Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs

Karteek Alahari, Pushmeet Kohli, Philip H. S. Torr

Dynamic Hybrid Algorithms for MAP Inference in Discrete MRFs

Karteek Alahari, *Student Member, IEEE,* Pushmeet Kohli, *Member, IEEE,* and Philip H. S. Torr, *Senior Member, IEEE*

Abstract—In this paper, we present novel techniques that improve the computational and memory efficiency of algorithms for solving multi-label energy functions arising from discrete MRFs or CRFs. These methods are motivated by the observations that the performance of minimization algorithms depends on: (a) the initialization used for the primal and dual variables; and (b) the number of primal variables involved in the energy function. Our first method (dynamic α-expansion) works by 'recycling' results from previous problem instances. The second method simplifies the energy function by 'reducing' the number of unknown variables present in the problem. Further, we show that it can also be used to generate a good initialization for the dynamic α-expansion algorithm by 'reusing' dual variables.
We test the performance of our methods on energy functions encountered in the problems of stereo matching, and colour and object based segmentation. Experimental results show that our methods achieve a substantial improvement in the performance of α-expansion, as well as other popular algorithms such as sequential tree-reweighted message passing, and max-product belief propagation. We also demonstrate the applicability of our schemes for certain higher order energy functions, such as the one described in [1], for interactive texture based image and video segmentation. In most cases we achieve a 10-15 times speed-up in the computation time. Our modified α-expansion algorithm provides similar performance to Fast-PD [2], but is conceptually much simpler. Both α-expansion and Fast-PD can be made orders of magnitude faster when used in conjunction with the 'reduce' scheme proposed in this paper.

Index Terms—Markov Random Fields, Multi-label Problems, Energy Minimization, Approximate Algorithms.

1 Introduction

MANY problems in computer vision such as image segmentation, stereo matching, image restoration, and panoramic stitching involve inferring the maximum a posteriori (MAP) solution of a probability distribution defined by a discrete MRF or CRF [3], [4], [5], [6]. The MAP solution can be found by minimizing an energy or cost function. In the last few years, driven by its applicability, energy minimization has become a very active area of research [6]. Although, minimizing a general MRF energy function is an NP-hard problem [3], there exist a number of powerful algorithms which compute the exact solution for a particular family of energy functions in polynomial time. For instance, max-product (min-sum) belief propagation exactly minimizes energy functions defined over graphs with no loops [7]. Similarly, certain submodular energy functions can be minimized by solving an st-mincut problem [8], [9], [10], [11].

Efficient algorithms have also been proposed for functions which do not fall under the above classes [3], [12], [13]. Expansion and swap move making algorithms, sequential tree-reweighted message passing, and belief propagation are examples of popular methods for solving these functions. They have been shown to give excellent results on discrete MRFs typically used in computer vision [3], [6]. However, these algorithms can take a considerable amount of time to solve problems which involve a large number of variables. As computer vision moves towards the era of large videos and gigapixel images, computational efficiency is becoming increasingly important. Indeed, the last few years have seen a lot of attention being devoted to increasing the performance of minimization algorithms [2], [14], [15], [16].

We make two contributions to improve the efficiency of energy minimization algorithms. Our first contribution is a method which works by reusing results from previous problem instances, providing a simpler alternative to the recent work of [2] on dynamic energy minimization. Our second contribution is a method which simplifies the energy minimization problem by reducing the number of variables in the energy function. Further, it can also be used to speed-up the inference of the optimal values of the remaining variables. We also demonstrate the applicability of these methods for certain higher order energy functions [1].

Recycling Solutions: Our first method is inspired by the dynamic computation paradigm [2], [15], [16]. It improves the performance of the α-expansion algorithm by reusing results from previous problem instances. The idea of dynamic computation has been used in the recent work of [15], [16] on minimizing submodular energy functions. In particular, [16] showed how flow can be reused in maxflow algorithms, and [15] showed how cuts (or previous labelling) can be reused. However,

- *K. Alahari and P. H. S. Torr are with the Department of Computing, Oxford Brookes University, Oxford, U.K.*
  E-mail: see http://cms.brookes.ac.uk/research/visiongroup
- *P. Kohli is with Microsoft Research Cambridge.*

This work was supported by the EPSRC research grants EP/C006631/1(P) and GR/T21790/01(P), the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. P. H. S. Torr is in receipt of Royal Society Wolfson Research Merit Award.

these methods are only applicable for the special case of dynamic MRFs[1] that are characterized by submodular energy functions. Our work extends these methods to non-submodular multi-label energy functions. It is most similar to the interesting Fast-PD algorithm proposed by Komodakis *et al*. [2], which generalizes the work of [17] and [16]. Fast-PD works by solving the energy minimization problem by a series of graph cut computations. This process is made efficient by reusing the primal and dual solutions of the linear programming (LP) relaxation of the energy minimization problem, achieving a substantial improvement in the running time. Our modified dynamic α-expansion algorithm is conceptually much simpler and easier to implement than Fast-PD whilst giving similar performance. Our method of initializing the α-expansion algorithm can make both methods orders of magnitude faster.

Simplifying energy functions: Most energy minimization problems encountered while solving computer vision problems are composed of *easy* and *difficult* components [18], [19]. For instance, the variables labelled by the QPBO algorithm [18], [20] constitute the easy component, while the rest constitute the difficult component. The globally optimal labels for variables constituting the easy component of the MRF energy function can be found in a few iterations of the minimization algorithm, while those of the difficult part typically cannot be found in polynomial time (in the number of variables). Energy minimization algorithms generally do not take advantage of this decomposition, and process all the random variables at every iteration.

We propose a novel strategy which solves a given discrete MRF in two phases. In the first phase a partially optimal solution of the energy function is computed [18], [19], [20]. In such solutions, not all variables are assigned a label. However, the set of variables which are assigned a label, are guaranteed to take the same labelling in at least one of the optimal solutions of the energy function. This is referred to as the property of *partial optimality*. Using the partial solutions to fix values of these variables results in a *projection* (cf. section 2) of the original energy function [11]. In the second phase we minimize this simplified energy which depends on fewer variables, and is easier and faster to minimize compared to the original energy function. This approach is applicable to many popular energy minimization approaches such as α-expansion, BP, Fast-PD and TRW-S. We also show how to achieve a substantial speed-up in the minimization of the simplified energy by reusing results from computations performed to find the partially optimal solution.

Outline of the Paper: In section 2, we provide the notation used in the paper and review the basics of energy minimization and submodular functions. Algorithms for approximate energy minimization [3], [12] and computing partially optimal solutions [19], [20] are briefly described in the same section. Section 3 presents our two methods to improve the running time of algorithms for minimizing multi-label energy functions. Specifically, it describes methods to: (a) recycle the primal and dual solutions to obtain a good initialization for the new problem instance, and (b) reduce energy functions and reuse the resulting residual graphs. Section 4 shows the applicability of our methods for certain higher order energy functions, such as those containing the $\mathcal{P}^n$ model potentials proposed by Kohli *et al*. [1], for interactive texture based image and video segmentation. We also present a scheme to compute partially optimal solutions for this model. In section 5, we evaluate the performance of our methods on the problems of colour and object based segmentation [21], [22], and stereo matching [6]. A few examples of these problems are shown in Fig. 1. Summary and discussion are provided in section 6.

1. MRFs that vary over time [15], [16].

2 Preliminaries

The notation and basic definitions relevant to our work are provided here. Consider a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, where each variable $X_i \in \mathbf{X}$ takes a value from the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$. A labelling $\mathbf{x}$ refers to any possible assignment of labels to the random variables and takes values from the set $\mathcal{L}^n$. The label set corresponds to disparities in the case of stereo matching problem, and segments in the case of the segmentation problem.

An energy function $E : \mathcal{L}^n \rightarrow \mathbb{R}$ maps any labelling $\mathbf{x} \in \mathcal{L}^n$ to a real number $E(\mathbf{x})$ called its energy or cost. Energy functions are the negative logarithm of the posterior probability distribution of the labelling. Maximizing the posterior probability is equivalent to minimizing the energy function and leads to the MAP solution, which is defined as $\mathbf{x_{map}} = \arg\min_{\mathbf{x}\in\mathbf{L}} E(\mathbf{x})$.

Energy functions typically used in computer vision can be decomposed into a sum over unary (ϕ_i) and pairwise (ϕ_{ij}) potential functions as:

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \phi_{ij}(x_i, x_j), \qquad (1)$$

where $\mathcal{V}$ is the set of all random variables and $\mathcal{E}$ is the set of all pairs of interacting variables. The unary potential $\phi_i(x_i)$ represents the cost of the assignment: $X_i = x_i$, while the pairwise potential $\phi_{ij}(x_i, x_j)$ represents that of the assignment: $X_i = x_i$ and $X_j = x_j$.

