A Bundle Approach To Efficient MAP-Inference by Lagrangian Relaxation

Jörg Hendrik Kappes[1]
[1]IPA and [2]HCI,
Heidelberg University, Germany
kappes@math.uni-heidelberg.de

Bogdan Savchynskyy[2]
bogdan.savchynskyy@iwr.uni-heidelberg.de
Christoph Schnörr[1,2]
schnoerr@math.uni-heidelberg.de

Abstract

Approximate inference by decomposition of discrete graphical models and Lagrangian relaxation has become a key technique in computer vision. The resulting dual objective function is convenient from the optimization point-of-view, in principle. Due to its inherent non-smoothness, however, it is not directly amenable to efficient convex optimization. Related work either weakens the relaxation by smoothing or applies variations of the inefficient projected subgradient methods. In either case, heuristic choices of tuning parameters influence the performance and significantly depend on the specific problem at hand.

In this paper, we introduce a novel approach based on bundle methods from the field of combinatorial optimization. It is directly based on the non-smooth dual objective function, requires no tuning parameters and showed a markedly improved efficiency uniformly over a large variety of problem instances including benchmark experiments.

Our code will be publicly available after publication of this paper.

1. Introduction

We consider the problem of finding the most likely configuration of a discrete graphical model.[1] Adopting the common assumption that the objective function factorizes according to a factor graph $G = (V, F, E)$, this amounts to the energy minimization problem

$$\min_{x\in X} J_G(x), \quad J_G(x) = \sum_{f\in F} \varphi_f(x_{\mathrm{ne}(f)}). \tag{1}$$

Since problem (1) is NP-hard, research has focused on approximate inference for larger problem sizes. Schlesinger [18] and Wainwright [24] proposed a linear programming (LP) relaxation based on the so-called local polytope. Although the corresponding LP can be solved in polynomial time, the number of variables and constraints can be quite large for a range of typical computer vision problems, such that standard solvers can not be applied.

[1]Precise definitions follow in subsequent sections.

Recently, Ravikumar *et al.* [15] and Schmidt *et al.* [20] presented *primal/primal-dual methods*, which partially overcomes this problem by exploiting the underlying graphical structure. However, since all primal variables have to be stored, corresponding to a point of the local polytope, the memory requirements are huge.

Schlesinger [18] analyzed the dual LP as a lower bound for (1) and proposed algorithms that can be interpreted as *block-coordinate descent*, *c.f.* [25] for a recent review. The TRW-S algorithm [10] generalizes this by considering arbitrary subtrees as subproblems. However, because the dual problem is intrinsically non-smooth, such methods can – and also often do – converge to non-optimal fixed points.

As an alternative a *projected subgradient* scheme was independently proposed by Komodakis [11] and Schlesinger [19]. Although this method guarantees convergence to the optimal dual solution, single subgradient directions defining the iterates may not improve the objective value, and choosing the step-size sequence is crucial for the overall performance. In order to find better update directions, Kappes *et al.* [7] used an 'averaged' update direction, motivated by a paper of Ruszczynski [16], and reported better convergence. Yet, subgradient-based methods can deal with a larger class of decompositions [7, 1, 26, 22] and can also provide tighter relaxations [12].

Another line of research [6, 4, 17] considered *smoothing of the objective function* in order to apply established gradient-based methods of smooth convex optimization. On the other hand, the evaluation of the dual becomes more complicated, which limits the set of feasible decompositions and generates additional computational costs. A more detailed consideration of this alternative research direction is beyond the scope of the present paper.

Contribution. As a case study, we consider the same primal and dual formulations related to problem (1), as e.g. in Komodakis [12], and introduce – to our knowledge for the first time in computer vision – bundle methods [9, 13] to replace simple subgradient-based iterations and heuristic step-size rules by more efficient, mathematically sound schemes.

The key idea is to construct and maintain a local upper approximation of the dual objective function, which then can be optimized by proximal-point methods. Thus, subgradients are no longer directly used to update the dual variables. Rather, they serve to improve a polyhedral approximation of the dual objective and thus to cope with the inherent non-smoothness in a less myopic way. Specifically, we use an aggregate bundle approach [8] and an adaptive method to adjust the trust region [9] with guaranteed convergence property. The resulting optimization scheme does not require parameter tuning. The evaluation of our approach in comparison to state-of-the-art methods based on subgradient schemes [12] showed significantly improved convergence rates using a single fixed parameter set without problem-specific tuning, uniformly over a broad range of problem instances including benchmark data. Our method is general enough to cope with problems beyond the case study underlying the present paper.

Organization. The problem and problem decompositions are defined and detailed in Section 2. In Section 3, we describe state-of-the-art solvers and our novel approach utilizing a bundle method. A comprehensive, competitive evaluation taking into account benchmark data is reported in Section 4.

