# Chapter 6
# Clustering and Segmentation

**Abstract** Clustering algorithms are used to find communities of nodes that all belong to the same group. This grouping process is also known as *image segmentation* in image processing. The clustering problem is also deeply connected to machine learning because a solution to the clustering problem may be used to propagate labels from observed data to unobserved data. In general network analysis, the identification of a grouping allows for the analysis of the nodes within each group as separate entities. In this chapter, we use the tools of discrete calculus to examine both the *targeted clustering problem* (i.e., finding a specific group) and the *untargeted clustering problem* (i.e., discovering all groups). We additionally show how to apply these clustering models to the clustering of higher-order cells, e.g., to cluster edges.

The clustering problem is to assign a set of labels to a set of cells such that all cells assigned to the same label belong to the same *group*. The most common type of cells to cluster are nodes, and so our discussion will be limited to node clustering, except in Sect. 6.4 where we address the clustering of higher-order cells. Clustering appears in several fields of study, including machine learning and image analysis. In the context of image analysis, the clustering problem is often called *image segmentation*. We will generally use the term *clustering* in this chapter unless we specifically discuss image clustering, in which we will use the term *segmentation*.

Formally, the clustering problem may be formulated by assigning elements $b$ of a label set $\mathcal{L}$ to the set of nodes, $\mathcal{V}$. Specifically, the goal of a clustering algorithm is to produce a segmentation function $\sigma : \mathcal{V} \to \mathcal{L}$. Inspired by image analysis, the term *object* will also be used interchangeably with *cluster* to refer to all nodes that are mapped to the same label through $\sigma$.

Clustering algorithms may be broadly categorized along two axes. The first axis ranges from *targeted clustering* to *untargeted clustering*. Targeted clustering algorithms seek to identify a specific set of objects and therefore require some mechanism for training or steering the algorithm to find the desired set of objects. Consequently, targeted clustering algorithms have a specific label set which is drawn

L.J. Grady, J.R. Polimeni, *Discrete Calculus*,
DOI 10.1007/978-1-84996-290-2_6, 

from to assign labels to each node. In contrast, an untargeted clustering algorithm seeks to label nodes as belonging to the same group if the group shares some properties, such as features of the data, or locality or connectivity derived from the graph. An untargeted clustering algorithm typically determines the number of labels to be used in order to satisfy its internal assumptions about node homogeneity, although in some cases the number of labels may be fixed. The traditional conception of *the* clustering problem has been the untargeted clustering problem, but interest has been increasing rapidly in targeted clustering algorithms.

All of the algorithms described here model the clustering problem with an objective function (often viewed as an energy function), and the purpose of the clustering algorithm is to find the clustering which optimizes the objective. However, some objective functions are difficult to optimize completely, or even fall into the class of NP-Hard problems. Consequently, the clustering calculated as a solution to one of these difficult objective functions may depend on the initialization to the algorithm. In these cases, a clustering algorithm, which would be untargeted if the objective could be completely optimized, can act as a targeted algorithm by initializing the clustering close to an intended target. We term algorithms of this variety as *semi-targeted* clustering, and treat these algorithms in a separate section.

The second axis for categorizing clustering algorithms ranges from *primal* algorithms (in which the nodes within a cluster are directly labeled) to *dual* algorithms (in which the cells comprising the boundaries of clusters are labeled). Primal algorithms are generally more popular, better developed, and easier to generalize. Dual algorithms work only under limited circumstances and depend on an embedding, but they constitute an important class of algorithms in image analysis that are of a fundamentally different character than the primal algorithms.

Several clustering algorithms in this chapter assume that each node is associated with some data (i.e., an attributed graph) and it is this data which is to be clustered. However, the predominant methodology is to transform this data into graph structure via the weights (using the functions in Chap. 4) and then apply the clustering algorithms to the weighted graph. Consequently, the clustering algorithms utilize both the network connectivity structure and the weights to determine a clustering and may therefore be applied to produce a clustering of any general network (even when the nodes are not associated with data).

This chapter is organized to consist of three main sections that describe the targeted, untargeted, and semi-targeted classes of clustering algorithms, with subsections detailing the primal and dual versions of these algorithms. A smaller fourth section addresses the extension of these algorithms to the clustering of higher-order cells. We conclude with a section providing some example applications of the presented algorithms.

## 6.1 Targeted Clustering

Targeted segmentation algorithms require information to be input about the desired output object or set of objects to cluster. This prior information can take different

forms, such as a partial labeling of the nodes or a probability, assigned to a set of nodes, of belonging to each label. In the field of image processing, a targeted segmentation algorithm might have the goal of segmenting a particular tumor in a medical image, or an incoming missile in a military application. Such an algorithm could also be used interactively to extract an object from a photograph for editing. In a World Wide Web application, the goal of a targeted clustering algorithm might be to extract a list of websites in the same cluster as a target website. An example in social networking would be to extract all members of the group in which a particular individual is a member. In the context of machine learning, targeted clustering algorithms are related to both *supervised learning* and *semi-supervised learning* algorithms, with the difference that these algorithms must generalize to all unseen data (nodes). However, in the context of machine learning, the targeted clustering algorithms described here may be viewed as examples of *transductive* learning algorithms that simply focus on labeling a known set of data points. Section 6.5.3 contains more information about this view of clustering as a machine learning algorithm.

We begin by addressing primal algorithms for targeted segmentation since primal algorithms comprise the majority of existing techniques for both targeted and untargeted methods. Additionally, primal algorithms are generally much more straightforward to describe and implement than dual algorithms.

### *6.1.1 Primal Targeted Clustering*

The basic components of a targeted segmentation algorithm are a known label set comprised of a finite number of labels, an energy for which the extrema describe "good" clusters, and additional information about how the labels relate to the node set. Examples of additional information exploited in a primal algorithm include:

1. A method for assigning membership probabilities for each label to a subset of nodes.
2. Known labels for a subset of the nodes.
3. A set of edges which the boundary is known to cross.

We will address each of these types of information sequentially.

In all of the subsequent discussion on targeted primal clustering algorithms, we can formulate our goal as solving for a probability $x_{i,b}$ that node $v_i$ belongs to label $b$. (Without loss of generality, we assume that each label is represented by an integer from $0 \leq b < |\mathcal{L}|$.) Unless otherwise noted, the segmentation function $\sigma(v_i) = b$ is obtained via choosing the most likely label for each node, i.e.,

$$\sigma(v_i) = \underset{b}{\operatorname{argmax}}\{x_{i,b}\}. \tag{6.1}$$

Consequently, our focus will be on finding the membership probabilities $x_{i,b}$ for each node $v_i$ and label $b$, since this set of membership probabilities defines the clustering via (6.1).

#### 6.1.1.1 Probabilities Assigned to a Subset

Consider a set of nodes $\mathcal{V}_S \subseteq \mathcal{V}$, such that for each node $v_i \in \mathcal{V}_S$, we have a prior probability that $v_i$ is assigned to label $b$, given by $\bar{s}_{i,b}$. We may now apply *any filtering technique* to produce a clustering (see Chap. 5 for a discussion on filtering on cell complexes). The approach for applying a filtering technique to compute a targeted clustering is to begin by extending the vector of priors $\bar{\mathbf{s}}$ beyond the subset $\mathcal{V}_S$ to priors $\mathbf{s}$ defined over all nodes in the complex, $\mathcal{V}$, via setting entries corresponding to the nodes not in the subset $\mathcal{V}_S$ to zero, i.e.,

$$s_{i,b} = \begin{cases} \bar{s}_{i,b} & \text{if } v_i \in \mathcal{V}_S, \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

Any filtering technique may then be applied to produce segmentation probabilities by treating the prior probabilities $\mathbf{s}$ as noisy data in a filtering technique. We saw several approaches for filtering noisy data in Chap. 5 which we now apply to find clustering probabilities $\mathbf{x}$. The first method that we apply from Chap. 5 is Taubin's method, in which we set the initial conditions $x_{i,b}^{[0]} = s_{i,b}$ and produce the final values of $x_{i,b}$ iteratively via Taubin's filtering. Specifically, to apply Taubin's filtering method (see Chap. 5) to find the $\mathbf{x}$ for each label, we employ the iteration rule

$$\mathbf{x}_b^{[2k+1]} = \mathbf{x}_b^{[2k]} - \lambda \mathbf{L}\mathbf{x}_b^{[2k]} = \mathbf{x}_b^{[2k]} - \lambda \mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x}_b^{[2k]}, \tag{6.3}$$

$$\mathbf{x}_b^{[2k+2]} = \mathbf{x}_b^{[2k+1]} + \mu\, \mathbf{L}\mathbf{x}_b^{[2k+1]}. \tag{6.4}$$

Similarly, we could minimize the Basic Energy Model from Chap. 5,

$$\mathcal{E}_{\text{BEM}}[\mathbf{x}_b] = \mathbf{1}^\mathsf{T}(\mathbf{G}^{-1})^p |\mathbf{A}\mathbf{x}_b|^p = \sum_{e_{ij}} w_{ij}^p |x_{i,b} - x_{j,b}|^p, \tag{6.5}$$

by applying an iterative mode ($p = 0$), median ($p = 1$), mean ($p = 2$) or minimax ($p = \infty$) filter to the initial probabilities. In other words, for each label $b$, the label probabilities $x_{i,b}$ (for node $v_i$) are filtered to produce final label probabilities and then each node is assigned the label for which is has the greatest probability (i.e., via (6.1)). For each label, the solution is initialized to $\mathbf{x}_b^{[0]} = \mathbf{s}_b$. Chapter 5 provides justification for why the mode, median, mean and minimax filters minimize the energy for the various values of $p$.