Limiting the energy functions to pairwise potentials severely restricts the representational power of these models. Researchers have recognized this fact and have used higher order models to improve the expressive power of MRFs and CRFs [1], [23], [24], [25]. Efficient methods to solve a certain class of higher order potential functions have also been presented [1], [26]. Given a neighbourhood system $\mathcal{N}$, a clique c is specified by a set of random variables $\mathbf{X}_c$ such that $\forall i, j \in c, i \in \mathcal{N}_j$ and $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ and $\mathcal{N}_j$ are the sets of all neighbours

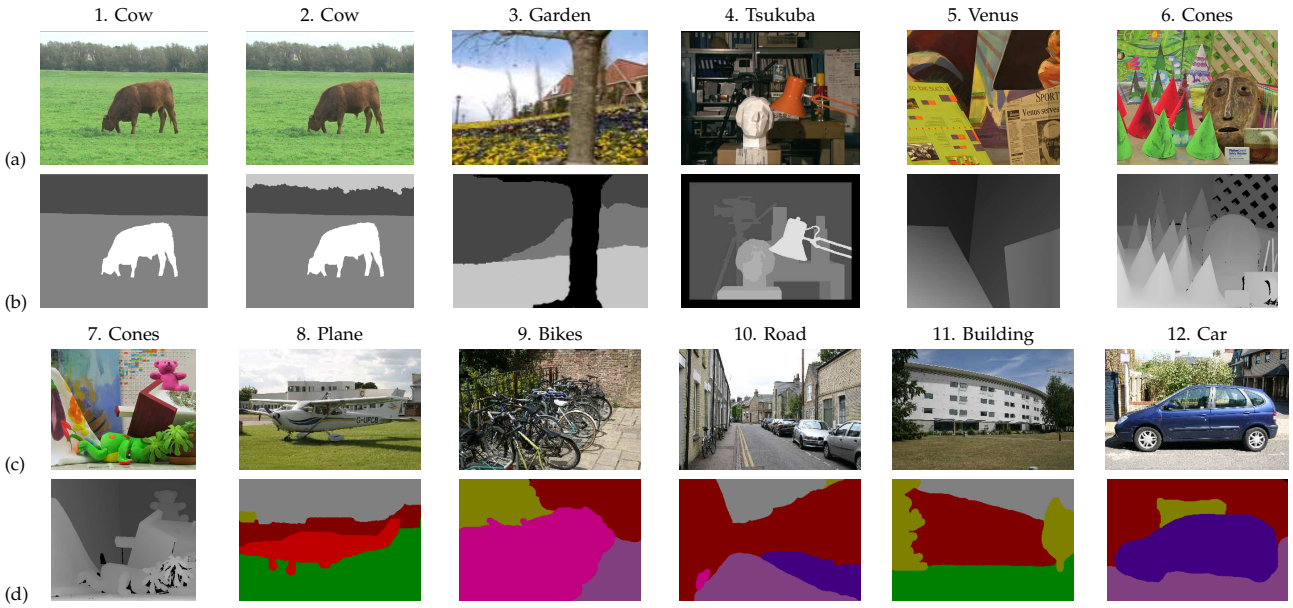


Fig. 1. *Some of the images (a,c) and their ground truth labellings (b, d) used in our experiments. 1-3 Colour-based segmentation problems with 3, 4, 4 labels respectively. 4-7 Stereo matching problems with 16, 20, 60, 60 labels respectively. 8-12 Object-based segmentation problems with 4, 5, 5, 7, 8 labels respectively. (This figure is best viewed in colour.)*

of variable X_i and X_j respectively. We denote the higher order potential energy functions as:

$$E_h(\mathbf{x}) = \sum_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c). \tag{2}$$

The term $\phi_c(\mathbf{x}_c)$ is known as the potential function of the clique c, where $\mathbf{x}_c = \{x_i, i \in c\}$. Note that restricting the clique size in equation (2) to at most 2 produces energy functions of the form (1).

Energy Projection: A *projection* of any function $f(.)$ is a function f^p obtained by fixing the values of some of the arguments of $f(.)$. For instance, fixing the value of the first t variables of the energy function $E(x_1, x_2, \ldots, x_n) : \mathcal{L}^n \rightarrow \mathbb{R}$ produces the projection $E^p(x_{t+1}, x_{t+2}, \ldots, x_n) : \mathcal{L}^{n-t} \rightarrow \mathbb{R}$.

Energy Reparameterization: Energy functions E_1 and E_2 are called *reparameterizations* of each other if and only if $\forall \mathbf{x}, E_1(\mathbf{x}) = E_2(\mathbf{x})$ [12], [20]. Note that this simply means that all possible labellings $\mathbf{x}$ have the same energy under both functions E_1 and E_2, and does not imply that E_1 and E_2 are composed of the same potential functions.

Submodular Functions: Submodular functions are discrete analogues of convex functions. They are particularly important because they can be minimized in polynomial time [20], [27]. Given an *ordering* over the label set $\mathcal{L}$, a function $f(.)$ is submodular if all its projections on two variables satisfy the constraint:

$$f^p(a,b) + f^p(a+1,b+1) \leq f^p(a,b+1) + f^p(a+1,b), \tag{3}$$

for all $a, b \in \mathcal{L}$. Kolmogorov and Zabih [11] showed that all submodular functions of binary variables which can be decomposed into potential functions with at most three variables as arguments can be minimized exactly by solving an st-mincut problem. Later, Ishikawa [10], Zalesky [28], Schlesinger and Flach [29] provided solutions for the multi-label case.

Most multi-label energy functions encountered in computer vision do not satisfy the constraint (3) and thus are not submodular. For instance, it can be clearly seen that the Potts model potential ψ defined as:

$$\psi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ \gamma & \text{otherwise,} \end{cases} \tag{4}$$

does not satisfy the constraint (3). For example, choosing $a = k$ and $b = k + 1$ in (3) violates the constraint.

A number of algorithms have been proposed to efficiently find approximate or *partially optimal* solutions of these energy functions [3], [13], [18], [19], [20]. Our techniques to improve the computational efficiency are based on these algorithms.

2.1 Approximate Energy Minimization

We now give a brief summary of popular and commonly used algorithms for approximate energy minimization.

2.1.1 *Move making algorithms*

The α-expansion and $\alpha\beta$-swap algorithms are widely used for approximate energy minimization [3], [6]. These algorithms work by starting from an initial labelling $\mathbf{x}$ and making a series of moves (label changes) which lower the energy iteratively. Convergence is achieved when the energy cannot be decreased further. An optimal move (one that decreases the energy of the labelling by the most amount) is made at every step.

An α-expansion move allows a random variable to either retain its current label or take a label α. One iteration of the algorithm involves performing expansions

for all $\alpha \in \mathcal{L}$ in some order successively. Boykov *et al*. [3] showed that the optimal expansion moves for energy functions of the form (1) can be computed in polynomial time by solving an st-mincut problem if the pairwise potential functions ϕ_{ij} define a *metric*.

An $\alpha\beta$-swap move allows a random variable whose current label is α or β to either take a label α or β. One iteration of the algorithm involves performing swap moves for all pairs of labels $\alpha, \beta \in \mathcal{L}$ in some order successively. Optimal swap moves for energy functions of the form (1) can be computed in polynomial time if ϕ_{ij} defines a *semi-metric* [3].

2.1.2 Message passing algorithms

These algorithms work by passing messages between nodes representing the different random variables of the model. Max-product belief propagation (BP) is a popular and well-known message passing algorithm for MAP inference [30]. It is guaranteed to produce the exact solution on graphs of tree topology. However, on loopy graphs it is not guaranteed to converge.

Wainwright *et al*. [13] proposed a new message passing algorithm called tree-reweighted message passing (TRW) which worked by solving the dual problem of maximizing a lower bound on the energy. However, their algorithm was not guaranteed to increase this bound in successive iterations, and also lacked convergence guarantees. Kolmogorov [12] developed a modification of this algorithm called sequential tree-reweighted message passing (TRW-S) with the property that the lower bound is guaranteed not to decrease. Other variants of message passing algorithms have also been proposed [31], [32], [33].