Notation. Single primal variables are denoted by $x_a \in X_a$ and indexed by nodes $a \in V$. We use the shorthands $x_A = (x_a)_{a\in A}$, $X_A = \prod_{a\in A} X_a$ for any $A \subseteq V$, and $x = x_V \in X$. Besides the vertices V, factor graphs $G = (V, F, E)$ comprise factor nodes F and an edge set $E \subseteq V \times F$. With each factor $f \in F$, we associate a function $\varphi_f : X_{ne(f)} \to \mathbb{R}$ that has as argument the variables indexed by the neighbors $ne(f) = \{a \in V \mid (a, f) \in E\}$ of f. For simplicity, we assume that for each $a \in V$ there exists a $f_a \in F$ with $\mathrm{ne}(f_a) = \{a\}$. Notation $P_C(\cdot)$ is used for the Euclidean projection onto the set C.

2. Problem Formulation

Problem (1) is defined over a discrete domain, $X = \prod_{a\in V} X_a$, $X_a = \{1, \dots, L_a\}$, and it is convenient to reformulate it as a linear (integer) program. Let $N = \sum_{f\in F} \prod_{a\in \mathrm{ne}(f)} |X_a|$, and let $\theta \in \mathbb{R}^N$ collect all possible function values as components $\theta_{f,i} = \varphi_f(i)$. Then problem (1) reads

$$\min_{\mu\in\mathcal{M}(G)} \langle \theta, \mu\rangle, \tag{2}$$

where the marginal polytope $\mathcal{M}(G)$ is given as the convex hull of all indicator vectors

$$\phi(x)_{f,i} = \begin{cases} 1 & \text{if } x_{\mathrm{ne}(f)} = i \\ 0 & \text{else,} \end{cases} \tag{3}$$

i.e. $\mathcal{M}(G) = \mathrm{conv}\{\phi(x) \mid x \in X\}$. A common approach to overcome the combinatorial complexity of $\mathcal{M}(G)$ is to consider the outer relaxation in terms of the first order local polytope [24]

$$\mathcal{L}(G) = \left\{ \mu \in [0,1]^N \;\middle|\; \begin{array}{l} 1 = \sum_{x_a\in X_a} \mu_{f_a,x_a} \;\forall a \in V, \\ \mu_{f_a} = [\mu_f]_a \;\forall f \in F, a \in \mathrm{ne}(f) \end{array} \right\} \tag{4}$$

where $[\mu_f]_a(x_a) = \sum_{\bar{x}\in X_{ne(f)}, \bar{x}_a = x_a} \mu_f(\bar{x}_a)$. The corresponding problem relaxation

$$\min_{\mu\in\mathcal{L}(G)} \langle \theta, \mu\rangle \tag{5}$$

can be solved in polynomial time. However, since even in this case the number of variables and constraints of the problem might be quite large for many real problems, solving (5) with standard solvers is not a feasible option due to memory restrictions. This motivates the decomposition of the problem into tractable subproblems and to solve (5) iteratively by fusing the partial solutions through convex programming.

2.1. Dual Decomposition

Dual decomposition is an abbreviation for the application of Lagrangian relaxation to the decomposition of (5) into subproblems. Dualizing the constraints that ensure consistency of the decomposed problem with (5) leads to an (un-)constrained non-smooth concave maximization problem. We describe these two steps next.

Model Decomposition. Let us define a second factor graph $\tilde{G} = (\tilde{V}, \tilde{F}, \tilde{E})$, together with a mapping $\rho : \tilde{V} \to V$. We say, that $\tilde{a} \in \tilde{V}$ is a duplicate of $a \in V$ if $\rho(\tilde{a}) = a$. We call $\tilde{G}$ a decomposition of G if the following holds:

$$\forall a \in V,\ \exists \tilde{a} \in \tilde{V} : \rho(\tilde{a}) = a, \tag{6}$$

$$\forall f \in F,\ \exists \tilde{f} \in \tilde{F} : \rho(\mathrm{ne}(\tilde{f})) = \mathrm{ne}(f), \tag{7}$$

$$\forall \tilde{f} \in \tilde{F},\ \exists f \in F : \rho(\mathrm{ne}(\tilde{f})) = \mathrm{ne}(f), \tag{8}$$

$$\forall f \in F,\ i \in X_{\mathrm{ne}(f)} : \varphi_f(i) = \sum_{\substack{\tilde{f} \in \tilde{F} \\ \mathrm{ne}(f) = \rho(\mathrm{ne}(\tilde{f}))}} \tilde{\varphi}_{\tilde{f}}(i). \tag{9}$$

Note that according to this definition various problem decompositions are feasible.

Unconstrained Dual Problem. While (6)–(8) require that each variable, factor and edge is represented in $\tilde{G}$ and vice versa, (9) ensures equality of the objective functions. As a result, problem (1) is equivalent to

$$\min_{\tilde{x}\in\tilde{X}} J_{\tilde{G}}(\tilde{x}) \quad \text{s.t. } \forall a, b \in \tilde{V}, \rho(a) = \rho(b) : \tilde{x}_a = \tilde{x}_b. \tag{10}$$

As before at the beginning of Section 2, we replace the combinatorial problem (10) by a LP,

$$\min_{\tilde{\mu}\in\mathcal{M}(\tilde{G})} \langle \tilde{\mu}, \tilde{\theta}\rangle \qquad \text{s.t.} \quad A\tilde{\mu} = 0. \tag{11}$$

Unlike with (2), however, here the feasible set $\mathcal{M}(\tilde{G})$ is tractable due to the decomposition in tractable subproblems, and A is chosen such that $A\,\tilde{\mu} = 0$ enforces the constraints of (10), $\forall a, b \in \tilde{V}, \rho(a) = \rho(b) : \tilde{\mu}_{f_a} = \tilde{\mu}_{f_b}$.

