Solving Multilabel Graph Cut Problems with Multilabel Swap

Peter Carr Richard Hartley
Australian National University and NICTA
Canberra, Australia

***Abstract*—Approximate solutions to labelling problems can be found using binary graph cuts and either the α-expansion or $\alpha - \beta$ swap algorithms. In some specific cases, an exact solution can be computed by constructing a multilabel graph. However, in many practical applications the multilabel graph construction is infeasible due to its excessively large memory requirements. In this work, we expand the concept of $\alpha - \beta$ swap to consider larger sets of labels at each iteration, and demonstrate how this approach is able to produce good approximate solutions to problems which can be solved using multilabel graph cuts. Furthermore, we show how α-expansion is a special case of multilabel swap, and from this new formulation, illustrate how α-expansion is now able to handle binary energy functions which do not satisfy the triangle inequality. Compared to α-β swap, multilabel swap is able to produce an approximate solution in a shorter amount of time. We demonstrate the merits of our approach by considering the denoising and stereo problems. We illustrate how multilabel swap can be used in a recursive fashion to produce a good solution quickly and without requiring excessive amounts of memory.**

***Index Terms*—graph cuts; optimization; labelling; Markov Random Field;**

I. Introduction

In this work we are concerned with algorithms for solving labelling problems, where one is given a set of variables $\mathbf{X} = \{X_1, X_2, \dots X_n\}$ and must assign each variable a label x_i from a set of $\ell + 1$ labels $\mathcal{L} = \{0, 1, 2, \dots, \ell\}$. In computer vision applications, the labelling problem is often treated within the context of a Markov Random Field; mainly because images tend to have inherent spatial relations — i.e., neighbouring pixels usually have the same label. Previous work [1] has demonstrated how the MRF formulation is quite useful for problems such as stereo, image enhancement and segmentation.

Roof dual [2] is able to solve a binary labelling problem in an efficient manner. However, in general, there is no guarantee that the algorithm will assign every variable a label. When this occurs, one can use the *probe* or *improve* [3] extensions to obtain a complete labelling which is respectively optimal or approximate. However, if the problem is submodular (or more correctly, if the problem can be made submodular by complementing some binary variables), roof dual will produce a label for each variable without needing the *probe* or *improve* extensions.

When the label set contains more than two labels, move-based algorithms such as α-expansion and $\alpha - \beta$ swap [1] can be used to find an approximate solution. If a decision about each variable is required (i.e., the binary move problem must be submodular), each algorithm has restrictions on the types of energy functions it can handle — in particular, according to [1], α-expansion requires the binary energy terms to satisfy the triangle inequality. It is a surprising outcome of this paper that α-expansion can also handle convex functions, which certainly do not satisfy the triangle inequality.

Alternatively, instead of using move algorithms, one can encode a multilabel problem using a series of binary variables [4]. If the multilabel problem is submodular, an exact solution can be recovered [5], [6], [7]. Generally, one is only guaranteed to obtain a permissible sub-range of labels for each variable (where the sub-range could be the entire label set). Again, the extensions *probe* and *improve* can be used to identify a single label for each variable. Although multilabel constructions have useful optimality properties, the approach is rarely used in practice. For a typical vision problem, the required computing resources for multilabel graphs usually exceed the capabilities of the computer.

Finally, we note that there are many other algorithms for estimating the optimal configuration(s) of an MRF. However, as Szeliski *et al.* [8] demonstrated, algorithms based on max-flow perform particularly well for vision problems. As a result, we will focus solely on these methods.

A. Contributions

We present the 'multilabel swap' algorithm as a suitable compromise between the optimality properties of multilabel encodings and the minimal memory requirements of move algorithms. For convenience, we will focus on submodular problems, but emphasize that this is not a necessary requirement of multilabel swap.

We explain how multilabel swap can be implemented using different strategies, and focus on a new iterative refinement scheme. Although the method operates in an iterative fashion, unlike α-expansion or α-β swap, it does not move from one configuration to the next. Instead, multiple restrictions of the energy function are considered, and each is minimized. The quality of the solution, speed of the algorithm, and its memory requirements are all dependent on the restrictions. A larger restriction results in a smaller multilabel graph and lower memory requirements. However, the explored state space is also smaller, which means more iterations may be necessary to find a good solution.

We show how α-expansion and α-β swap correspond to particular strategies of the multilabel swap algorithm. Here, however, α-expansion uses a new encoding to indicate whether a variable should switch to the label α. This new scheme permits the algorithm to handle some functions which do not satisfy the triangle inequality; a situation that had previously been deemed infeasible [1]. Specifically, our algorithm applies to functions that satisfy an ordered "reverse triangle inequality", and, in particular, it works for pairwise functions of label difference which are convex.

In vision applications, labelling problems with large label sets are usually discrete approximations to continuous optimization problems. We consider the denoising and stereo scenarios, and show how restrictions in the continuous domain can be applied in an iterative fashion. Although the conditions for an optimality guarantee are quite stringent, our experiments illustrate how multilabel swap is able to outperform α-β swap in terms of speed as well as producing an equal, if not better, solution while maintaining reasonable memory requirements.