2.2 Computing Partially Optimal Solutions

Some algorithms for minimization of non-submodular functions return a partial solution $\mathbf{x} \in (\mathcal{L} \cup \{\epsilon\})^n$ of the energy. We define the assignment $x_i = \epsilon$ to imply that no label has been assigned to random variable X_i. For instance, the QPBO algorithm [18], [20] for minimizing energy functions of binary variables returns a partially labelled solution $\mathbf{x}$ with the following property: that there exists a global minimum $\mathbf{x}^*$ of the energy function such that $x_p = x_p^*$ for all variables X_p that are labelled, *i.e.* $x_p \neq \epsilon$. This property of a partial solution is called *weak persistency*. There are certain partial solutions of the energy for which a *stronger* condition called *strong persistency* holds true. This property states that if a variable X_p is labelled, then it is assigned the same label in all global minima $\mathbf{x}^*$ of the energy, *i.e.* $x_p = x_p^*$ for all $\mathbf{x}^* \in \{\arg\min_{\mathbf{x}} E(\mathbf{x})\}$. The QPBO algorithm was extended to the multi-label case in [34] by converting the multi-label problem to a binary one. However, this approach is computationally expensive and defeats our aim of achieving fast energy minimization.

Another method for finding partially optimal solutions of multi-label energy functions was recently proposed by [19]. The key step of this algorithm is the construction of a submodular subproblem $\mathcal{P}_k$ for each label $l_k \in \mathcal{L}$.

3 Efficient Energy Minimization

We now present methods to improve the performance of algorithms for minimizing multi-label MRFs. For brevity, we explain the working of these techniques in the context of the α-expansion algorithm. However, our methods are general and are applicable to all popular algorithms such as $\alpha\beta$-swap, BP, Fast-PD and TRW-S (sequential TRW). Experimental results using all these algorithms are presented in the latter sections. We also limit our discussion to energy functions with unary and pairwise terms in this section. Methods for higher order terms are presented in Section 4.

The techniques proposed in this paper are inspired from the observations that the computation time of energy minimization algorithms primarily depends on (a) the initialization used, and (b) the number of variables involved in the energy function. Thus, our primary goals are:

1) To generate a good initialization for the current problem instance which results in a reduction in the amount of computation required for solving the problem.
2) To reduce the number of variables involved in the energy function in an efficient manner.

3.1 Recycling Primal and Dual Solutions

We achieve our first goal of obtaining a good initialization by reusing results from previous problem instances. We now explain the dynamic α-expansion algorithm. As discussed earlier, the α-expansion algorithm works by making a series of changes to the current solution to decrease its energy. In one iteration of the algorithm, it computes moves with respect to every label 'α' ($\in \mathcal{L}$). It finds the optimal changes (or move) to be made by minimizing a binary energy function using the st-mincut algorithm. The binary energy function corresponding to a particular 'α' move will be denoted by $E^\alpha(\mathbf{x}^\alpha)$. It is defined as:

$$E^\alpha(\mathbf{x}^\alpha) = \sum_{i \in \mathcal{V}} \phi_i^\alpha(x_i^\alpha) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha), \tag{5}$$

where $x_i^\alpha, x_j^\alpha \in \{0, 1\}$. The unary potential $\phi_i^\alpha(x_i^\alpha)$ is given by:

$$\phi_i^\alpha(x_i^\alpha) = \begin{cases} \phi_i(x_i = \alpha) & \text{if } x_i^\alpha = 0, \\ \phi_i(x_i = x_i^{\text{cur}}) & \text{if } x_i^\alpha = 1, \end{cases} \tag{6}$$

where x_i^{cur} is the current label assignment for X_i. The pairwise potentials, for the Potts model in (4), are defined as:

$$\phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha) = \begin{cases} 0 & \text{if } x_i^\alpha = 0, x_j^\alpha = 0, \\ \gamma(1 - \delta(x_i^{\text{cur}} - x_j^{\text{cur}})) & \text{if } x_i^\alpha = 1, x_j^\alpha = 1, \\ \gamma & \text{otherwise,} \end{cases} \tag{7}$$

where $\delta(x_i^{\text{cur}} - x_j^{\text{cur}}) = 1$, if $x_i^{\text{cur}} = x_j^{\text{cur}}$, and 0 otherwise.

The above function is pairwise and *submodular*, if the energy is metric [3]. The problem of minimizing any such function is equivalent to finding the st-mincut in a particular graph. The st-mincut is found by solving the dual problem of maxflow on the same graph. Thus, the primal solution of the above defined problem corresponds to the labels assigned to each variable x_i^{α}, while the dual solution corresponds to the feasible flow solution of the maxflow problem. Note that each α-expansion move requires a different graph.

Reusing Flow across Iterations: When solving an expansion move in a particular iteration, we reuse the flow from the corresponding move in the previous iteration to make the new computation faster. In the first iteration of the algorithm, we build one graph $G_i^1, i = 1, \ldots, k$, for each label expansion. The optimal expansion move for a given label l_i is computed by solving the st-mincut/maxflow problem on the graph G_i^1. Maxflow problems corresponding to all the labels are solved just as in standard α-expansion. In iterations $u > 1$ of the algorithm, instead of creating a new graph G_i^u for a label expansion, we reuse the corresponding graph G_i^{u-1} from the previous iteration exploiting the fact that the two graphs are similar. We use dynamic graph cuts proposed by Kohli and Torr [16] to achieve this. Given the solution of the maxflow problem on a graph, their method efficiently computes the maxflow in a modified version of the graph. Intuitively, the solution of the graph G_i^{u-1} is a good initialization for that of the graph G_i^u. The dynamic update step involves updating the flows and the residual edge capacities. After these update operations, the maxflow algorithm is performed on the residual graph. As the number of changes in the graphs decrease in the latter iterations, the number of update and maxflow computations decrease. Hence, the optimal moves in these iterations are computed efficiently.

For large problems, *i.e.* when the number of labels, k, or the number of pixels, n, is very large, maintaining multiple dual solutions may not be viable due to memory requirements. This issue can be overcome by working with a projected energy function obtained from a partially optimal solution (cf. section 3.2). Thus our method is not only time-efficient but also memory-efficient if the projected energy function involves a small subset of random variables. The recycle scheme for single MRFs is summarized as follows:

1) Construct graphs $G_i^1, i = 1, \ldots, k$, in the first iteration.
2) Compute the maxflow solutions to get the optimal moves.
3) For iterations $u > 1$,
   - Update graphs from iteration $u - 1$.
   - Compute the new maxflow solutions for the residual graphs.

Efficiently Solving Dynamic MRFs: For Dynamic MRFs [16], the task is to solve a problem where the data changes from one problem instance to the next. For instance, this occurs when solving a labelling problem on the image frames of a video sequence. The conventional method to solve such a problem is to use the standard α-expansion algorithm on each problem instance (*e.g.* each time instance) independently. This method is inefficient and would require a lot of computation time. Our method works by using both the primal and dual solutions. The primal solution is generated by reusing the labelling of the previous problem instance. Intuitively, if the data changes minimally from one problem instance to the next, the solution of a particular problem instance provides a good initialization for the subsequent instance.

Consider a labelling problem defined on a video sequence. The first frame in the video sequence is labelled using the single MRF method described above. The primal and dual solutions thus obtained are used to initialize the maxflow/st-mincut problems for the next frame. The labelling (primal solution) of a frame t is initialized with the solution obtained for frame $t-1$. The graphs $G_i^1(t), i = 1, \ldots, k$, corresponding to the first iteration for frame t are obtained by dynamically updating [16] the graphs from the last iteration for frame $t - 1$. With these initializations the maxflow problem for each label is solved as in the single MRF case. In summary,

1) Solve frame 1 as a 'single MRF'.
2) For all frames $t > 1$,
   - Initialize the labelling (primal) using the solution of frame $t - 1$.
   - Initialize the graph flow (dual) from the corresponding solutions for frame $t - 1$.
   - Solve as a 'single MRF'.

These techniques for α-expansion provide similar speed-ups as the Fast-PD algorithm as shown in Section 5.1.

3.2 Reducing Energy Functions

We now propose a method to simplify (reduce the number of unknown variables in) the MRF by solving the *easy* part. Our reduce strategy is applicable to popular energy minimization approaches such as α-expansion, BP, Fast-PD and TRW-S (see section 5). We also show how computations performed during this procedure can be used to efficiently initialize the dynamic α-expansion algorithm described in the previous section.

