MRF Labeling with a Graph-Shifts Algorithm

Jason J. Corso[1], Zhuowen Tu[2], and Alan Yuille[3]

[1] Computer Science and Engineering, University at Buffalo, State University of New York
[2] Center for Computational Biology, Laboratory of Neuro Imaging
University of California, Los Angeles
[3] Department of Statistics, University of California, Los Angeles
jcorso@cse.buffalo.edu

Abstract. We present an adaptation of the recently proposed graph-shifts algorithm for labeling MRF problems from low-level vision. Graph-shifts is an energy minimization algorithm that does labeling by dynamically manipulating, or shifting, the parent-child relationships in a hierarchical decomposition of the image. Graph-shifts was originally proposed for labeling using relatively small label sets (e.g., 9) for problems in high-level vision. In the low-level vision problems we consider, there are much larger label sets (e.g., 256). However, the original graph-shifts algorithm does not scale well with the number of labels; for example, the memory requirement is quadratic in the number of labels. We propose four improvements to the graph-shifts representation and algorithm that make it suitable for doing labeling on these large label sets. We implement and test the algorithm on two low-level vision problems: image restoration and stereo. Our results demonstrate the potential for such a hierarchical energy minimization algorithm on low-level vision problems with large label sets.

1 Introduction

Markov random field (MRF) models [2] play a key role in both low- and high-level vision problems [13]. Example low-level vision problems are image restoration and stereo disparity calculation. Fast and accurate labeling of MRF models remains a fundamental problem in Bayesian vision. The configuration space is combinatorial in the labels and the energy landscape is rife with local minima. This point is underscored by the recent comparative survey of methods for low-level labeling by Szeliski et al. [15].

In recent years, multiple new algorithms have been proposed for solving the energy minimization problem associated with MRF labeling. For example, graph cuts [3] is one such algorithm that guarantees achieving a *strong* local minimum for two-class energy functions. However, processing times for the graph cuts remain in the order of several minutes on modern hardware. Max Product Belief propagation [8] computes local maxima of the posterior, but it is not guaranteed to converge for the loopy graphs present in low-level vision. Efficient implementations can lead to running times in the order of seconds [6]. However, despite its high-regard and widespread use, it performed poorly in the recent benchmark study [15].

In this paper, we work with a recently proposed approach called graph-shifts [4, 5]. Graph-shifts is a class of algorithms that do energy minimization on dynamic, adaptive graph hierarchies. The graph hierarchy represents an adaptive decomposition of the input image; they are adaptive in the sense that the graph hierarchy and neighborhood structure is data-dependent in contrast to conventional pyramidal schemes [1] in which the hierarchical neighborhood structure is fixed. They are dynamic in the sense that the algorithm iteratively reduces the energy by changing the parent-child relationships, or *shifting*, in the graph, which results in a change in the underlying labels at the pixels. Graph-shifts stores a representation of the combinatorial energy landscape, and is able to efficiently compute the optimal energy reducing move at each iteration.

The original graph-shifts algorithm [4, 5] was defined on a conditional random field (CRF) [11, 12] with comparatively few labels (e.g., 8) and applied to high-level labeling problems in medical imaging. Recall that a CRF is a MRF with a broader conditioning on the observed data than is typical in MRF and MAP-MRF [9] formulations. But, in the low-level labeling problems considered in this paper, the label sets are much larger (e.g. 32, 256). The original graph-shifts algorithms scales linearly in pixels; however a factor linear in labels is incurred at each iteration. The memory requirement is quadratic in the labels. In practice, these complications lead to slower convergence as the number of labels grow. The main focus and contribution of this paper is how we adapt graph-shifts to work efficiently with large, ordered label sets (e.g., 256). This requires four improvements on the original graph-shifts algorithm: 1) an improved way of caching the binary energy terms, 2) efficient sorting of the potential shift list and 3) an improved spawn shift, and 4) new, efficient rules for keeping the hierarchy in synch with the energy landscape after shifting. We demonstrate this algorithm on standard benchmark data in image restoration and stereo calculation.

We consider labeling problems of the following form. Define a pixel lattice Λ with pixels $i \in \Lambda$, and associate a label (random variable) y_i with each pixel. The labels take values in a label set $\mathcal{L} = \{1, \ldots, k\}$ and represent various problem specific physical quantities we want to estimate, like intensities, disparities, etc. The joint probability over the labels on the lattice, $y \doteq \{y_i\}_{i \in \Lambda}$, is given by a Gibbs distribution:

$$P(y) = \frac{1}{Z} \exp \left[- \sum_i U(y_i, i) - \sum_{\langle i,j \rangle} V(y_i, y_j) \right] \tag{1}$$

where Z is the partition function, and $\langle i, j \rangle$ represents the set of neighbors on Λ.