II. Background

The most probable labelling $\mathbf{x}^\star$ of an MRF corresponds to the configuration with the minimum potential energy [9]. In a second-order MRF, the energy function $E : \mathcal{L}^n \to \mathbb{R} \cup \{+\infty\}$ is a summation over cliques of size one (individual pixels) and two (pairs of neighbouring pixels):

$$E(\mathbf{x}) = \sum_{i \in \mathcal{C}_1} E_i(x_i) + \sum_{(i,j) \in \mathcal{C}_2} E_{ij}(x_i, x_j) \qquad (1)$$

A. Binary Graph Cuts

When the variables $\mathbf{X}$ take binary values, the energy function (1) is a quadratic pseudo-boolean function $E : \mathbb{B}^n \to \mathbb{R} \cup \{+\infty\}$. Boykov *et al.* [1] demonstrated how the global minimum of certain pseudo-boolean functions can be found in polynomial time using the max-flow algorithm. The approach expresses the labelling problem as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{W})$ with vertices $\mathcal{V}$ corresponding to variables X_i and edge weights $\mathcal{W}$ to terms in the energy function (1), such that the energy of a configuration $\mathbf{x}$ is equal to the cost of a corresponding cut of the graph. The max-flow algorithm is used to find the minimum cut of the graph, which corresponds to a minimum configuration $\mathbf{x}^\star$ of $E(\mathbf{x})$. The max-flow algorithm requires a graph with non-negative edge weights, which means all quadratic terms must be submodular [10]:

$$E_{ij}(0,0) + E_{ij}(1,1) \le E_{ij}(0,1) + E_{ij}(1,0) \qquad (2)$$
$$\text{for all } i, j \in \mathcal{P}$$

B. Move Algorithms

Binary graph cuts can be employed to minimize a multilabel energy function by formulating the optimization strategy in terms of binary decisions. Boykov *et al.* [1] proposed two 'move' algorithms which start at a configuration $\mathbf{x}$ and try to find a labelling $\mathbf{x}'$ with a lower energy. Neither is guaranteed to find an optimal solution, but both do quite well in practice. The binary graphs used in each iteration of either algorithm must be submodular [11]. In α-expansion, the decision is whether a variable should take the label α:

$$E_{ij}(\alpha,\alpha) + E_{ij}(x_i,x_j) \le E_{ij}(x_i,\alpha) + E_{ij}(\alpha,x_j) \qquad (3)$$
$$\text{for all } \alpha, x_i, x_j \in \mathcal{L}$$
$$\text{for all } i, j \in \mathcal{P} \mid \{x_i, x_j\} \ne \alpha$$

In α-β swap, the decision is whether the variables labelled α or β should change their labels to β or α respectively:

$$E_{ij}(\alpha,\alpha) + E_{ij}(\beta,\beta) \le E_{ij}(\alpha,\beta) + E_{ij}(\beta,\alpha) \qquad (4)$$
$$\text{for all } \alpha, \beta \in \mathcal{L}$$
$$\text{for all } i, j \in \mathcal{P} \mid \{x_i, x_j\} \in \{\alpha, \beta\}$$

C. Multilabel Graph Cuts

The max-flow algorithm can be used to compute the optimal labelling of a multilabel function if all multilabel variables can be represented using a set of binary variables, and the resulting pseudo-boolean function is representable as a valid flow graph.

Previous work [5], [6], [7] has already addressed this problem. Although each presents a slightly different methodology, the results are equivalent. Since the details are available elsewhere, we only provide an overview of the graph construction here; mainly to establish consistent terminology for explaining our multilabel swap algorithm.

Each multilabel variable X_i is encoded using ℓ binary variables:

$$X_i \equiv \{Z_i^0, Z_i^1, Z_i^2, \ldots, Z_i^{\ell-1}\} \qquad (5)$$

where a binary variable Z_i^p represents the condition $x_i \le p$. There is an implicit restriction that all binary variables Z_i^0 through Z_i^{p-1} must be equal to zero, while the remaining Z_i^p to $Z_i^{\ell-1}$ must be equal to one. Fortunately, this constraint can be realized in a graphical representation.

Each binary variable Z_i^p is represented as a vertex in the graph. For a given multilabel variable X_i, we will refer to its set of binary variables (5) as a *chain*. Each chain is connected to the source and sink (see Figure 1), which permits $\ell+1$ pairs of forwards and reverse intra-chain edges. Therefore, one can associate assigning the label p to X_i with the edge from Z_i^{p-1} to Z_i^p (where the first boundary condition, -1, refers to the source and the second, $\ell+1$, to the sink). To ensure that X_i can only take one state at any given time — i.e., the restriction that Z_i^0 through Z_i^{p-1} are zero and Z_i^p to $Z_i^{\ell-1}$ are one — the edges in the reverse direction are assigned infinite weight.

Edge Weights Since there is a direct correspondence between states of X_i and forwards intra-chain edges, one can define these edge weights as:

$$w_i^{p-1,p} = E_i(p) \qquad (6)$$

Although there are alternative mappings [6], [7], the graphs are functionally equivalent, and each can be "reparametrized" [10] into the other form.

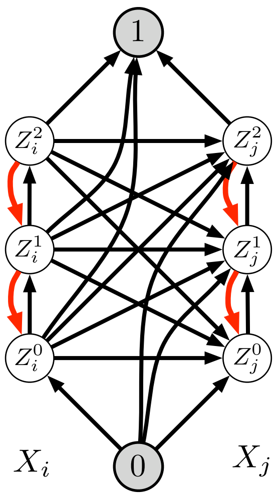


Fig. 1. *The above graph represents a pair of multilabel variables X_i and X_j which are assigned a label from $\mathcal{L}_4 = \{0, 1, 2, 3\}$. An edge from Z_i^{p-1} to Z_i^p represents $X_i = p$. Each chain is connected in reverse with infinite constraint edges (shown in red) to ensure only one state of X_i can be represented in any min-cut/max-flow partition. Every vertex in a particular chain is connected to all other vertices of neighbouring chains. The boundary effects of this pattern connect vertices in the middle of a chain directly to the source or sink. The graph can be simplified via re-weighting, but we leave it in this form for clarity.*

The binary terms $E_{ij}(p, q)$ produce inter-chain edges which connect vertices between chains corresponding to neighbouring variables:

$$w_{ij}^{pq} = E_{ij}(p+1, q) + E_{ij}(p, q+1) - E_{ij}(p, q) - E_{ij}(p+1, q+1) \quad (7)$$

