

# Reduce, Reuse & Recycle: Efficiently Solving Multi-Label MRFs

KartEEK Alahari<sup>1</sup> Pushmeet Kohli<sup>2</sup> Philip H. S. Torr<sup>1</sup>

<sup>1</sup>Oxford Brookes University\*

<sup>2</sup>Microsoft Research, Cambridge

## 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 CRFs or MRFs. 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  $\alpha$ -expansion) works by ‘recycling’ results from previous problem instances. The second method simplifies the energy function by ‘reducing’ the number of unknown variables, and can also be used to generate a good initialization for the dynamic  $\alpha$ -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  $\alpha$ -expansion, as well as other popular algorithms such as sequential tree-reweighted message passing, and max-product belief propagation. In most cases we achieve a 10-15 times speed-up in the computation time. Our modified  $\alpha$ -expansion algorithm provides similar performance to Fast-PD [15]. However, it is much simpler and can be made orders of magnitude faster by using the initialization schemes proposed in the paper. †*

## 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 [4, 19, 23]. 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 [23]. Although, minimizing a general MRF energy function is an NP-hard problem [13], 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 [5].

Similarly, certain submodular energy functions can be minimized by solving an st-mincut problem [3, 6, 7, 13].

Efficient algorithms have also been proposed for functions which do not fall under the above classes [4, 11, 24]. 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 [4, 23]. 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. The last few years have seen a lot of attention being devoted to increasing the performance of minimization algorithms.

We make two main 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 [15] on dynamic energy minimization. Our second contribution is a method which simplifies the energy function by reducing the number of variables. Further, it can also be used to speed-up the inference of the optimal values of the remaining variables.

**Recycling Solutions:** Our first method is inspired by the dynamic computation paradigm [8, 10, 15]. It improves the performance of the  $\alpha$ -expansion algorithm by reusing results from previous problem instances. The idea of dynamic computation has been used in the recent work of [8, 10] on minimizing submodular energy functions. In particular, [10] showed how flow can be reused in maxflow algorithms, and [8] showed how cuts (or previous labelling) can be reused. However, these methods are only applicable for the special case of dynamic MRFs<sup>1</sup> 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 work of Komodakis *et al.* [15] on the Fast-PD algorithm, which generalizes the work of [10]. Fast-PD works by solving the energy minimization problem by a series of graph cut computations. They show how this process can be 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 run-

\*<http://cms.brookes.ac.uk/research/visiongroup>

†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 PASCAL Network of Excellence, IST-2002-506778.

<sup>1</sup>MRFs that vary over time [8, 10].

ning time. Our modified dynamic  $\alpha$ -expansion algorithm is conceptually much simpler and easier to implement than Fast-PD whilst giving similar performance. Our method of initializing the  $\alpha$ -expansion algorithm can make both methods orders of magnitude faster.

**Simplifying energy functions using partially optimal solutions:** Most discrete MRFs encountered in computer vision problems are composed of *easy* and *difficult* components [12, 16]. 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. Energy minimization algorithms generally do not take advantage of this decomposition, and process all the random variables in 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 [1, 12, 16]. 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 [13]. 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. We also show how to achieve a substantial speed-up in the minimization of the simplified energy by initializing the corresponding dual variables based on the algorithm for establishing a partially optimal solution.

**Outline of the Paper:** In section 2, we provide the notation and definitions. Algorithms for approximate energy minimization [4, 12] and partially optimal solutions [1, 16] are briefly described in the same section. In section 3, we present methods to improve the running time of algorithms for minimizing multi-label energy functions. Section 4 compares the performance of our methods on the problems of colour and object based segmentation [2, 22], and stereo matching [23]. Summary and discussion are provided in section 5.

## 2. Preliminaries

The notation and basic definitions relevant to our work are provided here. Consider a set of random variables  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ , where each variable  $X_i \in \mathbf{X}$  takes a value from the label set  $\mathcal{L} = \{l_1, l_2, \dots, l_k\}$ . 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  $\mathbf{x}_{\text{map}} = \arg \min_{\mathbf{x} \in \mathcal{L}} E(\mathbf{x})$ .

