MAP-Inference on Large Scale Higher-Order Discrete Graphical Models by Fusion Moves

Jörg Hendrik Kappes*, Thorsten Beier* and Christoph Schnörr

Heidelberg Collaboratory for Image Processing, Heidelberg University

Abstract. Many computer vision problems can be cast into optimization problems over discrete graphical models also known as Markov or conditional random fields. Standard methods are able to solve those problems quite efficiently. However, problems with huge label spaces and or higher-order structure remain challenging or intractable even for approximate methods.
We reconsider the work of Lempitsky et al. 2010 on fusion moves and apply it to general discrete graphical models. We propose two alternatives for calculating fusion moves that outperform the standard in several applications. Our generic software framework allows us to easily use different proposal generators which spans a large class of inference algorithms and thus makes exhaustive evaluation feasible.
Because these fusion algorithms can be applied to models with huge label spaces and higher-order terms, they might stimulate and support research of such models which may have not been possible so far due to the lack of adequate inference methods.

1 Introduction

Many computer vision problems can be cast into optimization problems over discrete graphical models also known as Markov or conditional random fields. While standard methods are able to solve those problems quite efficiently, problems with huge label spaces and or higher-order structure are still challenging and even approximate methods do not scale well.

Consequently, research has focused on models with moderate order and small label spaces [1–3], models with huge but decomposable label spaces [4], or higher-order models that can be reformulated into second order models with additional auxiliary variables [5–7].

A more generic approach to deal with large label spaces has been suggested by Lempitsky et al. [8]. Starting with an initial labeling, they generate an alternative proposal and search for a better labeling within the subspace of labeling spanned by the current and the proposed labeling. This step is called *move*, since the current labeling is moved within the subspace without increasing the energy. Except for some special cases, e.g. [9], finding the optimal move for a given proposal is NP-hard. The common way to calculate a move exploits that the problem is binary and QPBO is used to calculate a labeling with a persistency certificate [10, 11]. For all persistent variables we can change the current

* these authors contributed equally to this work

label to the persistent one and do not increase the energy. This procedure has been generalized to higher-oder problems by reducing the higher-order binary subproblems two second-order ones and additional auxiliary variables [12–14].

A complementary part of fusion algorithms that need to be specified is the generation of proposal. Proposal generators can be generic or problem specific. As discussed in [8] a good proposal should have a high *quality* and the proposals should be *diverse* among each other to allow various moves.

Except for fusion with simple α-proposals in [1], fusion moves have not been considered in recent benchmarks [2, 1, 15]. This might be caused by the lack of a publicly available implementations and the option to choose *any* generator. Likewise, in many applications fusion moves with less generic *problem specific* proposal generators have been used.

Contribution: (1) The first *publicly available generic implementation of fusion moves*. It supports user defined proposal generators and is embedded into the OpenGM-Library [16]. **(2)** *Two novel methods for calculation fusion moves* that outperform QPBO in several settings. **(3)** We show how *improved any-time performance* of state-of-the-art methods can be obtained by embedding them into the fusion framework. **(4)** A *detailed evaluation* of proposal generators and fusion algorithms on recent published and new benchmark datasets.

Outline: We start in Sec. 2 with the mathematical formulation of the problem and fusion moves and present in Sec. 3 novel and state-of-the-art methods to calculate them. In Sec. 4 we present some generic proposal generators. In the experimental section 5 we evaluate the performance of fusion-methods and proposal generators on recent benchmark datasets and conclude in Sec. 6.

2 Problem Formulation

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 [17, 18]. 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

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

where the assignment of the variable x is also known as the labeling. We consider the problem to find a labeling with minimal energy, i.e.

$$\hat{x} \in \arg\min_{x\in X} J(x). \tag{2}$$

Algorithm 1 Fusion Based Algorithms

procedure FUSION-BASED-INFERENCE(GEN,FUSE,J,X)
  $x^0 \leftarrow$ initial state form X
  $n \leftarrow 0$ ▷ Number of moves
  $m \leftarrow 0$ ▷ Number of moves without progress
  while $m < m_{\max}$ and $n < n_{\max}$ **do**
    $n \leftarrow n + 1$
    $x' \leftarrow GEN(x^{n-1}, J, X)$ ▷ Generate proposal
    if $J(x^{n-1}) \leq J(x')$ **then**
      $x^n \leftarrow FUSE(x^{n-1}, x', J)$
    else
      $x^n \leftarrow FUSE(x', x^{n-1}, J)$
    end if
    if $J(x^n) \leq J(x^{n-1})$ **then**
      $m \leftarrow 0$ ▷ Reset counter
    else
      $m \leftarrow m + 1$ ▷ Increment counter
    end if
  end while
  return x^n
end procedure

This labeling is a maximum-a-posteriori (MAP) solution of a Gibbs distribution $p(x) = \exp\{-J(x)\}/Z$ defined by the energy $J(x)$. Here, Z normalizes the distribution.

To avoid the large labeling space X, fusion moves optimize only over the subspace $X' \subset X$, which is defined as the set of labelings spanned by the current x^{cur} and proposed x^{pro} labeling,

$$X'(x^{\text{cur}}, x^{\text{pro}}) = \{x \in X \,|\, \forall i : x_i \in \{x_i^{\text{cur}}, x_i^{\text{pro}}\}\}\,. \tag{3}$$

The set of all feasible moves, i.e. that decrease the energy, is given by

$$X^{\text{MOVE}}(x^{\text{cur}}, x^{\text{pro}}) = \{x \in X' \,|\, J(x) \leq J(x^{\text{cur}})\}\,. \tag{4}$$

Since finding the optimal move (optimal labeling in X^{MOVE}) is NP-hard we can not expect to find the optimal move in polynomial time. This is why we define and consider fusion-operators $\text{FUSE}(x, x', J)$ which return an element of $X^{\text{MOVE}}(x, x')$.