Figure 6.1 shows an example of how these filters may be applied to targeted image segmentation when the probabilities are assigned based on an intensity model of the target object. In this example, the image intensities inside the circle and outside the circle were both drawn from a uniform distribution with the same variance, but with a different mean inside and outside. In this case, the variance of the distributions was 2.5 times greater than the difference in mean between the distributions. The foreground priors were generated by assuming a Gaussian distribution for the intensities inside the circle with a mean equal to the minimum intensity in the image.

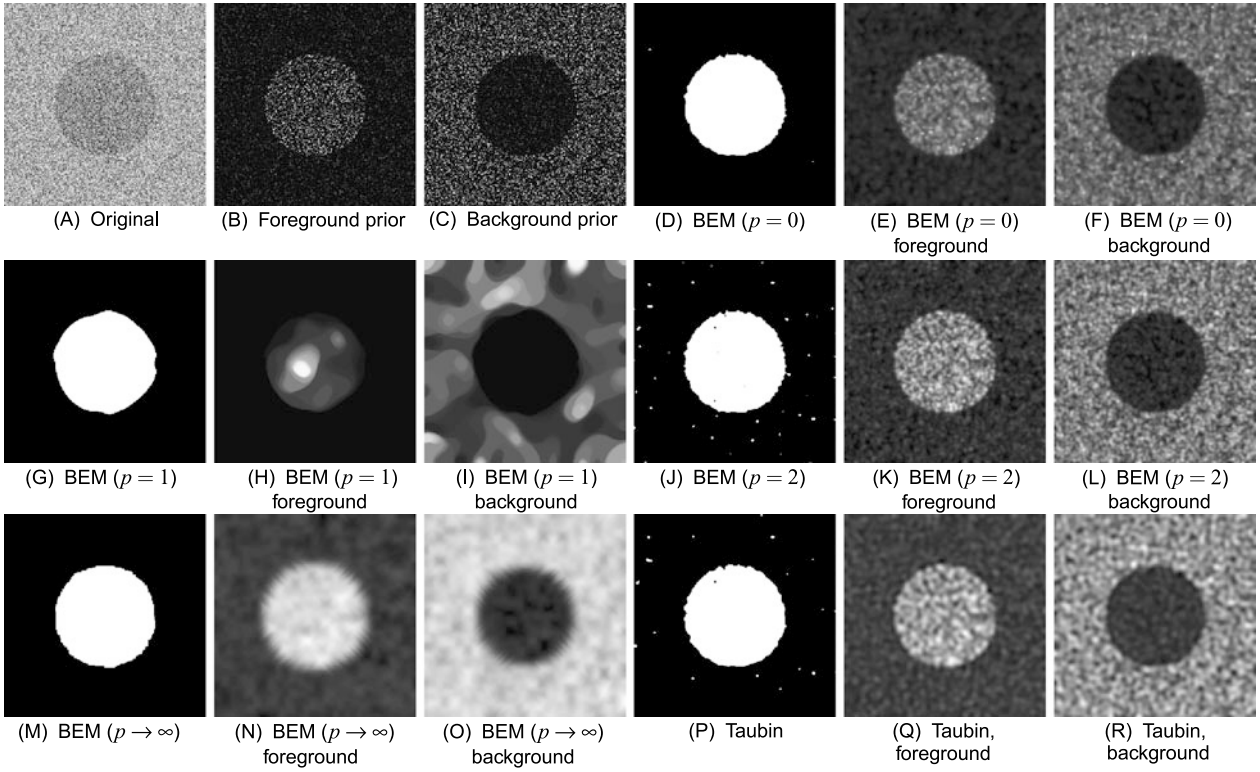


**Fig. 6.1** Targeted image segmentation using the Basic Energy Model and Taubin spectral filtering method. Each algorithm was initialized with known foreground and background intensity priors (obtained by modeling the foreground and background as a Gaussian distribution with a mean equal to the minimum/maximum intensity values respectively). These priors were filtered with each algorithm to produce foreground and background probabilities (a few iterations of each minimization was used for the BEM models). Each node was then assigned to the label (foreground or background) for which it has a higher probability

With this Gaussian prior model, each pixel in the foreground prior image was assigned the probability that it was drawn from the Gaussian model. The background priors were produced in the same manner, except that the mean of the Gaussian was chosen to be equal to the highest intensity pixel in the image. By simply smoothing these foreground and background priors (using the filtering methods described in Chap. 5) we are able to obtain a smoothed "probability" that each pixel belongs to the foreground or background, which may then be used to generate the final label for each pixel by comparing the relative foreground and background "probabilities". In this manner, *any* filtering algorithm may be used as a clustering algorithm *if* it is possible to assign a prior likelihood that each node belongs to a particular label.

In Chap. 5 we noted that the danger with variational problems of the form in the Basic Energy Model is that there is a trivial optimum—the solution where $\mathbf{x}_b$ is constant. In practice we could simply take a few iterations of our mode, median, mean or minimax filtering of $\mathbf{x}_b$ (as was done to generate the results in Fig. 6.1). An alternative to this iterative approach is to consider the Extended Basic Energy Model in which the prior information is represented by a second term that the solution must balance. In the context of clustering, the Extended Basic Energy Model is given by

$$\begin{aligned}\mathcal{E}_{\mathrm{EBEM}}[\mathbf{x}_b] &= \mathbf{1}^{\mathsf{T}}(\mathbf{G}^{-1})^p|\mathbf{A}\mathbf{x}_b|^p + \lambda\sum_a \mathbf{s}_a^{\mathsf{T}}|\mathbf{x}_b - \delta(a,b)|^p \\ &= \sum_{e_{ij}} w_{ij}^p|x_{i,b} - x_{j,b}|^p + \lambda\sum_{v_i}\sum_a s_{i,a}|x_{i,b} - \delta(a,b)|^p, \qquad (6.6)\end{aligned}$$

where $\delta(a,b)$ is a Kronecker delta function. It was shown in Chap. 5 that the strength of the regularization parameter $\lambda$ is inversely related to the number of iterations used for smoothing in the Basic Energy Model.

We may continue with the application of filtering methods to the clustering problem by considering the Total Variation model from Chap. 5. One benefit in image processing of the Total Variation model over the Basic Energy Model is that Total Variation has less tendency to exhibit gridding artifacts. In the context of clustering, the Total Variation Model is described by the energy functional

$$\begin{aligned}\mathcal{E}_{\mathrm{TV}}[\mathbf{x}_b] &= \mathbf{1}^{\mathsf{T}}\left(|\mathbf{A}^{\mathsf{T}}||\mathbf{A}\mathbf{x}_b|^2\right)^{\frac{p}{2}} + \lambda\sum_a \mathbf{s}_a^{\mathsf{T}}|\mathbf{x}_b - \delta(a,b)|^p \\ &= \sum_{v_{i,b}}\left(\sum_{e_{ij}}|x_{i,b} - x_{j,b}|^2\right)^{\frac{p}{2}} + \lambda\sum_{v_i}\sum_a s_{i,a}|x_{i,b} - \delta(a,b)|^p. \qquad (6.7)\end{aligned}$$

As before, this model is used to generate a clustering by finding a solution for each label $b$ and then comparing these solutions to produce a final labeling by assigning each node, $v_i$, to the label for which it has the maximum solution. More information on the optimization of this model is given in Chap. 5.

All of these models may be included to incorporate implicit boundaries via an edge weighting. These models are written with edge weights in Chap. 5 and we do not repeat this material here. The same weighting functions that were used to weight edges for filtering applications may also be applied for clustering.

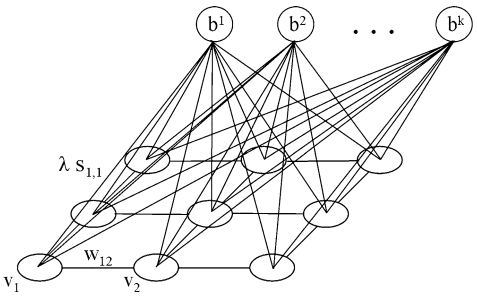


**Fig. 6.2** Mathematically, the use of label priors is equivalent to using $k$ labeled, "phantom" nodes that correspond to each label and are connected to each node. Despite the abuse of notation, note the labels $b$ are distinguished with superscript index indicating the different labels

#### 6.1.1.2 Known Labels for a Subset of Nodes

Many targeted clustering applications permit some method of assigning to each node a prior probability quantifying the likelihood that the node belongs to each label. In these situations, the targeted segmentation problem may be solved via any of the filtering methods reviewed in the previous section. However, another common form of targeting information is to know the labeling for a small subset of nodes. The nodes with a known label within this subset are often called **seeds** in the image processing literature. Seeded nodes can also take advantage of the filtering approaches to clustering discussed above by simply assigning the $x_i$ values for each seed node and performing the optimization with respect to this assignment. One major advantage of targeting with seeded nodes is that a global, nontrivial optimum may be found for each of the variational models (i.e., the Extended Basic Energy Model or the Total Variation Model) even if $\lambda = 0$. That is, the initial assignment of seed notes comprises a constraint that the solution is explicitly forced to satisfy. Therefore, if prior probabilities are not available or not reliable, they can be ignored by setting $\lambda = 0$. In this way, each model can be used completely parameter-free when seeds are available.