Equation 1 is the standard MRF. The clique potential functions, U and V, encode the local energy associated with various labelings at pixel sites. U is conditioned on the pixel location to permit the incorporation of some external data in the one-clique potential computation; for example, the input intensity field in image restoration (see section 5.1). The goal is to find a labeling y that maximizes $P(y)$, or equivalently minimizes the energy given by the sum over all clique potentials U and V. To simplify notation, we will consider problem as energy function minimization in the remainder of the paper.

The remainder of the paper is as follows: a short literature survey is in the next section. In section 3 we review the graph-shifts approach to energy minimization. Then, in section 4, we present our adaptations that make it possible to apply the algorithm on large label sets. In section 5 we analyze the proposed algorithm and give comparative results to efficient belief propagation [6] for the image restoration problem.

2 Related Work

Many algorithms have been proposed to solve the energy minimization problem associated with labeling the MRFs. The iterated conditional modes [2] is an early algorithm developed near the beginning of MRF research. It iteratively updates the pixel labels in a greedy fashion choosing the new labeling that gives the steepest decrease in the energy function. This algorithm converges slowly since it flips a single label at a time, and is very sensitive to local minima. Simulated annealing [9], on the other hand, is a stochastic global optimizer that given a *slow enough* cooling rate will always converge to the global minimum. However, in practice, the *slow enough* is a burden.

Some more recent algorithms are able to consistently approach the global minimum in compute times in the order of minutes. Graph cuts [3, 10] can guarantee a so-called *strong* local minimum for a defined class of (metric or semi-metric) energy functions. Max-product loopy belief propagation (BP) [8] computes a low energy solution by passing messages (effectively, max of conditional distributions) between neighbors in a graph. When (if) convergence is reached, BP will have computed a local max to the label posterior at each node. Although not guaranteed to converge for loopy graphs, it has performed well in a variety of low-level vision tasks [7], and can be effectively implemented for low-level vision tasks to run in just a few seconds [6]. Tree reweighted belief propagation (TRW) [16] is a similar message-passing algorithm that has the goal of computing the upper bound on the log partition function of any undirected graph. Although the recent comparative analysis [15] did not find a single best method, a modified version of the TRW approach did consistently outperform the other methods.

3 Graph-Shifts

Following the notation from [4], define a graph G to be a set of nodes $\mu \in \mathcal{U}$ and a set of edges. The graph is hierarchical and composed of multiple layers with the nodes at the lowest layer representing the image pixels. Call two connected nodes on the same layer neighbors using the predicate $N(\mu, \nu) = 1$ and $N(\mu, \nu) = 0$ otherwise. Two connected nodes on different (adjacent) layers are called parent-child nodes. Each node has a single parent (except for the nodes at the top layer, which have no parent) and has the same label as its parent. Every node has at least one child (except for the nodes at the bottom layer). Let $C(\mu)$ be the set of children of node μ and $A(\mu)$ be the parent of node μ. A node μ on the bottom layer (i.e. on the lattice) has no children, and hence $C(\mu) = \emptyset$. At the top of the graph is a special *root* layer with a single node $\overline{\mu}$ for each of the k labels. The label of the root nodes is fixed to a single value. Since all non-root nodes in the hierarchy can trace their ancestry back to a single root node, an instance of the graph G is equivalent to a labeling of the image.

The coarser layers are computed recursively by an iterative bottom-up coarsening procedure. We use the coarsening method defined in [4] without modification. The basic idea is that edges in the graph are randomly turned on or off based on the local intensity similarity. The *on* edges induce a connected components clustering, and each component defines a new node in the next coarse layer in the hierarchy. Thus, nodes at coarser layers in the hierarchy represent (roughly) homogeneous regions in the images. The procedure is adaptive and the resulting hierarchy is data-dependent. This is in contrast to traditional pyramidal schemes [1] which fix the coarse level nodes independent of the data. A manually tuned *reduction* parameter governs the amount of coarsening that happens at each layer in the bottom up procedure. In section 5, we give advice based on empirical experimentation on how to choose this parameter.