Conditions As before, the max-flow algorithm requires non-negative edge weights. If an intra-chain edge weight is negative, it can be made positive by pushing flow along the chain. However, it is impossible to conduct a similar re-weighting to the inter-chain edges. The positivity of these edge weights is determined by the binary terms $E_{ij}(x_i, x_j)$ alone. Therefore, to ensure a valid flow graph, the pairwise function must satisfy the multilabel submodularity condition:

$$E_{ij}(x_i, x_j) + E_{ij}(x_i{+}1, x_j{+}1) \leq E_{ij}(x_i, x_j{+}1) + E_{ij}(x_i{+}1, x_j) \quad (8)$$
$$\text{for all } x_i, x_j \in \mathcal{L} \text{ and all } i, j \in \mathcal{P}$$

When $E_{ij}(x_i, x_j)$ depends only on the difference in labels, (8) simplifies to the condition that the second difference of the function must always be non-negative. Equivalently, the function must be convex [5].

Graph Size For an $n \times m$ lattice with cliques of two and a label set of $\ell + 1$ labels, the above graph construction requires $n \times m \times \ell$ vertices, and potentially $\ell^2[n\times(m-1)+m\times(n-1)]$ non-zero edges between vertices. We neglect edges between each vertex and the source and sink terminals, as these can be represented efficiently within the vertex structure.

III. Multilabel Swap

The multilabel swap algorithm is based on regular multilabel graph cuts. The difference is that multilabel swap is defined over a smaller set of labels, which means the memory requirements are much lower. Each iteration of the algorithm begins by identifying a subset $\mathcal{L}_i \subseteq \mathcal{L}$ of permissible labels for each variable X_i. A smaller multilabel graph representing the restricted energy function is then constructed and its minimum configuration found using max-flow. The process is repeated several times, with each iteration using a different set of permissible labels. The best estimate $\hat{\mathbf{x}}$ of the optimal configuration corresponds to the solution of the iteration having the lowest energy.

Related Work Our multilabel swap method shares similarities with Boykov *et al.*'s $\alpha-\beta$ swap [1], Veksler's $\alpha-\beta$ range [12] and Zureiki *et al.*'s reduced graphs [13]. Unlike $\alpha - \beta$ swap, multilabel swap considers subsets of two or more labels. Moreover, in multilabel swap, a different subset of labels can be selected for each variable. Furthermore, unlike the $\alpha - \beta$ range algorithm, the subsets selected for any particular iteration of multilabel swap do not have to correspond to a sub-range of labels, nor do the sizes of the subsets have to be identical for all variables. For example, if the original problem contained sixteen labels $\mathcal{L} = \{0, 1, 2, \ldots, 15\}$, then in a particular iteration, the variable X_i could be restricted to a sub-range $\mathcal{L}_i = \{4, 5, \ldots, 12\}$ and its immediate neighbour X_j restricted to the subset $\mathcal{L}_j = \{0, 5, 8, 11\}$. In $\alpha - \beta$ swap, the selected subset of two labels is identical for every pixel in the image.

Zureiki *et al.* [13] employ a similar approach, but their method requires the smoothness function to be the distance metric — i.e., $E_{ij}(x_i, x_j) = \lambda_{ij}|x_i - x_j|$. Each label set $\mathcal{L}_i$ is determined by finding the N labels which produce the lowest evaluations of the individual unary functions $E_i(p)$ in isolation. A multilabel graph is then constructed using the restricted label set for each variable and defining new mappings between terms of the energy function and weights of the edges in the multilabel graph. In constrast, multilabel swap is able to handle any energy function which satisfies (8) — not just the distance metric. Our algorithm achieves this generalization by starting with the complete multilabel graph and producing a smaller graph by propagating the implications of each restriction.

LogCut [14] estimates an optimal labelling by determining the individual bit values in the integer representations of the labels. The bit values are estimated in sequential fashion from most significant to least significant bit. When estimating the n^{th} bit, the previous $n-1$ bits have already been determined, and the remaining $n + 1 \ldots \lceil \log_2 \ell \rceil$ bits are temporarily assigned values using a variety of different methods. The algorithm employs roof dual and its extensions to estimate the optimal n^{th} bit labelling. For increased robustness, the algorithm applies shifts to the labels (essentially changing the bit ordering) and repeats the optimization process.

Very recently (while our multilabel swap algorithm was being refined), Gould *et al.* proposed “Alphabet SOUP” [15], an algorithm which employs a similar idea of minimizing a series of restricted energy functions. The strategy uses message passing to solve each restricted problem, since the authors claim energy functions used in higher level image reasoning are typically not submodular nor limited to pairwise interactions. The authors do state that a max-flow based optimization strategy could be used. However, since message passing algorithms do not place conditions on the types of functions that can be minimized, Gould *et al.* did not consider whether the restriction of a submodular function remains submodular. This observation is quite important if a max-flow or roof dual optimization strategy is to be employed.

We will now formally describe our definition of a restriction and show how this leads to a simplified multilabel graph.

Algebraic Restriction Each iteration of multilabel swap identifies a subset of labels $\mathcal{L}_i$ for each variable X_i. We refer to the collection of these subsets as the permissible set $\mathbf{R} = \{\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \dots\}$ of configurations. A restriction $E^{\mathbf{R}}(\mathbf{x})$ of the energy function to the set of permissible configurations $\mathbf{R}$ is defined as:

$$E^{\mathbf{R}}(\mathbf{x}) = \begin{cases} E(\mathbf{x}) & \text{if } x_i \in \mathcal{L}_i \text{ for all } i \\ \infty & \text{otherwise} \end{cases} \tag{9}$$

In general, the original energy function $E(\mathbf{x})$ may have more than one minimal configuration. To handle this situation, we refer to the set of minimizers $\{\mathbf{x}^\star\} = \{\mathbf{x}_0^\star, \mathbf{x}_1^\star, \mathbf{x}_2^\star, \dots\}$, where $\mathbf{x}_i^\star$ is a labelling with minimum energy. If the entire set of minimizers is contained within the restricted domain $\mathbf{R}$, then the minimization of the restricted function is identical to the minimization of the unrestricted function.

