Graph Cuts via ℓ_1 Norm Minimization

Arvind Bhusnurmath, *Student Member*, *IEEE*, and Camillo J. Taylor, *Member*, *IEEE*

Abstract—Graph cuts have become an increasingly important tool for solving a number of energy minimization problems in computer vision and other fields. In this paper, the graph cut problem is reformulated as an unconstrained ℓ_1 norm minimization that can be solved effectively using interior point methods. This reformulation exposes connections between graph cuts and other related continuous optimization problems. Eventually, the problem is reduced to solving a sequence of sparse linear systems involving the Laplacian of the underlying graph. The proposed procedure exploits the structure of these linear systems in a manner that is easily amenable to parallel implementations. Experimental results obtained by applying the procedure to graphs derived from image processing problems are provided.

Index Terms—Convex optimization, computer vision, graph-theoretic methods, linear programming.

1 Introduction

Graph cuts have emerged as an important tool for solving a number of energy minimization problems encountered in computer vision and machine learning. In their seminal paper, Kolmogorov and Zabih [14] show that any energy function that satisfies a property called regularity can be minimized by finding the minimum cut of a graph whose edge weights are related to the energy function. The energy functions that are encountered in many computer vision problems satisfy this condition, which helps to explain the popularity of the approach.

Problems like image restoration [4], segmentation [6], [18], etc., have been reduced to graph cut problems. The graph cut methodology can also be applied to problems on 3D grids such as surface reconstruction [19]. Fig. 1 shows the typical structure of the resulting graphs. Here, the nodes s and t correspond to class labels, while the interior nodes correspond to the pixels in the image.

It is well known that, like many combinatorial optimization problems, the min-cut problem can be formulated as a linear program (LP) [16]. This paper presents an analysis that shows that this problem can be phrased as an unconstrained ℓ_1 norm minimization. This analysis allows us to draw connections between the graph cut problem and other ℓ_1 norm optimization problems such as those described by Koh et al. [13]. In many problems that are of interest to computer vision, the special structure of the problem can be exploited very effectively in this formulation.

Section 3 describes how the ℓ_1 norm minimization problem can be tackled using an interior point method. Using this approach, the original optimization problem is effectively reduced to the problem of solving a sequence of sparse linear systems involving the graph Laplacian. In this case, we can exploit the fact that these Laplacian matrices have a regular structure and a number of useful numerical properties which make them particularly amenable to solution by methods such as conjugate gradients. In fact, linear systems with this structure have been extensively studied in the context of solving the Poisson equation and related partial differential equations on 2D domains. The paper describes how techniques developed for these problems can be adapted to solve graph cut problems.

Importantly, the proposed optimization procedure can be carried out using vector operations that are highly amenable to parallelization. This means that the system is well suited to implementation on modern multicore CPUs and GPUs.

1.1 Related Work

Graph cut problems are usually solved using the equivalent maxflow formulation with Ford-Fulkerson or Push-relabel methods, which can be found in standard algorithms textbooks such as Cormen et al. [8]. However, as previously noted, most of the graphs that are encountered in vision problems tend to have an extremely structured form based on the underlying pixel grid. Boykov and Kolmogorov [5] exploit this fact and tune the Ford-Fulkerson algorithm to obtain a better performance. The basic idea is to employ two search trees, one emanating from the source and one from the sink, which are updated over the course of the algorithm. Parallel implementations using the push relabel approach on a GPU have also been described by Dixit et al. [9]. Their implementation offered some advantages over standard push relabel methods when the CPU and GPU were combined or when the maxflow problem was approximated. In contrast, our approach is based on a monotonically convergent continuous optimization scheme that is executed entirely on the GPU, avoiding costly GPU to CPU transfers.

Grady [11] formulates the interactive foreground background segmentation problem using the random walker framework and solves a system of equations involving the graph Laplacian that are very similar to the ones obtained in this work. This method is also implemented on the GPU in Grady et al. [12].

Sinop and Grady [20] have independently established connections between ℓ_1 norm and graph cuts. Their work shows that the Random Walker algorithm and the graph cuts algorithm both minimize energy. The random walker uses the ℓ_2 norm measure, while graph cuts uses the ℓ_1 norm. This paper establishes the same result through duality theory and also provides an implementation of the ℓ_1 norm minimization, which is highly parallelizable.