3.1 Energy in the Hierarchy

The original energy function (1) is defined at the pixel level only. Now, we extend this definition to propagate the energies up the hierarchy by recursing on the potentials:

$$\hat{U}(y_\mu, \mu) = \begin{cases} U(y_\mu, \mu) & \text{if } C(\mu) = \emptyset \\ \displaystyle\sum_{\nu \in C(\mu)} \hat{U}(y_\mu, \mu) & \text{otherwise} \end{cases} \tag{2}$$

$$\hat{V}(y_{\mu_1}, y_{\mu_2}) = \begin{cases} V(y_{\mu_1}, y_{\mu_2}) & \text{if } C(\mu_1) = C(\mu_2) = \emptyset \\ \displaystyle\sum_{\substack{\nu_1 \in C(\mu_1), \\ \nu_2 \in C(\mu_2) : \\ N(\nu_1, \nu_2) = 1}} \hat{V}(y_{\nu_1}, y_{\nu_2}) & \text{otherwise} \end{cases} \tag{3}$$

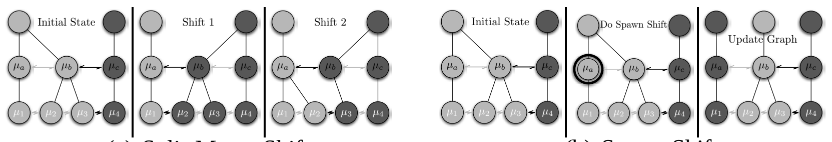
By defining the recursive energies in this form, we are able to explore the full label set $\mathcal{L}$ at each layer in the hierarchy rather than work on a reduced label set at each layer, which is typical of pyramidal coarse-to-fine approaches. By operating with the complete label set in the whole hierarchy, graph-shifts is able to quickly switch between scales at each iteration when selecting the next steepest shift to take (further discussion in section 5).

By using (2) and (3), we can compute the exact energy caused by any node in the hierarchy. Furthermore, the complete energy (1) can be rewritten in terms of the roots:

$$E(y) \doteq \sum_{i \in \mathcal{L}} \hat{U}(y_{\overline{\mu}_i}, \overline{\mu}_i) + \sum_{\substack{i,j \in \mathcal{L} \\ N(\overline{\mu}_i, \overline{\mu}_j) = 1}} \hat{V}(y_{\overline{\mu}_i}, y_{\overline{\mu}_j}) \tag{4}$$

3.2 The Graph Shift and Minimizing The Energy

A *graph shift* is defined as an operation that changes the label of a node by dynamically manipulating the connectivity structure of the graph hierarchy. There are two types of graph shifts: a split-merge shift and a spawn shift. During a split-merge shift (figure 1(a)), a node μ detaches itself from its current parent $A(\mu)$ and takes the parent of a neighbor $A(\nu)$. By construction, the shift also relabels the entire sub-tree rooted at μ such that $y_\mu = y_\nu$. During a spawn shift (figure 1(b)), a node μ creates (or spawns) a new top-level root node $\overline{\mu}$ and dynamically creates a chain connecting μ to $\overline{\mu}$ with one new node per layer. The new tree is assigned whatever label (one that none of μ's neighbors already had) was associated with the spawn shift. After making either shift, the hierarchy must be resynchronized with the changed energy landscape (section 4.3).



(a) Split-Merge Shift (b) Spawn Shift

Fig. 1. Toy examples of the split-merge and spawn shifts with two classes, light and dark gray.

The basic idea behind the graph-shifts algorithms is to select shift that would most reduce the energy at each iteration. Using (2) and (3), the exact energy gradient, or the *shift gradient*, can be computed as

$$\Delta E(\mu, y_\mu \to \hat{y}_\mu) = \hat{U}(\hat{y}_\mu, \mu) - \hat{U}(y_\mu, \mu) \;+ \sum_{\nu:N(\mu,\nu)=1} \Big[\hat{V}(\hat{y}_\mu, y_\nu) - \hat{V}(y_\mu, y_\nu)\Big] \;. \tag{5}$$

This directly leads to the graph-shifts algorithm. After initialization the graph hierarchy, iterate the following steps until convergence:

1. Compute and select the graph shift that most reduces the energy.
2. Apply this shift to the graph.
3. Update the graph hierarchy to resynchronize with the new energy landscape.

4 Adapting Graph-Shifts for Large Label Sets

This section describes the adaptations we make to the original graph-shifts algorithms to increase its efficiency when dealing with large label sets. The first two adaptations (section 4.1) consider how the shifts are stored in various caches up the hierarchy. The third one considers a reduced spawning label set (section 4.2). The fourth one, in section 4.3, discusses how to update the hierarchy and potential shift list after executing a shift.

4.1 Computing and Selecting Shifts

Here, we discuss two representational details that reduce the amount of computation required when computing potential shifts. First, though the energy recursion formulas (2) and (3) provide a mathematically convenient way of computing the energy at a given node, repeatedly recursing down the entire tree to the leaves to compute the energy at a node is often redundant. So, an *energy cache* is stored at each node in the hierarchy. The cache directly stores the unary energy $\hat{U}(y_\mu, \mu)$ for a node μ in a vector of k dimension. The unary cache can be efficiently evaluated in a bottom-up fashion at the leaves first and them *pushing* them up the hierarchy. The memory cost is $O(kn \log n)$ for n pixels.