Given a proposal generator *GEN*, a fusion-operator *FUSE*, an objective function J, and a state-space X we can define the class of *Fusion-Algorithms*, as shown in Alg. 1. They all monotonically decrease the energy. As stopping condition we will use the maximal number of moves $n_{\max}$ as well as the maximal length of a sequence of non-improving moves $m_{\max}$. Algorithms in this family are distinguished by the fusion operation and the proposal generator that they employ, which we will discuss in the next two sections.

Algorithm 2 Fusion Moves

Require: $J(x) \leq J(x')$
Ensure: $J(\hat{x}) \leq J(x)$

procedure $\text{FUSE}_{\text{QPBO}}(x, x', J)$
  $\bar{X} \leftarrow \{\bar{x} \in X | \forall i : \bar{x}_i \in \{x_i, x'_i\}\}$ ▷ Build Boolean subspace of X
  $\hat{x} \leftarrow QPBO(J(\cdot), \bar{X})$ ▷ Solve relaxation for persistent states
  $\hat{x}_i \leftarrow x_i \quad \forall \hat{x}_i = \frac{1}{2}$ ▷ Replace non-persistent states
  return $\hat{x}$
end procedure

procedure $\text{FUSE}_{\text{LF2}}(x, x', J)$
  $\bar{X} \leftarrow \{\bar{x} \in X | \forall i : \bar{x}_i \in \{x_i, x'_i\}\}$ ▷ Build Boolean subspace of X
  $LazyFlipper.setStartingPoint \leftarrow x$ ▷ Set starting point
  $LazyFlipper.searchDepth \leftarrow 2$ ▷ Set search depth
  $\hat{x} \leftarrow LazyFlipper(J(\cdot), \bar{X})$ ▷ Lazy Flipper improves the current state
  return $\hat{x}$
end procedure

procedure $\text{FUSE}_{\text{ILP}}(x, x', J)$
  $\bar{X} \leftarrow \{\bar{x} \in X | \forall i : \bar{x}_i \in \{x_i, x'_i\}\}$ ▷ Build Boolean subspace of X
  $RILP.setStartingPoint \leftarrow x$ ▷ Add the current best into the solution pool
  $\hat{x} \leftarrow RILP(J(\cdot), \bar{X})$ ▷ ILP improves the current state
  return $\hat{x}$
end procedure

procedure $\text{FUSE}_{\text{BASE}}(x, x', J)$
  return $\arg\min_{\bar{x} \in \{x, x'\}} J(\bar{x})$
end procedure

3 Fusion Move Operators

As discussed in the previous section an elementary part of fusion-algorithms is the fusion-operator *FUSE*. In this section we discuss different operators and present two novel fusion-operators. The corresponding pseudo code is shown in Alg. 2. The returned labeling is guaranteed to have an energy lower or equal to the energy of the current labeling and the proposed labeling.

QPBO Fusion: The standard fusion-operator $FUSE_{QPBO}$ was proposed by Lempitsky et al. [8] and generalized to the higher-order case by Ishikawa [12] and Fix et al. [13], which reduce in a preprocessing step the higher-order sub-problem into a second-order one. For the second-order problem the local polytope relaxation is solved by QPBO [11] and persistency is used to improve the current best labeling. While this can be done in polynomial time, there is in general no guaranty that we obtain persistency for any variable. However, empirically this fusion-operator works well and is therefore widely considered as state-of-the-art.

Lazy Flipping Fusion: An alternative ansatz is to improve the current labeling by local flipping. In the case when only one variable is flipped at the same time this boils down to ICM [19]. Lempitsky et al. [8] show that ICM-Fusion does not work well. However, Andres et al. have suggested a generalization of ICM to multi-variable flipping, called Lazy Flipper [20]. Lazy Flipper can handle higher-order terms directly, hence order reduction is not required. In the present work we use lazy flipping with search depth two defining the fusion-operator $FUSE_{LF2}$ and initialize it with the current best labeling. The initial labeling is sequentially improved by flips of less or equal than two variables until no further improvement is possible. Obviously, the final labeling will not be worse than the initial one. While Lazy Flipping does not require the existence of persistent variables, it stops if improvements can only be obtained by flipping too many variables simultaneously.

Optimal Fusion: Recently, Kappes et al. [21] have shown that many discrete optimization problems in computer vision can be solved exactly by first reducing the problem size by partial optimality and than solving the smaller remaining problem by advanced methods like integer linear programming (ILP). In the case that the remaining problem splits in several connected components, those can be handled independently which gives additional speed up. The fusion-operator $FUSE_{ILP}$ is defined by using QPBO [11] with the reduction of Fix [13] for higher-order models to obtain partial optimality and solving the connected components of the remaining problem by the Cplex ILP-solver [22]. By adding the current best solution in the solution pool of the ILP solver it is guaranteed that the final solution will not be worse. Furthermore, this provides a good starting point and an upper bound. Since the remaining ILPs can still be quite hard, we interrupted the solver after 100 seconds and return the best labeling from the solution pool. Consequently, in our experiments a move is optimal if it is calculated within 100 seconds.