Formally, define the set of seeded or "marked" nodes $\mathcal{V}_M \subset \mathcal{V}$ for which $\sigma(\mathcal{V}_M)$ is known. These labels are assumed to have been obtained via a different process such as user interaction or an automatic seeding. Using this information, we can fix $x_{i,b},\ \forall v_i \in \mathcal{V}_M$ via

$$x_{i,b} = \begin{cases} 1 & \text{if } \sigma(v_i) = b, \\ 0 & \text{if } \sigma(v_i) \neq b. \end{cases} \tag{6.8}$$

Fixing these values as Dirichlet boundary conditions on the set of seeded nodes allows for optimization of the above models to produce a nontrivial solution for $\mathbf{x}_b$ if, for each label $b$, $\sigma(v_i) = b$ for some seed node $v_i \in \mathcal{V}_M$ (i.e., each label is associated with at least one seed). These seeds could also be used with Taubin's method applied to segmentation by initially fixing the values in $\mathcal{V}_M$ and not updating the corresponding values of $\mathbf{x}_b$ during the iterations. See Appendix B for more information on optimization in the presence of Dirichlet boundary conditions.

Although we distinguish between the prior probability method and seeded method for producing a targeted clustering, it is possible to view the prior probability method as equivalent to the seeded method. Specifically, the prior likelihood

term in the Extended Basic Energy Model or the Total Variation Model may also be obtained via seeding. The "seed" in this case is a "phantom seed" representing each label, $b$, which is attached to node $v_i$ with weight $\lambda s_{i,b}$. This interpretation of the prior term is represented in Fig. 6.2 and has previously appeared in the literature [53, 159]. Since the method of specifying a targeted clustering via prior terms is equivalent to seeding (with the phantom seeds construction), we may henceforth treat only the case of the seeded model, since it is understood that all of the results apply also to the incorporation of prior terms. Consequently, in the context of discussing seeded targeted clustering algorithms, we will employ the term "Basic Energy Model" to refer either to the "Basic Energy Model" or "Extended Basic Energy Model" since the additional term may be viewed simply as another form of seeding.

Incarnations of the seeded Basic Energy Model with various values of $p$ have been heavily utilized in the literature. We review several targeted clustering algorithms that can be interpreted as special cases of the Basic Energy Model, along with one algorithm that can be interpreted as a seeded Total Variation Model.

### Max-flow/Min-cut

It was shown by Sinop and Grady [350] that when $p = 1$ the solution $\mathbf{x}_b$ given by the Basic Energy Model (6.6) may be found by computing a max-flow/min-cut between the seeds (both real and "phantom") labeled '1' and the seeds labeled '0'. Consequently, the use of max-flow/min-cut to perform targeted node clustering may be viewed as an instance of employing the Basic Energy Model for targeted clustering with norm $p = 1$. Since max-flow/min-cut may be used to optimize the Basic Energy Model when $p = 1$, this energy model may be interpreted as minimizing the boundary length between the labeled regions. In this case, the seeds are identified with the source/sink terminals in the traditional max-flow/min-cut problem.

Max-flow/min-cut techniques have a long history in clustering problems [246, 407, 415]. Appendix B and Refs. [55, 56, 173] contain more details on this optimization. In the context of image segmentation, this seeded max-flow/min-cut model has been known as "graph cuts" [53].

### Random Walker

Taking $p = 2$ in the Basic Energy Model (6.6) allows the solution at node $v_i$, $x_{i,b}$, to be interpreted as the probability that a random walker leaving node $v_i$ arrives at a seed (real or "phantom") labeled $b$ before arriving at a seed not labeled $b$ [161] (see Chap. 3). Consequently, the clustering algorithm that employs $p = 2$ is known as the "random walker" algorithm in the image segmentation literature. Alternately, the clustering obtained by this model is equivalent to the clustering obtained by computing the effective resistance between each node and the seeds of each label (viewed as a single node) and assigning the node to the label having the smallest effective resistance [161] (for more information about effective resistance, see Chap. 3). The

**Table 6.1** Targeted clustering using the Basic Energy Model with seeds. Different values of the norm parameter $p$ give different interpretations to the model when applied to clustering

| Choice of $p$ | 1 | 2 | $\infty$ |
|---|---|---|---|
| **name** | Max-flow (Graph cut) | Random walk | Geodesic |
| **objective function** | $\sum_{e_{ij}\in\mathcal{E}} w_{ij}\|x_i - x_j\|$ | $\sum_{e_{ij}\in\mathcal{E}} w_{ij}^2\|x_i - x_j\|^2$ | $\max_{e_{ij}\in\mathcal{E}} w_{ij}\|x_i - x_j\|$ |
| **objective function interpretation** | boundary cut | effective conductance | minimum Lipschitz extension |
| **optimization method** | maximum flow | solution of a sparse linear system | shortest path |
| **uniqueness** | not unique | unique | not unique |

earliest adoption of this model for clustering may have been Kodres [237] who used it to determine how to design a circuit layout. This clustering algorithm has also been applied to machine learning [424] and extended to directed graphs [349].

### Geodesic Segmentation

When $p = \infty$, it was shown in [350] that the problem can be recognized as a discrete formulation of the minimal Lipschitz extension [13]. Additionally, it was shown in [348] that a minimum of the Basic Energy Model (6.6) may be given by

$$x_{i,b} = d_{i,b}/(d_{i,b} + d_{i,\overline{b}}), \tag{6.9}$$

where $d_{i,b}$ is used to indicate the weighted length of the shortest path from $v_i$ to any seed labeled $b$ and $d_{i,\overline{b}}$ is used to indicate the weighted length of the shortest path from $v_i$ to any seed not labeled $b$. This clustering approach is often called **geodesic segmentation** in the image segmentation literature. Using shortest paths in this way for image segmentation have been popularized by several groups [8, 18, 94, 128] (albeit without explicit reference to the energy minimization interpretation).

### P-Brush

These three choices of the norm parameter $p$ in the Basic Energy Model were compared by Sinop and Grady [350] to find that a smaller value of $p$ produces a clustering that has less dependence on the location of the seeds, but the clustering obtained by using a larger value of $p$ has less dependence on the number of seeds. Table 6.1 compares the clustering algorithms generated by various values of $p$. The use of the Basic Energy Model (6.6) for clustering with fractional values of $p$ was recently examined in [106], which found that the clustering algorithms obtained by minimizing fractional values of $p$ effectively interpolate between the clustering algorithms corresponding to integer values of $p$. Additionally, the solutions given by the Basic Energy Model are pointwise continuous with respect to changes in $p$. This clustering algorithm with fractional values of the norm parameter $p$ was called **P-Brush**.

Power Watershed

The classical watershed algorithm from mathematical morphology [320, 337] may also be seen as an instance of the Basic Energy Model with only a slight modification. Couprie et al. [90] modified the Basic Energy Model from (6.6) to produce

$$\mathcal{E}_{\mathrm{PW}}[\mathbf{x}_b] = \mathbf{1}^{\mathsf{T}}(\mathbf{G}^{-1})^q|\mathbf{A}\mathbf{x}_b|^p = \sum_{e_{ij}} w_{ij}^q |x_{i,b} - x_{j,b}|^p. \tag{6.10}$$

Once again, we assume that some nodes are seeded (alternately, a prior term is included as in the Extended Basic Energy Model). The only difference between (6.10) and the Basic Energy Model in (6.6) is the exponent, $q$, on the edge weights. It was shown by Allène et al. [4] that the $p = 1$ model above (max-flow/min-cut) is also optimized by a watershed computation for a value of $q$ above some constant. Therefore, as $q \to \infty$, the model in (6.10) becomes the watershed clustering algorithm when $p = 1$. Viewed differently, Allène et al. [4] showed that as the power of the weights increases to infinity, then the max-flow/min-cut algorithm produces a clustering corresponding to the maximum spanning forest (MSF) used in a watershed computation [4]. Interpreted from the standpoint of the Welsch weighting function in Chap. 4, it is clear that we may associate $q = \frac{1}{\alpha}$ to understand that the watershed equivalence arises when the weighting function is employed using a particular range of parameter values. An important insight from this connection is that *when the value of $\alpha$ is sufficiently small, we can replace the expensive max-flow computation with an efficient maximum spanning forest computation*.

**Algorithm 6.1** Power Watershed algorithm, optimizing $q \to \infty$, $p \geq 1$

**Data**: A weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and a subset of foreground seeds $\mathcal{V}_{\mathrm{FG}}$ and background seeds $\mathcal{V}_{\mathrm{BG}}$

**Result**: A solution $\mathbf{x}$

Set $\mathbf{x}_{\mathrm{FG}} = 1$, $\mathbf{x}_{\mathrm{BG}} = 0$ and all other $\mathbf{x}$ values as unknown, mark all edges as unprocessed.

Sort the edges of $\mathcal{E}$ by decreasing order of weight.

**while** *any node has an unknown potential* **do**

- Find an edge (or a plateau) $E_{\mathrm{MAX}}$ in $\mathcal{E}$ which is both of maximal weight and **safe**; denote by $\mathcal{S}$ the set of nodes connected by $E_{\mathrm{MAX}}$.
- **if** $\mathcal{S}$ *contains any nodes with known potential* **then**
  - Find $\mathbf{x}_{\mathcal{S}}$ minimizing (6.10) (using the input value of $p$) on the subset $\mathcal{S}$ with the weights in $E_{\mathrm{MAX}}$ set to $w_{ij} = 1$, all other weights set to $w_{ij} = 0$ and the known values of $\mathbf{x}$ within $\mathcal{S}$ fixed to their known values. Consider all $\mathbf{x}_{\mathcal{S}}$ values produced by this operation as known.
- **else**
  - Merge all of the nodes in $\mathcal{S}$ into a single node, such that when the value of $\mathbf{x}$ for this merged node becomes known, all merged nodes are assigned the same value of $\mathbf{x}$ and considered known.