[4] suggests such a caching scheme for both the unary and the binary terms of the energy. However, storing a similar complete cache for the binary term is not plausible for large label sets because its cost is quadratic in the labels, $O(k^2 cn \log n)$ with c

being the average cardinality of the nodes. However, recall the binary term is a function of only the labels at the two nodes, and the sub-trees have the same labels as their parents. So, for the binary energy, the recursion formulas serve to count the length of the boundary between two nodes, which is then multiplied by the cost on the two labels:

$$\hat{V}(y_{\mu_1}, y_{\mu_2}) = V(y_{\mu_1}, y_{\mu_2})B(\mu_1, \mu_2) \tag{6}$$

$$B(\mu_1, \mu_2) = \begin{cases} N(\mu_1, \mu_2) & \text{if } C(\mu_1) = C(\mu_2) = \emptyset \\ \displaystyle\sum_{\substack{\nu_1 \in C(\mu_1) \\ \nu_2 \in C(\mu_2)}} B(\nu_1, \nu_2) & \text{otherwise.} \end{cases} \tag{7}$$

This form of the binary energy suggests caching the boundary length B between nodes in the graph. By doing so, we save on the k^2 factor for the cache and the resulting memory cost is $O(cn \log n)$. We discuss how to update this cache after a shift in 4.3.

Second, a complete list of potential shifts is maintained by the algorithm. After initializing the hierarchy, the full list is computed, and all those shifts with a negative gradient (5) are stored. To further reduce the list size, only a single shift is stored for any given node. The list is updated after each shift (section 4.3). Computing the best shift at each iteration results searching this list. [4] choose to store an unsorted list to save the $O(s \log s)$, for s shifts, cost of initially sorting the list at the expense of $O(s)$ search every iteration. However, an $O(s)$ is already paid during the initial computation. Hence, the "sorting cost" is only an additional $O(\log s)$ cost if we sort while computing the list. Searching every iterations is then only $O(1)$. Thus, we choose to sort the list.

4.2 An Improved Spawn Shift

The original spawn shift [5] requires the evaluation of a shift gradient when switching to potentially any new label. The cost of this evaluation grows with the number of labels, $O(k)$, but the cost of computing the best split-merge shift for a node is $O(c)$ using the caches. In the problems we consider $k \gg c$. We exploit the label ordering and search only a sub-range of the labels based on the current label. For node μ with label y_{μ}, we search the range $\{y_\mu - \kappa, y_\mu + \kappa\}$ where κ is a user selected parameter (we use 3).

4.3 Updating the Hierarchy After a Shift

A factor of crucial importance to the graph-shifts algorithm is dynamically keeping the hierarchy in synch with the energy landscape it represents. Since a shift is a very local change to the labeling, updating the hierarchy can be done quickly. After each shift, the following steps must be performed to ensure synchrony. Assume the shift occurs at level l, pixels correspond to level 0 and there are T levels in the hierarchy.

1. Update the unary caches at levels $t = \{l + 1, \ldots, T\}$.
2. Update the boundary length caches at levels $t = \{l + 1, \ldots, T\}$.
3. Recompute the shift for all affected nodes and update their entries in the potential shift list (removing them if necessary). Affected nodes will be present on all graph

levels. For nodes below l, any node in the subtree or an immediate neighbor of the subtree of the shifted node must be updated. For nodes above only the parents and their neighbors must be updated. This an $O(c \log n)$ number of updates.

Updating the unary caches is straightforward. For a split-merge shift where node μ shifts to ν and takes $A(\nu)$ as a new parent, the update equations are

$$\hat{U}\left(y, A(\mu)\right)' = \hat{U}\left(y, A(\mu)\right) - \hat{U}\left(y, \mu\right) \qquad \forall y \in \mathcal{L} \tag{8}$$

$$\hat{U}\left(y, A(\nu)\right)' = \hat{U}\left(y, A(\nu)\right) + \hat{U}\left(y, \mu\right) \qquad \forall y \in \mathcal{L} \ . \tag{9}$$

Consider a spawn shift where μ spawns a new sub-tree to the root level. Equation (8) applies to the old parent $A(\mu)$. The new parent $A^*(\mu)$ is updated by

$$\hat{U}\left(y, A^*(\mu)\right) = \hat{U}\left(y, \mu\right) \qquad \forall y \in \mathcal{L} \ . \tag{10}$$

Each of these equations must be applied recursively to the root level T in the hierarchy.