Problem (11) constitutes a relaxation of (10) since the polytope $\{\tilde{\mu} \in \mathcal{M}(\tilde{G}) \mid A\,\tilde{\mu} = 0\}$ contains also fractional vertices. Due to the construction of $\tilde{G}$, however, this relaxation is at least as tight as the standard relaxation (5) over the local polytope (4), *c.f.* [6, 24]. Finally, to get rid of the additional constraint $A\,\tilde{\mu} = 0$, we add them to the objective via Lagrangian multipliers [13],

$$\min_{\tilde{\mu}\in\mathcal{M}(\tilde{G})} \max_{\lambda} \langle\tilde{\theta}, \tilde{\mu}\rangle + \langle\lambda, A\,\tilde{\mu}\rangle, \tag{12}$$

and obtain the unconstrained dual problem

$$\max_{\lambda} f(\lambda), \quad f(\lambda) = \min_{\tilde{\mu}\in\mathcal{M}(\tilde{G})} \langle A^\top\lambda + \tilde{\theta}, \tilde{\mu}\rangle. \tag{13}$$

Note that strong duality holds and (11) and (13) have the same optimal value, *c.f.* [13].

2.2. Evaluating the Dual Function

The dual function $f(\lambda)$ in (13) is concave, piecewise-linear and non-differentiable. However, for suitable choices of $\tilde{G}$ we can calculate a subgradient $g \in \partial f(\lambda)$ and the objective value $f(\lambda)$. The subdifferential of $f(\lambda)$ is given by

$$\partial f(\lambda) = \operatorname{conv}\left\{A\tilde{\mu}^* \mid \tilde{\mu}^* \in \arg\min_{\tilde{\mu}\in\mathcal{M}(\tilde{G})} \langle A^\top\lambda + \tilde{\theta}, \tilde{\mu}\rangle\right\}$$

The common strategy to select $\tilde{G}$ is to decompose G into a set of disconnected subgraphs by duplication of nodes, such that $\tilde{G} = \tilde{G}_1 \cup \ldots \cup \tilde{G}_N$, $\mathcal{M}(\tilde{G}) = \{\tilde{\mu} \mid \tilde{\mu}_{\tilde{G}_i} \in \mathcal{M}(\tilde{G}_i)\}$, and inference is tractable for each single $\tilde{G}_i$, *c.f.* [7, 1, 22, 12, 6, 26]. In order to evaluate $f(\lambda)$, we have to solve the independent subproblems for $\tilde{\mu}_{\tilde{G}_i} = \phi(\tilde{x}_{\tilde{G}_i})$. This gives us a subgradient $g \in \partial f(\lambda)$ as well as a lower bound $b = f(\lambda)$ of $J_G(\cdot)$.

To obtain an upper bound we generate integer solutions for (1). If the subgraphs $\tilde{G}_i$ are spanning, *i.e.* they contain all original variables once, we can use the solution to subproblem $\tilde{x}_{\tilde{G}_i}$ directly to obtain an upper bound $u = J_G(\tilde{x}_{\tilde{G}_i})$. Otherwise we have to construct a full solution from the solutions to the subproblems, by filling the missing parts by solutions of other subproblems

The upper bound may not be tight (integrality gap > 0), *i.e.* the optimal primal solution is fractional. For special cases of decomposition a method to obtain a tighter upper bound exist [17], but its discussion is beyond the scope of this paper.

3. Dual Solvers

Although the dual problem (13) is a concave maximization problem, the non-differentiability of the objective renders the problem challenging. All corresponding optimization methods discussed below rely on calling a *primal oracle* that returns at each iteration step k:

- g^k, a subgradient of $f(\lambda^k)$;
- b^k, the value of $f(\lambda^k)$ as a lower bound for (1);
- x^k, a (possibly non-optimal) solution of (1);
- u^k, an upper bound of (1) obtained by $J(x^k)$.

We detail next several established methods (Section 3.1) and our own novel approach (Section 3.2) that can optimize (13) using only such oracle calls.

3.1. Projected Subgradient Method

Projected subgradient methods [21, 14, 2] are very popular because they are easy to implement. Commonly a constraint dual problem is considered, which makes a additional projection onto a simple set Λ necessary, *c.f.* [12] and appendix A.2. Dual variables are iteratively updated by a scaled subgradient and projected onto the feasible set,

$$\lambda^{k+1} = P_\Lambda(\lambda^k + \tau^k \cdot g^k). \tag{14}$$

The scaling sequence τ^k has to be specified by the user. We consider the following choices suggested in [12][2]: (i) *nonsummable diminishing step size* (15), (ii) *nonsummable diminishing step lengths* (16), and (iii) *adaptive step length* (17).

$$\tau^k = \frac{\gamma}{1+\alpha k} \tag{15}$$

$$\tau^k = \frac{\gamma}{1+\alpha k}\frac{1}{\|P_\Lambda(g^k)\|_2} \tag{16}$$

$$\tau^k = \frac{\gamma\cdot(-b^k + \min_k u^k)}{\|P_\Lambda(g^k)\|_2^2} \tag{17}$$

While (15),(16) guarantee convergence to the optima, they vary with the scaling of $J_G(\cdot)$ and often converge slowly. The adaptive choice (17), on the other hand, is invariant to scaling but is not guaranteed to converge. Moreover, as the gap might be overestimated, selecting values of γ that work well in the beginning and towards the end of the iteration is difficult in general. Finally, all three methods usually oscillate close to non-differentiable points of the objective function.