As discussed earlier, there are two main algorithms for obtaining partially optimal solutions of non-submodular multi-label energy functions. It would be interesting to compare these partially optimal solution algorithms for the segmentation and stereo problems, but is beyond the scope of this paper. We chose to use the algorithm proposed by Kovtun [19] because of its efficiency. The key step of this algorithm is the construction of k auxiliary problems $\mathcal{P}_m$, one for each label $l_m \in \mathcal{L}$. Kovtun

Fig. 2. *Pseudo-code for computing the partially optimal solution of an energy function. An auxiliary problem $\mathcal{P}_j$ for each label l_j is formulated as an st-mincut problem. The solution computed is used to project the energy function E by fixing the values of the labelled variables. After the iteration terminates we obtain a new energy function, E^p, comprising of all the unlabelled variables.*

Input: $\mathbf{X}$, $\mathcal{L} = \{l_1, \ldots, l_k\}$, E
Output: Partially optimal solution
s_j: Set of variables taking label l_j in the partially optimal solution

$E^p \leftarrow E$
for $j \leftarrow 1$ to k **do**
  $\mathcal{P}_j \leftarrow$ Auxiliary problem for label l_j
  $s_j \leftarrow$ Solve(E^p, $\mathcal{P}_j$) (cf. §3.2)
  $E^p \leftarrow$ Project(E^p, s_j)
end for

showed that the solution of problem $\mathcal{P}_m$ could be used to find variables that have the persistency property (as described in §2.2). Thus, by solving all subproblems $\mathcal{P}_m,\ \forall l_m \in \mathcal{L}$, a partial solution which satisfies strong persistency can be obtained.

Specifically, problem $\mathcal{P}_m$ is the minimization of the following binary energy function

$$E^m(\mathbf{x}^m) = \sum_{i \in \mathcal{V}} \phi_i^m(x_i^m) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^m(x_i^m, x_j^m), \qquad (8)$$

where $x_i^m, x_j^m \in \{0, 1\}$. The unary potential $\phi_i^m(x_i^m)$ is given by:

$$\phi_i^m(x_i^m) = \begin{cases} \phi_i(x_i = l_m) & \text{if } x_i^m = 0, \\ \phi_i(x_i = l_i^{\min}) & \text{if } x_i^m = 1, \end{cases} \qquad (9)$$

where $l_i^{\min} = \arg\min_{l \in \mathcal{L} - \{l_m\}} \phi_i(x_i = l)$. For the case of Potts model, the pairwise potentials are defined as[2]:

$$\phi_{ij}^m(x_i^m, x_j^m) = \begin{cases} 0 & \text{if } x_i^m = 0, x_j^m = 0, \\ 0 & \text{if } x_i^m = 1, x_j^m = 1, \\ \gamma & \text{otherwise.} \end{cases} \qquad (10)$$

$E^m(\mathbf{x}^m)$ defines a submodular energy function and can be minimized by solving an st-mincut problem. Let $\mathbf{x}^{m*}$ denote the optimal solution of the subproblem $\mathcal{P}_m$. We extract a partially optimal solution $\mathbf{x} \in (\mathcal{L} \cup \{\epsilon\})^n$ of the multi-label function $E(\mathbf{x})$ as:

$$x_i = \begin{cases} l_m & \text{if } x_i^m = 0, \\ \epsilon & \text{otherwise.} \end{cases} \qquad (11)$$

We repeat this process for all the labels $l_m \in \mathcal{L}$, and merge the solutions to obtain the final partially optimal solution of the original energy function $E(\mathbf{x})$.

2. Although the algorithm proposed in [19] only handles Potts model energies, it can be easily extended to general energy functions [35].

To make this procedure computationally efficient, we project the energy function after every subproblem computation. This involves fixing values of all variables whose optimal labels have already been extracted from the solution of previous subproblem $\mathcal{P}_m$. This reduces the number of unknown variables in the multi-label energy function and makes the computation of subsequent auxiliary problems faster. We summarize this approach in Fig. 2. Our hope is that after solving all auxiliary problems, we would be left with a projection of the original energy function which involves far fewer variables compared to the original function $E(\mathbf{x})$. The experiments described in the next section on MRFs commonly encountered in computer vision confirm this behaviour.

The energy function projection obtained from the procedure described above corresponds to the *difficult* component of the energy function. It depends on the variables whose optimal labels were not found. The original problem is now reduced to finding the labels of these variables. This can be done using any algorithm for approximate energy minimization. Results of this method are shown in Table 2. Next, we show how this process can be made efficient by reusing the solutions of subproblems solved during the partial optimality algorithm. Again, we will describe our technique using the α-expansion algorithm.

Reusing solutions from the partial optimality algorithm: Next we explain how to achieve computational efficiency for solving the difficult part of the MRF. From (5) and (8), it can be seen that the energy functions corresponding to the subproblems of the partial optimality and α-expansion algorithms have the same form. Thus we can reuse the solutions of the partial optimality subproblems to make the computation of the α-expansion moves faster. Specifically, we use the dual (flow) solutions of the partial optimality problems to generate an initialization for the expansion moves of the first iteration of the α-expansion algorithm (in a manner similar to that described in the previous section).

The speed-up obtained depends on the similarity of the two problems [2], [16]. Thus, by making the subproblems of the partial optimality and α-expansion algorithms similar, we can improve the running time. We note that for unassigned labels we have some choice as to their initialization, and a natural question arises as to whether any particular initialization is better. Consider the expansion and partial optimality subproblems with respect to a label $\alpha \in \mathcal{L}$, *i.e.* $l_m = \alpha$ in (9). From (6) and (9) it can be seen that the unary potentials of the partial optimality and α-expansion subproblems are identical if the current label assignment for X_i, $x_i^{\text{cur}} = l_i^{\min}$. This can be done by initializing the labelling for the α-expansion algorithm by initializing $x_i = l_i^{\min}$, where $l_i^{\min} = \arg\min_{l \in \mathcal{L}} \phi(x_i = l)$. The pairwise potentials may differ at most by the constant γ for the case $x_i^\alpha = 1, x_j^\alpha = 1$ (cf. (7) and (10)). This change makes the two problems similar and as shown in the experimental results in Fig. 6

results in a speed-up in running time.
Our method is summarized as follows:

1) Compute the partially optimal solution and project the energy function. (Reduce)
2) To label the remaining nodes using α-expansion,
   - Initialize the labelling of each node i to $l_i^{\min}$, where $l_i^{\min} = \arg\min_{l \in \mathcal{L}} \phi_i(x_i = l)$.
   - Update the residual graphs from the k auxiliary problems to construct graphs for the first α-expansion iteration. (Reuse)
   - Restart the maxflow algorithms to compute optimal moves, using flow recycling between expansion moves. (Recycle)

4 Solving $\mathcal{P}^n$ Potts Model Efficiently

We now consider the problem of minimizing energy functions which contain clique potentials which take the form of a $\mathcal{P}^n$ Potts model (introduced in [1]). The $\mathcal{P}^n$ Potts model potential for cliques of size n is defined as:

$$\phi_c(\mathbf{x}_c) = \begin{cases} \gamma_k & \text{if } x_i = l_k, \forall i \in c, \\ \gamma_{\max} & \text{otherwise,} \end{cases} \tag{12}$$

where $\gamma_{\max} > \gamma_k, \forall l_k \in \mathcal{L}$. These potentials can be solved using the α-expansion and $\alpha\beta$-swap move making algorithms. The optimal expansion/swap move is computed by minimizing a binary energy function using the st-mincut algorithm [1]. We describe our methods in the context of the α-expansion algorithm. The binary (higher order) energy function corresponding to a particular 'α' move will be denoted by $E_h^\alpha(\mathbf{x}^\alpha)$. It is defined as:

$$E_h^\alpha(\mathbf{x}^\alpha) = \sum_{i \in \mathcal{V}} \phi_i^\alpha(x_i^\alpha) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha) + \sum_{\substack{c \in \mathcal{C} \\ |c|>2}} \phi_c^\alpha(\mathbf{x}_c^\alpha), \tag{13}$$

where $x_i^\alpha, x_j^\alpha \in \{0,1\}$, $\mathbf{x}_c^\alpha = \{x_i^\alpha, \forall i \in c\}$. The unary potential $\phi_i^\alpha(x_i^\alpha)$ and the pairwise potential $\phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha)$ are given equations (6) and (7) respectively. The clique potential $\phi_c^\alpha(\mathbf{x}_c^\alpha)$ forms a $\mathcal{P}^n$ Potts model, and is given by:

$$\phi_c^\alpha(\mathbf{x}_c^\alpha) = \begin{cases} \gamma_\alpha & \text{if } x_i^\alpha = 0, \forall i \in c, \\ \gamma & \text{if } x_i^\alpha = 1, \forall i \in c, \\ \gamma_{\max} & \text{otherwise,} \end{cases} \tag{14}$$

where $\gamma = \gamma_\beta$ if $x_i^{\text{cur}} = \beta \in \mathcal{L}$, for all $i \in c$, and $\gamma = \gamma_{\max}$ otherwise. The move energy function is submodular and can be represented as an st-mincut graph. The reader is referred to [1] for details of the graph construction.