**Table 6.2** The targeted clustering algorithms obtained by minimizing the Power Watershed model in (6.10) for various choices of $p$ and $q$

| $p$ | $q$ | | | |
|---|---|---|---|---|
| | 0 | 1 | 2 | $\infty$ |
| 1 | collapse to seeds | Max-flow | Max-flow | Watershed |
| 2 | $\ell_2$-norm Voronoï | Random walker | Random walker | Power watershed, $p = 2$ |
| $\infty$ | $\ell_1$-norm Voronoï | Geodesic | Geodesic | Power watershed, $p = \infty$ |

Couprie et al. [90] went on to explore the clustering model in (6.10) when $q \to \infty$ for any value of $p$. Since this family of watersheds was characterized by the exponent $p$, they termed this clustering algorithm the **power watershed**. Algorithm 6.1 gives an algorithm for finding the solution to $\mathbf{x}_b$ that optimizes the Power Watershed model in (6.10). In Algorithm 6.1, if $\mathcal{E}_{\text{MSF}}$ is a set of edges forming a subset of an MSF, then an edge $e_i$ is considered *safe* if $\mathcal{E}_{\text{MSF}} \cup e_i$ is also a subset of an MSF. Note that this algorithm applies only to two labels. In a multilabel targeted clustering problem, the potential function $\mathbf{x}_b$ for each target label $b$ would be set to the value '1' for all nodes assigned to the "foreground" ('1') or to the value '0' for all nodes set to the "background", and the final labeling for node $v_i$ assigned by choosing the label with the largest value (as expressed in (6.1)). Table 6.2 gives a description of the segmentation algorithms obtained by minimizing the Power Watershed energy for various pairs of values for $p$ and $q$.

Continuous Max-flow

The seeded Total Variation Model has also been applied in the context of image processing. When $p = 1$, this model is equivalent to the **continuous max-flow** formulation of [358] that was subsequently applied to image segmentation [11]. Additionally, due to the interpretation of (6.7) as a minimization of total variation, the minimization of (6.7) has also been applied to image segmentation under the name "total variation segmentation" or "TVSeg" [386]. Fast algorithms for this minimization are given in [11] and [68, 98, 385]. When $p = 2$, the Basic Energy Model and the Total Variation Model are equivalent (i.e., the "random walker" algorithm). To the knowledge of the authors, the cases of $p = \infty$ or the separation of the exponent onto the weights (yielding a power watershed-like algorithm) have not been explored in conjunction with the Total Variation Model for targeted clustering.

#### 6.1.1.3 Negative Weights

A less direct method for specifying a clustering target is to assign *negative weights* to some of the edges which are known to lie between clusters. Negative weights are generally sufficient without seeds or prior probabilities to produce a nontrivial solution of any of the above models, since the value of the objective function

can dip below zero. Negative weights can be used to encode *repulsion* between two nodes, causing the difference between the values of their membership probabilities $\mathbf{x}_b$ values to grow [413, 414]. However, negative weights can cause difficulties in optimization since the objective function may become unbounded, yielding no useful solution. To avoid these unbounded scenarios, restrictions on $\mathbf{x}_b$ must be imposed, such as requiring that $0 \leq \mathbf{x}_b \leq 1$ [257]. Optimization of the Basic Energy Model, the Power Watershed Model or the Total Variation Model are substantially more difficult problems to solve with negative weights, with limited benefit demonstrated so far. Consequently, there has been less attention devoted to employing repulsion (via negative weights) for specifying a clustering target.

We have now covered the most prominent variational models applied to primal image segmentation. All of these targeted clustering algorithms were derived from filtering algorithms applied to prior knowledge (likelihood priors and/or seeds) of the segmentation labels. In the next section, we discuss targeted segmentation via the *dual* complex. Adopting a dual viewpoint departs from the previous discussion on filtering.

### *6.1.2 Dual Targeted Clustering*

Instead of clustering by labeling nodes on the primal graph, clustering on the dual graph seeks to find the set of edges defining the *boundary* of the labeled regions. Finding this set of edges has two disadvantages over the primal algorithms we have studied.

1. Cycles in the dual complex map to cutsets in the primal complex for only a limited set of complexes (e.g., planar graphs in two dimensions).
2. The dual complex *depends on the dimensionality of the embedding*, and therefore these algorithms may require modification if the embedding changes. For example, embedding in $\mathbb{R}^2$ can be substantially different than embedding in $\mathbb{R}^3$, since edges are dual to edges in two dimensions and edges are dual to faces in three dimensions.

Nonetheless, there are advantages of using a dual algorithm. Some dual algorithms are faster, and sometimes the inputs to the targeted segmentation algorithm are easier to specify in a dual lattice (i.e., boundary locations). Additionally, some objective functions are easier to express as functions of the dual elements (e.g., edges, faces) rather than the primal elements (e.g., nodes). A computational advantage of dual algorithms is that the number of boundary edges is typically quite small compared to the number of internal labeled nodes. Consequently, the boundary may be represented more efficiently, and perturbations of the boundary from an initialization may be accomplished efficiently. Due to the natural embedding of images (as two-dimensional or three-dimensional lattices), most of the work on dual clustering algorithms has appeared in this literature (in which boundary-focused algorithms are sometimes called a "boundary parameterization" or a "Lagrangian representation" [339]).

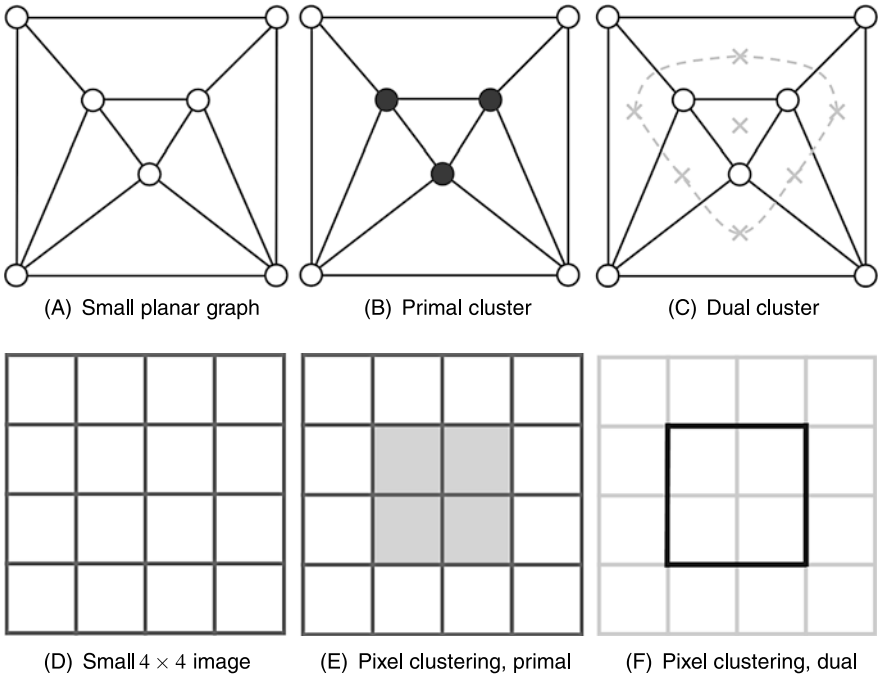


**Fig. 6.3** A comparison of primal clustering with dual clustering. (**A–C**) A small planar graph in which the primal clustering is represented as a binary labeling of each node (indicating the cluster membership of each node) and the dual representation of the clustering is as a closed boundary of edges which separate the inside of the cluster from the outside. (**D–F**) A set of pixels from a small $4 \times 4$ image. A primal clustering of these pixels is represented by assigning a binary labeling to each pixel to indicate cluster membership. In contrast, the dual representation of the clustering is obtained by identifying the edges *between* the pixels inside the cluster and those pixels outside the cluster

Figure 6.3 contrasts primal clustering with dual clustering on a small planar graph. A primal clustering algorithm assigns a *label* to each node in the primal graph, while a dual clustering algorithm identifies boundary edges in the dual graph. The first row of the figure gives an example of a small planar graph in which the primal clustering is represented as a binary labeling of each node (indicating the cluster membership of each node) while the dual representation of the clustering is as a closed boundary of edges which separate the inside of the cluster from the outside. The second row of the example shows a set of pixels from a small $4 \times 4$ image. A primal clustering of these pixels is represented by assigning a binary labeling to each pixel to indicate cluster membership. In contrast, the dual representation of the clustering is obtained by identifying the edges *between* the pixels inside the cluster and those pixels outside the cluster. In the image processing literature, these dual edges between the pixels have sometimes been called “cracks” or “bels” (for “boundary elements”) [129].

