Chapter5

Graph Cuts in Vision and Graphics: Theories and Applications

Y. Boykov and O. Veksler

Abstract

Combinatorial min-cut algorithms on graphs have emerged as an increasingly useful tool for problems in vision. Typically, the use of graph-cuts is motivated by one of the following two reasons. Firstly, graph-cuts allow geometric interpretation; under certain conditions a cut on a graph can be seen as a hypersurface in N-D space embedding the corresponding graph. Thus, many applications in vision and graphics use min-cut algorithms as a tool for computing optimal hypersurfaces. Secondly, graph-cuts also work as a powerful energy minimization tool for a fairly wide class of binary and non-binary energies that frequently occur in early vision. In some cases graph cuts produce globally optimal solutions. More generally, there are iterative techniques based on graph-cuts that produce provably good approximations which (were empirically shown to) correspond to high-quality solutions in practice. Thus, another large group of applications use graph-cuts as an optimization technique for low-level vision problems based on global energy formulations.

This chapter is intended as a tutorial illustrating these two aspects of graph-cuts in the context of problems in computer vision and graphics. We explain general theoretical properties that motivate the use of graph cuts, as well as show their limitations.

5.1 Introduction

Graph cuts remain an area of active research in the vision and graphics communities. Besides finding new applications, in the last years researchers have discovered and rediscovered interesting links connecting graph cuts with other combinatorial algorithms (dynamic programming, shortest paths [107, 477]), Markov random fields, statistical physics, simulated annealing and other regularization techniques [362, 113, 424], sub-modular functions [491], random walks and electric circuit theory [356, 357], Bayesian networks and belief propagation [790], integral/differential geometry, anisotropic diffusion, level sets and other variational methods [767, 109, 28, 477].

Graph cuts have proven to be a useful multidimensional optimization tool which can enforce piecewise smoothness while preserving relevant sharp discontinuities. This paper is mainly intended as a survey of existing literature and a tutorial on graph cuts in the context of vision and graphics. We present some basic background information on graph cuts and discuss major theoretical results, some fairly new and some quite old, that helped to reveal both strengths and limitations of these surprisingly versatile combinatorial algorithms. This chapter does not provide any new research results, however, some applications are presented from a point of view that may differ from the previous literature.

The organization of this chapter is as follows. Chapter 5.2 provides necessary background information and terminology. In their core, combinatorial min-cut/max-flow algorithms are binary optimization methods. Chapter 5.3 presents a simple binary problem that can help to build basic intuition on using graph cuts in computer vision. Then, graph cuts are discussed as a general tool for exact minimization of certain binary energies.

Most publications on graph cuts in vision and graphics show that, despite their binary nature, graph-cuts offer significantly more than "binary energy minimization". Chapter 5.4 shows that graph cuts provide a viable geometric framework for approximating continuous hypersurfaces on N-dimensional manifolds. This geometric interpretation of graph cuts is widely used in applications for computing globally optimal separating hypersurfaces. Finally, Chapter 5.5 presents generalized (non-binary) graph cuts techniques applicable to exact or approximate minimization of multi-label energies. In the last decade, such non-binary graph cut methods helped to significantly raise the bar for what is considered a good quality solution in many early vision problems.

5.2 Graph Cuts Basics

First, we introduce some basic terminology. Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph which consists of a set of nodes $\mathcal{V}$ and a set of directed edges $\mathcal{E}$ that connect them. The nodes set $\mathcal{V} = \{s, t\} \cup \mathcal{P}$ contains two special *terminal* nodes, which are called the *source*, s, and the *sink*, t, and a set of non-terminal nodes $\mathcal{P}$. In Figure 5.1(a) we show a simple example of a graph with the terminals s and t. Such N-D grids are typical for applications in vision and graphics.

Each graph edge is assigned some nonnegative weight or cost $w(p, q)$. A cost of a directed edge (p, q) may differ from the cost of the reverse edge (q, p). An edge is called a *t-link* if it connects a non-terminal node in $\mathcal{P}$ with a terminal. An edge is called a *n-link* if it connects two non-terminal nodes. A set of all (directed) n-links will be denoted by $\mathcal{N}$. The set of all graph edges $\mathcal{E}$ consists of n-links in

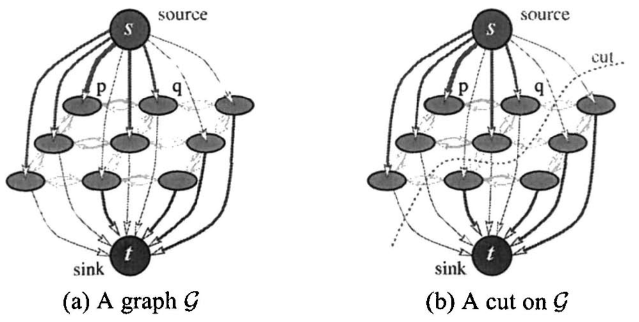


Figure 5.1. Graph construction in Greig et. al. [362]. Edge costs are reflected by thickness.

$\mathcal{N}$ and t-links $\{(s,p),(p,t)\}$ for non-terminal nodes $p \in \mathcal{P}$. In Figure 5.1 t-links are shown in red and blue, while n-links are shown in yellow.

5.2.1 The Min-Cut and Max-Flow Problem

An s/t cut C (sometimes we just call it a *cut*) is a partitioning of the nodes in the graph into two disjoint subsets $\mathcal{S}$ and $\mathcal{T}$ such that the source s is in $\mathcal{S}$ and the sink t is in $\mathcal{T}$. Figure 5.1(b) shows one example of a cut. The cost of a cut $C = \{\mathcal{S},\mathcal{T}\}$ is the sum of costs/weights of "boundary" edges (p,q) such that $p \in \mathcal{S}$ and $q \in \mathcal{T}$. If (p,q) is a boundary edge, then we sometimes say that cut C severs edge (p,q). The *minimum cut* problem is to find a cut that has the minimum cost among all cuts.