Recycling Solutions: Once the st-mincut graph corresponding to the higher order move energy is built, our methods for recycling primal and dual solutions (cf. §3.1) are directly applicable. When solving an expansion move in a particular iteration, we reuse the flow from the corresponding move in the previous iteration to make the new computation faster.

Computing Partially Optimal Solutions: We now propose a method to efficiently compute partially optimal solutions of certain higher order energy functions. As in §3.2, our method is based on the algorithm proposed by Kovtun [19]. An auxiliary problem P_m, for label $l_m \in \mathcal{L}$, is the minimization of the following higher order binary energy function:

$$E_h^m(\mathbf{x}^m) = \sum_{i \in \mathcal{V}} \phi_i^m(x_i^m) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^m(x_i^m, x_j^m) + \sum_{\substack{c \in \mathcal{C} \\ |c|>2}} \phi_c^m(\mathbf{x}_c^m), \tag{15}$$

where $x_i^m, x_j^m \in \{0,1\}$, $\mathbf{x}_c^m = \{x_i^m, \forall i \in c\}$. The unary potential $\phi_i^m(x_i^m)$ and the pairwise potential $\phi_{ij}^m(x_i^m, x_j^m)$ are given by equations (9) and (10) respectively. The $\mathcal{P}^n$ Potts clique potential $\phi_c^m(\mathbf{x}_c^m)$ is defined as:

$$\phi_c^m(\mathbf{x}_c^m) = \begin{cases} \gamma_m & \text{if } x_i^m = 0, \forall i \in c, \\ \min\limits_{k \in \mathcal{L}, k \neq m} \gamma_k & \text{if } x_i^m = 1, \forall i \in c, \\ \gamma_{\max} & \text{otherwise.} \end{cases} \tag{16}$$

It can be easily verified that $E_h^m(\mathbf{x}^m)$ is a submodular energy function [29]. We now provide the relevant notation to prove Theorem 1 in [19] for the case of $\mathcal{P}^n$ Potts model.

For every auxiliary problem P_m we consider any ordering of the label set $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ such that label l_m is the *highest* label. This allows us to define a partial ordering can be defined on the set of label pairs $(a, a') \in \mathcal{L} \times \mathcal{L}$. The maximum and minimum for any two label pairs (a, a') and (b, b') are defined as $(a, a') \vee (b, b') = (a \vee b, a' \vee b')$ and $(a, a') \wedge (b, b') = (a \wedge a', b \wedge b')$ respectively. Similarly, the maximum and minimum of any pair of labellings $\mathbf{x}_c$ and $\mathbf{x}'_c$ is denoted by $\mathbf{x}_c \bigvee \mathbf{x}'_c$ and $\mathbf{x}_c \bigwedge \mathbf{x}'_c$ respectively. We also define the *lowest* optimal labelling $\widehat{\mathbf{x}_c^m}$ as follows:

$$\widehat{\mathbf{x}_c} = \bigwedge_{\mathbf{x}_c^* = \arg\min_{\mathbf{x}_c} E(\mathbf{x}_c)} \mathbf{x}_c^*. \tag{17}$$

Using this notation, the submodularity condition, in equation (3), can be written as:

$$f(\mathbf{x}_c) + f(\mathbf{x}'_c) \geq f(\mathbf{x}_c \bigvee \mathbf{x}'_c) + f(\mathbf{x}_c \bigwedge \mathbf{x}'_c). \tag{18}$$

Let $\mathbf{y}^m \in \mathcal{L}^n$ denote the partially optimal solution after solving the auxiliary problem corresponding to label l_m (*i.e.* $E_h^m(\mathbf{x})$). In other words, the labelling $x_i^m = 0$ is equivalent to $y_i^m = l_m$, and $x_i^m = 1$ to the random variable X_i retaining the initial label.

Theorem 4.1: An arbitrary solution of the initial problem $\mathbf{x}^* = \arg\min_{\mathbf{x}} E_h(\mathbf{x})$ satisfies the following condition: $\mathbf{x}^* \bigwedge \widehat{\mathbf{y}^m} = \widehat{\mathbf{y}^m}$, where $\widehat{\mathbf{y}^m}$ denotes the lowest optimal labelling for the auxiliary problem.

In other words, this theorem states that the persistency property holds for our higher order energy function. We use the following Lemma from [19] to prove the theorem.

Lemma 4.2: Let $\widehat{\mathbf{x}}$ be the lowest optimal labelling for a submodular problem, and $\mathbf{x}^*$ be any arbitrary

labelling satisfying the condition $\mathbf{x}^* \bigwedge \widehat{\mathbf{x}} \neq \widehat{\mathbf{x}}$, then $E_h(\mathbf{x}^*) > E_h(\mathbf{x}^* \bigvee \widehat{\mathbf{x}})$[3].

Proof of Theorem 4.1: Our proof is similar to that given in [19]. Let us assume for any labelling $\mathbf{x}$, $\mathbf{x} \bigwedge \widehat{\mathbf{y}^m} \neq \widehat{\mathbf{y}^m}$. From Lemma 4.2 it follows that

$$E_h^m(\mathbf{x} \bigvee \widehat{\mathbf{y}^m}) < E_h^m(\mathbf{x}). \tag{19}$$

The following inequality is obtained from equations (12) and (16):

$$\phi_c^m(\mathbf{x}_c \vee \widehat{\mathbf{y}_c^m}) - \phi_c^m(\mathbf{x}_c) \geq \phi_c(\mathbf{x}_c \vee \widehat{\mathbf{y}_c^m}) - \phi_c(\mathbf{x}_c). \tag{20}$$

Also, from [19],

$$\begin{aligned} \phi_{ij}^m(x_i \vee \widehat{y_i^m}, x_j \vee \widehat{y_j^m}) - \phi_{ij}^m(x_i, x_j) \geq \\ \phi_{ij}(x_i \vee \widehat{y_i^m}, x_j \vee \widehat{y_j^m}) - \phi_{ij}(x_i, x_j). \end{aligned} \tag{21}$$

Using inequalities (19), (20) and (21) it can be easily shown that,

$$E_h(\mathbf{x} \bigvee \widehat{\mathbf{y}^m}) < E_h(\mathbf{x}), \tag{22}$$

which was to be proved. □

Thus, the persistency property holds for our higher order energy function. We extract a partially optimal solution of the multi-label function $E_h(\mathbf{x})$ using equation (11). The final partially optimal solution is obtained by repeating this process for all the labels, and merging the solutions.

5 Experiments

We evaluated our methods on a variety of multi-label MRF problems such as stereo matching [3], colour [21], object [22] and texture [1] based segmentation. The details of the unary and pairwise potentials of the energy functions used for formulating these problems are given below.

Colour-based Segmentation: For the colour-based segmentation problem, we used the energy function defined in [21]. The unary potential functions $\phi_i(x_i), i \in \mathcal{V}$ are defined using the RGB distributions $\mathcal{H}_a, a = l_1, \ldots, l_k$, of the k segments as follows:

$$\phi_i(x_i) = -\log p(x_i = a|\mathcal{H}_a), \tag{23}$$

The distributions $\mathcal{H}_a$ are obtained using user specified constraints. The pairwise potentials encourage contiguous segments while preserving the image edges [21], and take the form of a Generalized Potts model defined as:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \lambda_1 + \lambda_2 \exp(\frac{-g^2(i,j)}{2\sigma^2}) \frac{1}{\text{dist}(i,j)} & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j, \end{cases} \tag{24}$$

where λ_1, λ_2 and σ are parameters of the model. The terms $g(i,j)$ and $\text{dist}(i,j)$ give the difference in the RGB values and the spatial distance respectively between pixels i and j. We used the following parameter values for all our experiments with this energy function: $\lambda_1 = 5, \lambda_2 = 100$ and $\sigma = 5$. Segmentation results are shown on the well-known garden image and a cow image used in [15], [16].

(a) (b) (c) (d)

Fig. 3. *(a) The keyframe of the 'Dayton' video sequence and (b) its segmentation. (c) An image from the* MSRC-21 *database, and (d) the brush strokes marked by the user. The keyframe segments and brush strokes are used to learn the colour histogram models and the patch dictionaries.*

Stereo Matching: We used the standard energy function for stereo matching problem [19], [36]. The unary potentials of the energy are computed using a fixed size window-based method similar to the one used in [19]. The pairwise potentials take the form of a Potts model (4). Stereo matching results are shown on "Tsukuba", "Venus", "Cones", "Teddy" images from the Middlebury stereo data set [36]. The Potts model smoothness cost γ was set to 20 for all our experiments on this energy function.