A common algorithm for targeted clustering on the dual graph in two dimensions is known as intelligent scissors or live wire [129, 287]. This algorithm interactively builds an open contour on the dual graph up to the last step, at which time the contour is closed to enclose a region of nodes. In the intelligent scissors algorithm, a series of nodes in the dual graph are sequentially input into the algorithm in pairs (e.g., interactively) and a shortest path is computed between the pairs of nodes. Formally, assume that we have computed appropriate edge weights between nodes on the primal graph (in the same manner as in the primal algorithms, see Chap. 4). Define an indicator vector $\mathbf{y}^s$ of edges representing a line segment (chain of dual edges) $s$ of the cluster boundary consisting of the set of dual edges such that

$$y_i^s = \begin{cases} 1 & \text{if dual edge } e_i \text{ is a member of the boundary,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.11}$$

Therefore, the edges indicated by $\mathbf{y}$ serve to "surround" the target cluster of nodes in the primal graph. Given two dual nodes $v_i$ and $v_j$ that are known to lie on the desired boundary (produced either interactively by a user or automatically), the intelligent scissors/live wire algorithm finds a solution to

$$\begin{aligned} &\min_{\mathbf{y}^s} \mathbf{w}^\mathsf{T}\mathbf{y}^s, \\ &\text{s.t.} \quad \mathbf{A}^\mathsf{T}\mathbf{y}^s = \mathbf{p}, \end{aligned} \tag{6.12}$$

where $\mathbf{w}$ is the vector of dual distance edge weights (equal to the primal affinity edge weights in the two-dimensional case, since in two dimensions edges are dual to edges—see Chap. 2 for more details) and the boundary vector $\mathbf{p}$ is defined as

$$p_k = \begin{cases} +1 & \text{if } k = i, \\ -1 & \text{if } k = j, \\ 0 & \text{otherwise.} \end{cases} \tag{6.13}$$

The optimization in (6.12) may be performed quite efficiently using Dijkstra's shortest path algorithm. The constraints in (6.12) demonstrate the use of the incidence matrix $\mathbf{A}^\mathsf{T}$ as the boundary operator, since we may interpret the constraint that $\mathbf{A}^\mathsf{T}\mathbf{y}^s = \mathbf{p}$ as the requirement that the set of edges represented by $\mathbf{y}^s$ has endpoints given by $v_i$ and $v_j$. After $\mathbf{y}^s$ has been computed, a new dual node is input to the intelligent scissors/live wire algorithm to define a new $\mathbf{p}$ vector. A series of $\mathbf{y}^s$ line segments are computed using a sequential set of points which are then combined to form the final output $\mathbf{y} = \sum_s \mathbf{y}^s$. An important implementation detail in the intelligent scissors/live wire algorithm is that because $\mathbf{y}$ is strictly binary valued, the orientation of edge traversal must be encoded in the incidence matrix. Consequently, the incidence matrix must be modified to represent each edge twice with opposite orientation (see Chap. 4 for more details on this over-representation). In practice, the use of Dijkstra's algorithm obviates these details—Dijkstra's algorithm implicitly solves (6.12). Therefore, all that is necessary for a practical implementation is to

use Dijkstra's algorithm to compute the shortest path between each successive series of node pairs on the boundary until the boundary is closed. A second implementation detail is that the edges connecting the dual nodes to the "outside face" (e.g., the border of the image) must be assigned some weight to allow for the contour to close around a set of nodes on the border. One approach to weighting the edges on the outside face is to assign these edges a weight equal to the average weight inside the complex.

A very recent clustering algorithm that employs a dual formulation is the algorithm of Schoenemann et al. [334] which uses a dual formulation as a way of encoding the *curvature* of the cluster boundary. The dual formulation is employed because the discretization of curvature employed by Schoenemann et al. was based on the work of Bruckstein et al. [61], who showed how angles between successive line segments of a polygon could be used to approximate curvature of the polygon. By associating the dual edges with polygonal line segments, Schoenemann et al. [334] produced an optimization method that optimized the cluster boundary to have small curvature.

#### 6.1.2.1 Dual Algorithms in Three Dimensions

Since the dual complex changes with embedding and dimensionality, any dual algorithm must be altered to accommodate these changes. In order to apply the shortest-path targeted clustering algorithm to a 3-complex, we now transition from minimal paths to minimal surfaces. Unless otherwise stated, we will assume for simplicity that our 3-complex is a three-dimensional, 6-connected lattice. Fortunately, the dimensionality of the minimal path problem may be increased simply by using the dimension-appropriate incidence matrix (acting as the boundary operator) and boundary vector $\mathbf{p}$. This dimension-increased shortest path problem is therefore stated as: *Given the boundary of a two-dimensional surface (i.e., a closed contour or series of closed contours), find the minimal two-dimensional surface with the prescribed boundary*. We note that this problem may be considered as a discrete instance of Plateau's problem [363].

In this three-dimensional problem, the boundary operator is the edge–face incidence matrix defined in Chap. 2. Instead of the lower-dimension boundary vector, $\mathbf{p}$, we can now employ the vector $\mathbf{r}$ as a signed indicator vector of a closed contour with an associated ordering of vertices obtained via a traversal along the edges comprising the contour. Given a contour represented by an ordering of vertices $(v_a, v_b, v_c, \ldots, v_a)$ such that each neighboring pair of vertices is contained in the edge set, the contour may be represented with the vector

$$r_i = \begin{cases} +1 & \text{if the vertices comprising edge } e_i \text{ are contained in the contour} \\ & \text{with coherent orientation,} \\ -1 & \text{if the vertices comprising edge } e_i \text{ are contained in the contour} \\ & \text{without coherent orientation,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.14}$$

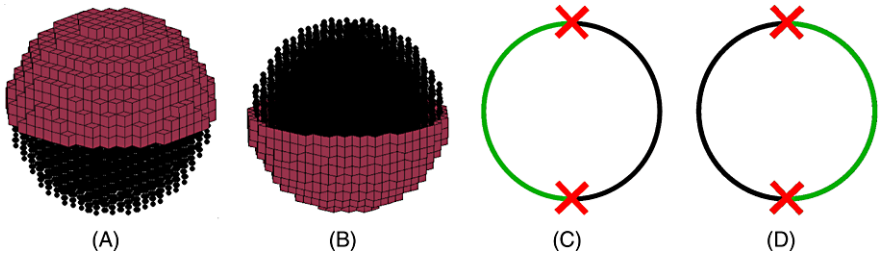


**Fig. 6.4** The discrete minimal surface given a boundary may not be unique. For example, if the surface boundary is given as a closed contour at the equator of a sphere, then either the upper (**A**) or lower (**B**) hemisphere is a valid minimum solution. This same lack of uniqueness may also appear in the shortest path problem. Analogously, if two endpoints were placed at antipodal points of a circle, the shortest path may be returned as either the left (**C**) or right (**D**) path around the circumference of the circle. However, in applications with real data, both the shortest path and minimal surface are typically unique for a given input

Therefore, the discrete minimal surface problem is

$$\begin{aligned} \min_{\mathbf{z}} \mathcal{Q}[\mathbf{z}] &= \sum_i w_i z_i, \\ \text{subject to} \quad & \mathbf{B}\mathbf{z} = \mathbf{r}, \end{aligned} \tag{6.15}$$

where $\mathbf{z}$ is an indicator vector indicating whether or not a face (in the dual complex) is in the minimal surface and $w_i$ is used to indicate the weights of a face. The surface represented by the solution to (6.15), $\mathbf{z}$, is a (discrete) minimal surface. Since the faces in the dual lattice correspond to edges in the primal lattice (where the image data is represented), any of the weighting functions in Chap. 4 may be used to produce the set of face weights.

The discrete minimal surface problem was extensively treated by Sullivan [364], who showed that a fast algorithm exists for its optimization. However, the equality constraints in (6.15) are pre-unimodular matrix (see Appendix B and [162]). Therefore, a generic linear programming solver could be used to produce an optimal integer solution of (6.15) even though linear programming uses a real-valued relaxation of the variable $\mathbf{z}$. For more details on this optimization, see Refs. [162, 364].

The solution to the discrete minimal surface problem may not be unique. However, the shortest path problem may also have a solution which is not unique. Figure 6.4 illustrates this issue. For example, a closed contour located precisely at the equator of the sphere in Fig. 6.5(A) could result in a solution indicating either the upper or the lower hemisphere. This is analogous to the one-dimensional case where multiple solutions exist that give shortest paths between two antipodal points on a circle.

Figure 6.5 gives three examples drawn from three-dimensional image segmentation to illustrate the properties of this discrete minimal surface algorithm. Firstly, we use the algorithm to find the surface of a black sphere in a white background,

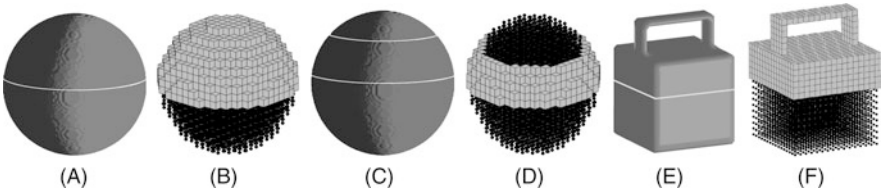


**Fig. 6.5** Minimal surface segmentation of synthetic three-dimensional images. Renderings of the original object (with the input contours) are shown, along with the solutions. The input volumes consisted of black voxels indicating voxels belonging to the object and white voxels indicating background. Black voxels are represented in the figure by *small black spheres*. The *white stripe* in each of the rendered views shows the input contour(s). In the solution visualizations, black dots are plotted at the center of the black (object) voxels and faces are shown to indicate the computed surface. (**A**, **B**) A sphere with an input contour along a parallel. Note that, unlike shortest paths (which require two endpoints), a single boundary contour input is sufficient to define a solution. (**C**, **D**) A sphere with two input contours at parallels of different heights. (**E**, **F**) A lunchbox shape with a handle on the top and a contour input around the middle of the object. The algorithm will correctly find minimal surfaces with topological changes

given an initial contour around one parallel. Secondly, we find the surface of the same sphere using a boundary consisting of contours around two parallels (i.e., a contour given on two slices). Finally, we segment a "lunch box" shape given a contour around the middle of the object. This experiment shows that the algorithm correctly handles changes in object genus without any special handling. In contrast to the shortest path problem in which two points are necessary to define a path, Fig. 6.5(A) shows that a single closed contour is sufficient to define the boundary of a surface. The applications in Sect. 6.5.1 illustrate the use of this segmentation algorithm on real three-dimensional image data.