Base Fusion: To determine the impact of fusion-operations, we also define a naive operator $FUSE_{BASE}$, which returns the better of the two labelings

$$\bar{x} = \arg\min_{\bar{x}\in\{x,x'\}} J(\bar{x}). \tag{5}$$

This fusion-operator does only profit from the proposal quality and not from their diversity.

4 Generating Proposals

The second major component of a fusion-algorithm is the generation of proposals. On the one hand, proposals should be of high quality with respect to the energy function $J(\cdot)$. On the other hand, they should be also diverse among each other and cheap to calculate. Proposal generators can be clustered into four groups: (i) inference-based generators, (ii) randomized generators, (iii) deterministic generators, and (iv) application specific generators.

Pseudo-code for (i)–(iii) is given in Alg. 3. We do not consider application specific generators in the present work because they are none generic and require more data than just the objective function.

Algorithm 3 Proposal Generators

```
1: procedure RandomGen(x, J, X)
Require: ∀i ∈ V : P_i(x_i)                          ▷ Shared for all moves
2:     for i ∈ V do
3:         x̂_i ∼_{P_i(x_i)} X_i
4:     end for
5:     return x̄
6: end procedure

7: procedure InfGen(x, J, X)
Require: INF ← INF(J, X)                            ▷ Shared for all moves
8:     INF.runOneStep
9:     X̄ ← INF.getLabeling
10:    return x̄
11: end procedure

12: procedure DeterministicGen(x, J, X)
Require: n ← 0                                      ▷ Shared for all moves
13:    X̄ ← gen(x, n, X)
14:    n ← n + 1
15:    return x̄
16: end procedure
```

Inference-Based Generators: For the cartographic label placement problem Lempitsky et al. [8] used the labelings that Loopy Belief Propagation (LBP) generates after each iteration as proposals. They obtained a result superior to state-of-the-art for this problem instance.

This result was not further generalized or tested for other problems in later work. However, it is very appealing since methods based on linear programming relaxations like TRWS [23], MPLP [24] or approximative message passing methods like LBP [25], BPS [23] provide after each iteration good proposals close to the optimal one. The diversity is generated by the heuristic rounding procedure. Fusion moves can profit from this diversity and overcome failures caused by greedy rounding if this failures are not present in all iterations.

We use the visitor concept of OpenGM [16] and inject the fusion operation after each algorithmic unit. This allows using any OpenGM-inference method as proposal generator with a few lines of code. In the present work we show results for TRWS, MPLP, BPS and LBP with different damping. MPLP and LBP can also deal with higher-order problems.

Randomized Generators: A general way to generate diverse proposals is to sample those from a distribution P. The disadvantage of such generators is that the proposals usually have bad quality. One can try to alleviate this by prior knowledge. We consider the following sampling distributions, which all defined independently for each variable. For problems with arbitrary structure

we consider *uniform random distributions* (P_U)

$$P_i(x_i) = \frac{1}{|X_i|}, \tag{6}$$

and *local marginal approximations* (P_L) which estimate for a given temperature T first order marginals from unary terms $\bar{f}_i$ by

$$P_i(x_i) \propto \exp\{-T \cdot \bar{f}_i(x_i)\}. \tag{7}$$

For $T \to 0$ the distribution becomes uniform and for $T \to \infty$ all its mass concentrated in the local mode. When local data terms a weak or misleading the distribution is not helpful.

We also follow the idea used in [12, 13], which blur the current labeling on the image grid and sample proposals around the "blurred labeling". Of course this is only useful if labels have the same meaning for all variables. Empirically we observe no advantage by repeating the blurring in each iteration if the standard variation of the Gaussian blur is large. We suggest to blur the unary terms instead of the labeling, this is also more robust to missing unary terms and uncertain information. Furthermore, blurring has to be done only once. For each variable we obtain a Gaussian blurred unary term label-wise

$$\bar{f}_i^B(x_i) = GaussianBlur_\sigma(\bar{f}(x_i))_i, \tag{8}$$

$$\bar{x}_i^B = \arg\max_{x_i} \bar{f}_i^B(x_i). \tag{9}$$

As in [12, 13] we sampling uniformly (P_{UB}) from

$$P_i(x_i) \propto \begin{cases} 1 & \text{if even round or } x_i \in [\bar{x}_i^B - 1.5\sigma, \bar{x}_i^B + 1.5\sigma] \\ 0 & \text{else} \end{cases} \tag{10}$$

Alternatively we can use the blurred unaries for a *local blurred marginal approximations* (P_{LB}) as in the non-blurred case

$$P_i(x_i) \propto \exp\{-T \cdot \bar{f}_i^B(x_i)\}. \tag{11}$$

Deterministic Generators: Deterministic generators provide very simple proposals with low workload. The proposals depend on the current labeling x and iteration n. For deterministic generators we determine the number of moves with no improvements $m_{\max}$ for which immediate termination will have no effect on the final solution. An example is the generalization of α-*Expansion* [26] where $m_{\max} = \max_{i \in V} |X_i|$. The proposal $\hat{x}$ in iteration n takes the label $\alpha(n) = n \mod m_{\max}$ if possible, i.e.

$$\hat{x}_i = \begin{cases} \alpha(n) & \text{if } \alpha(n) \in X_i \\ x_i & \text{else} \end{cases} \tag{12}$$

Another example are $\alpha\beta$-*Swaps* [26] which can be generalized to arbitrary discrete problems. Here in each step n variables that have the labels $\alpha(n)$ and

Table 1: Overview of the used models and the number of variables (# variables), number of labels (# labels), model order (order), number of instances (# instances) and temperature used for determine local marginals.