2 Theory

The goal of the min-cut problem is to divide the nodes in the graph shown in Fig. 1 into two disjoint sets, one containing s and the other containing t, such that the sum of the weights of the edges connecting these two sets is minimized. In the sequel, n will denote the number of interior nodes in the graph, while m will represent the total number of edges. This min-cut problem is typically solved by considering the associated max-flow problem. That is, if we view the edges as pipes and the associated edge weights as capacities, we can consider the problem of maximizing the total flow between the source node, s, and the sink node, t, subject to the constraint that each of the interior nodes is neither a sink nor a source of flow. The max-flow problem can be expressed as an LP, as shown in (1).

Let $\mathbf{x} \in \mathbb{R}^m$ denote a vector indicating the flow in each of the edges of the graph. A positive entry in this flow vector corresponds to a flow along the direction of the arrow associated with that edge, while a negative value corresponds to a flow in the opposite direction. In other words, the edges in our graph are undirected and the associated arrows merely represent the convention used to interpret the flow values.

• *The authors are with the GRASP Laboratory, University of Pennsylvania, 3330 Walnut St., Levine Hall, Philadelphia, PA 19104. E-mail: bhusnur4@seas.upenn.edu, cjtaylor@cis.upenn.edu.*

Manuscript received 10 Oct. 2007; revised 6 Feb. 2008; accepted 19 Mar. 2008; published online 31 Mar. 2008.
Recommended for acceptance by R. Zabih.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-10-0696.
Digital Object Identifier no. 10.1109/TPAMI.2008.82.

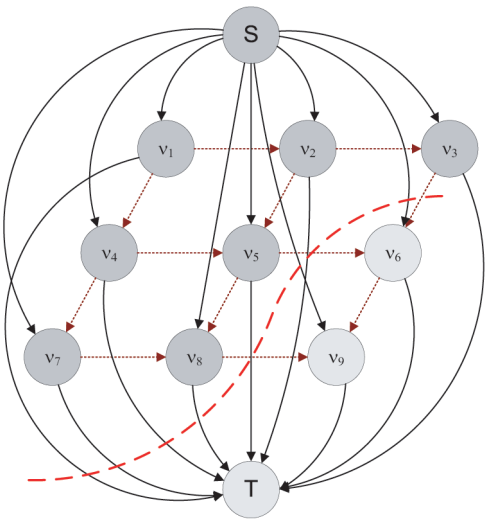

Fig. 1. The figure shows the typical grid-like graph found in vision problems. The dashed curve indicates the min cut.

The goal of the optimization problem is to maximize the inner product $\mathbf{c^T x}$, where $\mathbf{c} \in \mathbb{R}^m$ is a binary vector with $+1$ entries for all of the edges emanating from s and 0 entries elsewhere; this inner product effectively computes the net flow out of node s.

In order to express the constraint that the net flow associated with each of the interior nodes in the graph should be zero, we introduce the node edge incidence matrix $A \in \mathbb{R}^{n \times m}$ whose rows and columns correspond to interior nodes and graph edges, respectively. Each column of this matrix corresponds to an edge in the graph and will contain at most two nonzero entries, a $+1$ entry in the row corresponding to the node at the head of the arrow associated with that edge and a -1 for the node at the other end of the edge. Note that those columns corresponding to edges starting at s or terminating at t will only contain a single nonzero entry since the A matrix does not contain rows corresponding to the s and t nodes.

The product $A\mathbf{x} \in \mathbb{R}^n$ denotes the sum of the flows impinging upon each of the interior nodes due to the flow assignment $\mathbf{x}$. The constraint $A\mathbf{x} = \mathbf{0}$ reflects the fact that the net flow at each of the interior nodes should be zero. The vector $\mathbf{w} \in \mathbb{R}^m$ represents the nonnegative weights associated with each of the edges in the graph. The inequalities $-\mathbf{w} \leq \mathbf{x}$ and $\mathbf{x} \leq \mathbf{w}$ reflect the capacity constraints associated with each of the edges:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{st} \quad & A\mathbf{x} = 0 \\ & -\mathbf{w} \leq \mathbf{x} \leq \mathbf{w}. \end{aligned} \tag{1}$$

A careful reader will note that this formulation differs slightly from the one presented by Kolmogorov and Zabih [14], which makes use of a directed graph. However, it can be shown that this formulation allows us to represent precisely the same set of objective functions as the ones described in that work.