One of the fundamental results in combinatorial optimization is that the minimum s/t cut problem can be solved by finding a *maximum flow* from the source s to the sink t. Speaking informally, maximum flow is the maximum "amount of water" that can be sent from the source to the sink by interpreting graph edges as directed "pipes" with capacities equal to edge weights. The theorem of Ford and Fulkerson [324] states that a maximum flow from s to t saturates a set of edges in the graph dividing the nodes into two disjoint parts $\{\mathcal{S},\mathcal{T}\}$ corresponding to a minimum cut. Thus, min-cut and max-flow problems are equivalent. In fact, the maximum flow value is equal to the cost of the minimum cut.

5.2.2 Algorithms for the Min-Cut and Max-Flow Problem

There are many standard polynomial time algorithms for min-cut/max-flow[217]. These algorithms can be divided into two main groups: "push-relabel" style methods [350] and algorithms based on augmenting paths. In practice the push-relabel algorithms perform better for general graphs. In vision applications, however, the most common type of a graph is a two or a higher dimensional grid. For the grid graphs, Boykov and Kolmogorov [110] developed a fast augmenting path algorithm which often significantly outperforms the push relabel algorithm. Furthermore, its observed running time is linear.

While the (sequential) algorithm in [110] is very efficient, with the execution time of only a few seconds for a typical problem, it is still far from real time. A possible real time solution may come from a GPU acceleration that has become popular for improving the efficiency of algorithms allowing parallel implementations on pixel level. Note that push-relabel algorithm can be run in parallel over graph nodes [350]. In the context of image analysis problems, graph nodes typically correspond to pixels. Thus, pixel based GPU architecture is a seemingly perfect match for accelerating push-relabel algorithm for computing graph cuts in vision and graphics. This is a very promising direction for getting applications of graph cuts up to real time.

5.3 Graph Cuts for Binary Optimization

In this section we concentrate on graph cuts as a binary optimization tool. In fact, min-cut/max-flow algorithms are inherently binary techniques, and so binary problems constitute the most basic case for graph cuts. In Section 5.3.1 we discuss the earliest known example where graph cuts were used in vision, which also happens to be a particularly clear binary problem. The example illustrates that graph cuts can effectively enforce spatial coherence on images. Section 5.3.2 presents the general case of binary energy minimization with graph cuts.

5.3.1 Example: Binary Image Restoration

The earliest use of graph cuts for energy minimization in vision is due to Greig et.al. [362]. They consider the problem of binary image restoration. Given a binary image corrupted by noise, the task is to restore the original image. This problem can be formulated as a simple optimization over binary variables corresponding to image pixels. In particular, [362] builds a graph shown in Figure 5.1(a) where non-terminal nodes $p \in \mathcal{P}$ represent pixels while terminals s and t represent two possible intensity values. To be specific, source s will represent intensity 0 and sink t will represent intensity 1. Assume that $I(p)$ is the observed intensity at pixel p. Let $D_p(l)$ be a fixed penalty for assigning to pixel p some "restored intensity" label $l \in \{0, 1\}$. Naturally, if $I(p) = 0$ then $D_p(0)$ should be smaller than $D_p(1)$, and vice versa. To encode these "observed data" constraints, we create two t-links for each pixel node in Figure 5.1. The weight of t-link (s, p) is set to $D_p(1)$ and the weight of (p, t) is set to $D_p(0)$. Even though t-link weights should be non-negative, the restriction $D_p \geq 0$ for data penalties is not essential.

Now we should add regularizing constraints that help to remove image noise. Such constraints enforce spatial coherence between neighboring pixels by minimizing discontinuities between them. In particular, we create n-links between neighboring pixels using any (e.g. 4- or 8-) neighborhood system. The weight of these n-links is set to a smoothing parameter $\lambda > 0$ that encourages a minimum cut to sever as few n-links as possible.

Remember that a cut C (Figure 5.1(b)) is a binary partitioning of the nodes into subsets $\mathcal{S}$ and $\mathcal{T}$. A cut can be interpreted as a binary labeling f that assigns labels $f_p \in \{0,1\}$ to image pixels: if $p \in \mathcal{S}$ then $f_p = 0$ and if $p \in \mathcal{T}$ then $f_p = 1$. Obviously, there is a one-to-one correspondence between cuts and binary labelings of pixels. Each labeling f gives a possible image restoration result.

Consider the cost of an arbitrary cut $C = \{\mathcal{S}, \mathcal{T}\}$. This cost includes weights of two types of edges: severed t-links and severed n-links. Note that a cut severs exactly one t-link per pixel; it must sever t-link (p,t) if pixel p is in the source component $p \in \mathcal{S}$ or t-link (s,p) if pixel p is in the sink component $p \in \mathcal{T}$. Therefore, each pixel p contributes either $D_p(0)$ or $D_p(1)$ towards the t-link part of the cut cost, depending on the label f_p assigned to this pixel by the cut. The cut cost also includes weights of severed n-links $(p,q) \in \mathcal{N}$. Therefore,

$$|C| = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\substack{(p,q) \in \mathcal{N} \\ p \in \mathcal{S}, q \in \mathcal{T}}} w(p,q)$$

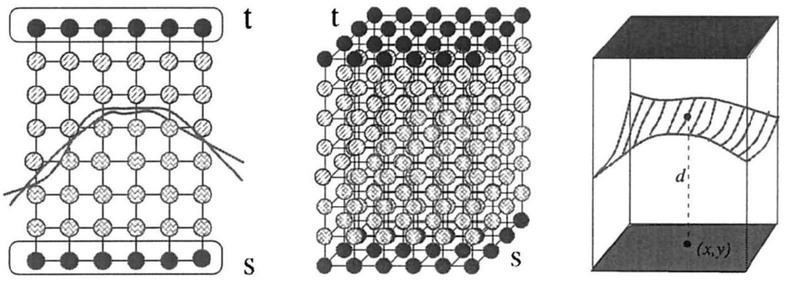
The cost of each C defines the "energy" of the corresponding labeling f:

$$E(f) := |C| = \sum_{p \in \mathcal{P}} D_p(f_p) \;+\; \lambda \cdot \sum_{(p,q) \in \mathcal{N}} \mathcal{I}(f_p = 0, f_q = 1), \qquad (5.1)$$

where $\mathcal{I}(\cdot)$ is the identity function giving 1 if its argument is true and 0 otherwise. Stated simply, the first term says that pixel labels f_p should agree with the observed data while the second term penalises discontinuities between neighboring pixels. Obviously, a minimum cut gives labeling f that minimizes energy (5.1).