modelname	# variables	# labels	order	# instances	used temperature
Field of Experts	38801	256	4	100	0.1
MRF Inpainting	65536	256	2	2	0.001
Protein Folding	1972	503	2	21	0.1
Protein Prediction	14441	2	3	8	1
DTF Inpainting	17856	2	2	100	0.1
Matching	21	21	2	4	0.1
Cell Tracking	41134	2	9	1	0.1

$\beta(n)$ are changed to $\beta(n)$ and $\alpha(n)$ if possible, respectively. Here $m_{\max} = 0.5 \cdot \max_{i \in V} |X_i| \cdot (\max_{i \in V} |X_i| - 1)$.

$$\hat{x}_i = \begin{cases} \alpha(n) & \text{if } x_i = \beta(n) \text{ and } \alpha(n) \in X_i \\ \beta(n) & \text{if } x_i = \alpha(n) \text{ and } \beta(n) \in X_i \\ x_i & \text{else} \end{cases} \tag{13}$$

5 Evaluation

We compare the combination of fusion operations and proposal generators for different graphical models benchmarks [2, 1, 15] and the FoE-dataset [27]. All this instances are or will be made publicly available in the OpenGM-format.

We run all combinations for 1000 iterations ($n_{\max} = 1000$) and maximal 900 seconds on a Core i7-2600K with 3.40 GHz single-threaded. We stop after 50 moves without improvement ($m_{\max} = 50$). Stopping condition of deterministic methods are the deterministic default. Due lack of space we add the complete results as supplementary material and show only selected combinations here. The used temperature parameter for the sampling distributions and an overview of the models is given in Tab. 1.

We report the energy value, averaged over all model instances, of the best labeling after 10, 60 and 600 seconds as well as for the final labeling. Additionally we report the mean runtime and the number of iterations or moves. The best value among all fusion-algorithms in each time slot is marked green, and the fusion-operation which give the best mean energy for a given proposal-generator blue. Additionally, we add results of state-of-the-art-methods to the tables, if those results were available. If the best of those beats all fusion algorithm it is marked red.

Field of Experts: Field of experts were introduced by Roth and Black [27], which use higher-order terms to expressive image priors that capture the statistics of natural scenes. Field of export models have become a standard benchmark for fusion moves [12, 13]. We follow the experimental setup used in [12, 13] and take the 100 test images from the BSD300 [28], downscale them by a factor of two and add Gaussian noise with standard deviation $\sigma = 20$. The energy function includes unary terms penalize the L_1-distance of the 256 labels/colors to

the noisy pixel color and fourth order experts learned and kindly provided by Roth and Black [27].

Classical QPBO-based fusion is clearly inferior to LazyFlipper-based, c.f. Tab. 2 and Fig. 1(a). For the α-expansion generator QPBO-fusion does a bad job as reported in [12]. When we switch to LazyFlipper-based fusion it is still not best but comparable to other combinations. Using optimal moves does not improve the results significantly. The moves are only marginal better but slower. Overall best results are obtained when sampling from the distributions base on non-blurred unary terms.

Table 2: For *field of experts* instances $\text{FUSION}_{\text{LF2}}$ overall performs best.

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
α-Exp-$\text{FUSION}_{\text{ILP}}$	115331.95	112908.90	108011.80	105001.75	941.28 sec	27.30
α-Exp-$\text{FUSION}_{\text{LF2}}$	109604.69	76950.74	35553.15	34958.88	709.57 sec	999.88
α-Exp-$\text{FUSION}_{\text{QPBO}}$	113027.96	107330.34	56267.95	54571.25	900.91 sec	541.81
P_{UB}-$\text{FUSION}_{\text{ILP}}$	107585.42	105930.25	37603.67	35351.69	903.09 sec	220.24
P_{UB}-$\text{FUSION}_{\text{LF2}}$	71918.21	38631.97	32925.36	32848.61	695.85 sec	1000.00
P_{UB}-$\text{FUSION}_{\text{QPBO}}$	97796.97	47536.08	33481.48	33090.60	872.46 sec	899.96
P_L-$\text{FUSION}_{\text{ILP}}$	87010.93	41320.95	32779.26	32637.81	899.81 sec	806.71
P_L-$\text{FUSION}_{\text{LF2}}$	54960.13	35583.31	32619.64	32586.99	701.58 sec	1000.00
P_L-$\text{FUSION}_{\text{QPBO}}$	57337.32	35918.37	32646.95	32613.20	688.93 sec	1000.00
P_U-$\text{FUSION}_{\text{ILP}}$	81230.66	41289.75	32936.93	32779.52	806.42 sec	999.44
P_U-$\text{FUSION}_{\text{LF2}}$	64828.40	38662.98	32882.46	32782.16	736.44 sec	996.45
P_U-$\text{FUSION}_{\text{QPBO}}$	63305.54	38500.68	32871.21	32797.15	699.14 sec	1000.00

Inpainting: We consider the two inpainting problems from [3] which have 256 labels. For these instances TRWS followed by local search is currently the leading method [1]. These methods make use of the convex regularizer and apply distance transform [29] for good any time performance. Fusion algorithms did not work well within 1000 iterations except TRWS is used as generator. This agrees with the results reported in [3] where α-expansion also needed much more iterations and has a simple explanation. The unaries and the regularizer

Table 3: For the *inpainting* problems fusion-algorithms improve the performance of TRWS. Random generators do not work well here.