Instead of tackling the LP described in (1) directly, we proceed by formulating the associated dual problem. More specifically, by adding Lagrangians λ corresponding to the capacity constraint and ν corresponding to the conservation constraint, we can compute the optimal value of our original primal problem by maximizing the associated Lagrangian dual function, which gives rise to the following dual problem:

$$\begin{aligned} \min_{\lambda,\nu} \quad & \mathbf{w}^T(\lambda_+ + \lambda_-) \\ \text{st} \quad & A^T\nu - \mathbf{c} = (\lambda_- - \lambda_+) \\ & \lambda_+ \geq 0,\ \lambda_- \geq 0. \end{aligned} \tag{2}$$

It can be seen that, for a fixed value of ν, the minimum value that $(\lambda_- + \lambda_+)_i$ attains is $|(A^T\nu - \mathbf{c})_i|$.

This property allows us to reformulate the optimization problem in (2) as follows:

$$\min_{\nu} \sum_{i=1}^{m} w_i \left|(A^T\nu - \mathbf{c})_i\right|, \tag{3}$$

which can be rewritten as

$$\min_{\nu} \left\|\mathrm{diag}(\mathbf{w})(A^T\nu - \mathbf{c})\right\|_1. \tag{4}$$

Notice that the resulting optimization problem is an *unconstrained* ℓ_1 norm minimization, where the decision variables correspond to the Lagrange multipliers $\nu \in \mathbb{R}^n$. Here, the symmetries of the undirected graph cut formulation allow us to derive a result that is stronger than the LP formulations available for the more general directed graph cut problem [8], [16]. The unconstrained formulation in (4) is advantageous in many ways. It underlines the connection between graph cuts and convex optimization and allows us to employ continuous optimization techniques that can exploit the structure of the problem.

It is possible to show that the ν_i variables in (4) will converge to binary values without any external prodding. This is shown in the Appendix. The fact that the node weights, ν, will converge to binary values at the global minimum is a useful property with important practical consequences. First, it means that the optimization procedure yields the node labels immediately, without the need for an intervening flow interpretation. Second, the fact that the weights tend toward discrete values makes it easy to employ rounding as the barrier method approaches convergence. It also reduces the numerical precision required to execute the algorithm; in practice, one can carry out the procedure using a *single-precision* floating-point arithmetic. Contrast this with the problems one encounters in applying the barrier method to the max flow LP formulation where numerical issues can make it difficult to determine whether a given link is saturated with flow or merely close to saturation [17].

3 Implementation

The resulting unconstrained ℓ_1 norm minimization problem described in (4) can itself be formulated as an LP by introducing an auxiliary variable $\mathbf{y} \in \mathbb{R}^m$, where $\mathbf{y} \geq (A^T\nu - \mathbf{c})$ and $\mathbf{y} \geq -(A^T\nu - \mathbf{c})$, as described in [3]. The associated LP is shown below:

$$\begin{aligned} & \min \quad \mathbf{w}^T\mathbf{y} \\ \text{st} \quad & \begin{bmatrix} A^T & -I \\ -A^T & -I \end{bmatrix} \begin{bmatrix} \nu \\ \mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \mathbf{c} \\ -\mathbf{c} \end{bmatrix}. \end{aligned} \tag{5}$$

This problem can be solved using the interior point method with logarithmic barrier potentials. In this approach, the original LP is replaced with the following convex objective function:

$$\phi(\nu, \mathbf{y}) = t(\mathbf{w}^T\mathbf{y}) - \sum_{i=1}^{m} \log(\mathbf{y}_i - \mathbf{z_i}) - \sum_{i=1}^{m} \log(\mathbf{y}_i + \mathbf{z_i}), \tag{6}$$

where $\mathbf{z} = (A^T\nu - \mathbf{c})$. The scalar t is used to weigh the original objective function against the barrier potentials associated with the linear constraints.

This objective function is minimized using Newton's method. On each iteration of this procedure, a locally optimal step, $[\Delta\nu\ \Delta\mathbf{y}]^T$, is computed by solving the following linear system:

$$\begin{bmatrix} AD_+A^T & AD_- \\ D_-A^T & D_+ \end{bmatrix}\begin{bmatrix} \Delta\nu \\ \Delta\mathbf{y} \end{bmatrix} = -\begin{bmatrix} A(\mathbf{d}_1 - \mathbf{d}_2) \\ t\mathbf{w} - (\mathbf{d}_1 + \mathbf{d}_2) \end{bmatrix}, \tag{7}$$

where $\mathbf{d}_{1i} = 1/(\mathbf{y}_i - \mathbf{z}_i)$, $\mathbf{d}_{2i} = 1/(\mathbf{y}_i + \mathbf{z}_i)$, D_+ and D_- are diagonal matrices whose diagonal entries are computed as follows: $D_{+ii} = (\mathbf{d}_{1_i}^2 + \mathbf{d}_{2_i}^2)$ and $D_{-ii} = (\mathbf{d}_{2_i}^2 - \mathbf{d}_{1_i}^2)$. By applying block elimination to factor out $\Delta\mathbf{y}$, the system can be further reduced to