Note that parameter λ controls the relative importance of the data constraints versus the regularizing constraints. Note that if λ is very small, an optimal labeling assigns each pixel p a label f_p that minimizes its own data cost $D_p(f_p)$. In this case, each pixel chooses its own label independently from the other pixels. If λ is big, then all pixels must choose one label that has a smaller average data cost. For intermediate values of λ, an optimal labeling f should correspond to a balanced solution with compact spatially coherent clusters of pixels who generally like the same label. Noise pixels, or outliers, should conform to their neighbors.

Before [362], exact minimization of energies like (5.1) was not possible. Researches still used them, but had to approach them with iterative algorithms like simulated annealing [341]. In fact, Greig et.al. published their result mainly to show that in practice simulated annealing reaches solutions very far from the global minimum even in simple binary cases. Unfortunately, the result of Greig et.al. remained unnoticed in the vision community for almost 10 years probably because the binary image restoration looked too restrictive as an application.



(a) a cut on a 2D grid (b) a cut and a separating hypersurface in 3D

Figure 5.2. s-t cut on a grid corresponds to binary partitioning of N-D space where the grid is embedded. Such space partitioning may be visualized via a separating hypersurface. As shown in (a), multiple hypersurfaces may correspond to the same cut. However, such hypersurfaces become indistinguishable as the grid gets finer.

5.3.2 General Case of Binary Energy Minimization

In general, graph construction as in Figure 5.1 can be used for other binary "labeling" problems. Suppose we are given a penalty $D_p(l)$ that pixel p incurs when assigned label $l \in \mathcal{L} = \{0, 1\}$ and we need to find a spatially coherent binary labeling of the whole image. We may wish to enforce spatial regularization via some global energy function that generalizes (5.1)

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \quad + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{5.2}$$

The question is: can we find a globally optimal labeling f using some graph cut construction? There is a definitive answer to this question for the case of binary labelings. According to [491], a globally optimal binary labeling for (5.2) can be found via graph cuts if and only if the pairwise interaction potential V_{pq} satisfies

$$V_{pq}(0,0) + V_{pq}(1,1) \leq V_{pq}(0,1) + V_{pq}(1,0)$$

which is called the regularity condition. The theoretical result in [491] is constructive and they show the corresponding graph. It has the same form as the graph of Greig et.al. in Figure 5.1, however, edge weights are derived differently.

5.4 Graph Cuts as Hypersurfaces

Solution of many problems in vision, image processing and graphics can be represented in terms of optimal hypersurfaces. This section describes a geometric interpretation of graph-cuts as hypersurfaces in N-D manifolds that makes them an attractive framework for problems like image segmentation, restoration, stereo, photo/video editing, texture synthesis, and others.

We show a basic idea allowing s-t cuts to be viewed as hypersurfaces, discuss interesting theories that make various connections between discrete graph cuts and hypersurfaces in continuous spaces, and we also provide a number of recently published examples where a hypersurface view of graph cuts has led to interesting applications in computer vision, medical imaging, or graphics.

5.4.1 Basic idea

Consider two simple examples in Figure 5.2. Throughout Section 5.4 we assume that a graph has no "soft" t-links, that is the source and the sink terminals are directly connected only to some of the graph nodes via infinity cost t-links. In fact, all nodes hardwired to two terminals can be effectively treated as multiple sources and multiple sinks that have to be separated by a cut. Figure 5.2 shows these sources and sinks in dark red and dark blue colors. Such sources and sinks provide hard constraints or boundary conditions for graph cuts; any feasible cut must separate sources from sinks. Other nodes are connected to the sources and sinks via n-links.

Without loss of generality (see Section 5.4.2), we can concentrate on feasible cuts that partition the simple 4- and 6- nearest neighbor grid-graphs in Figure 5.2 into two connected subsets of nodes: source component and sink component. Continuous 2D and 3D manifolds where the grid nodes are embedded can be split into two disjoint contiguous regions, one containing the sinks, and the other containing the sources. A boundary between two such regions are separating hypersurfaces shown in green color. As illustrated in Figure 5.2(a), there are many separating hypersurfaces that correspond to the same cut. They should all correctly separate the grid nodes of the source and the sink components, but they can "freely move" in the space between the grid nodes. Without getting into mathematical details, we will identify a class of all hypersurfaces corresponding to a given cut with a single hypersurface. In particular, we can choose a hypersurface that follows boundaries of "grid cells", or we can choose "the smoothest" hypersurface. Note that the finer the grid, the harder it is to distinguish two separating hypersurfaces corresponding to the same cut.