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
TRWS	26481554.50	26465539.50	26464769.50	26464759.00	632.40 sec	944.50
TRWS-LF2	∞	∞	∞	26463829.00	3009.52 sec	–
P_U-$\text{FUSION}_{\text{BASE}}$	420556187.50	420556187.50	420556187.50	420556187.50	2.40 sec	50.00
P_U-$\text{FUSION}_{\text{ILP}}$	60296247.50	38570409.50	34890334.50	34890334.50	196.09 sec	1000.00
P_U-$\text{FUSION}_{\text{LF2}}$	100770607.50	45696051.50	35241978.50	34985385.50	501.38 sec	1000.00
P_U-$\text{FUSION}_{\text{QPBO}}$	50696441.50	36367339.50	34904322.00	34904322.00	119.94 sec	1000.00
TRWS-$\text{FUSION}_{\text{BASE}}$	26481554.50	26465534.50	26465416.50	26465416.50	103.48 sec	163.00
TRWS-$\text{FUSION}_{\text{ILP}}$	26476904.00	26464727.50	26464158.00	26464158.00	217.59 sec	318.50
TRWS-$\text{FUSION}_{\text{LF2}}$	26482403.50	26465290.00	26464904.50	26464904.50	206.98 sec	276.00
TRWS-$\text{FUSION}_{\text{QPBO}}$	26476820.00	26464728.50	26464158.00	26464158.00	214.05 sec	318.50

are based on squared differences. This make them very picky and selective. This limits the set of improving moves for random proposals.

Protein Folding: The protein folding instances [30] have a moderate number of variables, but are fully connected and have for some variables huge label spaces. Recently it has been shown [15], that sequential Belief Propagation (BPS) gives very good results near optimality. Using BPS as generator fusion obtain better and faster results than BPS alone and advanced combinatorial methods like CombiLP [31]. For other generators the results are worse but still comparable with other methods and always improve the baseline significantly, c.f. Tab.4 and Fig. 1(c).

Table 4: For the *protein folding* instances BPS-FUSION leads to better results and is more than ten times faster than BPS.

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
BPS	−5958.72	−5958.72	−5958.72	−5958.72	25.34 sec	1000.00
LBP	−5817.90	−5841.98	−5872.91	−5872.91	183.53 sec	1000.00
TRWS	−5735.86	−5799.52	−5846.86	−5846.86	118.17 sec	675.48
CombiLP	∞	∞	−5822.45	−5911.12	568.86 sec	–
BPS-FUSION$_{\text{BASE}}$	−5958.37	−5958.37	−5958.37	−5958.37	1.63 sec	57.24
BPS-FUSION$_{\text{ILP}}$	−5959.82	−5959.82	−5959.82	−5959.82	1.69 sec	57.05
BPS-FUSION$_{\text{LF2}}$	−5959.48	−5959.48	−5959.48	−5959.48	1.70 sec	57.05
BPS-FUSION$_{\text{QPBO}}$	−5959.82	−5959.82	−5959.82	−5959.82	1.61 sec	57.05
LBP-0.5-FUSION$_{\text{BASE}}$	−5926.10	−5944.87	−5944.87	−5944.87	16.95 sec	86.24
LBP-0.5-FUSION$_{\text{ILP}}$	−5928.60	−5946.35	−5946.35	−5946.35	16.19 sec	80.67
LBP-0.5-FUSION$_{\text{LF2}}$	−5926.10	−5944.87	−5944.87	−5944.87	16.99 sec	86.24
LBP-0.5-FUSION$_{\text{QPBO}}$	−5928.60	−5945.28	−5945.28	−5945.28	16.11 sec	81.86

Protein Prediction: The protein prediction instances [32] include sparse third-order binary models. We beat the best performing method from the benchmark [15] which is LBP with damping 0.5 followed by Lazy Flipping of search depth 2, by using damped LBP as generator and QPBO or ILP for fusion, c.f. Tab.5 and Fig. 1(d).

Table 5: For the *protein-prediction* problems the FUSION$_{\text{ILP}}$ leads to better results even with random proposals.

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
LBP-0.5	53407.52	52974.98	52974.98	52974.98	60.97 sec	766.88
LBP-LF2	∞	∞	52942.95	52942.95	69.86 sec	–
LBP-0.5-FUSION$_{\text{BASE}}$	52971.53	52971.53	52971.53	52971.53	6.22 sec	110.50
LBP-0.5-FUSION$_{\text{ILP}}$	52827.89	52821.38	52821.38	52821.38	9.64 sec	110.12
LBP-0.5-FUSION$_{\text{LF2}}$	52971.53	52971.53	52971.53	52971.53	6.22 sec	110.50
LBP-0.5-FUSION$_{\text{QPBO}}$	52826.64	52826.64	52826.64	52826.64	6.20 sec	124.12
P_U-FUSION$_{\text{BASE}}$	97071.97	97071.97	97071.97	97071.97	0.76 sec	50.00
P_U-FUSION$_{\text{ILP}}$	95886.12	95787.15	55531.88	55509.32	380.11 sec	689.25
P_U-FUSION$_{\text{LF2}}$	58622.95	58622.81	58622.81	58622.81	13.62 sec	87.25
P_U-FUSION$_{\text{QPBO}}$	75582.10	66164.13	65933.02	65933.02	58.27 sec	955.00

DTF Chinese Characters: A challenging second-order binary problem is using decision tree fields (DTF) for inpainting [33, 1]. While advanced combinatorial solvers (MCBC) [21] give best performance [1], they are slow. The best fast solver in [1] was sequential belief propagation (BPS). Recently, Gorelick et al. presented a fast and accurate alternative based on local submodular approximations with trust region terms (LSA-TR) [34]. While we do not beat LSA-TR we improve other methods significantly. This indicates that fusion algorithms are also useful for hard problems – especially if ILP-Fusion is used – and improve final solutions and any-time performance, c.f. Tab. 6 and Fig. 1(b). Note that contrary to MCBC and LSA-TR, Fusion algorithms are not limited to binary models.