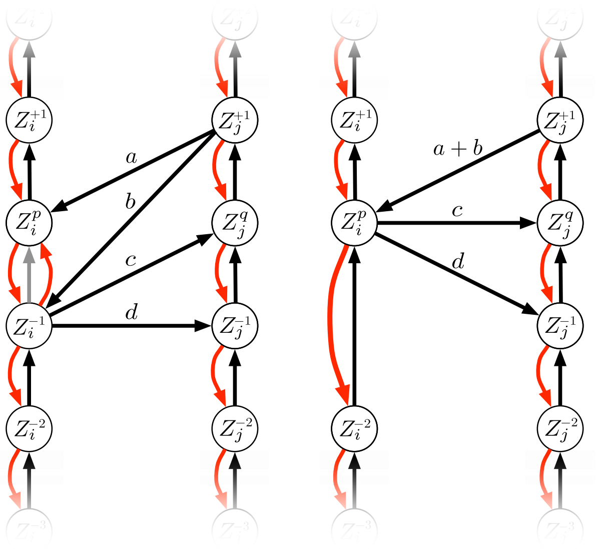
$$\arg\min_{\mathbf{x}} E(\mathbf{x}) = \arg\min_{\mathbf{x}} E^{\mathbf{R}}(\mathbf{x}) \tag{10}$$
$$\text{for all } \mathbf{x} \in \mathcal{L}^n \text{ where } \{\mathbf{x}^\star\} \in \mathbf{R}$$

If multiple minima exist, usually one is only concerned with finding a solution. Therefore, as long as the restriction $\mathbf{R}$ does not exclude the entire set $\{\mathbf{x}^\star\}$, the minimization of the restricted function $E^{\mathbf{R}}(\mathbf{x})$ will produce an optimal solution to the original problem.

Graphical Restriction In any iteration of multilabel swap, a restriction of the original function is produced by selecting a subset of labels $\mathcal{L}_i \subseteq \mathcal{L}$ for every multilabel variable X_i. If a particular label p is not within the restricted subset $\mathcal{L}_i$, the restricted energy function (9) evaluates to infinity for any configuration $\mathbf{x}$ where $\mathbf{x}_i = p$.

A restriction of the energy function (1) can be achieved by imposing a similar restriction on the the unary functions $E_i(\mathbf{x}_i)$. In this example, the unary function is redefined such that $E_i^{\mathbf{R}}(p) = \infty$ for any label $p \notin \mathcal{L}_i$. Realizing the restriction through the unary terms ensures the restricted function remains submodular (8), since the binary terms $E_{ij}(x_i, x_j)$ remain unaltered.

The restriction is implemented graphically by adding an edge with infinite weight from Z_i^{p-1} to Z_i^p (see Figure 2a). Since vertices Z_i^{p-1} and Z_i^p are now connected with infinite capacity in both directions, one of the vertices can be removed from the graph without affecting the flow. Generally, the choice of which vertex to remove is arbitrary. However, if one of the vertices corresponds to either the source or sink, the other vertex must be removed. In this example, we arbitrarily select Z_i^{p-1} as the vertex to eliminate. To remove the vertex from the graph, any edges originating from or terminating at Z_i^{p-1} must be redirected to Z_i^p (see Figure 2b).



(a) (b)

Fig. 2. *A restricted function is realized by setting the weight associated with a disallowed label to infinity and then removing the corresponding vertex. In this example, variable X_i is not allowed to take label p. An edge with infinite weight, as shown in (a), is added from Z_i^{p-1} to Z_i^p. As illustrated in (b), edges originating from Z_i^{p-1} now emanate from Z_i^p. Similarly, edges which terminated at Z_i^{p-1} are now incident on Z_i^p. If an edge already exists in the new location, the weight of the redirected edge is added to the existing edge.*

The above process can be repeated for every disallowed label associated with each multilabel variable. The resulting graph will have $|\mathcal{L}_i| - 1$ binary vertices for each multilabel variable X_i, and $(|\mathcal{L}_i| - 1) \times (|\mathcal{L}_j| - 1)$ edges between each pair of multilabel variables. If the subsets of labels are suitably small for each variable, the graph of the restricted function will fit within the memory capabilities of current mainstream computers.

Optimality If the restriction to a specific subset of possible configurations $\mathbf{x}^{\mathbf{R}}$ does not exclude all of the minimizers $\{\mathbf{x}^\star\}$ of the original unrestricted function, then an optimal labelling of the restricted problem will correspond to a solution of the unrestricted problem (10). However, without knowing the solutions to the original problem, it is impossible to determine whether an arbitrary restriction excludes the entire set of optimal configurations $\{\mathbf{x}^\star\}$. As a result, like $\alpha - \beta$ swap, it is impossible to guarantee the performance of the algorithm.

A. Strategies

The quality of the solution and the speed at which it is acquired will be dependent on the size and number of restrictions, as well as the underlying characteristics of the energy function. Here, we provide a list of possible strategies which may be useful for certain vision problems:

Data Priority A natural approach is to neglect the pairwise terms and optimize each unary $E_i(x_i)$ term individually. This configuration can then be used as a starting point for generating restrictions. In each swapping iteration, one could select a subset of labels centred around the minima of $E_i(p)$, as well as labels corresponding to the minima of its neighbours (since most vision models assume neighbouring pixels have similar values). The process would then continue in an iterative fashion using labels clustered around x_i and x_j.