Since the boundary length terms involve two nodes, they result in more complicated update equations. For a shift from node μ to ν the update equations for level $l+1$ are

$$B\left(A(\mu), A(\nu)\right)' = B\left(A(\mu), A(\nu)\right) - \sum_{\substack{\eta:\ A(\eta)=A(\nu),\\ N(\mu,\eta)=1}} B(\mu, \eta) \tag{11}$$

$$\begin{aligned} B\left(A(\nu), A(\omega)\right)' &= B\left(A(\nu), A(\omega)\right) + B(\mu, \omega) \\ B\left(A(\mu), A(\omega)\right)' &= B\left(A(\mu), A(\omega)\right) - B(\mu, \omega) \\ &\forall \omega\colon\ A(\omega) \neq A(\nu), N(\mu, \omega) = 1 \end{aligned} \tag{12}$$

where $A(\mu)$ is the ancestor of μ before the shift takes place. The second term on the righthand side of (11) arises because μ can be a neighbor to more than one child of $A(\nu)$. When, μ shifts to become a child of $A(\nu)$, then it will a become sibling of such a node and the boundary length from $A(\mu)$ to $A(\nu)$ must account for it. Again, the updates must be applied recursively to the top of the hierarchy. These equations are also applicable in the case of a spawn shift with the additional knowledge, that if $B(A(\nu), \cdot)$ is 0, then a new edge must be created in the graph connecting these two nodes.

5 Experiments

We consider two low-level labeling problems in this section: image restoration and stereo. We also present a number of evaluative results on the efficiency and robustness of the graph-shifts algorithm for low-level MRF labeling. In all of the results, unless otherwise stated, a truncated linear binary potential function was used. It is defined on two labels and is fixed by two parameters, β_1, β_2:

$$V(y_i, y_j) = \min(\beta_1 ||y_i - y_j||, \beta_2) \ . \tag{13}$$

5.1 Image Restoration

Image restoration is the problem of removing the noise and other artifacts of an acquired image to restore it back to its original, or ideal state. The label set comprises the 256 gray-levels. To analyze this problem, we work with the well-known penguin image; it's ideal image is given on the right in figure 2. In figure 3, we present some restoration results for various possible potential functions on images that have been perturbed by independent Gaussian noise of three increasing variances (in each row). In the following potentials, let x_i be the inputted intensity in the corrupted image. The second column shows a Potts model on both the unary and binary potential functions. The third column shows a truncated linear unary potential and a Potts binary potential with the fourth column showing both truncated linear potentials. Truncated quadratic results are shown in figure 4 in comparison with EBP. In these results, we can see that the graph-shifts algorithm is able to find a good minimum to the energy function, and it's clear that the stronger potential functions are giving better results. In the truncated linear terms here, $\alpha_1 = \beta_1 = 1$, $\alpha_2 = 100$, and $\beta_2 = 20$. The two terms were equally weighted.

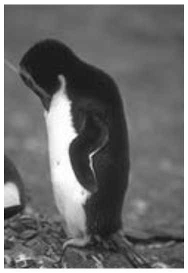


Fig. 2. The ideal image.

$$\text{Potts} \qquad U_P(y_i, i) = \delta(x_i, y_i) \tag{14}$$
$$\text{Truncated Linear} \qquad U_L(y_i, i) = \min(\alpha_1 ||x_i - y_i||, \alpha_2) \tag{15}$$
$$\text{Truncated Quadratic} \qquad U_Q(y_i, i) = \min(\alpha_1 ||x_i - y_i||^2, \alpha_2) \tag{16}$$

Figure 4 and table 1 present a comparison of the image restoration graph-shifts with the efficient belief propagation (EBP) [6] algorithm. Here, we use a truncated quadratic energy function (16) with exactly the same energy function and parameters: $\alpha_1 = \beta_1 = 1$, $\alpha_2 = 100$ and $\beta_2 = 20$. In these restoration results, we use the sum of squared differences error between the original (ideal) image and the restored image that is outputted by each algorithm to measure the accuracy. Note, however, that the energy minimization algorithms are not directly minimizing the sum-of-squared differences error function. When computing the SSD error on these images we disregarded the outside row and column since the EBP implementation cannot do labeling near the image borders.

Table 1. Quantitative comparison of time and SSD error between efficient belief propagation and graph-shifts. Speed is not directly comparable as BP is in C++ but graph-shifts is in Java.