Table 6: For the *DTF Chinese characters* fusion based methods has not beaten LSA-TR. However, we get quite close and improve standard methods. **Results was taken from the original papers and not reproduced.*

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
TRWS	−49512.31	−49514.04	−49514.06	−49514.06	112.37 sec	856.13
BPS-TAB	−49536.02	−49537.63	−49538.16	−49538.16	78.65 sec	1000.00
LSA-TR*	−49547.61	−49547.61	−49547.61	−49547.61	0.21 sec	−−
MCBC-pct*	−−	−−	−−	−49550.10	2053.89 sec	−−
α-Exp-FUSION$_{\text{BASE}}$	−49434.39	−49434.39	−49434.39	−49434.39	0.01 sec	2.00
α-Exp-FUSION$_{\text{ILP}}$	−49434.39	−49434.39	−49527.97	−49528.00	273.90 sec	4.40
α-Exp-FUSION$_{\text{LF2}}$	−49495.76	−49496.83	−49496.83	−49496.83	13.39 sec	3.50
α-Exp-FUSION$_{\text{QPBO}}$	−49499.09	−49501.69	−49501.69	−49501.69	7.63 sec	11.53
BPS-FUSION$_{\text{BASE}}$	−49535.10	−49535.10	−49535.10	−49535.10	5.17 sec	81.73
BPS-FUSION$_{\text{ILP}}$	−49504.33	−49504.36	−49542.08	−49543.30	447.73 sec	40.79
BPS-FUSION$_{\text{LF2}}$	−49535.69	−49535.69	−49535.69	−49535.69	6.27 sec	75.30
BPS-FUSION$_{\text{QPBO}}$	−49535.82	−49535.82	−49535.82	−49535.82	4.90 sec	74.58
TRWS-FUSION$_{\text{BASE}}$	−49512.19	−49512.21	−49512.21	−49512.21	8.59 sec	71.63
TRWS-FUSION$_{\text{ILP}}$	−49476.91	−49482.33	−49535.98	−49537.55	543.30 sec	41.83
TRWS-FUSION$_{\text{LF2}}$	−49528.15	−49529.41	−49529.41	−49529.41	16.06 sec	63.29
TRWS-FUSION$_{\text{QPBO}}$	−49531.64	−49532.29	−49532.29	−49532.29	17.03 sec	69.55

Matching: We also consider the matching instances from [1] which are small but very hard. In [1] it has been shown that α-expansion proposals are not an adequate proposal choice. This is no longer true for other proposals including random ones. However fusion moves often run into a labeling which is hard to escape. If such a labeling is feasible, i.e. represents a one-to-one match, a proposal has to support a cyclic swap of the labels in order to fulfill the one-to-one matching constraint and improve the energy in order to escape. Consequently, it is less likely to find global optimal solutions.

Table 7: For *matching* problems results could only be marginally improved, since the feasible move space is small in most iterations.

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
TRWS	43.38	43.38	43.38	43.38	0.35 sec	253.00
MPLP-C	21.22	21.22	21.22	21.22	4.63 sec	145.25
LBP-0.5-FUSION$_{\text{BASE}}$	26.87	26.87	26.87	26.87	0.16 sec	77.25
LBP-0.5-FUSION$_{\text{ILP}}$	24.56	24.56	24.56	24.56	0.19 sec	78.25
LBP-0.5-FUSION$_{\text{LF2}}$	26.87	26.87	26.87	26.87	0.16 sec	77.25
LBP-0.5-FUSION$_{\text{QPBO}}$	27.80	27.80	27.80	27.80	0.13 sec	66.00
P_U-FUSION$_{\text{ILP}}$	43.36	43.36	43.36	43.36	1.04 sec	243.00
P_U-FUSION$_{\text{LF2}}$	55.22	55.22	55.22	55.22	0.30 sec	216.00
P_U-FUSION$_{\text{QPBO}}$	50.78	50.78	50.78	50.78	0.03 sec	232.25
TRWS-FUSION$_{\text{BASE}}$	43.38	43.38	43.38	43.38	0.08 sec	59.25
TRWS-FUSION$_{\text{ILP}}$	40.97	40.97	40.97	40.97	0.37 sec	67.75
TRWS-FUSION$_{\text{LF2}}$	42.00	42.00	42.00	42.00	0.13 sec	67.25
TRWS-FUSION$_{\text{QPBO}}$	40.97	40.97	40.97	40.97	0.09 sec	67.75

Cell-Tracking: The tracking model considered in [1] include binary variables and terms of order up to 9. While ILP-solvers solves this instance to optimality very efficiently one should not expect that this will hold for larger models. In such scenarios relaxations would be an alternative but those suffer from the soft-constraints and labelings generated by rounding might violate those. In such situations Fusion can help a lot and provide early close-to-optimal solutions.