Model Priority Alternatively, one could down-play the importance of the unary terms and conduct a sparse sampling of the solution space. In this coarse to fine strategy, an initial restriction is applied to the set of labels such that only every n^{th} label is considered. One can then consider a finer (yet possibly still relatively coarse) sampling around the best label for each X_i. We refer to this method as 'iterative multilabel swap' and will discuss it in detail in the following section.

Alpha-Beta Swap The α-β swap algorithm identifies two labels α and β for the entire image and allows any variable labelled either α or β the opportunity to switch its label. If a particular variable X_i is not labelled α or β, it retains its original label $\mathbf{x}_i$. The same result can be achieved in a multilabel graph by defining the permissible label set for each variable as:

$$\mathcal{L}_i = \begin{cases} \{x_i\} & \text{if the current } x_i \notin \{\alpha, \beta\} \\ \{\alpha, \beta\} & \text{otherwise.} \end{cases} \tag{11}$$

Alpha-Expansion α-expansion is also a specific case of our multilabel swap framework. We define the permissible label set for each variable as:

$$\mathcal{L}_i = \{\alpha, x_i\} \quad \text{where } x_i \text{ is the current label.} \tag{12}$$

This allows a choice of α or the current label x_i at each variable X_i. Our general multilabel swap algorithm will solve the minimization problem that arises.

As we mentioned earlier, α-expansion is commonly believed [1] to require the triangle inequality:

$$E_{ij}(\alpha, \alpha) + E_{ij}(x_i, x_j) \leq E_{ij}(x_i, \alpha) + E_{ij}(\alpha, x_j) \tag{3}$$

Cost functions $E_{ij}(x_i, x_j) = f_{ij}(|x_i - x_j|)$ when $f_{ij}(\cdot)$ is convex do not satisfy this requirement. However, convexity is required for multilabel graphs [5]. Therefore, an explanation of how our multilabel swap algorithm can handle α-expansion is necessary.

In Boykov *et al.*'s original formulation, the binary state '0' represented the variable X_i keeping its label x_i, and '1' implied X_i should switch to the label α. In our approach, the binary state '0' means X_i should take the minimum of $\{x_i, \alpha\}$, whereas '1' implies the variable should take the maximum of the two labels.

We consider two variables currently labelled x_i and x_j and let α be another label value. At each iteration, the binary energy function $E^B(\mathbf{x})$ (which determines whether a variable takes the label α or not) must be submodular:

$$E^B_{ij}(0,0) + E^B_{ij}(1,1) \leq E^B_{ij}(0,1) + E^B_{ij}(1,0) \tag{2}$$

Suppose that the ordering of the labels is $x_i \leq \alpha \leq x_j$ and consider the term $E^B_{ij}(1,0)$. Since the first parameter '1' in this case means "select α" (the greater of α and x_i), and the second argument '0' also means "select α" (the lesser of α and x_j), we see that $E^B_{ij}(1,0)$ is equivalent to $E_{ij}(\alpha, \alpha)$ in the original multilabel cost function. Continuing in this fashion with the other terms, the required submodularity condition becomes:

$$E_{ij}(\alpha, \alpha) + E_{ij}(x_i, x_j) \geq E_{ij}(x_i, \alpha) + E_{ij}(\alpha, x_j) \tag{13}$$

Note that this is the opposite of the usual condition (3) for α-expansion. Equation (13) is satisfied if $E_{ij}(x_i, x_j) = f_{ij}(|x_i - x_j|)$ and $f_{ij}(\cdot)$ is a convex function.

A similar analysis, for the orderings $\alpha \leq x_i \leq x_j$ and $x_i \leq x_j \leq \alpha$, leads to the usual triangle inequality (3). However, this inequality will always be satisfied when $\alpha \leq x_i \leq x_j$ or $x_i \leq x_j \leq \alpha$ and $f_{ij}(\cdot)$ is a monotonically increasing function. If $f_{ij}(\cdot)$ is both convex and increasing, then it will satisfy all of the conditions just identified and α-expansion will work.

IV. Iterative Multilabel Swap

Many large multilabel problems are actually discrete approximations to continuous domain optimization problems. Image enhancement algorithms posed as MRF labelling problems, for instance, associate labels with pixel values — which are quantized light intensities. For these types of problems, we will associate the continuous domain interval $[0.0, 1.0]$ with a label set of $\ell + 1$ labels, such that the labels span the entire range in ℓ steps. As shown in Figure 3, the sets of labels generated via $\ell = 2^n$ have a straightforward mapping between labels in immediately smaller and larger sets. Alternatively, one can envision a function defined on a smaller label set as a restriction of an equivalent function defined on a larger label set. For instance, a function defined in $\mathcal{L}_5$ is the same as a function defined in $\mathcal{L}_9$ but restricted to even labels only.

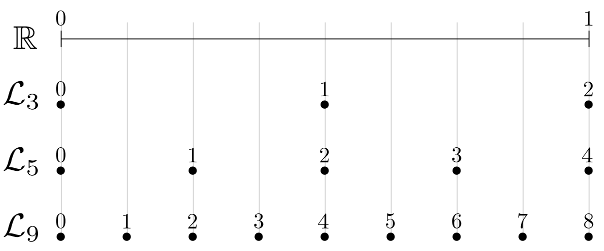


Fig. 3. *In many applications, the label set $\mathcal{L}$ will correspond to the continuous interval* $[0.0, 1.0]$. *The family of label sets $\{\mathcal{L}_3, \mathcal{L}_5, \mathcal{L}_9, \dots\}$ generated by $\ell = 2^n$ have a convenient mapping between labels in each set. The label 2 in $\mathcal{L}_5$ corresponds to the label 4 in $\mathcal{L}_9$.*

