), 302–324. doi:10.1007/s11263-008-0202-0.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Cambridge: MIT Press.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *PAMI*, *28*(10), 1568–1583.

Kolmogorov, V., & Rother, C. (2006). Comparison of energy minimization algorithms for highly connected graphs. In *ECCV* (pp. 1–15).

Kolmogorov, V., & Zabih, R. (2002). What energy functions can be minimized via graph cuts? In *ECCV*. http://dl.acm.org/citation.cfm?id=645317.649315.

Komodakis, N., & Paragios, N. (2008). Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *ECCV*.

Komodakis, N., Paragios, N., & Tziritas, G. (2011). MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(3), 531–552.

Komodakis, N., & Tziritas, G. (2007). Approximate labeling via graph cuts based on linear programming. *IEEE PAMI*, *29*(8), 1436–1453. doi:10.1109/TPAMI.2007.1061.

Kovtun, I. (2003). Partial optimal labeling search for a np-hard subclass of (max, +) problems. In B. Michaelis & G. Krell (Eds.), *DAGM-Symposium*, Lecture Notes in Computer Science (Vol. 2781, pp. 402–409). Heidelberg: Springer.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford: Oxford University Press.

Lellmann, J., & Schnörr, C. (2011). Continuous multiclass labeling approaches and algorithms. *SIAM Journal of Imaging Sciences*, *4*(4), 1049–1096.

Lempitsky, V., Rother, C., Roth, S., & Blake, A. (2010). Fusion moves for Markov random field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(8), 1392–1405. doi:10.1109/TPAMI.2009.143.

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., & Xing, E. P. (2011). An augmented lagrangian approach to constrained MAP inference. In *ICML* (pp. 169–176).

Nieuwenhuis, C., Toeppe, E., & Cremers, D. (2013). A survey and comparison of discrete and continuous multi-label optimization approaches for the Potts model. *International Journal of Computer Vision*, *104*, 223–240. doi:10.1007/s11263-013-0619-y.

Nowozin, S., & Lampert, C. H. (2011). Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, *6*(3–4), 185–365.

Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B., & Kohli, P. (2011). Decision tree fields. In *ICCV* (pp. 1668–1675). IEEE.

Orabona, F., Hazan, T., Sarwate, A., & Jaakkola, T. (2014). On measure concentration of random maximum a-posteriori perturbations. In *Proc. ICML*.

Osokin, A., Vetrov, D., & Kolmogorov, V. (2011). Submodular decomposition framework for inference in associative markov networks with global constraints. In *CVPR* (pp. 1889–1896).

Otten, L., & Dechter, R. (2011). Anytime AND/OR depth-first search for combinatorial optimization. In *Proceedings of the Annual Symposium on Combinatorial Search (SOCS)*.

Papandreou, G., & Yuille, A. (2011). Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *Proc. ICCV*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Prua, D., & Werner, T. (2013). Universality of the local marginal polytope. In *CVPR* (pp. 1738–1743). IEEE.

Rother, C., Kolmogorov, V., Lempitsky, V. S., & Szummer, M. (2007). Optimizing binary MRFs via extended roof duality. In *CVPR*.

Rother, C., Kumar, S., Kolmogorov, V., & Blake, A. (2005). Digital tapestry. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 589–596). IEEE Computer Society, Washington, DC, USA. doi:10.1109/CVPR.2005.130.

Savchynskyy, B., Kappes, J. H., Swoboda, P., & Schnörr, C. (2013). Global MAP-optimality by shrinking the combinatorial search area with convex relaxation. In *NIPS*.

Savchynskyy, B., & Schmidt, S. (2013). Getting feasible variable estimates from infeasible ones: MRF local polytope study. In *Workshop on Inference for Probabilistic Graphical Models at ICCV 2013*.

Savchynskyy, B., & Schmidt, S. (2014). Getting feasible variable estimates from infeasible ones: MRF local polytope study. In *Advanced structured prediction*. MIT Press.

Savchynskyy, B., Schmidt, S., Kappes, J. H., & Schnörr, C. (2012). Efficient MRF energy minimization via adaptive diminishing smoothing. *UAI*, *2012*, 746–755.

Schlesinger, M. (1976). Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, *4*, 113–130.

Sontag, D., Choe, D. K., & Li, Y. (2012). Efficiently searching for frustrated cycles in MAP inference. In N. de Freitas & K. P. Murphy (Eds.) *UAI* (pp. 795–804). AUAI Press.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., et al. (2008). A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE PAMI*, *30*(6), 1068–1080. doi:10.1109/TPAMI.2007.70844.

Tarlow, D., Batra, D., Kohli, P., & Kolmogorov, V. (2011). Dynamic tree block coordinate ascent. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Verma, T., & Batra, D. (2012). Maxflow revisited: An empirical comparison of maxflow algorithms for dense vision problems. In *BMVC* (pp. 1–12).

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2005). MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Transactions on Information Theory*, *51*(11), 3697–3717.

Werner, T. (2007). A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(7), 1165–1179. doi:10.1109/TPAMI.2007.1036.

Wesselmann, F., & Stuhl, U. (2012). Implementing cutting plane management and selection techniques. Tech. rep., University of Paderborn. http://www.optimization-online.org/DB_HTML/2012/12/3714.html.

Woodford, O. J., Torr, P. H. S., Reid, I. D., & Fitzgibbon, A. W. (2009). Global stereo reconstruction under second order smoothness priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(12), 2115–2128.

Yanover, C., Schueler-Furman, O., & Weiss, Y. (2008). Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology*, *15*(7), 899–911.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2004). Constructing free energy approximations and generalized belief propagation algorithms. MERL Technical Report, 2004–040. http://www.merl.com/papers/docs/TR2004-040.