Thus, any feasible cut on a grid in Figure 5.2 corresponds to a separating hypersurface in the embedding continuous manifold. Obviously, the opposite is also true; any separating hypersurface corresponds to a unique feasible cut. Generalization of examples in Figure 5.2 would establish correspondence between $s - t$ graph-cuts and separating hypersurfaces in case of "fine" locally connected grids embedded in N-D spaces. Following ideas in [109], one can set a cost (or area) of each continuous hypersurface based on the cost of the corresponding cut. This defines a *cut metric* introduced in [109] for continuous N-D manifold embedding a graph. By changing weights of n-links at graph nodes located in any particular point in space, one can tune local costs of all separating hypersurfaces that pass through such locations. In practical applications a cut metric can be easily tuned to attract (repel) hypersurfaces to (from) certain locations on N-D manifolds. A cut metric is a simple, yet sufficiently general tool. In particular, according to

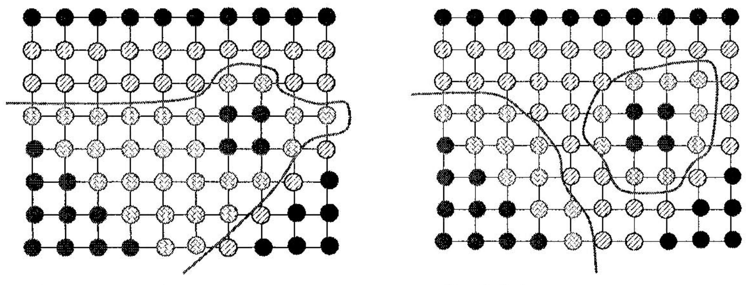


Figure 5.3. Separating hypersurfaces can have different topological properties for the same set of hard constraints. Separating hypersurfaces in (a) and (b) correspond to two distinct feasible $s - t$ cuts. Min-cut/max-flow algorithms compute a globally optimal hypersurface/cut without any restrictions on its topological properties as long as the sources and the sinks are separated.

[109] a cut metric on 2D and 3D manifolds can approximate any given continuous Riemannian metric. Finally, standard combinatorial algorithms for computing minimum cost $s - t$ cuts (see Section 5.2.2) become numerical tools for extracting globally optimal separating hypersurfaces.

5.4.2 Topological properties of graph cuts

The adjective "separating" implies that a hypersurface should satisfy certain hard constraints or boundary conditions; it should separate source and sink grid cells (seeds). Note that there are many freedoms in setting boundary conditions for graph cuts. Depending on hard constraints, topological properties of separating hypersurfaces corresponding to $s - t$ cuts may vary.

For example, we can show that the boundary conditions in Figure 5.2 guarantee that any feasible cut corresponds to topologically connected separating hypersurface. For simplicity, we assume that our graphs are connected, that is, there are no "islands" of disconnected nodes. In Figure 5.2 all source and all sink nodes form two connected components. In such cases a minimum cost cut must partition the graph into exactly two connected subsets of nodes; one containing all sources and the other containing all sinks. Assuming that the minimum cost cut creates three or more connected components implies that some of these components contain neither sources, nor sinks. This contradicts minimality of the cut; linking any "no-source/no-sink" subset back to the graph corresponds to a smaller cost feasible cut.

Examples in Figure 5.3 illustrate different topological properties for separating hypersurfaces in more general cases where multiple disjoint components of sources and sinks (seeds) are present. Note that feasible $s - t$ cuts may produce topologically different separating hypersurfaces for the same set of boundary conditions.

In fact, controlling topological properties of separating hypersurfaces by setting up appropriate hard constraints is frequently a key technical aspect of applications using graph cuts. As discussed in Section 5.4.3, appropriate positioning of sources and sinks is not the only tool to achieve desired topology. As shown in Figure 5.4, certain topological properties of separating hypersurfaces can be enforced via infinity cost n-links.

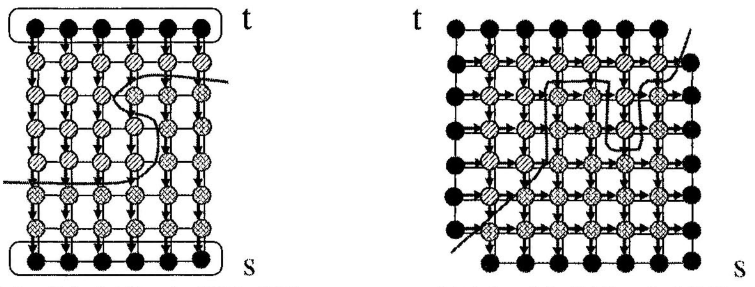
5.4.3 Applications of graph cuts as hypersurfaces

Below we consider several examples from recent publications where graph cuts are used as a method for extracting optimal hypersurfaces with desired topological properties.

Methods for object extraction [107, 96, 683, 903] take full advantage of topological freedom of graph-cut based hypersurfaces. In particular, they allow to segment objects of arbitrary topology. The basic idea is to set as sources (red seeds) some image pixels that are known (a priori) to belong to an object of interest and to set as sinks (blue seeds) some pixels that are known to be in the background. A separating hypersurface should coincide with a desirable object boundary separating object (red) seeds from background (blue) seeds, as demonstrated in Figure 5.3. A cut metric can be set to reflect image gradient. Pixels with a high image gradient would imply a low cost of local n-links and vice versa. Then, minimal separating hypersurfaces tend to adhere to object boundaries with high image gradients. Another practical strength of object extraction methods based on graph cuts is that they provide practical solutions for organ extraction problems in N-D medical image analysis [107]. One limitation of this approach to object extraction is that it may suffer from a bias to "small cuts", but this can often be resolved with proper constraining of the solution space.

Stereo was one of the first applications in computer vision where graph cuts were successfully applied as a method for optimal hypersurface extraction. Two teams, Roy&Cox [693, 692] and Ishikawa&Geiger [425], almost simultaneously proposed two different formulations of the stereo problem where disparity maps are interpreted as separating hypersurfaces on certain 3D manifolds. Their key technical contribution was to show that disparity maps (as optimal hypersurfaces) can be efficiently computed via graph cuts.