Table 8: For the *cell-tracking* instance we obtain faster good results only marginally worse than the optimum.

algorithm	value				time	it
	(10 sec)	(60 sec)	(600 sec)	(end)	(end)	(end)
LBP	107515639.76	107515319.56	107515319.56	107515319.56	80.70 sec	1000.00
ILP	45364196.24	7514421.21	7514421.21	7514421.21	13.78 sec	0.00
LBP-0.5-FUSION$_{\text{BASE}}$	7822517.15	7822517.15	7822517.15	7822517.15	10.00 sec	89.00
LBP-0.5-FUSION$_{\text{ILP}}$	7518000.15	7514751.98	7514751.98	7514751.98	26.69 sec	234.00
LBP-0.5-FUSION$_{\text{LF2}}$	7822517.15	7822517.15	7822517.15	7822517.15	9.83 sec	89.00
LBP-0.5-FUSION$_{\text{QPBO}}$	10324281.39	10314354.13	10314354.13	10314354.13	24.96 sec	227.00
LBP-FUSION$_{\text{BASE}}$	7518099.53	7518099.53	7518099.53	7518099.53	11.83 sec	111.00
LBP-FUSION$_{\text{ILP}}$	7515318.79	7515029.55	7515029.55	7515029.55	17.49 sec	145.00
LBP-FUSION$_{\text{LF2}}$	7518099.53	7518099.53	7518099.53	7518099.53	12.11 sec	111.00
LBP-FUSION$_{\text{QPBO}}$	7516031.12	7515029.55	7515029.55	7515029.55	15.96 sec	145.00
P_U-FUSION$_{\text{BASE}}$	58794439.99	58794439.99	58794439.99	58794439.99	2.17 sec	50.00
P_U-FUSION$_{\text{ILP}}$	14033539.27	7791724.31	7531572.24	7531572.24	643.67 sec	304.00
P_U-FUSION$_{\text{LF2}}$	9281131.45	9278699.79	9278699.79	9278699.79	18.91 sec	109.00
P_U-FUSION$_{\text{QPBO}}$	11217379.70	9008429.54	8437145.94	8437145.94	156.95 sec	1000.00

6 Conclusions

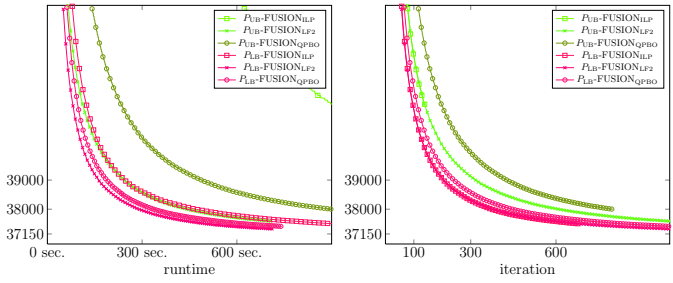
Fusion algorithms are very powerful and their performance on discrete graphical models has been apparently underestimated in the past. We showed that the performance of any inference method can be improved by embedding it as a proposal generator into a fusion algorithm. This leads to better solutions as well as to better any-time performance by compensating rounding artefacts, c.f. Fig. 1. The additional computational costs are usually negligible.

Concerning proposal generators, inference based generators are overall superior, since the proposals are of high quality. However, for large scale or higher-order models they are sometimes no longer applicable, e.g. for field of experts, or much slower, e.g. for protein folding, than random or deterministic ones. Here randomized generators work often reasonable. Application specific or more advanced generators might be able to further close this gap with small additional computational costs.

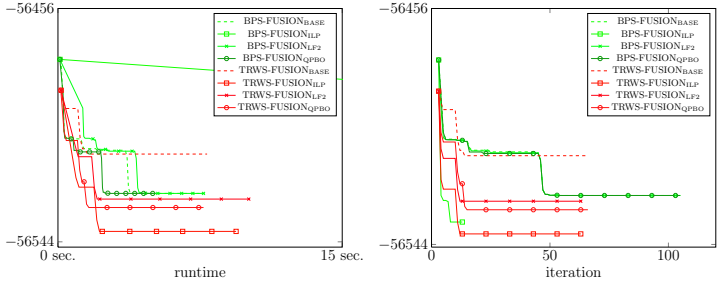
The quality of fusion algorithms can be also improved by fusion operators different from QPBO-Fusion. We presented two powerful alternatives: Integer linear programming solvers can be used to calculate the optimal moves in each step.

This can lead to much better results when the persistency of QPBO is small, e.g. DTF or protein prediction. Lazy Flipping based fusion does also not suffer from small persistency but requires that the global move can be obtained by a sequence of local moves. When this is the case, as for the field of expert instances, Lazy Flipping fusion gives the best trade-off between runtime and energy improvement. Another interesting observation in this context is that optimal moves are not always desirable. Contrary to non-optimal moves optimal moves, can tend to run into "dead ends" for which only a small number of proposals generate moves which allow to escape. Such a proposal might not be generated within $m_{\max}$ iterations and the algorithm stops too early. Furthermore, fusion is a greedy procedure and an optimal fusion move might not be optimal in the long run. For example for some protein folding instances QPBO fusion is sometimes marginal better than ILP fusion for the same number of iterations. However, except for these outliers and on average optimal moves performs better than QPBO-based moves – at least in the long run.

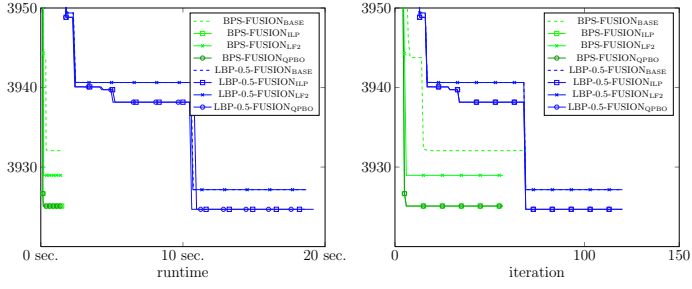
Finally we would like to remark that contrary to the standard QPBO-based fusion-operator the presented alternatives can deal with more than one proposal. Consequently the subproblems would be multi-label problems and X' larger, which allows more powerful moves.