$$\big(A\mathrm{diag}(\mathbf{d})A^T\big)\Delta\nu = -A\mathbf{g}, \tag{8}$$

where

$$\mathbf{d}_i = 2/\big(\mathbf{y}_i^2 + \mathbf{z}_i^2\big) \tag{9}$$

and

$$\mathbf{g}_i = \frac{2\mathbf{z}_i}{\mathbf{y}_i^2 - \mathbf{z}_i^2} + \frac{2\mathbf{y}_i\mathbf{z}_i}{\mathbf{y}_i^2 + \mathbf{z}_i^2}\left(t\mathbf{w}_i - \frac{2\mathbf{y}_i}{\mathbf{y}_i^2 - \mathbf{z}_i^2}\right). \tag{10}$$

Once $\Delta\nu$ has been obtained from (8), the $\Delta\mathbf{y}$ component of the step can be computed using the following expression:

$$\Delta\mathbf{y} = D_+^{-1}\big((\mathbf{d}_1 + \mathbf{d}_2) - t\mathbf{w} - D_-A^T\Delta\nu\big). \tag{11}$$

The entire interior point optimization procedure is outlined in pseudocode as Algorithm 1. Let s denote the s-link weights and t denote the t-link weights. The input to this procedure is the vector of edge weights, $\mathbf{w}$. As an initial solution to our interior point method, we take the weight of the source edge that is incident on the node and divide that by the sum of the source edge weight and the sink edge weight, $\nu = s/(s+t)$, which gives a purely data driven labeling.

Algorithm 1. Solve min-cut: $\min_\nu \|\mathrm{diag}(\mathbf{w})(A^T\nu - \mathbf{c})\|_1$

```
choose t, μ and set ν = s/(s + t)
choose y such that y ≥ |A^T ν − c|
while change in ℓ1 norm since last (outer) iteration above threshold1 do
    while change in ℓ1 norm since last (inner) iteration above threshold2 do
        Compute d from (9)
        Compute g from (10)
        Solve (A diag(d) A^T)Δν = −Ag to get Δν
        Compute Δy from (11)
        If necessary, scale step by β so that ν + βΔν, y + βΔy are feasible.
    end while
    t = μ * t
end while
```

Note that the principal step in this procedure is the solution of the sparse linear system given in (8), which means that the original ℓ_1 norm minimization problem has been reduced to the problem of solving a sequence of sparse linear systems.

At this point, we note that the matrix $L = (A\mathrm{diag}(\mathbf{d})A^T)$ corresponds to a weighted graph Laplacian [1], where the vector $\mathbf{d}$ indicates the weights that are to be associated with each of the edges of the graph. In fact, the matrix L corresponds to the Graph Laplacian without the rows and columns associated with the s and t nodes.

The matrix is symmetric by construction and, since the entries in $\mathbf{d}$ are all positive, it is also positive definite. The entries along the diagonal of this matrix L_{ii} correspond to the sum of the weights of the edges impinging on the corresponding interior node in the graph—including the links to the s and t nodes. For the off diagonal elements, L_{ij}, it can be shown that $-L_{ij}$ will correspond to the weight of the edge connecting nodes i and j. This value will be zero if the two nodes are not connected. From these two observations, we can conclude that the matrix will be strictly diagonally dominant since the diagonal entries will include the weights associated with the links to the s and t nodes, which do not make an appearance in any of the off-diagonal entries:

$$L = \begin{pmatrix} A_1 & B_1 & & & \\ B_1 & A_2 & B_2 & & \\ & B_2 & A_3 & B_3 & \\ & & B_3 & A_4 & \cdots \\ & & & \cdots & \cdots \end{pmatrix}. \tag{12}$$

The resulting sparse, banded matrix reflects the topology of the underlying grid and takes the block tridiagonal form shown in (12), where the A_i submatrices are tridiagonal and the B_i submatrices are diagonal. Matrices with this structure are frequently encountered in the process of solving partial differential equations, such as Poisson's equation, on two-dimensional domains.

The numerical properties of the matrix L make the resulting linear system amenable to solution by the method of conjugate gradients [10]. Iterative techniques are preferred over direct techniques like Cholesky decomposition in this case because of the size of the matrix and the storage that would be required for the resulting factors.