	Time (ms)		SSD Error	
Variance	Graph-Shifts	Efficient BP	Graph-Shifts	Efficient BP
10	18942	2497	2088447	2524957
20	24031	2511	2855951	3105418
30	24067	2531	5927968	3026555

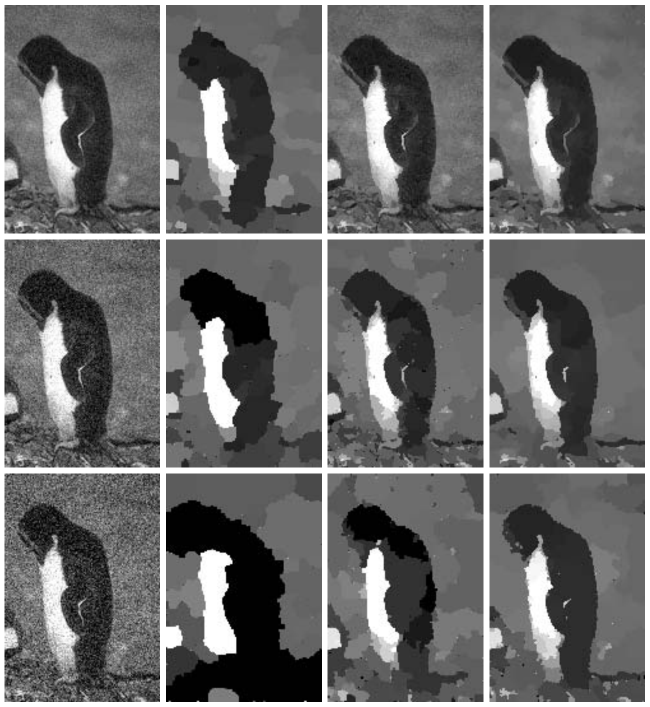


Fig. 3. Visual comparison of the performance of different energy functions with the same graph-shifts parameters. The images on the left are the input noisy images (with variances of 10, 20, and 30 in the rows). The remaining columns are Potts+Potts,Truncated Linear+Potts,Truncated Linear+Truncated Linear in the unary + binary terms, respectively.

From inspecting the scores in table 1, we find the two algorithms are both able to find good minima for the two inputs with smaller noise. The graph-shifts minimum achieves lower SSD error scores for these two images. This could be due to the extra high-frequency information near the bottom of the image that it was able to retain, but the EBP implementation smoothed it out. However, for the higher noise variance, EBP is able to converge to a similar minimum and its SSD error is much low than graph-shifts in this case. We also see that the two algorithms run in the order of seconds (they are run on the same hardware). However, the speed comparison is not completely fair: EBP is implemented in C++ while graph-shifts is implemented in Java (one expects at least a factor of two speedup). We note that some of the optimizations suggested in the EBP algorithm [6] can also apply in the computation and evaluation of the graph shifts to further increase efficiency. The clear message from the time scores is that the graph-shifts approach is of the same order of magnitude as current state of the art inference algorithms (seconds).

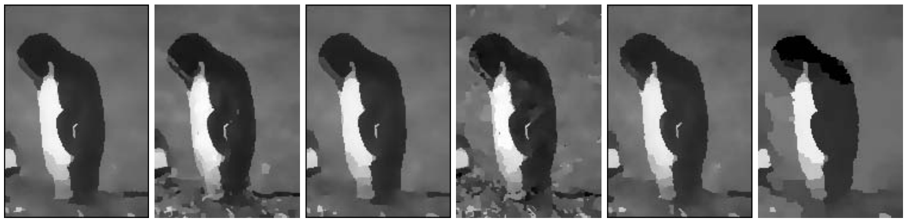


Fig. 4. Visual comparison between the proposed graph-shifts algorithms and the efficient belief propagation [6]. The two pairs of images have noise with variance 10, 20, and 30. Images in each pair are the EBP restoration followed by the graph-shifts restoration. See Fig. 3 for input images.

5.2 Stereo

We present results on two images from the Middlebury Stereo Benchmark [14]: the sawtooth image has 32 disparities with 8 levels of subpixel accuracy or a total of 256 labels, and the Tsukuba image has 16 labels. The energy functions we use here to model the stereo problem remain very simple MRFs. The unary potential is a truncated linear function on the pixel-wise difference between the left image $I_L(u, v)$ and the right image $I_R(u - y_i, v)$, where $i = (u, v)$:

$$U_S(y_i, i) = \min(\alpha_1 ||I_L(u, v) - I_R(u - y_i, v)||, \alpha_2) \ . \tag{17}$$

Figure 5 shows the two results. The parameters are $\alpha_1 = \beta_1 = 1$, $\alpha_2 = 100$ and $\beta_2 = 20$ for the tsukuba image and $\alpha_1 = \beta_1 = 1$, $\alpha_2 = \beta_2 = 10$. The inferred disparity, on the right, is very close the ground truth nearly everywhere in the image. These results are in the range of the other related algorithms [3, 6]. However, graph-shifts can compute them in only seconds. Even without a specific edge/boundary model, which many methods in the benchmark use, the graph-shifts minimizer is able to maintain good boundaries. For lack of space, we cannot discuss the stereo results in more detail.

5.3 Evaluation

We use the truncated linear unary and binary potentials on the penguin image (for restoration) with a variance of 20 for the noise in all results in this section unless otherwise noted. The parameters on the potentials are $(1, 100)$ and $(1, 20)$ for unary and binary respectively.