Object-based Segmentation: For this problem we used the energy function defined in [22]. The unary potentials of this energy are based on shape-texture, colour, and location. They are learnt using a boosting procedure with textons and shape filter features. The pairwise potentials take the form of a contrast sensitive Potts model (24). The reader is referred to [22] for more details on the energy function. We evaluated our algorithms on energy functions corresponding to some images of the MSRC-21 database.

Texture-based Segmentation: Given a set of distinct textures, such as texton histograms [37] or a dictionary of RGB patches, together with their object class labels, the task is to segment an image. The unary and pairwise terms are defined as in the object-based segmentation example. The higher order potential is defined as a function of the difference between a patch and a dictionary of $n_p \times n_p$ RGB patches. In the case of a video sequence (Dynamic MRF), the patches are computed using one user segmented keyframe, while user marked strokes are used in the case of an image (Single MRF). Sample images are shown in Fig. 3. More details of the potentials can be found in [1].

The following sections describe the results of primal and dual, and partially optimal solution initializations. Standard, publicly available implementations are used for comparison[4]. All experiments were performed on a Intel Core 2 Duo, 2.4 GHz, 3GB RAM machine. Source code for the proposed methods is available online[5].

3. The lemma can be proved using the submodularity condition in equation (18) and the definition of lowest optimal labelling. See [19] for more details.

4. We thank V. Kolmogorov, N. Komodakis and M. Pawan Kumar for providing the original implementation of their methods for comparison.

5. See `http://cms.brookes.ac.uk/research/visiongroup`

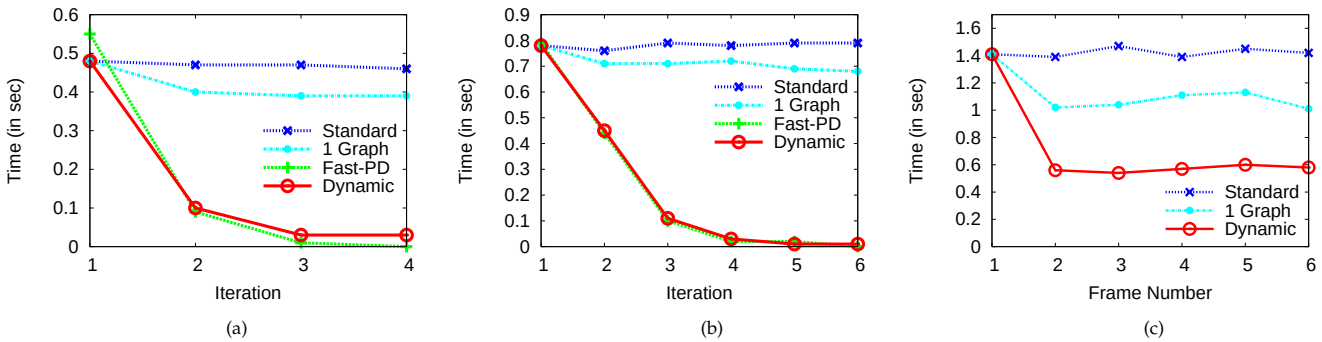


Fig. 4. *Reusing primal and dual solutions for (a), (b) single and (c) dynamic* MRF *problems: Comparison of run-times of standard and dynamic versions of α-expansion, and Fast-PD are shown for (a) object-based segmentation problem: 'Building' image from the TextonBoost dataset [22], (b) stereo matching problem: Tsukuba (Left image), and (c) colour-based segmentation problem: cow video sequence [15], [16]. In (a), (b) reusing the dual solution provides a speed-up of at least 4-10 times in subsequent iterations. In some cases, the first iteration of Fast-PD was slightly slower compared to both versions of α-expansion algorithm, but the overall computation time was better than 'standard' and comparable to 'dynamic'. For example, times for the 'Building' image are: Fast-PD: 0.65s, dynamic: 0.64s, standard: 1.88s. Note that the run-times of Fast-PD and our dynamic version are very similar in (a) and (b). In (c) the dynamic version reuses primal and dual solutions from the previous frames in the video sequence and results in 3-4 times speed-up. We also show that the strategy of maintaining only one graph while recycling solutions (denoted by '1 Graph') provides insignificant speed-up (see text).*

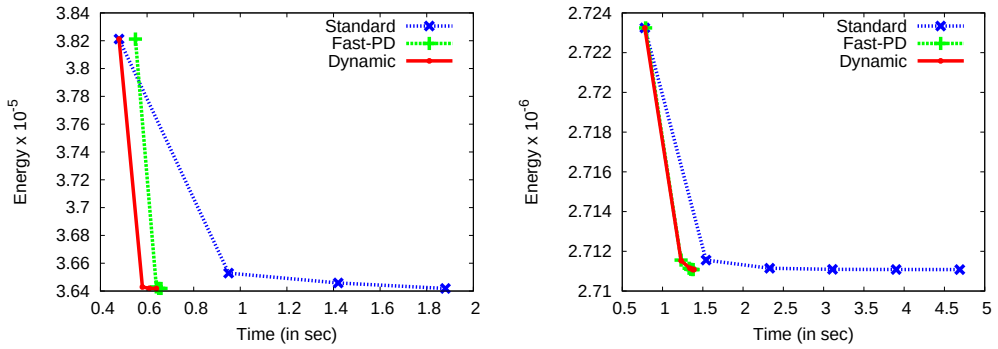
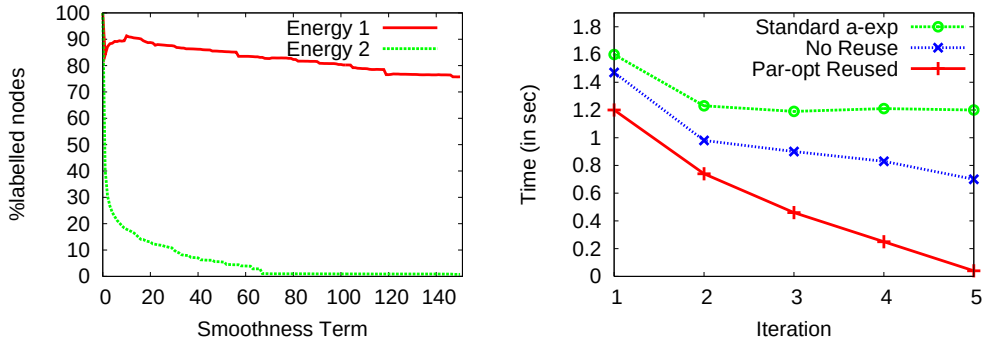


Fig. 5. *Comparison of run-times and solution energy of standard and dynamic versions of α-expansion and Fast-PD are shown for (a) 'Building' image, (b) Tsukuba (Left image). Although there is a small change in energy after iteration 1, Standard α-expansion spends much more time compared to our Dynamic version to obtain a new lower energy solution. The time vs energy plot for Fast-PD is very similar to dynamic α-expansion, except for iteration 1 in (a), where Fast-PD takes 0.07 seconds more than our dynamic algorithm.*

5.1 Dynamic α-expansion

We now discuss the effect of various primal and dual solution initializations on the α-expansion algorithm. We tested a simple of way of using the flow/cut from the solution of the previous expansion move (*i.e.* with a different label) as an initialization for the current move. From (5) it can be observed that the energy functions corresponding to two consecutive moves are substantially different. Hence, this scheme provides no significant speed-up. Fig. 4 confirms this expected behaviour.

Fig. 4(a), Fig. 4(b) show the results of the proposed 'recycle' strategy for two single MRF examples. The primal and dual solutions are recycled across iterations (cf. §3.1). The standard and dynamic versions take the same time in the first iteration, as no flow is recycled. In the subsequent iterations, the dynamic version provides a speed-up of 4-10 times. Similar results were observed for other problems as well. The approach of initializing both primal and dual solutions in a dynamic MRF was tested on the cow video sequence [15], [16]. These run-times for a sequence of 6 images are shown in Fig. 4(c). Our initialization method provides a speed-up of 3-4 times in this case. The graphs also compare the dynamic methods with Fast-PD [2]. Note that our methods resulted in very similar run-times compared to Fast-PD. Fig. 5 shows a comparison of run-time and solution energy for standard and dynamic versions of α-expansion. From Fig. 4 and Fig. 5 we see that the speed-up achieved by our dynamic version is due the fact that small changes in energy can be computed very efficiently. Table 1 shows the speed-up obtained for the $\mathcal{P}^n$ Potts model. Our method for



(a) (b)

Fig. 6. *(a) The percentage of nodes labelled by the partially optimal solution algorithm by varying the smoothness cost for two energy functions. The Tsukuba stereo matching problem with energy functions given in [19] (Energy 1) and [6] (Energy 2) is used as the example here. For the smoothness cost $\gamma = 20$, only 13% of the nodes are labelled in the case of 'Energy 2'. (b) The energy function in [6] (Energy 2) with smoothness cost $\gamma = 20$ is used for this experiment on the Tsukuba sequence. The speed-up obtained by reusing the flows from the partially optimal solution auxiliary problems (Par-opt) for this smoothness cost is shown. Reusing the flows provides a run-time improvement of at least 5 times in the last two iterations, and more than 2 times overall improvement. Note that even when the partially optimal solution algorithm fails, we obtain a significant speed-up.*

TABLE 1

Running times (in seconds) for various examples (from [1]) using the $\mathcal{P}^n$ Potts model. Results are shown for the α-expansion algorithm. 'α-exp' refers to the times obtained using the standard alpha expansion algorithm and 'dyn α-exp' refers to the dynamic version (which recycles primal and dual solutions). 'opt α-exp' refers to the optimized version which computes the partially optimal solution followed by α-expansion on the energy projection. It is observed that both 'dyn' and 'opt' methods provide a speed-up of at least 3-6 times compared to the standard method. The numbers in () denote the number of labels in the problem.