3.2. Bundle Methods

We suggest an alternative ansatz based on bundle methods [8, 9, 13]. The basic idea is to bound the dual function $f(\lambda)$ from above by a polyhedral approximation $\hat{f}(\lambda)$ constructed from a set of points λ^i, objective values $f(\lambda^i)$ and

[2]Contrary to [12] we use $\|P_\Lambda(g^k)\|_2^2$ instead of $\|g^k\|_2^2$ because the latter choice performs worse in our experiments.

subgradients $g^i \in \partial f(\lambda^i)$ constituting a bundle $\mathcal{B}$, such that $f(\lambda) \leq \hat{f}(\lambda)$ and $f(\lambda^i) = \hat{f}(\lambda^i)$,

$$\hat{f}(\lambda) = \min_{(f(\lambda'),\lambda',g')\in\mathcal{B}} \{f(\lambda') + \langle g', \lambda - \lambda'\rangle\}. \tag{18}$$

We use the proximal point algorithm to generate a sequence

$$\lambda^{k+1} = \arg\max_\lambda \hat{f}(\lambda) - \frac{w^k}{2}\|\lambda - \bar{\lambda}\|_2^2 \tag{19}$$

where $w > 0$ is intended to keep λ^{k+1} in the region around our current solution estimate $\bar{\lambda}$ in which $\hat{f}(\lambda)$ should be close to $f(\lambda)$.

As long as λ^{k+1} does not lead to significant progress we do not change $\bar{\lambda}$ and tighten the approximation by adding $(f(\lambda^{k+1}), g^{k+1}, \lambda^{k+1})$ to the bundle $\mathcal{B}$. This is commonly called a *null step*. Otherwise we take a so-called *serious step* and replace $\bar{\lambda}$ by λ^{k+1}. To decide if a serious or null step should be performed, we check the relative improvement of $f(\lambda)$ and $\hat{f}(\lambda)$ over $f(\bar{\lambda})$. If this ratio is larger than a predefined constant m_L, then the polyhedral approximation is tight enough to take a serious step.

The belief that problem (19) cannot be solved efficiently is indeed misleading. Due to the construction of $\hat{f}(\cdot)$ we can reformulate the convex problem (19) to take the form

$$\arg\max_{\lambda,v} v - \frac{w}{2}\|\lambda - \bar{\lambda}\|_2^2 \tag{20}$$
$$\text{s.t. } f(\lambda') + \langle g', \lambda - \lambda'\rangle \geq v \quad \forall (f(\lambda'), \lambda', g') \in \mathcal{B},$$

which in turn can be solved efficiently for small $|\mathcal{B}|$ by taking the corresponding dual problem into account, *c.f.* A.1.

Furthermore, bundle methods provide finite convergence for piecewise-linear functions [8, Sec. 5], *i.e.* the problem can be solved to precision $\epsilon = 0$ in a finite number of oracle calls.

Accordingly, two aspects of the **Bundle Algorithm 1** detailed above are important: dynamical management of the bundle in order to keep $\mathcal{B}$ small, and selection of the weights w^k. We discuss these two aspects next.

Bundle Update. An unbounded bundle size quickly makes (19) intractable. To manage the bundle we compare two different strategies, (i) a bounded bundle size and (ii) an aggregated bundle.

So the first strategy limits the size of the bundle and keep only the best one if the maximal bundle size is reached. In theory this simple strategy does not affect convergence [13] if

$$|\mathcal{B}| \leq \max\{\dim(\partial f(\lambda)) | \lambda \in \mathbb{R}^m\} + 1 \leq m + 1.$$

Unfortunately, the size of the dual space m is typically large. However, even with small bundle sizes we did not

[3]technical conditions for convergence, *c.f.* [9]

Algorithm 1 Bundle Method

```
$\lambda^0 = 0, u^0 > 0, M > 0, \epsilon \geq 0, m_L \in (0, 0.5)$ [3]
$\mathcal{B} = \{(f(\lambda^0), g \in \partial f(\lambda^0), \lambda^0)\}$
for $k = 0 \to k_{\max}$ do
    Find the solution $\lambda^{k+1}$ of (19)
    if $f(\lambda^{k+1}) - f(\bar{\lambda}) \geq m_L \left(\hat{f}(\lambda^{k+1}) - f(\bar{\lambda})\right)$ then
        $\bar{\lambda} = \lambda^{k+1}$
    end if
    Calculate $(g^{k+1}, b^{k+1}, x^{k+1}, u^{k+1})$ for $\lambda^{k+1}$
    Update Bundle $\mathcal{B} \leftarrow \mathcal{B} \cup \{(f(\lambda^{k+1}), g^{k+1}, \lambda^{k+1})\}$
    Update Weight $w^{k+1}$
    if $f(\bar{\lambda}) = \hat{f}(\lambda^{k+1})$ then converged to optimum
end for
```

get any problems in our experiments, but in general one can not guarantee convergence any more.