A distinct advantage of the conjugate gradients technique is that the steps in this algorithm can be readily parallelized. Each conjugate gradient step involves one matrix vector multiplication, two inner products, and three Scalar Alpha X Plus Y (SAXPY) operations. All of these operations are amenable to implementation on the parallel architectures found on modern GPUs and multicore processors, as shown in [2] and [15]. In this case, we can exploit the fact that the matrix we seek to invert, L, has a regular structure that further simplifies the matrix vector multiplication operation required on each iteration of the conjugate gradients procedure.

The conjugate gradients algorithm can be accelerated by choosing an appropriate symmetric preconditioning matrix, C, and then applying conjugate gradients to solve the system $(CAC)(\boldsymbol{C}^{-1}\mathbf{x}) = C\mathbf{b}$, as described in [10]. The goal here is to choose a matrix C in such a way that the preconditioned matrix $CAC \approx I + B$, where B is a low rank matrix.

Concus et al. [7] describe preconditioning strategies that are specifically designed for the types of matrices that we seek to invert. Section 4 presents results that illustrate how effective these strategies can be in improving the convergence rate of the solver.

Experiments were also carried out using a simpler preconditioning strategy, where the matrix C is chosen as follows: $C = \mathrm{diag}(\mathbf{a})$, $\mathbf{a}_i = 1/\sqrt{A_{ii}}$. This is the Jacobi preconditioner. When this preconditioner is applied to a diagonally dominant matrix, such as L, it produces a normalized variant, where the diagonal elements are all 1 and the off diagonal elements all have magnitudes less than 1. Multiplying with this preconditioner does not affect the fill pattern of the matrix.

Koh et al. [13] also describe how the preconditioned conjugate gradients method can be used to solve ℓ_1 regularized logistic regression problems. In this work, we are able to exploit the structure of the Graph Laplacian matrix, L, for further speedups.

4 Results

Experiments were carried out on graphs derived from actual image processing problems in order to determine how well the proposed scheme would work in practice. Since the scheme