Our iterative multilabel swap algorithm assumes the solution to the problem with a smaller label set will be somewhat similar to the solution using a larger label set. For instance, a solution $\hat{\mathbf{x}}_{\ell+1}$ to a problem involving $\ell + 1$ labels has an equivalent configuration $\tilde{\mathbf{x}}_{2\ell+1}$ obtained by doubling each value of $\hat{\mathbf{x}}_{\ell+1}$. If the solution $\hat{\mathbf{x}}_{2\ell+1}$ at this new resolution is within the proximity of $\tilde{\mathbf{x}}_{2\ell+1}$, then one only needs to consider the sub-range of labels centred around $\tilde{\mathbf{x}}_{2\ell+1}$. For now, we will define proximity as the immediate neighbouring configurations at the lower scale. In this particular case, our iterative approach assumes the solution at a higher scale lies within the hypercube in $\mathbb{R}^n$ centred around the solution at the lower scale:

$$\hat{\mathbf{x}}_{2\ell+1} \in [\tilde{\mathbf{x}}_{2\ell+1} - \mathbf{2}, \tilde{\mathbf{x}}_{2\ell+1} + \mathbf{2}] \tag{14}$$

As a result, at each iteration we only consider a subset of five sequential labels for each variable (see Figure 4). For example, if the solution to a function defined on the label set $\mathcal{L}_5$ produced a label for pixel X_i of 2, at the next iteration (a function defined on the label set $\mathcal{L}_9$) the subset of labels $\mathcal{L}_i$ for pixel X_i would be $\{2, 3, 4, 5, 6\}$.

Unlike the general multilabel swap algorithm, the size of the graph remains fixed over all iterations, as every restricted subset of labels for each variable at each iteration will only contain five elements. However, although the graph remains the same size, almost every edge weight will change between iterations (since each edge represents a different label at each iteration). Therefore, the optimization technique [16] to start each iteration using the search trees of the previous calculation and augmenting as necessary will give little improvement, as the problem at the current iteration will bare little resemblance to that of the previous iteration.

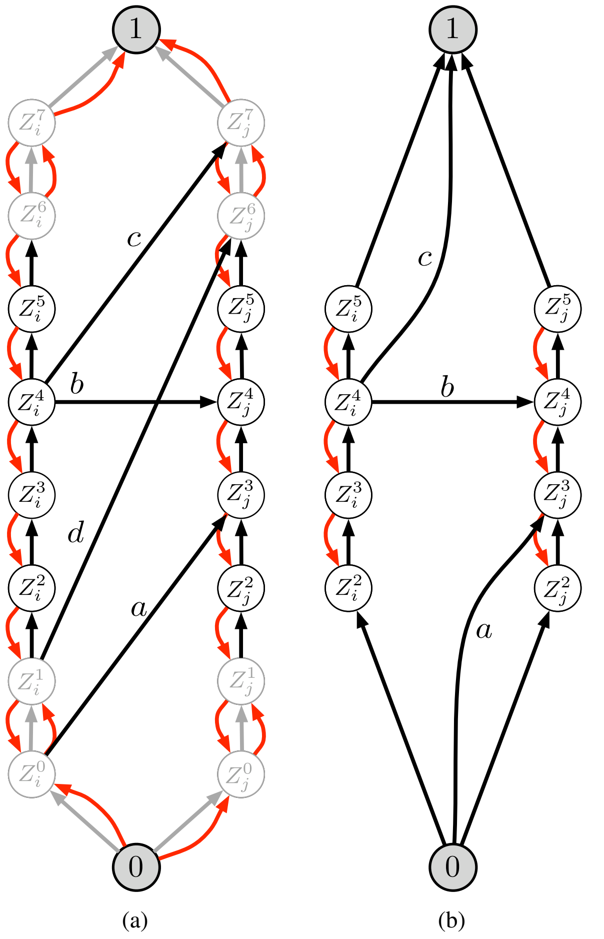


Fig. 4. *In iterative multilabel swap, the permissible label set is defined by the previous solution. Here, we illustrate the case when both X_i and X_j were assigned the label 2 from $\mathcal{L}_5$. The permissible label set in $\mathcal{L}_9$ is $\{2, 3, 4, 5, 6\}$. The restrictions are shown in (a), and the simplified graph is illustrated in (b). Any edge originating from Z_i^6 or Z_i^7 will not be in the minimum cut, since these two vertices are restricted to the set $\mathcal{V}_1$. Similarly, since Z_j^0 and Z_j^1 are constrained to $\mathcal{V}_0$, edges terminating at these vertices will not be in the minimum cut. The remaining four cases are illustrated above. Edges which redirect to produce a direct connection between the source and sink, such as Z_i^1 to Z_j^6 in (a), are omitted, as these represent constant terms in the energy expression and do not affect the minimization process.*

V. Experiments

We evaluate iterative multilabel swap — the strategy which samples the solution space in a coarse to fine manner — by examing its performance in denoising and stereo applications. Both situations are discrete approximations to continuous optimization problems. In denoising, one is given a noisy input image and must estimate the the true intensity of each pixel. In stereo, one is provided two views of a scene and must estimate the disparity between pixels in each view.

In all experiments, we compare the results of iterative multilabel swap with α-β swap, our convex formulation of α-expansion and, when feasible, multilabel graphs. Unless stated otherwise, both α-expansion and $\alpha - \beta$-swap are run for three iterations. An initial configuration is determined by optimizing the individual $E_i(x_i)$ functions independently.