	Time (in seconds)		
	α-exp	dyn α-exp	opt α-exp
Dayton (3)	1.31	0.49	0.21
Garden (4)	1.20	0.44	0.19
Bench (3)	1.76	0.59	0.38
Beach (4)	1.59	0.51	0.25

recycling solutions provides a speed-up of at least 3-5 times compared to standard α-expansion.

5.2 Using Partially Optimal Solutions

We now show the results of our partially optimal solution based method (cf. §3.2) on a variety of energy minimization algorithms for the problems defined above. Specifically, α-expansion, BP and TRW-S are used for the experiments. Optimized versions of BP and TRW-S refer to the computation of partially optimal solution followed by running the corresponding algorithm on the projected energy function. A comparison of the run-times for all these algorithms is shown in Table 2. It is observed that our method achieves a speed-up is 10-15 times for most of the examples. In some cases (*e.g.* Cow image with 3 labels), the speed-up is more than 100 times for optimized versions of TRW-S and BP algorithms. The amount of speed-up depends on the strength of the pairwise terms and the number of labels in the problem. The speed-up increases with a decrease in both the number of labels and the strength of the pairwise terms. This is because the pairwise potential of the partial optimality auxiliary problem (10) is closely related to that in the original problem (4). Images with highly textured regions also show orders of magnitude speed-up for segmentation and stereo problems. Table 1 shows the speed-up obtained for the $\mathcal{P}^n$ Potts model for various examples (from [1]). Using partially optimal solutions provides a speed-up of at least 4-6 times compared to standard α-expansion.

An analysis of the partially optimal solution algorithm shows that in some cases very few nodes may be labelled. One such case is when the smoothness cost γ is very high, as shown in Fig. 6(a). For illustration purposes we chose the Tsukuba stereo problem, which showed the most significant change in the number of labelled nodes. We used two energy functions [6], [19] on the stereo problem to demonstrate the effect of varying the smoothness term. The unary potential in [19] is computed using a normalized cross correlation approach on pixel windows of size 15×15, while [6] uses the sub-pixel window approach proposed by [38]. The pairwise potential in both cases is the Potts model given by (4). As the smoothness cost is increased, the percentage of labelled nodes decreases, and the projected component of the energy function remains large. The decrease is more dramatic using the energy function in [6]. This effect is perhaps because the partially optimal solution algorithm relies on strong unary potentials. In the case of [6], a large smoothness term dominates the unary

TABLE 2

Running times for various single MRF *problems: Comparison of the run-times (in seconds) of the standard and optimized (opt) versions of α-expansion (α-exp),* BP, TRW-S *is shown. The optimized version refers to computing the partial solution followed by solving the energy projection with the corresponding algorithm. The optimized versions are significantly faster in all the examples. The speed-up obtained depends on the nature and difficulty of the problem. The run-times shown for both* BP *and* TRW-S *versions correspond to the first 70 iterations. The number of iterations was chosen such that acceptable qualitative results (segmentation or stereo map) were obtained for all the problems. Some of the smaller problems produce results after 30-40 iterations, while others take 70-80 iterations. A better comparison of time vs energy is shown in Fig. 5 and Fig. 8. The numbers in () denote the number of labels in each problem.*

	Time (in seconds)						
	α-exp	Fast-PD	opt α-exp	BP	opt BP	TRW-S	opt TRW-S
Colour-based Segmentation:							
Cow (3)	2.53	1.31	**0.21**	95.93	**0.32**	98.36	**0.33**
Cow (4)	3.75	1.72	**0.38**	108.32	**0.42**	111.69	**0.43**
Garden (4)	0.28	0.14	**0.04**	5.59	**0.17**	5.89	**0.21**
Stereo:							
Tsukuba (16)	5.74	1.47	**0.84**	38.19	**4.47**	41.74	**4.67**
Venus (20)	11.87	3.07	**3.03**	67.04	**14.97**	71.46	**16.02**
Cones (60)	42.23	9.48	**4.36**	173.35	**29.41**	182.66	**30.70**
Teddy (60)	44.25	9.56	**8.27**	172.30	**60.35**	182.50	**63.77**
Object-based Segmentation:							
Plane (4)	0.39	0.35	**0.15**	9.41	**0.29**	9.89	**0.30**
Bikes (5)	0.82	0.54	**0.22**	10.69	**0.64**	11.19	**0.70**
Road (5)	0.91	0.51	**0.18**	10.67	**0.60**	11.26	**0.62**
Building (7)	1.32	0.89	**0.38**	12.70	**2.57**	13.52	**2.66**
Car (8)	0.99	0.53	**0.11**	13.68	**0.23**	14.42	**0.24**

(a) (b) (c)

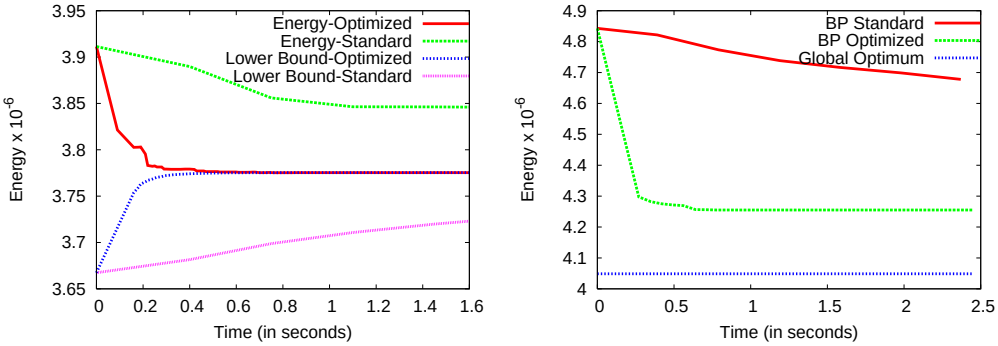
Fig. 7. *A sample result of object-based segmentation is shown in (a) Plane. Some of the stereo matching results are shown in (b) Tsukuba-Left and (c) Teddy-Left. The first row shows the original images. The second row shows the partially optimal solution. The regions marked in red denote the unlabelled pixels, which have low texture detail. Our method provides more than 6$\times$ speed-up even when majority of the nodes are unlabelled in the Teddy example. (This figure is best viewed in colour.)*

potentials, and leads to many unlabelled nodes. Thus, only a small improvement in run-time performance is achieved. *However, our strategy of reusing the flow from the partially optimal solution auxiliary problems always provides improved performance in these cases (see Fig. 6(b)).*

Segmentation and stereo matching results of some of the images used in our experiments are shown in Fig. 7. Note that even when majority of the nodes are unlabelled in the partially optimal solution, *e.g.* Teddy sequence in Fig. 7(c), our method provides more than 6 times speed-up. The proposed method is not only computationally efficient, but also provides a lower energy solution empirically in the case of TRW-S and BP. Furthermore, the optimality of the solutions is not compromised. Fig. 8(a) compares the energies of the solutions and lower bounds obtained using standard and optimized versions of TRW-S. The optimized version using the energy function projection converges to the global optima of the energy in only 0.64 seconds. Fig. 8(b) compares the energies of the solution obtained using the standard and optimized BP algorithms. Optimized BP converges to a low energy (although not the global optima), in 0.85 seconds, while standard BP converges to a much higher energy in 11.12 seconds. Standard BP solves the original (large) problem and converges to a local optima. On the other hand, optimized BP solves the projected energy function defined on a subset of nodes and converges to a better local optima. Empirically we observe that BP is more likely to provide a better local optima on the smaller problem (defined by the projected energy function), which is easier to solve compared to the original large problem. The solutions corresponding to these energies are shown in Fig. 9.