The second strategy was proposed by Kiwiel [8]. He suggests to replace the bundle by a single *aggregate subgradient* without losing convergence properties.

As stated in [13] there are no theoretical results that establish differences between different bundle sizes. In other words, a larger bundle can speed up or slow down convergence.

Weight Update. The most naive choice for w^k is a constant sequence. Although this often works surprisingly good in applications, it is inefficient for two reasons. If we choose the constant too small than the bundle method mimicks the cutting-plane approach and builds a non-local model around the current working point. As a consequence each serious step is followed by many null steps. On the other hand, if we choose the constant too large than almost all steps are small serious steps slowing down convergence as well. Therefore we use an update method suggested by Kiwiel that guarantees overall convergence, *c.f.* [9, Proc. 2.2] . While Kiwiel's update method does not take the estimated primal-dual gap into account, we suggest an alternative adaptive sequence of the form:

$$w^k = P_{[w_{\min}, w_{\max}]}\left(\gamma \cdot \frac{\min_k u^k - \max_k b^k}{\|g^k\|_2}\right) \tag{21}$$

To ensure convergence we set $w^{k+1} = w^k$ at null steps. Our update method (21) is motivated by incorporating the adaptive subgradient scheme (17) into the bundle framework. Additional bounds $w_{\min}, w_{\max}$ are required for stabilization and guaranteed convergence [13], respectively.

3.3. Implementation Details

For the experiments we use our C++ implementation, which makes use of the concept of views for a memory efficient implementation[4]. For acyclic subproblems we use dy-

[4]For none of the experiments we require more than 1 GB of memory.

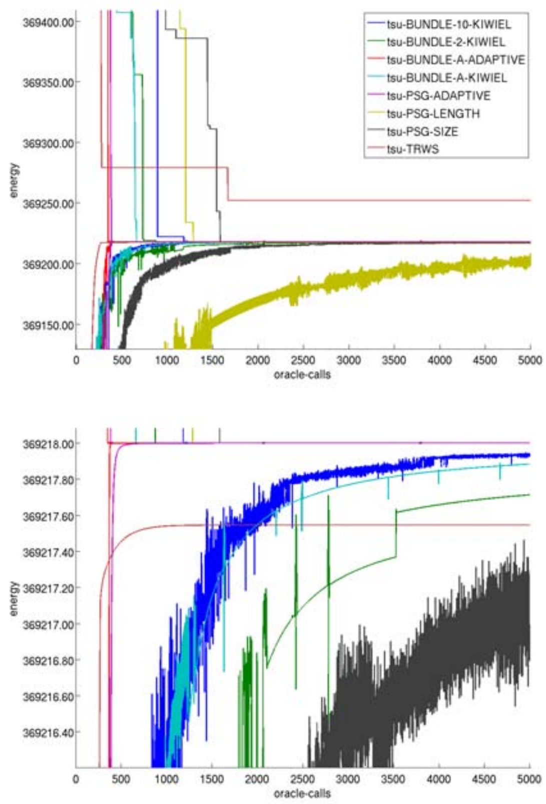

Figure 1. Energy values of the best estimated integer solutions u^k and lower bounds b^k. For the *Tsukuba dataset* all bundle methods outperform the competing subgradient methods, while the adaptive bundle and subgradient methods based on the primal-dual-gap are superior to all others. *The lower bounds and integer solutions for all version of our bundle aproach are better than any result reported in [23] including TRW-S.* Since the energy of Tsukuba is integer we obtain guaranteed global optimality after 370 oracle calls of BUNDLE-A-ADAPTIVE, 397 oracle calls of PSG-ADAPTIVE, and 1206 oracle calls of BUNDLE-A-KIWIEL. Also the convergence behavior of our bundle is superior to subgradient methods, as the close-up plot clearly shows. Furthermore, BUNDLE-A-ADAPTIVE converge fast and found as only method the *optimal* lower bound, up to double precision.