A. *Denoising*

We assume the 8-bit noisy input image $\mathbf{I}$ is corrupted with zero-mean additive Gaussian noise having standard deviation $\sigma = 30$. The goal is to recover the true noiseless image $\mathbf{I}^\star$. The unary term arises from the negative log probability of the Gaussian noise model [9]:

$$E_i(x_i) = \frac{1}{2\sigma^2}\left(\frac{x_i}{\ell} - I(i)\right)^2 \tag{15}$$

Typically, a pair of neighbouring pixels i and j in the true image $\mathbf{I}^\star$ will have similar intensities, and one should minimize the square difference between labels x_i and x_j. However, edges are also quite prevalent, and any pairwise function should also be discontinuity preserving [9], [10]. As a result, we employ the Huber function $H(x, T)$ [17], which is convex for small differences and linear for large differences (which means it is discontinuity preserving):

$$H(x, T) = \begin{cases} x^2 & \text{if } |x| < T, \\ 2T\,|x| - T^2 & \text{otherwise.} \end{cases} \tag{16}$$

The parameter T was arbitrarily set at 0.04, implying differences less than 10 units at 8-bit depth are penalized using the square difference, whereas larger differences are only penalized linearly. The value of λ was manually tuned to 2, and the parameter $\frac{1}{2\sigma^2}$ was omitted, as its effect is incorporated into λ. As a result, the pairwise cost function was:

$$E_{ij}(x_i, x_j) = 2H\left(\frac{x_i - x_j}{\ell}, 0.04\right) \tag{17}$$

We first denoised a small 128×128 image using 33 grey levels, so that the solutions produced by each algorithm could be evaluated relative to the optimal solution, which was recovered using a multilabel graph (see Table I). All algorithms were able to obtain a good answer. Iterative multilabel swap was able to recover an optimal labelling, while convex α-expansion was clearly the fastest.

TABLE I
Comparison of energies, execution time and memory requirements for the four algorithms when denoising a small 128×128 image (RMS = 19.90) using 33 grey levels. In this circumstance, iterative multilabel swap was able to find an optimal labelling, while both α-β swap and α-expansion produced good approximate solutions. Alpha-expansion was significantly faster than the other algorithms. All of the iterative algorithms had manageable memory requirements.

Method	Energy -	RMS [intensity]	Time [s]	Memory [MB]
Multilabel Graph	186.56	13.12	9.0	500
$\alpha - \beta$ Swap	187.43	12.98	7.6	2
α-expansion	187.13	13.11	1.7	2
Iter. Multilabel Swap	186.56	13.12	5.9	20

We then evaluated the three iterative algorithms on a more practical example: enhancing a 512×512 image using 129 grey levels (see Figure 5). As Table II shows, all of the algorithms are again able to reach similar energies. Once again, iterative multilabel swap produces the best answer and α-expansion is significantly faster than the others.

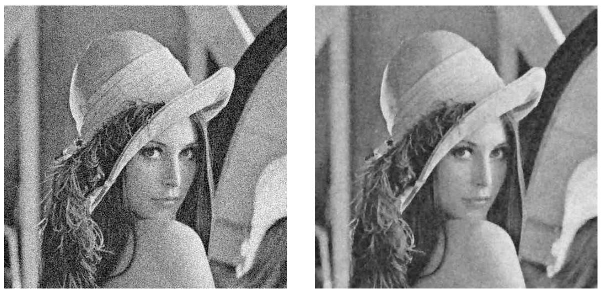


Fig. 5. *Synthetic zero-mean additive Gaussian noise with standard deviation $\sigma = 30$ intensity units was was added to an 8-bit image 512×512 input image. All algorithms enhanced the image using 129 labels, and arrived at similar energies. As Table II shows, iterative multilabel swap produced the best labelling and α-expansion was significantly faster than the other methods. The solution provided by iterative multilabel swap is shown on the right.*

TABLE II
The noisy 512×512 image (RMS = 29.48) was enhanced using 129 grey levels. All algorithms produced competitive answers. Iterative multilabel swap produced the best enhancement, but convex α-expansion was significantly faster. Since all of the iterative algorithms use memory efficiently, the specific requirements of each algorithm were not measured.

Method	Energy -	RMS [intensity]	Time [s]
$\alpha - \beta$ Swap	3384.24	9.79	2108.1
α-expansion	3381.59	9.73	107.6
Iter. Multilabel Swap	3376.24	9.69	1203.9

B. Stereo

Given a pair of stereo images $(\mathbf{L}, \mathbf{R})$ the task is to produce a disparity map $\mathbf{D}$ which indicates how each pixel i in $\mathbf{L}$ maps to a pixel $i - \mathbf{D}(i)$ in $\mathbf{R}$, where $i - \mathbf{D}(i)$ represents the horizontal translation in a rectified stereo pair [18].

Usually, the unary term is just the difference in appearance between a pair of pixels. However, for increased robustness, we evaluate the difference in appearance between two uniformly weighted 3×3 windows around i in $\mathbf{L}$ and $i - \mathbf{D}(i)$ in $\mathbf{R}$:

$$E_i(x_i) = \|\mathbf{L}(i) - \mathbf{R}(i - x_i)\|^2 \tag{18}$$

We model the scene as a collection of fronto-parallel surfaces, which corresponds to a binary energy term:

$$E_{ij}(x_i, x_j) = \lambda(x_i - x_j)^2 \tag{19}$$

As before, we first compare the three algorithms using a small problem. Here, we use the 'bull' trial from the Middlebury 2001 data set [19]. The image is reduced to 216×190 and we use 17 possible disparity values, which means disparities are estimated to the nearest pixel. The setting $\lambda = 0.5$ was tuned manually. The results are summarized in Table III. Iterative multilabel swap obtains an optimal labelling and in the shortest amount of time.

TABLE III
Results for the reduced stereo problem of a 216×190 image using 17 disparity labels. None of the iterative algorithms is able to find an optimal solution. Of the iterative algorithms, multilabel swap produces the best labelling in the shortest amount of time.

Method	Energy -	RMS [pixels]	Time [s]	Memory [MB]
Multilabel Graph	10.96	1.09	5.5	900
α-β Swap	15.73	1.05	36.8	6
α-expansion	10.97	1.09	13.1	6
Iter. Multilabel Swap	10.96	1.09	4.4	60

Finally, we use the three iterative algorithms to estimate the disparity map for the full resolution 433×381 images using 33 possible disparity values (see Figure 6). As with the smaller problem, disparities are estimated to the nearest pixel. Again the parameter $\lambda = 0.5$ was tuned manually. Iterative multilabel swap was able to find a good configuration in the smallest amount of time (see Table IV). Alpha-expansion, however, was able to find a slightly better solution.