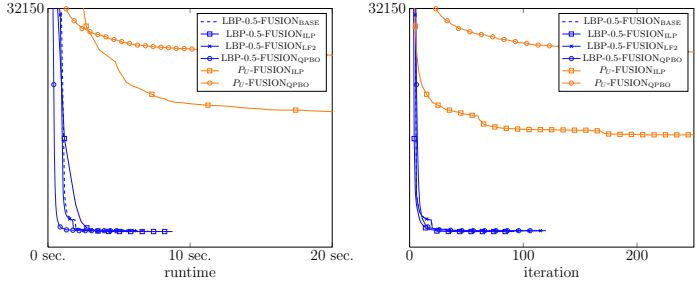
(a) FoE - instance: 101085



(b) DTF Chinese Characters - instance: 0001



(c) Protein Folding - instance: pdb1b25



(d) Protein Prediction - instance: 1

Fig. 1: Energy improvement for selected instances and methods over time **(left)** and over iterations **(right)**.

References

1. 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.: A comparative study of modern inference techniques for discrete energy minimization problems. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2013)
2. Elidan, G., Globerson, A.: The probabilistic inference challenge (PIC2011). http://www.cs.huji.ac.il/project/PASCAL/
3. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. IEEE PAMI **30**(6) (2008) 1068–1080
4. Goldluecke, B., Strekalovskiy, E., Cremers, D.: Tight convex relaxations for vector-valued labeling. SIAM Journal on Imaging Sciences **6**(3) (2013) 1626–1664
5. Kohli, P., Ladicky, L., Torr, P.H.: Robust higher order potentials for enforcing label consistency. International Journal of Computer Vision **82**(3) (2009) 302–324
6. Kim, S., Nowozin, S., Kohli, P., Yoo, C.D.: Higher-order correlation clustering for image segmentation. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS). (2011)
7. Delong, A., Osokin, A., Isack, H., Boykov, Y.: Fast approximate energy minimization with label costs. International Journal of Computer Vision **96** (Jan. 2012) 1–27
8. Lempitsky, V., Rother, C., Roth, S., Blake, A.: Fusion moves for markov random field optimization. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(8) (2010) 1392–1405
9. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? In: ECCV. (2002)
10. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Appl. Math. **123**(1-3) (November 2002) 155–225
11. Rother, C., Kolmogorov, V., Lempitsky, V.S., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: CVPR. (2007)
12. Ishikawa, H.: Transformation of general binary mrf minimization to the first-order case. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(6) (2011) 1234–1249
13. Fix, A., Gruber, A., Boros, E., Zabih, R.: A graph cut algorithm for higher-order Markov random fields. In: ICCV. (2011)
14. Kahl, F., Strandmark, P.: Generalized roof duality. Discrete Applied Mathematics **160**(16-17) (2012) 2419–2434
15. Kappes, J.H., Andres, B., Hamprecht, F.A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B.X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., Rother, C.: A comparative study of modern inference techniques for structured discrete energy minimization problems. CoRR **abs/1404.0533** (2014)
16. Andres, B., Beier, T., Kappes, J.H.: OpenGM2 (2012) http://hci.iwr.uni-heidelberg.de/opengm2/.
17. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
18. Nowozin, S., Lampert, C.H.: Structured learning and prediction in computer vision. Foundations and Trends in Computer Graphics and Vision **6**(3–4) (2011) 185–365

19. Besag, J.: On the Statistical Analysis of Dirty Pictures. Journal of the Royal Statistical Society. Series B (Methodological) **48**(3) (1986) 259–302
20. Andres, B., Kappes, J.H., Beier, T., Köthe, U., Hamprecht, F.A.: The lazy flipper: Efficient depth-limited exhaustive search in discrete graphical models. In: Proceedings of the European Conference on Computer Vision (ECCV). (2012)
21. Kappes, J.H., Speth, M., Reinelt, G., Schnörr, C.: Towards efficient and exact MAP-inference for large scale discrete computer vision problems via combinatorial optimization. In: CVPR. (2013)
22. IBM: ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (2013)
23. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(10) (2006) 1568–1583
24. Globerson, A., Jaakkola, T.: Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In: NIPS. (2007)
25. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. Int. J. Comput. Vision **70**(1) (October 2006) 41–54
26. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(11) (2001) 1222–1239
27. Roth, S., Black, M.J.: Fields of experts. International Journal of Computer Vision **82**(2) (2009) 205–229
28. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV. (2001)
29. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. International Journal of Computer Vision **70**(1) (2006) 41–54
30. Yanover, C., Schueler-Furman, O., Weiss, Y.: Minimizing and learning energy functions for side-chain prediction. Journal of Computational Biology **15**(7) (2008) 899–911
31. Savchynskyy, B., Kappes, J.H., Swoboda, P., Schnörr, C.: Global MAP-optimality by shrinking the combinatorial search area with convex relaxation. In: NIPS. (2013)
32. Jaimovich, A., Elidan, G., Margalit, H., Friedman, N.: Towards an integrated protein-protein interaction network: A relational markov network approach. Journal of Computational Biology **13**(2) (2006) 145–164
33. Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B., Kohli, P.: Decision tree fields. In: ICCV, IEEE (2011) 1668–1675
34. Gorelick, L., Boykov, Y., Veksler, O., Ayed, I.B., Delong, A.: Submodularization for binary pairwise energies. In: CVPR, IEEE (2014) in press.