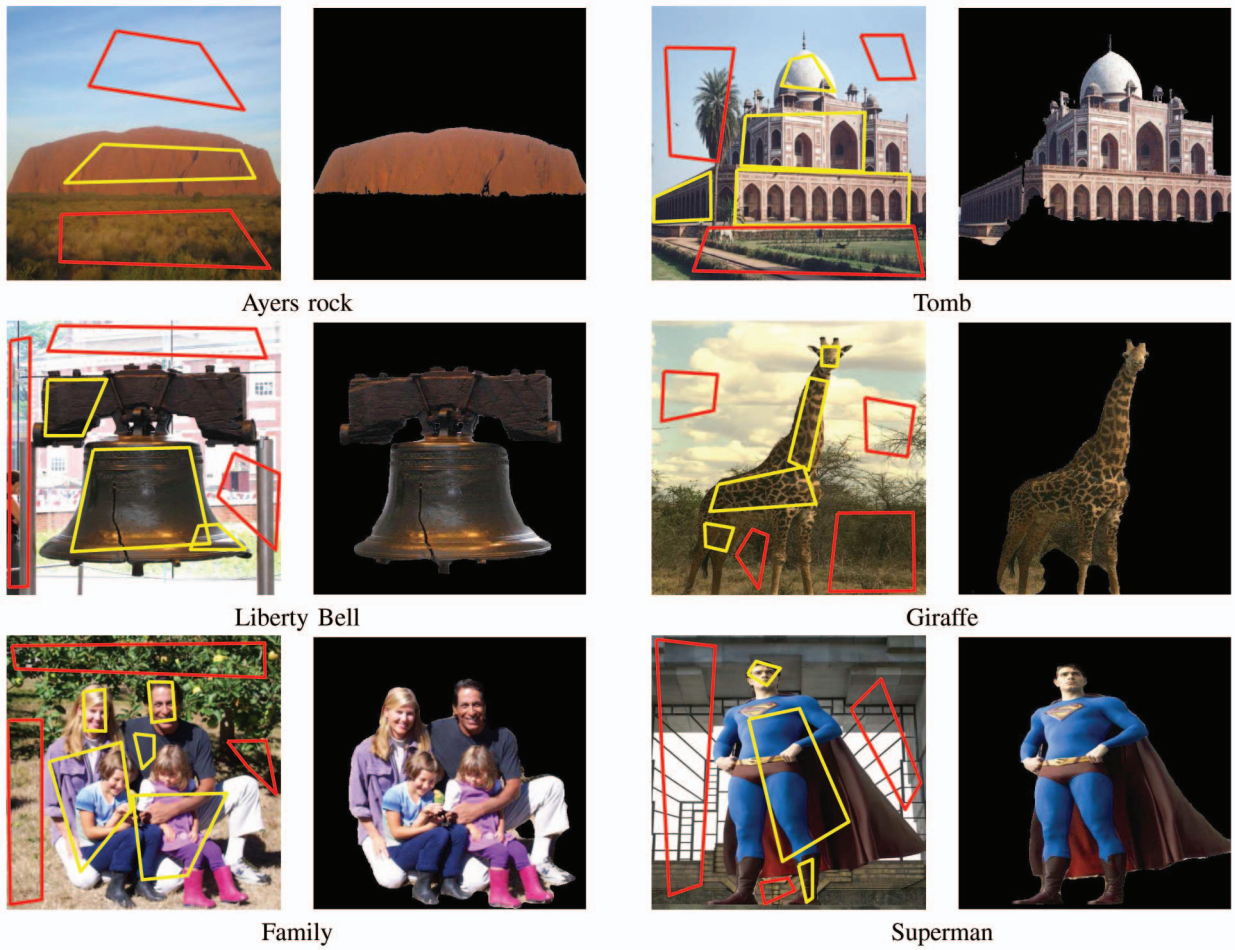

Fig. 2. These segmentation examples were used to test the graph cut implementation described in the previous sections. The graph weights were computed using a variant of the Grab Cut procedure described in [18].

essentially reduces the mincut problem to the problem of solving a sequence of sparse linear systems, one can gauge the computational effort required to resolve a given graph cut by recording the total number of conjugate gradient iterations that are performed in the course of driving the system to convergence. Three variants of the scheme were used in these experiments; the first variant employed the conjugate gradients method without any preconditioning, the second made use of the Jacobi (diagonal) preconditioner described in the previous section, while the third used the preconditioning strategy described by Concus et al. [7].

The method was applied to the foreground/background segmentation problems shown in Fig. 2. In these experiments, the underlying weighted graphs were constructed using the GrabCut algorithm described by Rother et al. [18]. All of the images in question are 512×512. Table 1 shows the number of conjugate gradient iterations taken by each of the three variants of the optimization procedure performing a single graph cut operation.

TABLE 1
Iterations Taken for Separating the Foreground in Each Image in Fig. 2, Using Different Preconditioning Strategies

Image name	CG iters	with diagonal preconditioner	with preconditioner in [7]
Ayers rock	411	74	19
Tomb	226	120	31
Liberty Bell	272	105	12
Giraffe	351	143	23
Family	279	132	25
Superman	435	95	23

These results demonstrate that the preconditioning schemes are, in fact, quite successful at accelerating the convergence of the conjugate gradients procedure in this situation. The diagonal preconditioner reduces the number of iterations required by a factor of 0.27 on average, while the Concus and Golub preconditioner reduces the complexity even further.

In all cases, the implementation converges in less than 15 Newton steps, which is consistent with the observed performance of interior point methods.

The proposed scheme was implemented on an Nvidia GeForce 8800 GTX GPU using Nvidia's newly released Compute Unified Device Architecture (CUDA). The Jacobi preconditioner was employed because of its relative simplicity. The implementation was applied to the segmentation problems shown in Fig. 2 and the timings achieved on these 512×512 problems are summarized in Table 2. For purposes of comparison, we also applied the flow-based graph cut method proposed by Boykov and Kolmogorov [5] to the same problems and recorded the timings achieved with a 2.66 GHz Intel Core 2 Duo processor with a 4 Mbyte cache. Although the two implementations use completely different hardware, these experiments provide some basis for comparison.

It is instructive to note how close the algorithm gets to the final solution within the first few Newton steps. Intermediate results are shown in Fig. 4. Fig. 3 shows the reduction in the ℓ_1 norm as a function of increasing Newton iterations. We tabulate the number

TABLE 2
Time Taken to Extract the Foreground of Images on the GPU versus a Flow-Based Method on the CPU

Image name	Boykov (in ms)	CUDA (in ms)
Ayers rock	60	62
Tomb	65	92
Liberty Bell	63	57
Giraffe	74	92
Family	73	86
Superman	61	71

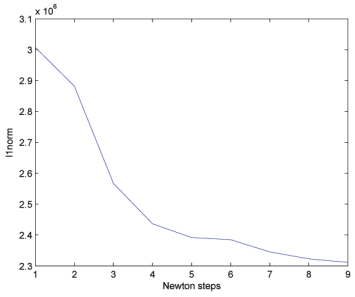

Fig. 3. ℓ_1 norm value at the end of each Newton step for the superman image.

of conjugate gradient iterations taken in each Newton step in Table 3. It is clear from the figure and the table that this is an algorithm with diminishing returns. The later stages take up more iterations (hence, more time) and do not seem to be changing the segmentation quality too much. This pattern was observed consistently in all of our experiments. One can, therefore, obtain acceptable segmentations in approximately half the time given in Table 2 by exploiting this observation. Note that this approach to approximation is justified by the monotonically decreasing ℓ_1 norm and differs from the push relabel approximation described by Dixit et al. [9].