## 6.2 Untargeted Clustering

Untargeted clustering is the traditional clustering problem in which the goal is to divide a graph into a hierarchy of clusters where the number of clusters is unknown. This problem is not well defined since there is no agreed-upon criteria for determining what constitutes a "good" cluster. Even for those algorithms with a well-defined, seemingly simple criteria for a good clustering, the problem is generally NP-Hard (e.g., $k$-means is NP-Hard [5, 269]). Consequently, nearly every untargeted clustering algorithm defines its own meaning of what constitutes a good clustering, and then supplies a heuristic that produces a suboptimal solution to the stated objective.

Untargeted clustering has a huge range of uses, including data discovery, compression [225, 332], parallelization [392], image processing [216, 303, 382], the efficient solution of PDEs (via domain decomposition) [351, 381], sparse matrix ordering (nested dissection) [229, 281] and identification of neural substructures [41, 190]. The collection of techniques for untargeted clustering is far too vast to provide a comprehensive review here (for a recent review on graph clustering algorithms see

Ref. [330]). Instead, we present only a few techniques that fit well into the theme of this book. In this section, we mainly address *primal* untargeted segmentation algorithms (which comprise almost all untargeted clustering algorithms), although dual untargeted algorithms are treated briefly at the end of this section.

We begin by noting that any of the *targeted* clustering algorithms can be used to produce an untargeted clustering algorithm. Specifically, a targeted clustering algorithm may be converted to an untargeted clustering algorithm by the following steps.

1. Select the number of clusters $K$.
2. Randomly select $K$ nodes and assign these nodes each a different label.
3. Apply the targeted segmentation algorithm with these seeds.
4. Move the seed location to the "center" of each labeled cluster.
5. Continue applying the targeted segmentation and moving the seed locations until all of the seed locations no longer move.

Essentially, this procedure is an adaptation of Lloyd's algorithm for $k$-means clustering [263, 264] to any of the targeted clustering methods. Note that convergence of this procedure may not be guaranteed, and therefore in practice it may be beneficial to limit the number of iterations.

One common approach for designing untargeted clustering algorithms is to define an algorithm that partitions the graph into two clusters and then recursively applies the partitioning on each cluster until some measure of partition quality is met and the recursion exits. Although there exist good reasons for theoretical and practical concerns with such a recursive approach (see Ref. [347]), the recursion approaches have the advantage of not requiring prior knowledge of the total number of clusters and they do work reasonably well in practice. We will now review a standard approach for dividing a graph into two clusters with the understanding that this partitioning may be applied recursively to obtain multiple clusters.

### *6.2.1 Primal Untargeted Clustering*

In all of the subsequent discussion on primal bipartitioning algorithms, we can formulate our goal as solving for a 0-cochain $\mathbf{x}$, with coefficients $x_i \in \mathbb{R}$, that determine whether node $v_i$ belongs to label '0' or '1'. The segmentation function is defined here as $\sigma(v_i) \mapsto \{0, 1\}$ obtained by thresholding the cochain $\mathbf{x}$, i.e.,

$$\sigma(v_i) = \begin{cases} 1, & \text{if } x_i \geq \theta, \\ 0, & \text{otherwise}, \end{cases} \tag{6.16}$$

where the threshold $\theta$ is set manually, automatically or in some application-dependent manner (see Ref. [352] for some standard possibilities for choosing a threshold $\theta$). Our initial focus will be on the production of the inclusion cochain $\mathbf{x}$ for each node before continuing to a discussion of how to set the threshold $\theta$.

If we reconsider the Basic Energy Model (6.6) in the context of untargeted bipartitioning, we see that the absence of the targeting information causes the energy to have a trivial minimum at $\mathbf{x} = k$ for some constant $k$. One approach for addressing this trivial minimum is to add an extra term that attempts to expand the cluster. This additional force is sometimes known in the image processing literature as a *balloon force* [83, 84]. Specifically, consider the Untargeted Basic Energy Model

$$\begin{aligned}\mathcal{E}_{\text{UBEM}}[\mathbf{x}] &= \mathbf{G}_1^{-1}|\mathbf{A}\mathbf{x}|^p - \lambda \mathbf{G}_0^{-1}|\mathbf{x}|^q \\ &= \sum_{e_{ij}} w_{ij}|x_i - x_j|^p - \lambda \sum_{v_i} w_i |x_i|^q, \end{aligned} \tag{6.17}$$

for some $\lambda > 0$. This energy can be made arbitrarily low by setting $\mathbf{x} = k$ for some constant value $k$. A variety of strategies for avoiding this problem may be adopted. For each combination of $p$ and $q$, we discuss how this problem has been overcome in the past. The gradient of $\mathcal{E}_{\text{UBEM}}[\mathbf{x}]$ equals zero when $\mathbf{x}$ satisfies

$$p\mathbf{A}^{\mathsf{T}}\mathbf{G}_1^{-1}|\mathbf{A}\mathbf{x}|^{p-1} = \lambda q \mathbf{G}_0^{-1}|\mathbf{x}|^{q-1}. \tag{6.18}$$

Although several combinations of $p$ and $q$ values have not been investigated in the literature, there are some cases which lead to algorithms that have had an important impact on the automatic clustering community. Optimization of the Untargeted Basic Energy Model produces a solution which may be thresholded to provide a bipartition into two clusters. The standard approach of using the Untargeted Basic Energy Model to produce more than two clusters is to recursively bipartition these clusters until some measure of the partition quality (usually the value of $\mathcal{E}_{\text{UBEM}}[\mathbf{x}]$) fails to be satisfied.

We begin by examining the Untargeted Basic Energy Model for $p = 2$ and $q = 2$. In this case, then the minimum taken in (6.18) is given by

$$\mathbf{A}^{\mathsf{T}}\mathbf{G}_1^{-1}\mathbf{A}\mathbf{x} = \mathbf{L}\mathbf{x} = \lambda \mathbf{G}_0^{-1}\mathbf{x}. \tag{6.19}$$

When $\mathbf{G}_0^{-1} = \mathbf{I}$ then the solution $\mathbf{x}$ is an eigenvector of $\mathbf{L}$. As an eigenvector problem, we can view (6.19) as a minimization of the Rayleigh quotient

$$\lambda = \frac{\mathbf{x}^{\mathsf{T}}\mathbf{L}\mathbf{x}}{\mathbf{x}^{\mathsf{T}}\mathbf{G}_0^{-1}\mathbf{x}}. \tag{6.20}$$

The first eigenvector of $\mathbf{L}$ is $\mathbf{x} = k$ for some constant $k$. However, since $\mathbf{L}$ is symmetric, the eigenvectors will be orthogonal. Therefore, by taking $\mathbf{x}$ to be the eigenvector corresponding to the second smallest eigenvalue we have an optimum to the Untargeted Basic Energy Model (6.17) when optimized in the space orthogonal to $\mathbf{x} = k$. Consequently the problem of an unbounded solution for the Untargeted Basic Energy Model in (6.17) is resolved by adopting this eigenvector because the optimization has been effectively performed in the space orthogonal to the problematic (constant) solution. The second smallest eigenvalue is often called the **Fiedler value**

(although Fiedler originally called it the "algebraic connectivity" [133]) and the corresponding eigenvector is known as the **Fiedler vector**. There are several reasons to support the Fiedler vector as a method for graph clustering: (i) A smaller value of $\lambda$ represents less perturbation from the targeted model, allowing the smoothness term to have the greatest effect; (ii) It was shown by Fiedler [134, 135] that labeling nodes above a threshold as foreground and below the threshold as background guarantees that both foreground and background components are connected [134, 135]; and (iii) The Fiedler value can be used to bound the isoperimetric constant of a graph [81]. Automatic clustering by thresholding the Fiedler vector is called **spectral clustering**, which has been rediscovered several times in the clustering literature [114, 184, 310]. We can view spectral clustering from the continuous calculus perspective as an instance of the *Helmholtz equation*, e.g.,

$$\nabla^2 x = \lambda x. \tag{6.21}$$

Recall that the Helmholtz equation describes the harmonic frequencies of an ideal membrane. Consequently, we can interpret the solution to (6.19) as the lowest frequency nontrivial harmonic which is then thresholded to produce a partitioning. Figure 6.6 shows two examples of graphs which are mapped to heights that correspond to the values of the lowest frequency eigenvector (i.e., the Fiedler vector). The first example shows an elongated lattice for which the harmonic reflects over the principle axis of symmetry. The second example shows a more separated graph for which each cluster isolates well as one component of the harmonic. An additional geometric interpretation of the spectral clustering approach is as a mapping of the nodes to the real line in order to both locate each node (on the real line) at the average location of its neighbors as well as maintain a unit average distance between all pairs of nodes [14].

We may again consider the spectral clustering algorithm (6.19) in the case that $\mathbf{G}_0^{-1} = \mathbf{D} = \text{diag}(\mathbf{d})$ where $\mathbf{d}$ is the vector of node degrees. In this case, the solution is the *generalized eigenvector* of $\mathbf{L}$ and $\mathbf{D}$. This variant of the spectral clustering algorithm (6.19) first appeared in the image processing literature as **Normalized Cuts** [345] in which the authors claimed that using $\mathbf{G}_0^{-1} = \mathbf{D}$ significantly improved the quality of spectral clustering applied to image segmentation, presumably because of the large range of weight values in a weighted graph derived from an image. The use of $\mathbf{G}_0^{-1} = \mathbf{D}$ for clustering has been additionally supported both theoretically and empirically by Coifman et al. [85].