Figure 5.3-left shows how the time to converge varies with changing the reduction criterion. As the graph reduction factor increases, we see an improvement in both the time to converge and the SSD error. Recall the graph reduction factor is related inversely to the amount of coarsening that occurs in each layer. So, with a larger factor, the taller the hierarchy we find and the stronger the homogeneity properties in each graph node. Thus, the shifts that are taken by the graph-shifts with larger graph reduction are more *targetted* and result in fewer total necessary shifts. Figure 5.3-right demonstrates robustness to variation in the parameters of the energy functions. As we vary the truncation

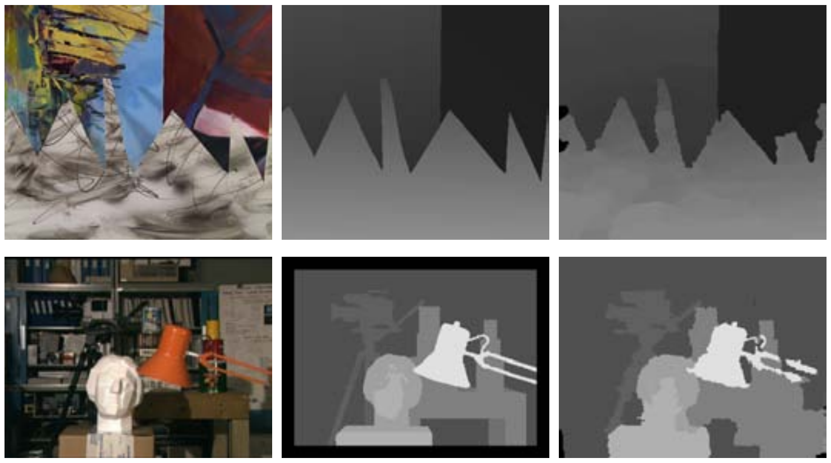


Fig. 5. Results on computing stereo with the graph-shifts algorithm on an MRF with truncated linear potentials. Left column is the left image of the stereo pair, middle column is the ground truth disparity, and the right column is the inferred disparity.

parameter on the unary potential the time to converge stays roughly constant, but the SSD error changes (as expected).

Figure 7 shows four different graphs that explore the actual graph-shift process. Each graph shows the shift number on the horizontal access and the vertical axis shows one of the following measurements (left-to-right) the level at which each shift occurs, the mass of the shift (number of pixels whose label changes), the shift gradient (5) and the SSD error of the current labeling. We show the SSD error to demonstrate that it is being reduce by minimizing the MRF; the SSD error is not the objective function directly being minimized. The upper-left plot highlights a significant difference in the energy minimization procedure created by the graph-shifts algorithm and the traditional multi-level coarse-to-fine approach. We see that as the algorithm proceeds it is greatly varies in which level to select the current shift; recall that graph-shifts will select the shift (at any level) with the current steepest negative shift gradient. This *up-and-down* action contrasts the coarse-to-fine approach which would complete all shifts at the top level and the proceeds down until the pixel level.

6 Conclusion

In this paper we present an adaptation of the recently proposed graph-shifts algorithm to the case of MRF labeling with large label sets. Graph-shifts does energy minimization by dynamically changing the parent-child relationships in a hierarchical decomposition of the image, which encodes the underlying pixel labeling. Graph-shifts is able to efficiently compute the optimal shift at every iteration. However, this efficiency comes from keeping the graph in synch with the underlying energy. The large label sets make

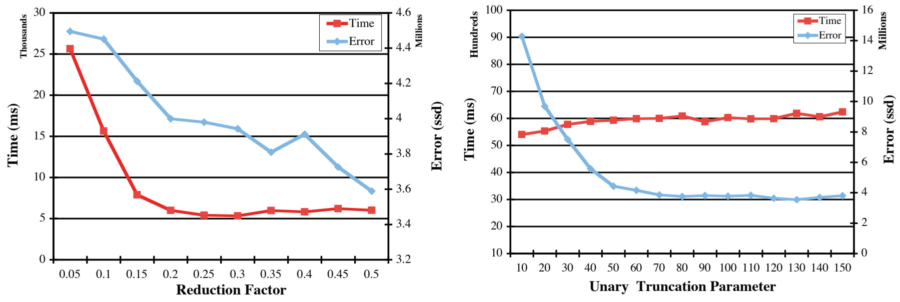


Fig. 6. Evaluation plots. (left) How does the convergence time vary with the height of the hierarchy (reduction parameter)? (right) How robust is the convergence speed when varying parameters of the potential functions?