Interestingly, an analysis of the current GPU implementation shows that it is currently memory bound rather than compute bound. On this application, the 8800 GPU delivers an effective performance of 16 GFlops, which is a small fraction of its theoretical peak performance of 330 GFlops. This is due to the fact that the underlying BLAS operations have low arithmetic intensity. On future multicore systems with larger on-chip memories, improved caching mechanisms, or greater memory bandwidth, the performance of the scheme will improve proportionately.

5 Conclusion

This paper describes how the graph cut problems of interest in computer vision can be rephrased as unconstrained ℓ_1 norm minimization problems. This perspective allows us to recognize connections between graph cut problems and other ℓ_1 norm optimization problems.

In the process of solving these optimization problems with an interior point method, one ends up solving a series of linear systems similar to those encountered in solving Poisson's equation on a grid. We can exploit the regularity and structure of these systems to develop optimized solvers that are amenable to parallelization.

Appendix

Convergence of the Node Labels to 0/1 Values

Proposition 1.1. *The unconstrained Lagrangian variables ν in the formulation given in problem (4) converge to either 0 or 1.*

TABLE 3
Number of Conjugate Gradient Iterations for Each Newton Step in the Superman Image

t	Newton step number	CG iters	Cumulative CG its
1	1	5	5
1	2	5	10
1	3	5	15
1	4	5	20
1	5	8	28
1	6	19	47
10	7	6	53
10	8	15	68
100	9	8	76
1000	10	19	95

Proof. We first note that the unconstrained ℓ_1 norm minimization arose out of the dual problem derived from the linear programming formulation of the maxflow problem. In the optimal flow assignment, every vertex is either connected to s or to t via a path consisting only of unsaturated edges.[1] Let us call such a path an unsaturated path.

The Langrangians ν correspond to the labeling of the pixels. Let us label the source as 1, while the label corresponding to the sink is 0.

Now, consider what happens when an edge i, connecting nodes j and k, is unsaturated in the final optimal flow assignment. We apply the KKT conditions and, in particular, complementary slackness [3] to the formulation in the primal-dual pair of problems (1) and (2). The complementary slackness conditions state that, when optimality is reached, the product of the constraint and the Lagrange multiplier corresponding to that constraint will be zero. Therefore, we get the following for edge i:

$$\begin{aligned}(x-w)_i \neq 0 &\Rightarrow (\lambda_+)_i = 0\\ (-x-w)_i \neq 0 &\Rightarrow (\lambda_-)_i = 0\\ &\Rightarrow (A^T\nu - c)_i = 0.\end{aligned} \tag{13}$$

This implies that if edge i is unsaturated, then the Lagrange multipliers associated with it will be 0. Now, depending upon the type of edge that i is, we have three cases:

1. $c_i = 1$ for an s edge. The corresponding row in A^T has a single $+1$ entry and, therefore, ν will have to be 1 for the pixel that is being connected to the source by this edge.
2. $c_i = 0$ for an internal edge. Here, it can be seen that both pixels being connected by this edge will have to have the same label.
3. $c_i = 0$ for a t edge. The row in A^T has a single -1 entry and, therefore, the pixel that is being connected to the sink by this edge will have to be labeled 0.

Therefore, all nodes connected to s by an unsaturated edge will be labeled 1 and this label propagates along every unsaturated edge. This leads to the conclusion that all nodes connected to s by an unsaturated path will be labeled 1 and all nodes connected to t by an unsaturated path will be labeled 0. When optimality is reached, every internal node has to belong to one of these two sets and, hence, every internal node is either going to be labeled 0 or 1. □