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

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

where  $\mathcal{V}$  is the set of all random variables and  $\mathcal{E}$  is the set of all pairs of interacting variables. However, see [9] for potential functions with three or more 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  $X_j = x_j$ .

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

**Energy Reparameterization:** Energy functions  $E_1$  and  $E_2$  are called *reparameterizations* of each other if and only if  $\forall \mathbf{x}, E_1(\mathbf{x}) = E_2(\mathbf{x})$  [1, 11]. 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 [1]. Given an *ordering* over the label set  $\mathcal{L}$ , a function  $f(\cdot)$  is submodular if all its projections on two variables satisfy the constraint:

$$f^p(a, b) + f^p(a+1, b+1) \leq f^p(a, b+1) + f^p(a+1, b), \quad (2)$$

for all  $a, b \in \mathcal{L}$ . Kolmogorov and Zabih [13] 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 [7], Zalesky [25], Schlesinger and Flach [21] provided solutions for the multi-label case.

Most multi-label energy functions encountered in computer vision do not satisfy the constraint (2) and thus are not submodular. For instance, it can be clearly seen that the Potts model potential  $\psi$  defined as:

$$\psi_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j, \\ \gamma & \text{otherwise,} \end{cases} \quad (3)$$

does not satisfy the constraint (2). For example, choosing  $a = k$  and  $b = k + 1$  in (2) violates the constraint.

A number of algorithms have been proposed to efficiently find approximate or *partially optimal* solutions of

these energy functions [1, 4, 12, 16, 24]. 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.

**Move making algorithms:** The  $\alpha$ -expansion and  $\alpha\beta$ -swap algorithms are widely used for approximate energy minimization [4, 23]. 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  $\alpha$ -expansion move allows a random variable to either retain its current label or take a label  $\alpha$ . One iteration of the algorithm involves performing expansions for all  $\alpha \in \mathcal{L}$  in some order successively. An  $\alpha\beta$ -swap move allows a random variable whose current label is  $\alpha$  or  $\beta$  to either take a label  $\alpha$  or  $\beta$ . One iteration of the algorithm involves performing swap moves for all pairs of labels  $\alpha, \beta \in \mathcal{L}$  in some order successively.

**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 [18]. Other message passing algorithms have also been proposed. Tree-reweighted message passing (TRW), which is motivated from the problem of maximizing a lower bound on the energy [11, 24], and dual decomposition [14] are a few examples.

### 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. The assignment  $x_i = \epsilon$  implies that no label has been assigned to random variable  $X_i$ . For instance, the QPBO algorithm [1, 12] for minimizing energy functions of binary variables returns a partially labelled solution  $\mathbf{x}$  with the 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  $\mathbf{x}^* \in \{\arg \min_{\mathbf{x}} E(\mathbf{x})\}$ .

### 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  $\alpha$ -expansion algorithm. However, our methods are general and are applicable to all popular algorithms such as

$\alpha\beta$ -swap, BP and TRW-S (sequential TRW). Experimental results using all these algorithms are presented in the latter sections.

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  $\alpha$ -expansion algorithm. As discussed earlier, the  $\alpha$ -expansion algorithm works by making a series of changes to the current solution to decrease its energy. A set of changes is called a *move*. In one iteration of the algorithm, it makes moves with respect to each label ' $\alpha$ ' ( $\in \mathcal{L}$ ). It finds the optimal changes to be made (or move) by minimizing a binary energy function using the st-mincut algorithm. The binary energy function corresponding to a particular ' $\alpha$ ' move will be denoted by  $E^\alpha(\mathbf{x}^\alpha)$ . It is defined as:

$$E^\alpha(\mathbf{x}^\alpha) = \sum_{i \in \mathcal{V}} \phi_i^\alpha(x_i^\alpha) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha), \quad (4)$$

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

$$\phi_i^\alpha(x_i^\alpha) = \begin{cases} \phi_i(x_i = \alpha) & \text{if } x_i^\alpha = 0, \\ \phi_i(x_i = \bar{\alpha}) & \text{if } x_i^\alpha = 1, \end{cases} \quad (5)$$

where  $\bar{\alpha}$  is the current label assignment for  $X_i$ . The pairwise potentials are defined as:

$$\phi_{ij}^\alpha(x_i^\alpha, x_j^\alpha) = \begin{cases} 0 & \text{if } x_i^\alpha = 0, x_j^\alpha = 0, \\ \gamma(1 - \delta(x_i - x_j)) & \text{if } x_i^\alpha = 1, x_j^\alpha = 1, \\ \gamma & \text{otherwise,} \end{cases} \quad (6)$$

where  $\delta(x_i - x_j) = 1$ , if  $x_i = x_j$ , and 0 otherwise.