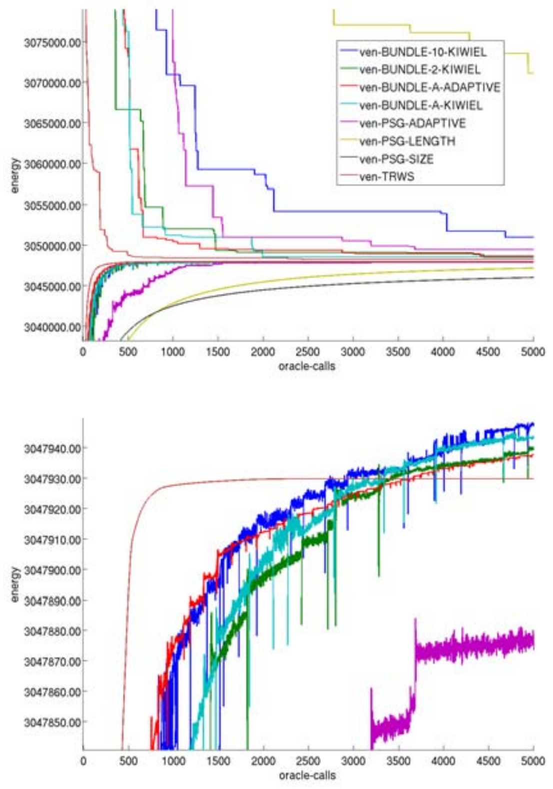

Figure 2. Energy values of the best estimated integer solutions u^k and lower bounds b^k. For the *Venus dataset* all bundle approaches outperform the projected subgradient methods. The lower bounds are even better than the one reported for TRW-S, as the close-up plot clearly shows. Contrary to Tsukuba a gap remains which indicates that the LP-relaxation is not tight for this problem. As a consequence, the adaptive rules based on gap estimates performed worse, since these estimates are necessarily worse.

namic programming, and for submodular binary subproblems the max-flow code of Boykov and Kolmogorov [3]. For the projected subgradient method we use our own code. For the bundle method we integrate the ConicBundle-Library [5] into our framework. Neither our library nor the ConicBundle-Library are yet optimized for specific problem instances, as done in [23], and we make no use of warm-start mechanisms so far. Furthermore, we solved the subproblems sequentially, but our code to be released[5] will support parallel optimization.

[5]`http://hci.iwr.uni-heidelberg.de/opengm2/`

4. Experimental Evaluation

Compared Algorithms. For the projected subgradient methods with scaling sequences (PSG-SIZE), (PSG-LENGTH) and (PSG-ADAPTIVE) defined by (15), (16) and (17) respectively, we used grid search to find good parameters. We also tested the averaged subgradient method of Ruszczynski [16] used by Kappes *et al*. [7], and found parameter values leading to equal but not substantially better performance compared to pure subgradient methods. Possibly our limit of 100 oracle calls for the grid search missed better parameter values. This indicates the importance of *robust parameter choices* which seems difficult to achieve for this class and methods and motivates our present work.

For the bundle methods we use aggregate bundle (BUNDLE-A) and bundles of maximal size of 2 (BUNDLE-2) and 10 (BUNDLE-10). When the maximal bundle sized was reached the bundle was reduced by the least violated constraints, *c.f*. [5]. The adaptive weight rule of Kiwiel (BUNDLE-*-KIWIEL) was used for all three bundle methods. Since we did not observe a significant difference between these methods, we applied our adaptive weight rule (21) only to aggregated bundle (BUNDLE-A-ADAPTIVE). The exact parameter values used are listed in the appendix A.3.

As a representative of block coordinate ascent methods we chose TRW-S [10]. A single iteration of TRW-S has the complexity of two oracle calls. While TRW-S is one of the fastest available methods it can – and in fact did – get stuck in non-optimal fixed points.

For the stereo problems Tsukuba and Venus from the MRF-benchmark [23], we applied a decomposition into two spanning trees in each case.

Figures 1 and 2 show the energy of the best estimated integer solutions u^k and lower bounds b^k. In order to visualize oscillations we show the lower bound estimated after the k-th oracle call rather than the best bound found so far. The apparent oscillation of bundle methods is due to null-steps – the subsequence of serious steps only is monotone.

Performance. For the Tzukuba dataset adaptive methods dominate all others since the integrality gap is nearly zero. The adaptive bundle method clearly outperforms the projected subgradient method and solves the dual problem to optimality after 2260 oracle calls, up to double precision[6].

Since for the Venus dataset an integrality gap remains, adaptive methods no longer dominate. They showed slow progress at later iterations when the ratio $(\min_k u^k - \max_k b^k)/(b^* - \max_k b^k)$ becomes large. However, bundle methods performed better than subgradient methods and returned contrary to subgradient methods better bounds than TRW-S after 5000 oracle calls[7].

The average run-time for a single oracle call was 2.40 seconds for Tsukuba and 4.72 seconds for Venus. A dual update step via aggregated bundle (0.71 and 1.20 sec) takes less than two times the costs of a projected subgradient step (0.46 and 0.75 sec), hence is quite effective in terms of the resulting better overall performance. When we consider fixed bundle size, a single dual step becomes slightly more time consuming, 1.44 and 2.59 sec for BUNDLE-2 and 2.80 and 4.70 sec for BUNDLE-10. We point out that our code is very general and not optimized for any special instance.

As a second scenario we followed Strandmark [22] and optimized binary sub-modular problems by domain-decomposition. We confined ourself to the binary problems from the MRF-benchmark [23] which can be solved to global optimality, to demonstrate clearly the advantages of our approach over projected subgradient methods. We decomposed each image in a left and right image part and coupled them by an overlapping stripe of 2 pixels width. Subproblems were solved by a standard max-flow solver without warm-start techniques. Again we tuned the parameters for projected subgradient methods by grid search as well as the parameters for our adaptive weighting.