For example, Roy&Cox [693, 692] proposed a framework for stereo where disparity maps are separating hypersurfaces on 3D manifolds similar to one in Figure 5.2(b). Points of this bounded rectangular manifold are interpreted as points in 3D "disparity space" corresponding to a pair of rectified stereo images. This disparity space is normally chosen with respect to one of the images, so that each 3D point with coordinates (x, y, d) represents correspondence between pixel (x, y) in the first stereo image and pixel $(x + d, y)$ in the second image. Then, solution of stereo problem is a hypersurface $d = f(x, y)$ on 3D manifold in Figure 5.2(b) that represents a disparity map assigning certain disparity d to each pixel (x, y) in the first image. Note that hypersurface $d = f(x, y)$ separates the bottom and the top (facets) of 3D manifold in Figure 5.2(b). Then, an optimal



(a) Infeasible folding in [693, 692] (b) Infeasible folding in [425]

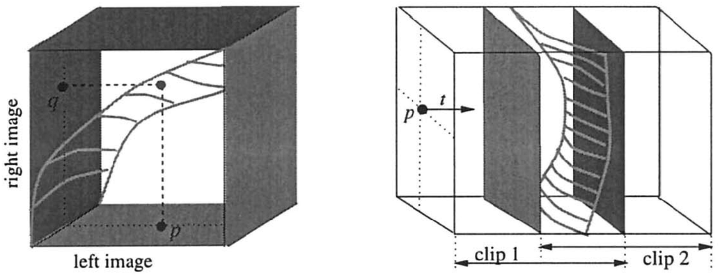
Figure 5.4. Graph-cuts approach allows to impose certain additional topological constraints on separating hypersurfaces, if necessary. For example, [426, 111] proposed infinity cost directed n-links, shown in brown color in (a), that forbid folds on separating hypersurfaces in Figure 5.2. In particular, a hypersurface in Figure 5.2(b) without such folds corresponds to a disparity map $d = f(x, y)$ according to [693, 692]. Also, [425] impose monotonicity/ordering constraint on their disparity maps by adding infinity cost directed n-links (in brown color) that make illegal topological folds shown in (b). For clarity, examples in (a) and (b) correspond to single slices of 3D manifolds in Figure 5.2(b) and 5.5(a).

disparity map can be computed using graph cuts as an efficient discrete model for extracting minimal separating hypersurfaces.

According to [693], cut metric on 3D "disparity space" manifold in Figure 5.2(b) is set based on color consistency constraint between two stereo cameras. Weights of n-links at node (x, y, d) are set as follows: if intensities of pixels (x, y) and $(x + d, y)$ in two cameras are similar then the likelihood that two pixels see the same 3D object point is high and the cost of n-links should be small. Later, [426, 692, 111] suggested anisotropic cut metric where vertical n-links are based on the same likelihoods as above but horizontal n-links are fixed to a constant encouraging smoother disparity maps that avoid unnecessary disparity level jumps.

In general, separating hypersurfaces in Figure 5.2(b) can have folds that would make them inappropriate as disparity maps $d = f(x, y)$. If a minimum hypersurface computed via graph cuts has a fold then we did not find a feasible disparity map. Therefore, [426, 111] propose a set of hard constraints that make topological folds (see Figure 5.4(a)) prohibitively expensive. Note that additional infinity cost vertical n-links (directed down) make folds infeasible. This topological hard constraint takes advantage of the "directed" nature of graph cuts; a cost of a cut includes only severed directed edges that go from the (red) nodes in the source component to the (blue) nodes in the sink component. A cut with an illegal fold in Figure 5.4(a) includes one infinity cost n-link.

Ishikawa&Geiger [425] also solve stereo by computing optimal separating hypersurfaces on a rectangular 3D manifold. However, their interpretation of the manifold and boundary conditions are different. As shown in Figure 5.5(a), they interpret a separating hypersurface $z = f(x, y)$ as a "correspondence mapping"



(a) Hypersurface as correspondence (b) Hypersurface separates two video clips

Figure 5.5. Two more examples of graph cuts as separating hypersurfaces. Formulation of stereo problem in [425] computes pixel correspondences represented by a separating hypersurface on a 3D manifold in (a). A smooth transition between two video clips is performed in [499] via graph cuts computing globally optimal separating hypersurface in a 3D region of overlap between two clips in (b).

between pixels $p = (x, y)$ in the left image and pixels $q = (f(x, y), y)$ in the right image (of a rectified stereo pair). Assignment of correspondences may be ambiguous if a hypersurface has folds like one in Figure 5.4(b). In order to avoid ambiguity, [425] introduce monotonicity (or ordering) constraint that is enforced by directed infinity cost n-links shown in brown color. Note that a cut in Figure 5.4(b) severs two brown n-links that go from a (red) node in a source component to a (blue) node in a sink component. Thus, the cost of the cut is infinity and the corresponding separating hypersurface with a fold becomes infeasible.

Similar to [693, 692], the cut metric on manifold in Figure 5.5(a) is based on color consistency constraint: a 3D points (x, y, z) on the manifold has low n-link costs if intensity of pixel (x, y) in the left image is close to intensity of pixel (z, y) in the right image. Note that hyperplanes parallel to diagonal crossection (from bottom-left to top-right corners) of manifold in Figure 5.5(a) give correspondence mappings with constant stereo disparity/depth levels. Thus, spatial consistency of disparity/depth map can be enforced with anisotropic cut metric where diagonal n-links (from left-bottom to right-top corner) are set to a fixed constant representing penalty for jumps between disparity levels.

Another interesting example of graph-cuts/hypersurface framework is a method for video texture synthesis in [499]. The technique is based on computing a seamless transition between two video clips as illustrated in Figure 5.5(b). Two clips are overlapped in 3D (pixel-time) space creating a bounded rectangular manifold where transition takes place. A point in this manifold can be described by 3D coordinates (x, y, t) where $p = (x, y)$ is a pixel and t is time or video frame number. The transition is represented by a separating hypersurface $t = f(x, y)$ that specifies for each pixel when to switch from clip 1 to clip 2. During transition a frame may have a mix of pixels from each clip. The method in [499] suggest a specific cut metric that for each point (x, y, t) in the overlap region depends on intensity difference between two clips. Small difference indicates a good moment (in space and time) for seamless transition between the clips and n-links at such (x, y, t) points are assigned a low cost. Note that "seamless transition" is a purely visual effect and it may be achieved with any separating hypersurface in Figure 5.5(b). In this case there is no real need to avoid hypersurfaces with "folds" which would simply allow pixels to switch between clip 1 and clip 2 a few times.