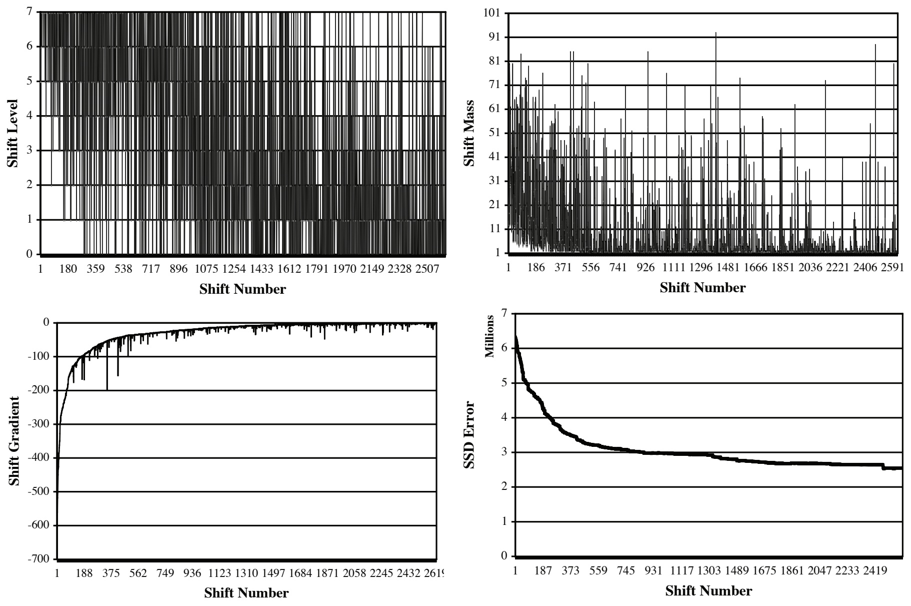


Fig. 7. These four graphs show statistics about the graph-shift process during one of the image restoration runs. See text for full explanation.

ensuring this synchrony difficult. We made four suggestions for adapting the original graph-shifts algorithm to maintain its computational efficiency and quick run-times (order of seconds) for MRF labeling with large label sets. The results on image restoration and stereo are an indication of the potential in such a hierarchical energy minimization algorithm. The results also indicate that the quality of the minimization depends on the properties of the hierarchy, like height and homogeneity of nodes. In future work, we plan to develop a methodology to systematically optimize these for a given problem.

References

1. Anandan, P.: A Computational Framework and an Algorithm for the Measurement of Visual Motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.
2. Besag, J.: Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Royal Stat. Soc., B*, 36:192–236, 1974.
3. Boykov, Y., Veksler, O., and Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.
4. Corso, J. J., Tu, Z., Yuille, A., and Toga, A. W.: Segmentation of Sub-Cortical Structures by the Graph-Shifts Algorithm. In N. Karssemeijer and B. Lelieveldt, editors, *Proceedings of Information Processing in Medical Imaging*, pages 183–197, 2007.
5. Corso, J. J., Yuille, A. L., Sicotte, N. L., and Toga, A. W.: Detection and Segmentation of Pathological Structures by the Extended Graph-Shifts Algorithm. In *Proceedings of Medical Image Computing and Computer Aided Intervention (MICCAI)*, volume 1, pages 985–994, 2007.
6. Felzenszwalb, P. F. and Huttenlocher, D. P.: Efficient Belief Propagation for Early Vision. *International Conference on Computer Vision*, 70(1), 2006.
7. Freeman, W. T. , Pasztor, E. C., and Carmichael, O. T.: Learning Low-Level Vision. *International Journal of Computer Vision*, 2000.
8. Frey, B. J. and MacKay, D.: A Revolution: Belief Propagation in Graphs with Cycles. In *Proceedings of Neural Information Processing Systems (NIPS)*, 1997.
9. Geman, S. and Geman, D.: Stochastic Relaxation, Gibbs Distributions, and Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.
10. Kolmogorov, V. and Zabih, R.: What Energy Functions Can Be Minimized via Graph Cuts? In *European Conference on Computer Vision*, volume 3, pages 65–81, 2002.
11. Kumar, S. and Hebert, M.: Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification. In *International Conference on Computer Vision*, 2003.
12. Lafferty, J., McCallum, A., and Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of International Conference on Machine Learning*, 2001.
13. Li, S. Z.: *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2nd edition, 2001.
14. Scharstein, D. and Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(2):7–42, 2002.
15. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C.: A Comparative Study of Energy Minimization Methods for Markov Random Fields. In *Ninth European Conference on Computer Vision (ECCV 2006)*, volume 2, pages 16–29, 2006.
16. Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S.: Tree-Reweighted Belief Propagation Algorithms and Approximate ML Estimation by Pseudo-Moment Matching. In *AISTATS*, 2003.