6 Summary

This paper proposes techniques for improving the performance of algorithms for solving multi-label MRFs. As there are no disadvantages in using them and many advantages we would expect them to become standard. Our methods work by recycling solutions from previous problem instances, and reducing energy functions utilizing algorithms for generating partially optimal solutions. Our work on reusing the dual (flow) solution for computing optimal label moves across successive iterations



(a) (b)

Fig. 8. *(a) Energy of the solution and lower bound obtained by running* TRW-S *algorithm on the Road image example [22]. Note that optimized* TRW-S *algorithm finds better energies (lower solution energy and higher lower bound) at any given point in time. It also finds an optima in only 0.64 seconds. Standard* TRW-S *converged to this energy after 37.24 seconds. Thus, the optimized version is more than 50 times faster. (b) Solution energies obtained by running standard and optimized* BP *algorithm on the Building image example [22]. Optimized* BP *refers to the computation of partially optimal solution followed by running the* BP *algorithm on the projected energy function. It finds an energy closer to the global optimum, while standard* BP *does not reach this energy even after 30 seconds.*

(a) (b) (c) (d)

Fig. 9. *(a) Building image [22], and (b) the global optimum solution computed by the* TRW-S *algorithm. Solutions obtained using (c) standard* BP*, and (d) optimized* BP *with an 8-neighbourhood. Neither the optimized nor the standard versions converge to the optimal solution. However, optimized* BP *is closer to the optima.*

of the α-expansion algorithm results in a dynamic algorithm. It can be seen as an extension of the work of [15], [16] for minimizing multi-label non-submodular energy functions. Experimental results show that our methods provide a substantial improvement in the performance of α-expansion, TRW-S, and BP algorithms. Our method also provides similar or better performance compared to Fast-PD. We expect that our techniques for simplifying energy functions, and the subsequent recycling of computations performed during this procedure can be used to make Fast-PD faster. This is a topic for future research.

REFERENCES

[1] P. Kohli, M. P. Kumar, and P. H. S. Torr, "P^3 & beyond: Move making algorithms for solving higher order functions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1645–1656, 2009.

[2] N. Komodakis, G. Tziritas, and N. Paragios, "Fast, approximately optimal solutions for single and dynamic MRFs," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, 2007.

[3] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, 2001.

[4] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," in *Proc. European Conf. Computer Vision*, vol. 3, 2002, pp. 82–96.

[5] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake, "Digital tapestry," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 589–596.

[6] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother, "A comparative study of energy minimization methods for Markov random fields," in *Proc. European Conf. Computer Vision*, vol. 2, 2006, pp. 16–29.

[7] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *NIPS*, 2000, pp. 689–695.

[8] 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.*, vol. 26, no. 9, pp. 1124–1137, 2004.

[9] D. Freedman and P. Drineas, "Energy minimization via graph cuts: Settling what is possible," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 939–946.

[10] H. Ishikawa, "Exact optimization for Markov random fields with convex priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1333–1336, 2003.

[11] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 147–159, 2004.

[12] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1568–1583, 2006.

[13] M. J. Wainwright, T. Jaakkola, and A. S. Willsky, "MAP estimation via agreement on trees: message-passing and linear programming," *IEEE Trans. on Information Theory*, vol. 51, no. 11, pp. 3697–3717, 2005.

[14] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, 2004, pp. 261–268.

[15] O. Juan and Y. Boykov, "Active graph cuts," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 1023–1029.

[16] P. Kohli and P. H. S. Torr, "Effciently solving dynamic Markov random fields using graph cuts," in *Proc. Int'l Conf. Computer Vision*, vol. 2, 2005, pp. 922–929.

[17] N. Komodakis and G. Tziritas, "A new framework for approximate labeling via graph cuts," in *Proc. Int'l Conf. Computer Vision*, 2005, pp. 1018–1025.

[18] V. Kolmogorov and C. Rother, "Minimizing non-submodular functions with graph cuts: a review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, 2007.

[19] I. Kovtun, "Partial optimal labeling search for a NP-Hard subclass of (max,+) problems," in *DAGM Symposium*, 2003, pp. 402–409.

[20] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, pp. 155–225, 2002.

[21] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in *Proc. Int'l Conf. Computer Vision*, vol. 1, 2001, pp. 105–112.

[22] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proc. European Conf. Computer Vision*, vol. 1, 2006, pp. 1–15.

[23] R. Paget and I. D. Longstaff, "Texture synthesis via a noncausal nonparametric multiscale Markov random field," *IEEE Trans. Image Processing*, vol. 7, no. 6, pp. 925–931, 1998.

[24] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 860–867.

[25] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black, "Efficient belief propagation with learned higher-order Markov random fields," in *Proc. European Conf. Computer Vision*, vol. 2, 2006, pp. 269–282.

[26] P. Kohli, L. Ladicky, and P. H. S. Torr, "Graph cuts for minimizing robust higher order potentials," in *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

[27] S. Iwata, S. T. McCormick, and M. Shigeno, "A strongly polynomial cut canceling algorithm for the submodular flow problem," in *ICPO*, vol. LNCS 1610, 1999, pp. 259–272.

[28] B. A. Zalesky, "Efficient determination of Gibbs estimators with submodular energy functions, http://arxiv.org/abs/math/0304041v1," 2003.

[29] D. Schlesinger and B. Flach, "Transforming an arbitrary minsum problem into a binary one," Dresden University of Technology, Tech. Rep. TUD-FI06-01, April 2006.

[30] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1998.

[31] M. I. Schlesinger and V. Hlavac, *Ten Lectures on Statistical and Structural Pattern Recognition*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2002.

[32] T. Werner, "A linear programming approach to max-sum problem: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 7, pp. 1165–1179, 2007.

[33] N. Komodakis, N. Paragios, and G. Tziritas, "MRF optimization via dual decomposition: Message-passing revisited," in *Proc. Int'l Conf. Computer Vision*, 2007.

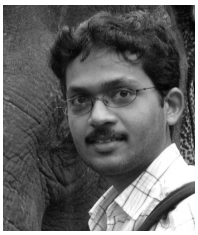
[34] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. H. S. Torr, "On partial optimality in multi-label mrfs," in *ICML*, 2008, pp. 480–487.

[35] I. Kovtun, "Image segmentation based on sufficient conditions for optimality in NP-complete classes of structural labeling problems," Ph.D. dissertation, IRTC ITS Nat. Academy of Science Ukraine, Kiev, 2004, in Ukranian.

[36] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithm," *Int'l J. Computer Vision*, vol. 47, pp. 7–42, 2002.

[37] F. Schroff, A. Criminisi, and A. Zisserman, "Single-histogram class models for image segmentation," in *ICVGIP*, 2006.

[38] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 4, pp. 401–406, 1998.



Karteek Alahari received the BTech (Honours) and the Masters degrees in computer science from IIIT, Hyderabad, in 2004 and 2005 respectively. He is currently a PhD student at Oxford Brookes University, Oxford, and is a member of the Brookes Vision Group and an associate member of the Visual Geometry Group at the University of Oxford. He is a student member of the IEEE.

Pushmeet Kohli is a researcher in the Machine Learning and Perception group at Microsoft Research Cambridge, and an associate of Trinity Hall, University of Cambridge. He completed his PhD studies at Oxford Brookes University under the supervision of Prof. Philip Torr. His PhD thesis, titled "Minimizing Dynamic and Higher Order Energy Functions using Graph Cuts", was the winner of the British Machine Vision Association's Sullivan Doctoral Thesis Award, and was a runner-up for the British Computer Society's Distinguished Dissertation Award. Before joining Microsoft Research Cambridge, Pushmeet was a visiting researcher at Microsoft Research Bangalore. He previously worked in the Foundation of Software Engineering Group at MSR Redmond. Pushmeet has worked on a number of problems in Computer Vision, Machine Learning and Discrete Optimization. His papers have appeared in SIGGRAPH, PAMI, IJCV, ICCV, CVPR, ICML and ECCV.

Philip H. S. Torr received the PhD (DPhil) degree from the Robotics Research Group of the University of Oxford under Professor David Murray of the Active Vision Group. He worked for another three years at Oxford as a research fellow and is currently a visiting fellow in engineering science at the University of Oxford, working closely with Professor Zisserman and the Visual Geometry Group. He left Oxford to work for six years as a research scientist for Microsoft Research, first in Redmond, Washington, in the Vision Technology Group, then in Cambridge, United Kingdom, founding the vision side of the Machine Learning and Perception Group. He is now a professor of computer vision and machine learning at Oxford Brookes University. He is a senior member of the IEEE. He is a Royal Society Wolfson Research Merit Award holder.