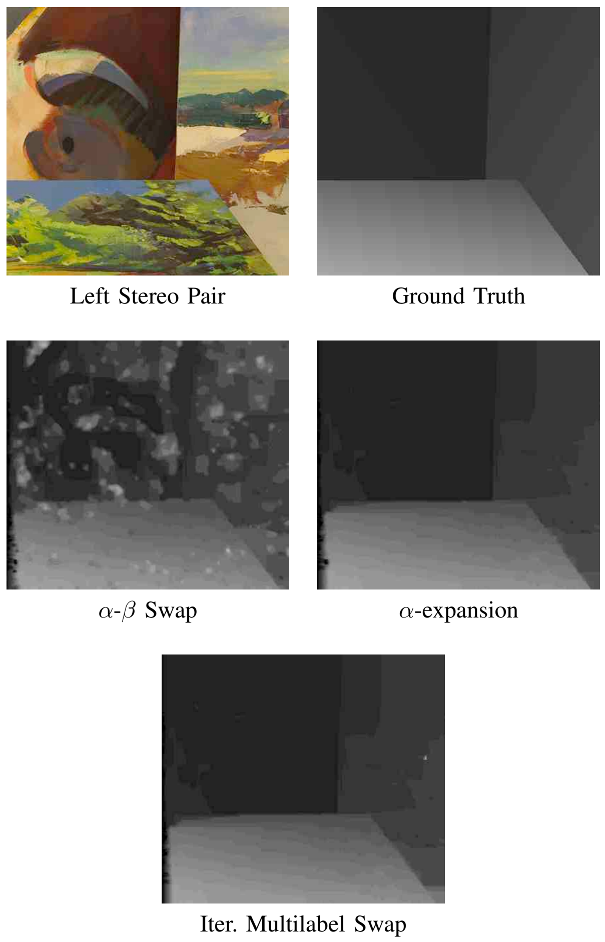


Fig. 6. *The disparities of the* 433×381 *left image were estimated by minimizing the costs defined in* (18) *and* (19)*. Each algorithm used 33 possible disparity values. As Table IV shows, iterative multilabel swap is able to obtain a good solution in the shortest amount of time.*

TABLE IV

Comparison of energies, execution time and RMS error for the full-resolution 433×381 *stereo experiment using 33 disparity values. In this scenario, iterative multilabel swap is able to find a good solution. Alpha-expansion finds a marginally better answer but takes approximately three times longer to do so. Alpha-beta swap is unable to find a good solution, and takes the longest time.*

Method	Energy	RMS	Time
	-	[pixles]	[s]
$\alpha - \beta$ Swap	61.93	2.25	512.9
α-expansion	39.67	1.52	100.6
Iter. Multilabel Swap	40.31	1.52	34.1

VI. Conclusions

Our multilabel swap algorithm is able to efficiently solve multilabel graph problems without requiring significant amounts of memory. Although there is no optimality guarantee, in practice the algorithm is able to find a global minimum given a sufficient number of iterations.

Multilabel swap is a generalization of both α-expansion and α-β swap. A limited solution space is considered by restricting the original energy function. As we have demonstrated, the restriction of a submodular function remains submodular. Furthermore, α-expansion can be used on convex problems by employing the minimum and maximum of a pair of labels. The alternative formulation, which considers the labels directly, can handle concave functions.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *PAMI*, vol. 23, no. 11, pp. 1222–1239, 2001.
[2] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 155–225, 2002.
[3] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, "Optimizing binary MRFs via extended roof duality," in *CVPR*, 2007.
[4] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. H. S. Torr, "On partial optimality in multi-label MRFs," in *ICML*, 2008.
[5] H. Ishikawa, "Exact optimization for markov random fields with convex priors," *PAMI*, vol. 25, no. 10, pp. 1333–1336, 2003.
[6] D. Schlesinger and B. Flach, "Transforming an arbitrary minsum problem into a binary one," Dresden University of Technology, Tech. Rep. TUD-FI06-01, 2006.
[7] J. Darbon, "Global optimization for first order markov random fields with submodular priors," in *International Workshop on Combinatorial Image Analysis*, 2008.
[8] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparitive study of energy minimization methods for markov random fields with smoothness-based priors," *PAMI*, vol. 30, no. 6, pp. 1068–1080, 2008.
[9] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001.
[10] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *PAMI*, vol. 26, no. 2, pp. 147–159, 2004.
[11] P. Kohli, M. P. Kumar, and P. H. S. Torr, "$\mathcal{P}^3$ & beyond: Solving energies with higher order cliques," in *CVPR*, 2007.
[12] O. Veksler, "Graph cut based optimization for MRFs with truncated convex priors," in *CVPR*, 2007.
[13] A. Zureiki, M. Devy, and R. Chatila, "Stereo matcing using reduced graph cuts," in *ICIP*, 2007.
[14] V. Lempitsky, C. Rother, and A. Blake, "Logcut - efficient graph cut optimization for markov random fields," in *ICCV*, 2007.
[15] S. Gould, F. Amat, and D. Koller, "Alphabet soup: A framework for approximate energy minimization," in *CVPR*, 2009.
[16] P. Kohli and P. H. S. Torr, "Dynamic graph cuts for efficient inference in markov random fields," *PAMI*, vol. 29, no. 12, pp. 2079–2088, 2007.
[17] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision (Second Edition)*. Cambridge University Press, 2003.
[18] M. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *PAMI*, vol. 25, no. 8, pp. 993–1008, 2003.
[19] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, 2002.