The above function is pairwise and *submodular*, if the energy is metric. 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  $x_i^\alpha$ , while the dual solution corresponds to the feasible flow solution of the maxflow problem.

**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  $G_i^1, i = 1, \dots, k$ , 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  $\alpha$ -expansion. In iterations  $u > 1$  of the algorithm, instead of creating a new graph  $G_i^u$  for a label expansion, we dynamically update [10] the corresponding graph  $G_i^{u-1}$  from the previous iteration. This 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 decreases 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. We overcome this issue 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. The recycle scheme for single MRFs is summarized as follows:

1. Construct graphs  $G_i^1, i = 1, \dots, k$ , 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 [10], 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  $\alpha$ -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  $G_i^1(t), i = 1, \dots, k$ , corresponding to the first iteration for frame  $t$  are

obtained by dynamically updating [10] 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  $\alpha$ -expansion provide similar speed-ups as the Fast-PD algorithm as shown in Section 4.1.

### 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. We also show how computations performed during this procedure can be used to efficiently initialize the dynamic  $\alpha$ -expansion algorithm described in the previous section.

As discussed earlier, there are many algorithms for obtaining partially optimal solutions of non-submodular energy functions. We chose to use the algorithm recently proposed by Kovtun [16] 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 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

$$E^m(\mathbf{x}^m) = \sum_{i \in \mathcal{V}} \phi_i^m(x_i^m) + \sum_{(i,j) \in \mathcal{E}} \phi_{ij}^m(x_i^m, x_j^m), \quad (7)$$

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

$$\phi_i^m(x_i^m) = \begin{cases} \phi_i(x_i = l_m) & \text{if } x_i^m = 0, \\ \phi_i(x_i = l_{\min}) & \text{if } x_i^m = 1, \end{cases} \quad (8)$$

where  $l_{\min} = \arg \min_{l \in \mathcal{L} - \{l_m\}} \phi_i(x_i = l)$ . For the case of Potts model, the pairwise potentials are defined as<sup>2</sup>:

$$\phi_{ij}^m(x_i^m, x_j^m) = \begin{cases} 0 & \text{if } x_i^m = 0, x_j^m = 0, \\ 0 & \text{if } x_i^m = 1, x_j^m = 1, \\ \gamma & \text{otherwise.} \end{cases} \quad (9)$$

$E^m(\mathbf{x}^m)$  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:

$$x_i = \begin{cases} l_m & \text{if } x_i^m = 0, \\ \epsilon & \text{otherwise.} \end{cases} \quad (10)$$

<sup>2</sup>Although the algorithm proposed in [16] only handles Potts model energies, it can be easily extended to general energy functions [17].



Figure 1. *Some of the images used in our experiments. 1-2 Colour-based segmentation problems with 4 labels each. 3-6 Stereo matching problems with 16, 20, 60, 60 labels respectively. 7-11 Object-based segmentation problems with 4, 5, 5, 7, 8 labels respectively.*

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  $E(\mathbf{x})$ .