We now examine the Untargeted Basic Energy Model (6.17) when $p = 2$ and $q = 1$. In this case, the minimum of the Untargeted Basic Energy Model (given by (6.18)) is taken when $\mathbf{x}$ satisfies

$$\mathbf{A}^\mathsf{T}\mathbf{G}_1^{-1}\mathbf{A}\mathbf{x} = \mathbf{L}\mathbf{x} = \lambda\, \mathbf{g}_0^{-1}, \tag{6.22}$$

where $\mathbf{g}_0^{-1}$ represents the vector consisting of the diagonal elements of $\mathbf{G}_0^{-1}$, i.e., $\mathbf{G}_0^{-1} = \text{diag}(\mathbf{g}_0^{-1})$. Unfortunately, because $\mathbf{L}$ is singular the solution of this problem is undefined. This singularity corresponds to the unbounded solution $\mathbf{x} = k$ discussed above for the Untargeted Basic Energy Model. The typical solution to this

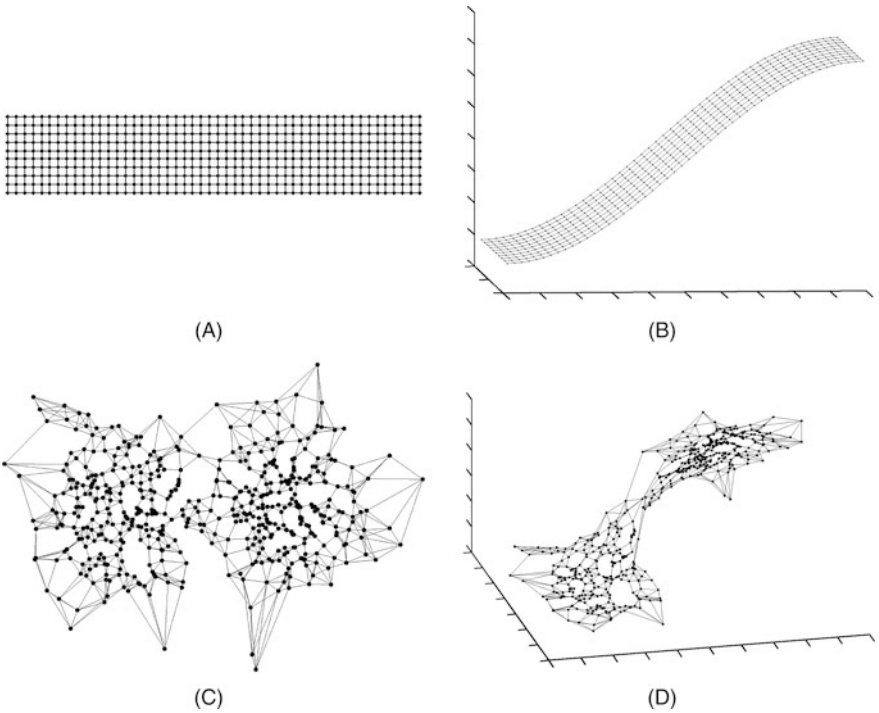


**Fig. 6.6** The first nontrivial harmonic of two example graphs. From a physics standpoint, we may consider the graphs as a network of springs for which the Fiedler vector provides the lowest frequency harmonic. Additionally, the lattice graph may be viewed as a finite differences approximation to a membrane. (**A**, **C**) Example graphs, (**B**, **D**) Corresponding embeddings in which the first nontrivial harmonic is mapped to the height of each node

singularity employed by spectral partitioning is to perform the optimization of $\mathbf{x}$ in the space orthogonal to $\mathbf{x} = k$. However, an alternative approach to this problem was suggested in [168, 169] that a single reference node, $v_r$, be chosen such that $x_r = 0$ is fixed. By fixing a reference node to one partition, the problem in (6.22) takes a unique solution which may then be thresholded to produce the clustering into two partitions. Note that the clustering obtained in this manner does not depend on the value of $\lambda$ and we therefore ignore this value. As with spectral clustering above, both $\mathbf{G}_0^{-1} = \mathbf{I}$ and $\mathbf{G}_0^{-1} = \mathbf{D}$ have been employed with the choice of $\mathbf{G}_0^{-1} = \mathbf{D}$ generally producing better clusters [169]. Additionally, it was proved by Grady and Schwartz [169] that the partitioning performed in this way guaranteed that the partition connected to the reference node is connected. Since the solution to (6.22) may be interpreted as the steady-state DC potentials for a resistive network with $\mathbf{c}_0$ representing currents injected into each node, the reference node was called the **ground node** [169]. Figure 6.7 shows the equivalent circuit for which the solution to (6.22) gives the steady-state electrical potentials. We can view (6.22) from the continuous

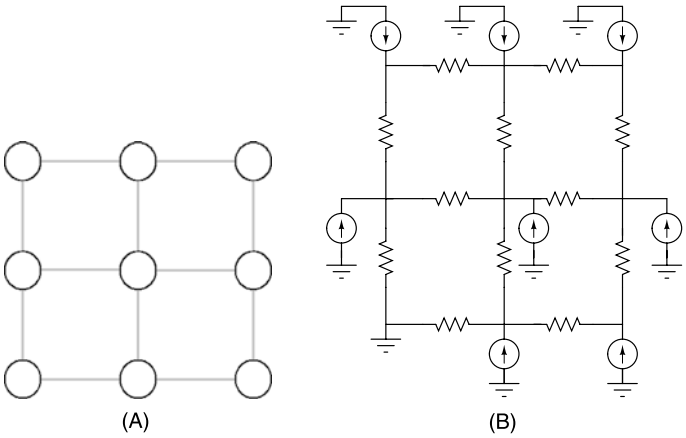


**Fig. 6.7** An example of a simple graph (**A**), and its equivalent circuit (**B**). Solving (6.22) (using the node in the lower left as ground) for the graph depicted in (A) is equivalent to connecting this node to ground in the circuit depicted in (B) and then measuring the potential values at each node

calculus perspective as an instance of the *Poisson equation*,

$$\nabla^2 x = \lambda c_0, \tag{6.23}$$

subject to the internal Dirichlet boundary condition that $x_r = 0$. This algorithm was called the **isoperimetric algorithm** by Grady and Schwartz [169] due to the connection with a relaxed version of the isoperimetric ratio. See Chap. 8 for more information about the isoperimetric ratio of a graph and Appendix B for more information about the optimization of a ratio. The primary advantages of this isoperimetric algorithm over spectral clustering are that solving a symmetric positive definite linear system of equations is more efficient than solving a (generalized) eigenvector problem, the solution to the eigenvector problem may not be unique, and the spectral algorithm overly tries to produce balanced clusters [179]. See Ref. [169] for more details on this comparison.

The Untargeted Basic Energy Model has also been studied for the case of $p = 1$ and $q = 1$ with the further restriction that the solution $\mathbf{x}$ is binary. This approach has the advantage that no threshold of $\mathbf{x}$ needs to be chosen and that it is possible to choose an optimal eigenvalue $\lambda$ automatically rather than manually. See Kolmogorov et al. [242] for examples of this model in the context of image segmentation and more information about its optimization.

### *6.2.2 Dual Untargeted Clustering*

Dual untargeted clustering algorithms have received little, if any, treatment in the literature. A primal algorithm has the advantage that it is independent of the embedding. This independence was forfeited in the targeted clustering problem in re-

turn for a unique method of targeting a cluster (i.e., by providing a partial boundary). Therefore, in the untargeted clustering problem, it is worth briefly examining whether anything is gained by a dual formulation. One possible advantage of a dual formulation is that it is easy to use the boundary operator to impose the constraint that a solution is a closed boundary. For example, if our dual graph were planar, then we could express the space of closed two-dimensional contours as any contour satisfying the constraint that

$$\mathbf{A}^{\mathsf{T}}\mathbf{y} = \mathbf{0}, \tag{6.24}$$

where $\mathbf{y}$ is a 1-cochain describing the set of edges in the boundary. An attractive aspect of the condition in (6.24) is that the $\mathbf{A}^{\mathsf{T}}$ matrix is totally unimodular, which means that an integer solution may be obtained under this condition even if the optimization were over a linear functional of real-valued $\mathbf{y}$ variables. A second attractive aspect of the condition (6.24) is that the nullspace of $\mathbf{A}^{\mathsf{T}}$ is known, allowing for an easy change of variables to create an unconstrained optimization problem. However, despite the ease of the dual constraint to describe a closed contour, this form seems to have been explored only very recently in the literature [293].

## 6.3 Semi-targeted Clustering

In the introduction to this chapter we commented that some clustering algorithms establish an objective function for the clustering which cannot be feasibly optimized. In these circumstances the clustering optimization may be initialized, leading to a solution similar to the initialization. Consequently, even though the objective function may describe an untargeted clustering problem, the dependence of the optimization on the initialization may lead to a clustering that is somewhat targeted. In this section, we consider a primal semi-targeted clustering algorithm, the $k$-means algorithm and its generalization to the Mumford–Shah model.

Traditionally, the Mumford–Shah model has appeared only in the field of image processing. Therefore, the connections between $k$-means and the Mumford–Shah have received little attention in the literature. However, the recent work on formulating (and optimizing) the Mumford–Shah model on an arbitrary graph [97, 121, 163, 418] has made it more clear how to interpret the Mumford–Shah model as a generalization of the $k$-means model. In the next section, we adopt this approach explicitly by building from the $k$-means model to the full, piecewise smooth, Mumford–Shah model.