5.4.4 Theories connecting graph-cuts and hypersurfaces in R^n

In this section we discuss a number of known results that established theoretically solid connections between cuts on discrete graphs and hypersurfaces in continuous spaces. It has been long argued in computer vision literature that discrete algorithms on graphs, including graph cuts, may suffer from metrication artifacts. Indeed, 4- and 6- nearest neighbor connections on 2D and 3D grids may produce "blocky" segments. Such geometric artifacts are due to "Manhattan distance" metrication errors. It turns out that such errors can be easily corrected, resolving the long-standing criticism of graph cuts methods. Boykov&Kolmogorov [109] showed that regular grids with local neighborhood systems of higher order can produce a cut metric that approximates any continuous Riemannian metric with arbitrarily small error. Using powerful results from integral geometry, [109] shows that weights of n-links from a graph node embedded at point p of continuous N-D manifold are solely determined by a given $N \times N$ positive-definite matrix $D(p)$ that defines local metric/distance properties at point p according to principles of Riemannian geometry. This result is quite intuitive as weights of n-links at this graph node define local measure for area/distance for hypersurfaces according to the corresponding cut metric. It is also interesting that results in [109] apply to arbitrary Riemannian metrics including anisotropic cases where local metric could be direction-sensitive.

So far in Section 5.4 we followed the general approach of [109] where hypersurfaces on N-D manifolds have implicit representation via cuts on embedded graphs. As illustrated in Figure 5.2, a cut only "implies" a separating hypersurface. A specific hypersurface can be obtained through additional conventions, as discussed in Section 5.4.1. More recently, [477] proposed an explicit approach to hypersurface representation by graph cuts that, in a way, is dual to [109]. The basic idea in [477] is to bisect a bounded N-D manifold with a large number of (random) hyperplanes. These hyperplanes divide the manifold into small cells (polyhedra) which can be thought of as irregular voxels. Then, [477] build an irregular "random-grid" graph where each cell is represented by a node. Two cells are connected by an n-link if and only if they touch through a common facet. Clearly, there is a one-to-one correspondence between a set of all n-links on the graph and a set of all facets between cells. A cut on this graph explicitly represents a unique hypersurface formed by facets corresponding to severed n-links. Obviously, a cost of any cut will be equal to the area of the corresponding hypersurface (in any metric) if weights of each n-link is equal to the area of the corresponding facet (in that metric). Thus, the model for representing hypersurfaces via graph-cuts in [477] can be applied to any metric. In their case, min-cut/max-flow algorithms will compute a minimum separating hypersurface among all explicitly represented hypersurfaces satisfying given boundary conditions.

Cuts on a graph in [477] represent only a subset of all possible hypersurfaces on an embedding manifold. If one keeps bisecting this bounded manifold into finer cells then the number of representable hypersurfaces increases. [477] proves that bisecting the manifold with a countably infinite number of random hyperplanes would generate small enough cells so that their facets can represent any continuous[1] hypersurface with an arbitrarily small error. This demonstrates that their approach to graph-cut/hypersurface representation is also theoretically solid.

Intuitively speaking, theoretical results in [109] and [477] imply that both approaches to representing continuous hypersurfaces via discrete graph cuts models have reasonable convergence properties and that minimum cost cuts on finer graphs "in the limit" produce a minimum separating hypersurfaces for any given metric. Results such as [109] and [477] also establish a link between graph cuts and variational methods such as level-sets [729, 616, 702, 617] that are also widely used for image segmentation.

There is (at least) one more interesting theoretical result linking graph cuts and hypersurfaces in continuous spaces that is due to G. Strang [767]. This result was established more than 20 years ago and it gives a view somewhat different from [109, 477]. Strang describes a continuous analogue of the min-cut/max-flow paradigm. He shows that maximum flow problem can be redefined on a bounded continuous domain Ω in the context of a vector field $\bar{f}(p)$ representing the speed of a continuous stream/flow. A constraint on discrete graph flow that comes from edge capacities is replaced by a "speed limit" constraint $|\bar{f}(p)| \leq c(p)$ where c is a given non-negative scalar function[2]. Discrete flow conservation constraint for nodes on a graph has a clear continuous interpretation as well: a continuous stream/flow is "preserved" at points inside the domain if vector field $\bar{f}$ is divergence-free $div \bar{f} = 0$. Strang also gives appropriate definition for sources and sinks on the boundary of the domain[3]. Then, the continuous analogue of the maximum flow problem is straightforward: find a maximum amount of water that continuous stream $\bar{f}$ can take from sources to sinks across the domain while satisfying all the constraints.

The main topic of this sections connects to [767] as follows. Strang defines a "real" cut on Ω as a hypersurface γ that divides the domain into two subsets. The minimum cut should separate sources and sinks and have the smallest possible cost $\int_\gamma c$ which can be interpreted as a length of hypersurface γ in isotropic metric defined by a scalar function c. Strang also establishes duality between continuous versions of minimum cut and maximum flow problems that is analogous to the discrete version established by Ford and Fulkerson [324]. On a practical note,

[1]piece-wise twice differentiable, see [477] for more details.

[2]More generally, it is possible to set an anisotropic "speed limit" constraint $\bar{f}(p) \in c(p)$ where c is some convex set defined at every point $p \in \Omega$.

[3]Sources and sinks can also be placed inside the domain. They would correspond to points in Ω where $div \bar{f}$ is non-null, t.e. where stream $\bar{f}$ has an in-flow or out-flow.

a recent work by Appleton&Talbot [28] proposed a finite differences approach that, in the limit, converges to a globally optimal solution of continuous min-cut/max-flow problem defined by Strang. Note, however, that they use graph cuts algorithms to "greatly increase the speed of convergence".