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. 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 figure 3. 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  $\alpha$ -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 equations (4) and (7), it can be seen that the energy functions corresponding to the subproblems of the partial optimality and  $\alpha$ -expansion algorithms have the same form. Thus we can reuse the solutions of the partial optimality subproblems to make the computation of the  $\alpha$ -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  $\alpha$ -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 [10, 15]. Thus, by making the subproblems of the partial optimality and  $\alpha$ -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 equation (8). From equations (5) and (8) it can be seen that the unary potentials of the partial optimality and  $\alpha$ -expansion subproblems are identical if  $\bar{\alpha} = l_{\min}$ . This can be done by initializing the labelling for the  $\alpha$ -expansion algorithm by setting  $x_i = l_{\min}$ , where  $l_{\min} = \arg \min_{l \in \mathcal{L}} \phi(x_i = l)$ . The pairwise potentials may differ at most by the constant  $\gamma$  for the case  $x_i^\alpha = 1, x_j^\alpha = 1$  (cf. equations (6) and (9)). This change makes the two problems similar and as shown in the experimental results in Figure 4 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  $\alpha$ -expansion,
  - Initialize the labelling of each node  $i$  to  $l_{\min}$ , where  $l_{\min} = \arg \min_{l \in \mathcal{L}} \phi_i(x_i = l)$ .
  - Update the residual graphs from the  $k$  auxiliary problems to construct graphs for the first  $\alpha$ -expansion iteration. (Reuse)
  - Restart the maxflow algorithms to compute optimal moves, using flow recycling between expansion moves. (Recycle)

## 4. Experiments

We evaluated our methods on a variety of multi-label MRF problems such as stereo matching [4], colour [2] and object [22] 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 [2]. The unary potential functions  $\phi_i(x_i), i \in \mathcal{V}$  are defined using the RGB distributions  $\mathcal{H}_a, a = 1, \dots, l_k$ , of the  $k$  segments as follows:

$$\phi_i(x_i) = -\log p(x_i = a | \mathcal{H}_a), \quad (11)$$

The distributions  $\mathcal{H}_a$  are obtained using user specified constraints. The pairwise potentials encourage contiguous segments while preserving the image edges [2], and take the form of a Generalized Potts model defined as:

$$\phi_{ij}(x_i, x_j) = \begin{cases} \lambda_1 + \lambda_2 \exp\left(\frac{-g^2(i,j)}{2\sigma^2}\right) \frac{1}{\text{dist}(i,j)} & \text{if } x_i \neq x_j, \\ 0 & \text{if } x_i = x_j, \end{cases} \quad (12)$$

where  $\lambda_1, \lambda_2$  and  $\sigma$  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 use 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 [8, 10].

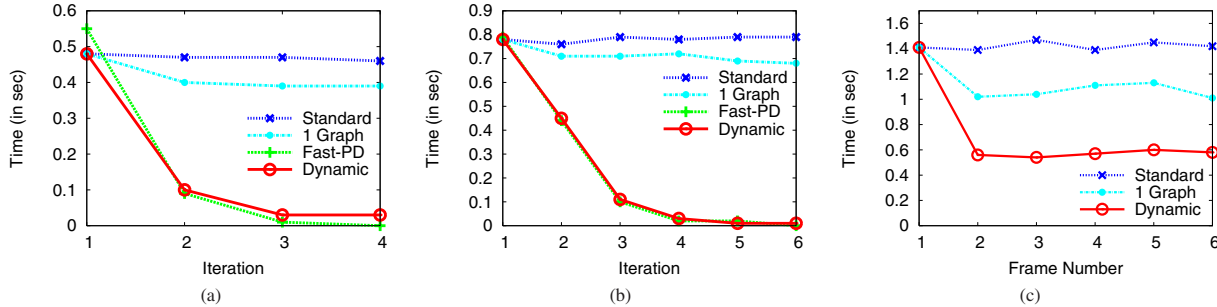


Figure 2. Reusing primal and dual solutions for (a), (b) single and (c) dynamic MRF problems: Comparison of run-times of standard and dynamic versions of  $\alpha$ -expansion, and Fast-PD are shown for (a) object-based segmentation problem: ‘Building’ image from the TextonBoost dataset [22], (b) stereo matching problem: Tsukuba (Left image), and (c) colour-based segmentation problem: cow video sequence [8, 10]. 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  $\alpha$ -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).

**Stereo Matching:** We used the standard energy function for stereo matching problem [16, 20]. The unary potentials of the energy are computed using a fixed size window-based method similar to the one used in [16]. The pairwise potentials take the form of a Potts model (3). Stereo matching results are shown on “Tsukuba”, “Venus”, “Cones”, “Teddy” images from the Middlebury stereo data set [20]. The Potts model smoothness cost  $\gamma$  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 (12). 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.

The following sections describe the results of primal and dual, and partially optimal solution initializations. Standard, publicly available implementations are used for comparison<sup>3</sup>. All experiments were performed on a Intel Core 2 Duo, 2.4 GHz, 3GB RAM machine. Source code for the proposed methods will be made available online.