1. Here, we tacitly assume that the maxflow solution is unique.

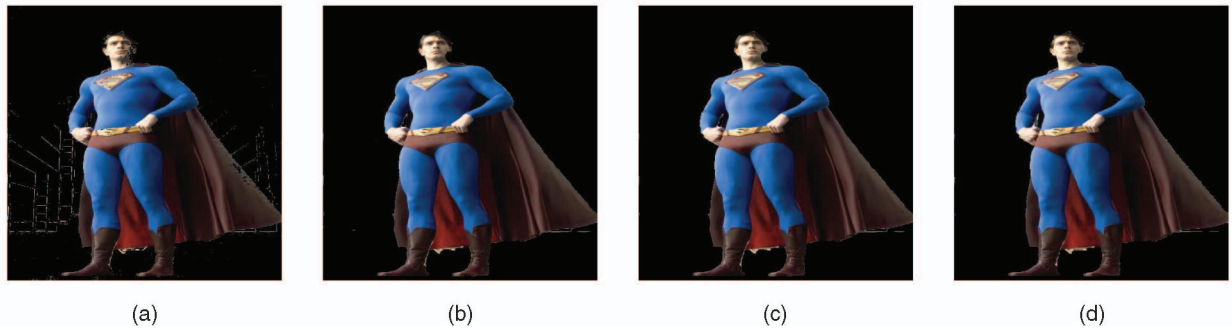

Fig. 4. The various stages of the graph cuts algorithm on the superman image. (a) Initial. (b) After six Newton steps. (c) After seven Newton steps. (d) Final result.

REFERENCES

[1] N. Biggs, *Algebraic Graph Theory*, second ed. Cambridge Math. Library, 1993.
[2] J. Bolz, I. Farmer, E. Grinspun, and P. Schroder, "Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid," *Proc. ACM SIGGRAPH '03, ACM Trans. Graphics*, pp. 917-924, 2003.
[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, 2004.
[4] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222-1239, Nov. 2001.
[5] Y. Boykov and V. Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124-1137, Sept. 2004.
[6] Y. Boykov and M. Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images," *Proc. Eighth Int'l Conf. Computer Vision*, pp. 105-112, 2001.
[7] P. Concus, G. Golub, and G. Meurant, "Block Preconditioning for the Conjugate Gradient Method," *SIAM J. Scientific Computing*, vol. 6, no. 1, pp. 220-252, 1985.
[8] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, second ed. MIT Press, 2002.
[9] N. Dixit, R. Keriven, and N. Paragios, "GPU-Cuts: Combinatorial Optimisation, Graphic Processing Units and Adaptive Object Extraction," Laboratoire Centre Enseignement Recherche Traitement Information Systèmes (CERTIS), Ecole Nationale des Ponts et Chaussees (ENPC), Mar. 2005.
[10] G. Golub and C. Van Loan, *Matrix Computations*, third ed. Johns Hopkins Univ. Press, 1998.
[11] L. Grady, "Random Walks for Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768-1783, Nov. 2006.
[12] L. Grady, T. Schiwetz, S. Aharon, and R. Westermann, "Random Walks for Interactive Organ Segmentation in Two and Three Dimensions: Implementation and Validation," *Proc. Int'l Conf. Medical Image Computing and Computer-Assisted Intervention*, pp. 773-780, 2005.
[13] K. Koh, S. Kim, and S. Boyd, "An Interior-Point Method for Large-Scale l1 Regularized Logistic Regression," *J. Machine Learning Research*, vol. 8, pp. 1519-1555, July 2007.
[14] V. Kolmogorov and R. Zabih, "What Energy Functions Can Be Minimized via Graph Cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147-159, Feb. 2004.
[15] J. Kruger and R. Westermann, "Linear Algebra Operators for GPU Implementation of Numerical Algorithms," *Proc. ACM SIGGRAPH '03, ACM Trans. Graphics*, pp. 908-916, 2003.
[16] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization—Algorithms and Complexity*. Prentice Hall, 1982.
[17] M. Resende and P. Pardalos, *Advances in Linear and Integer Programming*, Oxford Lecture Series in Math. and Its Applications, 1996.
[18] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut—Interactive Foreground Extraction Using Iterated Graph Cuts," *Proc. ACM SIGGRAPH '04, ACM Trans. Graphics*, pp. 309-314, 2004.
[19] S. Sinha and M. Pollefeys, "Multi-View Reconstruction Using Photo-Consistency and Exact Silhouette Constraints: A Maximum Flow Formulation," *Proc. 10th Int'l Conf. Computer Vision*, pp. 349-356, 2005.
[20] A.K. Sinop and L. Grady, "A Seeded Image Segmentation Framework Unifying Graph Cuts and Random Walker which Yields a New Algorithm," *Proc. 11th Int'l Conf. Computer Vision*, pp. 1-8, 2007.