5.5 Generalizing Graph Cuts for Multi-Label Problems

In this section, we show that even though graph cuts provide an inherently binary optimization, they can be used for multi-label energy minimization. In some cases, minimization is exact, but in more interesting cases only approximate minimization is possible. There is a direct connection between the exact multi-label optimization and a graph cut as a hypersurface interpretation of Section 5.4. We begin by stating the general labeling problem, then in Section 5.5.1 we describe the case when optimization can be performed exactly. Finally, Section 5.5.2 describes the approximate minimization approaches and their quality guarantees.

Many problems in vision and graphics can be naturally formulated in terms of multi-label energy optimization. Given a set of sites $\mathcal{P}$ which represent pixels/voxels, and a set of labels $\mathcal{L}$ which may represent intensity, stereo disparity, a motion vector, etc., the task is to find a labeling f which is a mapping from sites $\mathcal{P}$ to labels $\mathcal{L}$. Let f_p be the label assigned to site p and f be the collection of such assignments for all sites in $\mathcal{P}$.

We can use the same general form of energy (5.2) that was earlier introduced in the context of binary labeling problems. The terms $D_p(l)$ are derived from the observed data and it expresses the label preferences for each site p. The smaller the value of $D_p(l)$, the more likely is the label l for site p. Since adding a constant to $D_p(l)$ does not change the energy formulation, we assume, without loss of generality, that $D_p(l)$'s are nonnegative. The pairwise potential $V_{pq}(l_p, l_q)$ expresses prior knowledge about the optimal labeling f. In general, prior knowledge can be arbitrarily complex, but in graph cuts based optimization, we are essentially limited to different types of spatial smoothness priors. Typically $V_{pq}(l_p, l_q)$ is a nondecreasing function of $||l_p - l_q||$[4]. Different choices of $V_{pq}(l_p, l_q)$ imply different types of smoothness, see Sections 5.5.1 and 5.5.2 .

5.5.1 *Exact Multi-Label Optimization*

In this section, we describe the only known case of exact multi-label minimization of energy (5.2) via graph cuts. The corresponding graph construction is not covered by the general theoretical result in [491], which applies to binary labeling cases only. We have to make the assumption that labels are linearly ordered. This assumption limits the applicability of the method. For example, it cannot be directly used for motion estimation, since motion labels are 2 dimensional and

[4]Here we used the norm $||\cdot||$ notation because, in general, l_p may be a vector

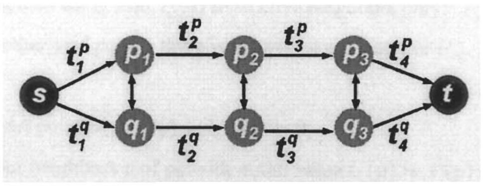


Figure 5.6. Part of the graph construction for energy minimization in 5.3 , $|\mathcal{L}| = 4$

cannot be linearly ordered[5]. Without loss of generality, assume that labels are integers in the range $\mathcal{L} = \{1, ..., k\}$. Let $V_{pq} = \lambda_{pq}|f_p - f_q|$. Then the energy is:

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} \lambda_{pq}|f_p - f_q|, \tag{5.3}$$

In vision, [425, 111] were the first to minimize energy (5.3) with a minimum cut on a certain graph $\mathcal{G}$. In fact, this graph is topologically similar to a graph of Roy&Cox [693] where separating hypersurface on 3D manifold gives a stereo disparity map, see Section 5.4.3.

The graph is constructed as follows. As usual, vertices $\mathcal{V}$ contain terminals s and t. For each site p, create a set of nodes $p_1, ..., p_{k-1}$. Connect them with edges $\{t_1^p, ..., t_k^p\}$, where $t_1^p = (s, p_1)$, $t_j^p = (p_{j-1}, p_j)$, and $t_k^p = (p_{k-1}, t)$. Each edge t_j^p has weight $K_p + D_p(j)$, where $K_p = 1 + (k-1)\sum_{q \in \mathcal{N}_p} \lambda_{pq}$. Here $\mathcal{N}_p$ is the set of neighbors of p . For each pair of neighboring sites p, q and for each $j \in \{1, ..., k-1\}$, create an edge (p_j, q_j) with weight λ_{pq}. Figure 5.6 illustrates the part of $\mathcal{G}$ which corresponds to two neighbors p and q. For each site p, a cut on $\mathcal{G}$ severs at least one edge t_i^p. The weights for t_i^p are defined sufficiently large so that the minimum cut severs exactly one of them for each p. This establishes a natural correspondence between the minimum cut and an assignment of a label to p. If the minimum cut severs edge t_i^p, assign label i to p. It is straightforward to show that the minimum cut corresponds to the optimum f [111].

Ishikawa [424] generalized the above construction to minimize any energy function with convex V_{pq}'s. His construction is similar to the one in this section, except even more edges between p_i's and q_j's have to be added. Unfortunately, a convex V_{pq} is not suitable for the majority of vision applications, especially if the number of labels is large. Typically, object properties tend to be smooth everywhere except the object boundaries, where discontinuities may be present. Thus in vision, a piecewise smooth model is more appropriate than the everywhere smooth model. However using a convex V_{pq} essentially corresponds to the everywhere smooth model. The penalty that a convex V_{pq} imposes on a sharp jumps in labels is so large, that in the optimal f discontinuities are smoothed out with a "ramp". It is much cheaper to create a few small jumps in f rather than one large jump.

[5]Iterative application of the algorithm described here was used for motion in [694]

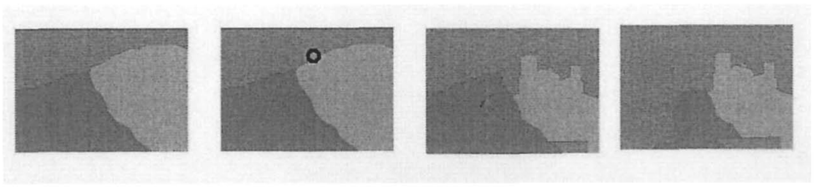


Figure 5.7. From left to right: a labeling f, a labeling within one standard move of f (the changed site is highlighted by a black circle), labeling within one green-yellow swap of f, labeling within one green expansion of f.