#### 4.1. Dynamic $\alpha$ -expansion

We now discuss the effect of primal and dual solution initializations on the  $\alpha$ -expansion algorithm. Figures 2(a), 2(b) show the result of recycling the dual (flow) solution in a single MRF on two examples. 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. Sim-

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

Figure 3. Running times for various single MRF problems: Comparison of the run-times (in seconds) of the standard and optimized (opt) versions of  $\alpha$ -expansion ( $\alpha$ -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 numbers in ( ) denote the number of labels in each problem.

ilar 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 [8, 10]. These run-times for a sequence of 6 images are shown in Figure 2(c). The primal-dual initialized (dynamic) version provides a speed-up of 3-4 times. In the case of dynamic MRFs, we observed that using only primal or only dual initializations provides a very small improvement in computation time. The graphs also compare the dynamic methods with Fast-PD [15]. Note that our methods resulted in very similar run-times compared to Fast-PD.

We also tested a simple 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

<sup>3</sup>We thank the authors of TRW-S and Fast-PD for providing the original implementation of their methods for comparison.

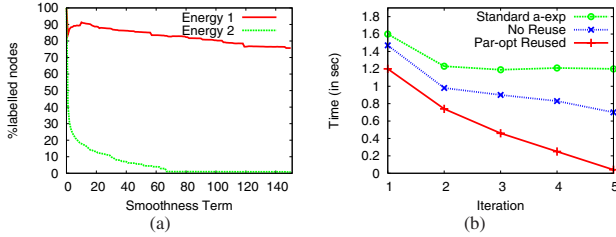


Figure 4. (a) The percentage of nodes labelled by the partially optimal solution algorithm by varying the smoothness cost for two energy functions. The Tsukuba stereo matching problem with energy functions given in [16] (Energy 1) and [23] (Energy 2) is used as the example here. For the smoothness cost  $\gamma = 20$ , only 13% of the nodes are labelled in the case of ‘Energy 2’. (b) The 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.

equation (4) it can be observed that the energy functions corresponding to two consecutive moves are substantially different. Hence, this scheme provides no significant speed-up. Figure 2 confirms this expected behaviour.

## 4.2. 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,  $\alpha$ -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 Figure 3. It is observed that the run-time 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.

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  $\gamma$  is very high, as shown in figure 4(a). As the smoothness cost increases, the percentage of labelled nodes decreases and the projected component of the energy function remains large. 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 figure 4(b)).

Segmentation and stereo matching results of some of the images used in our experiments are shown in Figure 7. Note that even when majority of the nodes are unlabelled in the partially optimal solution, e.g. Teddy sequence in Figure 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

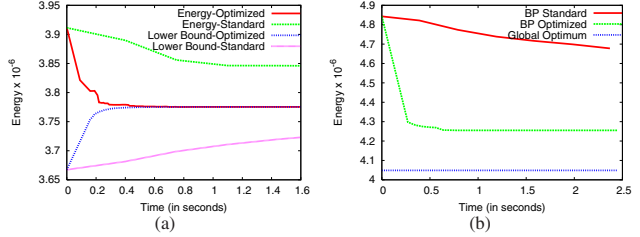


Figure 5. (a) Energy of the solution and lower bound obtained by running TRW-S algorithm on the Road image example [22]. Note that optimized TRW-S algorithm finds better energies (lower solution energy and higher lower bound) at any given point in time. It also finds an optima in only 0.64 seconds. Standard 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.

case of TRW-S and BP. Furthermore, the optimality of the solutions is not compromised. Figure 5(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. Figure 5(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. The solutions corresponding to these energies are shown in Figure 6.

## 5. 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 of the  $\alpha$ -expansion algorithm results in a dynamic algorithm. It can be seen as an extension of the work of [8, 10] for minimizing multi-label non-submodular energy functions. Experimental results show that our methods provide a substantial improvement in the performance of  $\alpha$ -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.