Table 1. Number of iterations k until $u^k - b^k < 1 \,/ < 10^{-7}$. The symbol '-' indicates that this condition was not met after 100 iterations. Overall, our bundle method with adaptive weights performs best. Notably even the bundle method with Kiwiel's heuristic performs competitive and robust without any parameter tuning.

	person	flower	sponge
PSG-SIZE	16/16	13/-	5/5
PSG-LENGTH	24/26	-/-	3/3
PSG-ADAPTIVE	-/-	6/6	2/2
BUNDLE-A-KIWIEL	26/26	15/15	4/4
BUNDLE-2-KIWIEL	24/28	16/16	4/4
BUNDLE-10-KIWIEL	10/10	14/14	4/4
BUNDLE-A-ADAPTIVE	6/7	7/7	2/2

Table 1 displays the number of oracle calls until optimality was achieved and the dual solver had converged. Again the bundle method with adaptive weight gives the best overall results, but the bundle method with Kiwiel's rule also showed a robust behavior without any parameter tuning.

Summary. *Overall, and largely independent of the particular choice of the bundle management, bundle methods outperformed established subgradient methods under comparable conditions. They also avoided non-optimal fixed points that sometimes attract established block coordinate methods.*
Using a fixed bundle size of 10 performed slightly better in terms of oracle calls but caused longer runtime time. However, since the differences were small and there is room for speeding up the bundle-solver, this finding is preliminary.

5. Conclusion and Further work

We introduced and presented a novel approach to the global optimization of non-smooth dual objective functions in connection with decompositions of discrete graphical inference problems in computer vision. Our method significantly outperforms state-of-the-art methods established in computer vision under competitive conditions. Furthermore, we presented a parameter-free rule for selecting the trust region in terms of bundle updates, that led to improved convergence rates in our benchmark experiments.

There are clear indications that better estimates of the primal bound may further improve the convergence of the adaptive version of our approach.

[6]The found optimal value was 369217.99999905284, the best value after 5000 oracle calls was 369217.9999905776 for PSG-ADAPTIVE, 369217.88337642595 for BUNDLE-A-KIWIEL, and 369217.54575490614 for TRW-S.

[7]The best lower bound on the optimal primal solution was 3047948.37739271 found by BUNDLE-10-KIWIEL.

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

References

[1] D. Batra, A. Gallagher, D. Parikh, and T. Chen. Beyond trees: MRF inference via outer-planar decomposition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2496 – 2503, june 2010. 1, 3

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 3

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:1124–1137, September 2004. 5

[4] T. Hazan and A. Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energy. In *In The 24th Conference on Uncertainty in Artificial Intelligence (UAI*, 2008. 1

[5] C. Helmberg. The ConicBundle Library for Convex Optimization v0.3.10. `http://www-user.tu-chemnitz.de/~helmberg/ConicBundle/`, 2011. 5, 8

[6] J. K. Johnson, D. Malioutov, and A. S. Willsky. Lagrangian relaxation for MAP estimation in graphical models. In *45th Annual Allerton Conference on Communication, Control and Computing*, September 2007. 1, 3

[7] J. H. Kappes, S. Schmidt, and C. Schnörr. MRF inference by k-fan decomposition and tight Lagrangian relaxation. In *European Conference on Computer Vision (ECCV)*, volume 6313, pages 735–747. Springer Berlin / Heidelberg, 2010. 1, 3, 5

[8] K. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983. 2, 3, 4

[9] K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Program.*, pages 105–122, 1990. 1, 2, 3, 4

[10] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Patt. Anal. Mach. Intell.*, 28(10):1568–1583, 2006. 1, 6

[11] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *IEEE 11th International Conference on Computer Vision, ICCV 2007*, pages 1–8. IEEE, 2007. 1

[12] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:531–552, 2011. 1, 2, 3

[13] C. Lemaréchal. Lagrangian relaxation. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*, pages 112–156. Springer, 2001. 1, 3, 4

[14] B. Polyak. *Introduction to Optimization*. Optimization Software Inc., New York, 1987. 3

[15] P. D. Ravikumar, A. Agarwal, and M. J. Wainwright. Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *Journal of Machine Learning Research*, 11:1043–1080, 2010. 1

[16] A. Ruszczynski. A merit function approach to the subgradient method with averaging. *Optimization Methods Software*, 23:161–172, February 2008. 1, 5

[17] B. Savchynskyy, J. H. Kappes, S. Schmidt, and C. Schnörr. A study of Nesterov's scheme for Lagrangian decomposition and MAP labeling. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 1, 3

[18] M. Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4:113–130, 1976. 1

[19] M. Schlesinger and V. Giginyak. Solution to structural recognition (max,+)-problems by their equivalent transformations. *Control Systems and Machines*, (1,2), 2007. 1

[20] S. Schmidt, B. Savchynskyy, J. H. Kappes, and C. Schnörr. Evaluation of a first-order primal-dual algorithm for MRF energy minimization. In *EMMCVPR*, volume 6819 of *LNCS*, pages 89–103. Springer, 2011. 1

[21] N. Shor. *Minimization methods for non-differentiable functions*. Springer series in computational mathematics. Springer-Verlag, 1985. 3

[22] P. Strandmark, F. Kahl, and T. Schoenemann. Parallel and distributed vision algorithms using dual decomposition. *Computer Vision and Image Understanding*, 2011. 1, 3, 6

[23] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30:1068–1080, June 2008. 5, 6