Of all the convex V_{pq}, the one in (5.3) works best for preserving discontinuities. Nevertheless in practice, it oversmooths disparity boundaries [837].

5.5.2 *Approximate Optimization*

The potential V_{pq} in the previous section is not discontinuity preserving because V_{pq} is allowed to grow arbitrarily large. One way to construct a discontinuity preserving V_{pq} is to cap its maximum value. Perhaps the simplest example is the Potts model $V_{pq} = \lambda_{pq} \cdot \mathcal{I}(f_p \neq f_q)$ [113]. We have already seen Potts V_{pq} in Section 5.3.1[6], and it corresponds to the piecewise constant prior on f. Unfortunately, energy minimization with Potts V_{pq} is NP-hard [113], however graph cuts can be used to find an answer within a factor of 2 from the optimum [113].

In this section, we describe two approximation methods, the expansion and the swap algorithms [113]. According to the results in [491], the swap algorithm may be used whenever $V_{pq}(\alpha,\alpha) + V_{pq}(\beta,\beta) \leq V_{pq}(\alpha,\beta) + V_{pq}(\beta,\alpha)$ for all $\alpha,\beta \in \mathcal{L}$, which we call the swap inequality. The expansion algorithm may be used whenever $V_{pq}(\alpha,\alpha)+V_{pq}(\beta,\gamma) \leq V_{pq}(\alpha,\gamma)+V_{pq}(\beta,\alpha)$ for all $\alpha,\beta,\gamma \in \mathcal{L}$, which we call the expansion inequality. Any V_{pq} which satisfies the expansion inequality also satisfies the swap inequality, hence the expansion inequality is more restrictive.

Both swap and expansion inequalities admit discontinuity preserving V_{pq}'s. The truncated linear $V_{pq}(\alpha,\beta) = min(T, ||\alpha - \beta||)$ satisfies the expansion inequality. The truncated quadratic $V_{pq}(\alpha,\beta) = min(T, ||\alpha - \beta||^2)$ satisfies the swap inequality. Here T is a positive constant, which is the maximum penalty for a discontinuity. The truncated linear and truncated quadratic V_{pq} correspond to a piecewise smooth model. Small deviations in labels incur only a small penalty, thus the smoothness is encouraged. However sharp jumps in labels are occasionally permitted because the penalty T is not too severe to prohibit them.

[6]In the binary case, it is typically called the Ising model.

5.5.2.1 Local Minimum with Respect to Expansion and Swap Moves

Both the expansion and the swap algorithms find a local minimum of the energy function. However, in discrete optimization, the meaning of "a local minimum" has to be defined. For each f, we define a set of moves M_f. Intuitively, these are the moves to other labelings that are allowed from f. Then we say that f is a local minimum with respect to the set of moves, if for any $f' \in M_f$, $E(f') \geq E(f)$. Most discrete optimization methods (e.g. [341, 81]) use *standard* moves, defined as follows. Let $H(f, f')$ be the number of sites for which f and f' differ. Then for each f, standard moves are $M_f = \{f' | H(f, f') \leq 1\}$. Thus a standard move allows to change a label of only one site in f, and hence $|M_f|$ is linear in the number of sites, making it is easy to find a local minimum with respect to the standard moves. The result, however is very dependent on the initial point since a high dimensional energy has a huge number of such local minima. In particular, the solution can be arbitrarily far from the global minimum.

We now define the swap moves. Given a labeling f and a pair of labels α and β, a move $f^{\alpha\beta}$ is called an α-β swap if the only difference between f and $f^{\alpha\beta}$ is that some sites that were labeled α in f are now labeled β in $f^{\alpha\beta}$, and some sites that were labeled β in f are now labeled α in $f^{\alpha\beta}$. M_f is then defined as the collection of α-β swaps for all pairs of labels $\alpha, \beta \in \mathcal{L}$.

We now define the expansion moves. Given a labeling f and a label α, a move f^{α} is called an α-expansion if the only difference between f and f^{α} is that some sites that were not labeled α in f are now labeled α in f^{α}. M_f is then defined as the collection of α-expansions swaps for all labels $\alpha \in \mathcal{L}$. Figure 5.7 shows an example of standard move versus α-expansion and α-β swap. Notice that a standard move is a special case of an α-expansion and a α-β swap. However there are α-expansion moves which are not α-β swaps and vice versa.

The expansion (swap) move algorithm finds a local minimum with respect to expansion (swap) moves. The number of expansion (swap) moves from each labeling is exponential in the number of sites. Thus direct search for an optimal expansion (swap) move is not feasible. This is where graph cuts are essential. It is possible to compute the optimal α-expansion or the optimal α-β swap with the minimum cut on a certain graph. This is because computing an optimal α-expansion (optimal α-β swap) is a binary minimization problem which happens to be regular [491] when the expansion (swap) inequality holds.

The expansion (swap) algorithms are iterative. We start with an initial labeling f. We then cycle in random order until convergence over all labels $\alpha \in \mathcal{L}$ (pairs of $\alpha, \beta \in \mathcal{L}$), find the optimal f^{α} ($f^{\alpha\beta}$) out of all α-expansions (α-β-swaps), and change current labeling to f^{α} ($f^{\alpha\beta}$). Obviously this cannot lead to an increase in energy, and at convergence we found the local minimum with respect to expansion (swap) moves. Thus the key step is how to find the optimal α-expansion (α-β swap), which is performed by finding a minimum cut on a certain graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The actual graph constructions can be found in [113].

The criteria for a local minimum with respect to the expansions (swaps) are so strong that there are significantly fewer of such minima in high dimensional spaces compared to the standard moves. Thus the energy function at a local minimum is likely to be much lower. In fact, it can be shown that the local minimum with respect to expansion moves is within a constant factor of optimum. The best approximation is in case of the Potts model, where this factor is 2. It is not surprising then that most applications based on graph cuts use the expansion algorithm with the Potts model [111, 88, 489, 490, 895, 499, 521, 403, 10, 900].