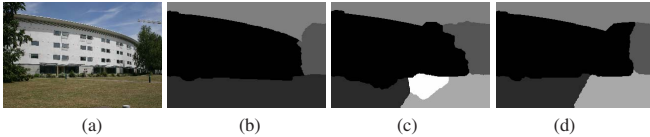


Figure 6. (a) Building image [22], and (b) the global optimum solution computed by the TRW-S algorithm. Solutions obtained using (c) standard BP, (d) and optimized BP with an 8-neighbourhood. Neither the optimized nor the standard versions converge to the optimal solution. However, optimized BP is closer to the optima.

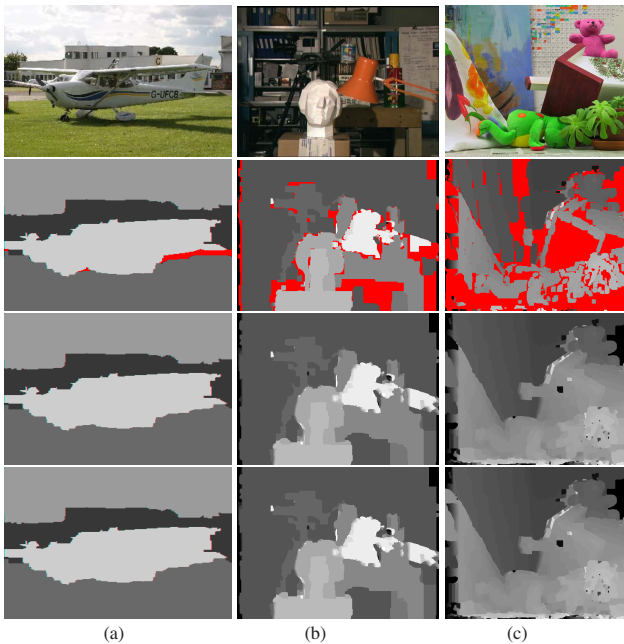


Figure 7. A sample result of object-based segmentation is shown in (a) Plane. Some of the stereo matching results are shown in (b) Tsukuba-Left and (c) Teddy-Left. The first row shows the original images. The second row shows the partially optimal solution. The regions marked in red denote the unlabelled pixels. The third and fourth rows show  $\alpha$ -expansion and TRW-S solutions on the projected energy function. 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.)

## References

- [1] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.
- [2] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, volume 1, pages 105–112, 2001.
- [3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.
- [4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.
- [5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. In *CVPR*, volume 1, pages 261–268, 2004.
- [6] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *CVPR*, volume 2, pages 939–946, 2005.
- [7] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *PAMI*, 25(10):1333–1336, 2003.
- [8] O. Juan and Y. Boykov. Active graph cuts. In *CVPR*, volume 1, pages 1023–1029, 2006.
- [9] P. Kohli, M. P. Kumar, and P. H. S. Torr.  $P^3$  & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.
- [10] P. Kohli and P. H. S. Torr. Efficiently solving dynamic Markov random fields using graph cuts. In *ICCV*, volume 2, pages 922–929, 2005.
- [11] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10):1568–1583, 2006.
- [12] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts: a review. *PAMI*, 2007.
- [13] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26:147–159, 2004.
- [14] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.
- [15] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic MRFs. In *CVPR*, 2007.
- [16] I. Kovtun. Partial optimal labeling search for a NP-Hard subclass of (max,+) problems. In *DAGM Symposium*, pages 402–409, 2003.
- [17] I. Kovtun. *Image segmentation based on sufficient conditions for optimality in NP-complete classes of structural labeling problems*. PhD thesis, IRTC ITS Nat. Academy of Science Ukraine, Kiev, 2004. In Ukrainian.
- [18] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1998.
- [19] C. Rother, S. Kumar, V. Kolmogorov, and A. Blake. Digital tapestry. In *CVPR*, volume 1, pages 589–596, 2005.
- [20] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithm. *IJCV*, 47:7–42, 2002.
- [21] D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology, April 2006.
- [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 *ECCV*, volume 1, pages 1–15, 2006.
- [23] 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 *ECCV*, volume 2, 2006.
- [24] 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*, 51(11):3697–3717, 2005.
- [25] B. A. Zalesky. Efficient determination of Gibbs estimators with submodular energy functions, <http://arxiv.org/abs/math/0304041v1>, 2003.