[24] M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA, 2008. 1, 2, 3

[25] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, July 2007. 1

[26] J. Yarkony, A. Ihler, and C. Fowlkes. Planar cycle covering graphs. In *UAI*, 2011. 1, 3

A. Appendix

A.1. Solving the Trust-Region Problem

With the notation

$$\mathcal{B} = \left\{ (f(\lambda^i), \lambda^i, g^i) \mid i = 1, \ldots, N \right\}$$

for the bundle set we can reformulate problem (20) into

$$\max_{\lambda} \min_{\xi} \sum_{i=1}^{N} \xi_i \left(f(\lambda^i) + \langle g^i, \lambda - \lambda^i \rangle \right) - \frac{w}{2} \|\lambda - \bar{\lambda}\|_2^2$$
$$\text{s.t. } \xi_i \in [0,1], \sum_{i=1}^{N} \xi_i = 1 \tag{22}$$

Instead of (22) we solve its dual

$$\min_{\xi} \max_{\lambda} \sum_{i=1}^{N} \xi_i \left(f(\lambda^i) + \langle g^i, \lambda - \lambda^i \rangle \right) - \frac{w}{2} \|\lambda - \bar{\lambda}\|_2^2$$
$$\text{s.t. } \xi_i \in [0,1], \sum_{i=1}^{N} \xi_i = 1, \tag{23}$$

for which the inner maximization over λ can be solved explicitly for any choice of ξ,

$$\lambda_{\min} = \sum_{i=1}^{N} \frac{\xi_i g^i}{w}.$$

Substituting $\lambda_{\min}$ into (23) we obtain

$$\min_{\xi} \sum_{i=1}^{N} \xi_i \left(f(\lambda^i) + \langle g^i, \lambda_{\min} - \lambda^i \rangle \right) - \frac{w}{2} \|\lambda_{\min} - \bar{\lambda}\|_2^2$$
$$\text{s.t. } \xi_i \in [0,1], \sum_{i=1}^{N} \xi_i = 1. \tag{24}$$

This quadratic program in ξ can be solved efficiently by interior point methods. We use the implementation included in the Conic Bundle Library [5], which is more general than required in (24).

A.2. Constrained Dual Problem

For the construction of the constraint dual we force consistency, by an additional variable x.

$$\min_{\tilde{x} \in \tilde{X}, x \in X} J_{\tilde{G}}(\tilde{x}) \qquad \text{s.t.} \forall a \in \tilde{V} : \tilde{x}_a = x_{\rho(b)} \tag{25}$$

Again we change to the LP-formulation and relax $\mu \in \mathcal{M}(G)$ to $\mu \in \mathbb{R}^N$

$$\min_{\tilde{\mu} \in \mathcal{M}(\tilde{G}), \mu \in \mathbb{R}^N} \langle \tilde{\mu}, \tilde{\theta} \rangle \qquad \text{s.t. } B\,\mu = \tilde{\mu} \tag{26}$$

and include the constraint $B\,\mu = \tilde{\mu}$ via Lagrangian multipliers

$$\min_{\tilde{\mu} \in \mathcal{M}(\tilde{G}), \mu \in \mathbb{R}^N} \max_{\lambda} \langle \tilde{\theta}, \tilde{\mu} \rangle + \langle \lambda, B\,\mu - \tilde{\mu} \rangle \tag{27}$$

By calculating the derivative for μ,

$$\frac{\partial}{\partial \mu} \max_{\lambda} \langle \tilde{\theta}, \tilde{\mu} \rangle + \langle \lambda, B\,\mu - \tilde{\mu} \rangle = B^\top \mu$$

we are able to replace μ but have to enforce that $\lambda \in \{\lambda | B^\top \lambda = 0\} = \Lambda$. Overall this ends in the following constraint dual problem

$$\max_{\lambda \in \Lambda} \min_{\tilde{\mu} \in \mathcal{M}(\tilde{G})} \langle B^\top \lambda + \tilde{\theta}, \tilde{\mu} \rangle. \tag{28}$$

The Euclidean projection on Λ can be calculated by

$$P_\Lambda(\lambda) = (I - B(B^\top B)^{-1} B^\top)\,\lambda$$

A.3. Parameters

For the bundle method as well as for the update rule of Kiwiel, we used the default parameters of the ConicBundle library. While tuning them for special problem classes would give further improvements, we found it more appealing to keep them fixed. The parameter for deciding if a null-step or serious-step is performed $m_L = 0.1$ as well as the parameter used in Kiwiel's update to decide if the trust region should be increased, $m_r = 0.5$, were not changed in all experiments. The relative precision condition for termination was set to $\epsilon = 0$. For our own adaptive update rule we choose $w_{\min} = 10^{-10}$ and $w_{\max} = 10^1$ and set $\gamma = 0.1$ for stereo and $\gamma = 1$ for binary problems. For the nonsummable diminishing step size rule we set $\alpha = 0.01$, $\gamma = 1$ for stereo and $\gamma = 32$ for binary problems. For the nonsummable diminishing step length rule we choose $\alpha = 0.01$ and $\gamma = 100$. For the adaptive step length rule we choose $\gamma = 0.1$ for stereo and $\gamma = 1$ for binary problems.