### *6.3.1 The k-Means Model*

The $k$-means model is probably the most common method for the untargeted clustering of data. Given a predetermined number of labels $k$, the $k$-means algorithm seeks to find a clustering that is a partitioning of the node set into $k$ disjoint subsets

of nodes, defined by the sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k$, such that the intersection of any two sets is empty and the union of all sets is $\mathcal{V}$. We can denote this partitioning with the partitioning function $Z$, which maps the nodes into the collection of partitions $\mathcal{P} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k\}$, i.e., $Z : \mathcal{V} \rightarrow \mathcal{P}$. The optimum clustering in the $k$-means algorithm is defined as the optimization of

$$\mathcal{E}_{\text{KM}}[Z] = \sum_i \sum_{v_j \in \mathcal{R}_i} \|\tilde{c}_{\mathcal{R}_i} - \tilde{s}_j\|^2, \tag{6.25}$$

where $\tilde{s}_j$ is the **data tuple** at node $v_j$ and $\tilde{c}_{\mathcal{R}_i}$ is the mean data tuple inside $\mathcal{R}_i$ representing the *centroid* of the data within region $\mathcal{R}_i$.[1] Therefore, the $k$-means algorithm attempts to find the $k$-way partitioning of the data such that the data tuples in each partition have minimal within-cluster sum-of-squares difference from the mean tuple. Note that here we use tuple-valued data to handle cases in which multivariate data is available (e.g., RGB or colorscale image data), however in the simple univariate case these tuples can be replaced with scalars.

The classical method for optimizing the $k$-means model is Lloyd's algorithm [263, 264], which consists of the following steps.

1. Randomly partition the data nodes into $k$ regions.
2. Repeat until convergence:
   a. Update each centroid tuple, $\tilde{c}_{\mathcal{R}_j}$, to the mean of the data tuples of all nodes in $\mathcal{R}_j$.
   b. Assign each point, $v_i$, to the label for which its data tuple $\tilde{s}_i$ is closest to the label centroid, i.e., $\sigma(v_i) = \text{argmin}_j\{\|\tilde{c}_{\mathcal{R}_j} - \tilde{s}_i\|_2\}$.

Since the output of Lloyd's algorithm depends on the initialization, the algorithm is typically run to convergence multiple times with different initializations in order to find a good partitioning.

The $k$-means algorithm is fast and simple to implement, which accounts for its tremendous popularity. However, some data clustering problems have a *spatial* component as well as a data component. For example, the $k$-means algorithm above would cluster data tuples of an image (e.g., RGB image data) without accounting for the spatial arrangement of the pixels. However, based on the low-frequency models established for data in Chap. 5 and cluster labels developed above, neighboring nodes should be more likely to take the same label. Under this model, an outlier gray pixel inside a uniform region of black pixels is likely to be an artifact and can reasonably be assigned to the same label as the surrounding black pixels, while an outlier gray pixel inside a uniform region of white pixels is likely to be a artifact and can reasonably be assigned to the same label as the surrounding white pixels. However, the definition of the $k$-means algorithm above has no mechanism to

[1] In the rest of this chapter we treated the data as univariate in order to simplify the exposition, with the understanding that all of the machinery could also be applied to multivariate data. However, since $k$-means is almost exclusively applied to multivariate data we have adopted a multivariate view of data in this section. Therefore, it is assumed that each node (data point) is associated with a tuple of data, rather than a scalar.

account for the neighborhood structure of the data values.[2] Consequently, we can modify the $k$-means algorithm by adding a spatial regularization term which penalizes cases where neighboring nodes (defined by an edge set) are assigned different labels. This penalty may be viewed as encoding the *boundary length* of each region $\mathcal{R}_j$. Additionally, the penalty could be weighted using any of the weighting methods where the weights are derived from the data as described in Chap. 4, but this direction has not been pursued in the literature and we therefore limit our exposition to the unweighted (i.e., the unity-weighted) case with the understanding that boundary length could be trivially modified to incorporate data weights. The energy functional describing this modified $k$-means algorithm is given by

$$\mathcal{E}_{\text{MPCV}}[Z] = \sum_i \sum_{v_j \in \mathcal{R}_i} \|\tilde{c}_{\mathcal{R}_i} - \tilde{s}_j\|^2 + \nu \sum_{e_{ij} \in \mathcal{E}} \delta(\sigma(v_i) - \sigma(v_j)), \tag{6.26}$$

where $\delta(\cdot)$ represents the Kronecker delta function. The value of the trade-off parameter $\nu$ is a free parameter that can be used to weight the relative importance of spatial coherency in the $k$-means algorithm. Unfortunately, the spatial regularization term in (6.26) makes this energy quite a bit more difficult to optimize than the standard $k$-means energy of (6.25) when $k > 2$. Recent work has addressed approximation methods for optimizing the boundary for multiple labels [17, 122]. Therefore, we simplify the energy functional of the modified $k$-means algorithm (6.26) by restricting the number of classes to two (i.e., $k = 2$) and optimizing instead

$$\mathcal{E}_{\text{CV}}[\mathcal{R}] = \mathbf{r}^{\mathsf{T}} \|\tilde{c}_{\mathcal{R}} - \tilde{\mathbf{s}}\|_2^2 + (\mathbf{1} - \mathbf{r})^{\mathsf{T}} \|\tilde{c}_{\bar{\mathcal{R}}} - \tilde{\mathbf{s}}\|_2^2 + \nu(\mathbf{1}^{\mathsf{T}} |\mathbf{A}\mathbf{r}|), \tag{6.27}$$

where $\tilde{\mathbf{s}}$ represents the $|\mathcal{V}| \times 1$ vector of tuples $\tilde{s}_j$, or a tuple-valued vector over the nodes. Since there are only two labels, we can represent them by the set $\mathcal{R}$ and $\bar{\mathcal{R}}$, in which case $\mathbf{r}$ is a node vector (0-chain) acting as an indicator function for the set $\mathcal{R}$. Similarly, we use $\tilde{c}_{\mathcal{R}}$ to represent the data centroid tuple in set $\mathcal{R}$ and $\tilde{c}_{\bar{\mathcal{R}}}$ to represent the data centroid tuple in the complement set, $\bar{\mathcal{R}}$. Thus the expression $\|\tilde{c}_{\mathcal{R}} - \tilde{\mathbf{s}}\|_2^2$ is used to represent a $|\mathcal{V}| \times 1$ vector where each entry $j$ equals $\|\tilde{c}_{\mathcal{R}} - \tilde{s}_j\|_2^2$. In the image processing literature this special case energy model is known by different names as the *piecewise constant Mumford–Shah* functional [289] or the **Chan–Vese model** [70], although it is important to note that both of these models were formulated in a continuous setting, and the first definitions of the models on a general graph appeared much more recently [97, 121, 418]. The expression of this model for multiple classes (i.e., $k > 2$), described in (6.26), has been known as the *multi-phase Chan–Vese model* and is given by the energy $\mathcal{E}_{\text{MPCV}}$ defined above.

The two-class model in (6.27) may be optimized in a manner similar to Lloyd's algorithm for optimizing $k$-means. The only difference with Lloyd's algorithm is

[2] Some authors have tried to incorporate spatial location into $k$-means by using the pixel coordinates as part of the feature vector in the application of $k$-means. This device can mitigate the problem described here in certain circumstances, but does not generalize to applications in which the network has no embedding or when the embedding is complicated, as in the gene expression example in Sect. 6.5.4 or the geospatial example in Sect. 5.9.4.

that the labeling step is more complicated, because the boundary term must now be accounted for. Specifically, by viewing $\mathbf{r}$ as a labeling vector, then (6.27) can be considered as a min-cut problem since

$$\text{Boundary}(\mathbf{r}) = \nu \mathbf{1}^{\mathsf{T}} |\mathbf{A}\mathbf{r}| = \sum_{e_{ij} \in \mathcal{E}} \nu |r_i - r_j|, \tag{6.28}$$

and

$$\mathbf{r}^{\mathsf{T}} \|\tilde{c}_{\mathcal{R}} - \tilde{\mathbf{s}}\|_2^2 = \sum_{v_i \in \mathcal{V}} r_i \|\tilde{c}_{\mathcal{R}} - \tilde{s}_i\|_2^2, \tag{6.29}$$

which may also be viewed as a min-cut (a unary term) via the construction

$$\text{Cut}(\tilde{c}_{\mathcal{R}}) = \sum_{v_i \in \mathcal{V}} \|\tilde{c}_{\mathcal{R}} - \tilde{s}_i\|_2^2 |r_i - 0|, \tag{6.30}$$

where $\mathcal{E}_0$ is a set of auxiliary edges connecting each node to a phantom terminal (as in Fig. 6.2). Similarly,

$$\text{Cut}(\tilde{c}_{\bar{\mathcal{R}}}) = \sum_{e_{ij} \in \mathcal{E}_1} \|\tilde{c}_{\bar{\mathcal{R}}} - \tilde{s}_i\|_2^2 |1 - r_i|. \tag{6.31}$$

Therefore, the optimization of the two-class model in (6.27) is enacted by the following steps.

1. Randomly initialize the set $\mathcal{R}$ and compute the centroid $\tilde{c}_{\mathcal{R}}$ of $\mathcal{R}$, and the centroid $\tilde{c}_{\bar{\mathcal{R}}}$ of set $\bar{\mathcal{R}}$.
2. Repeat until convergence:
   a. Find the labeling vector $\mathbf{r}$ that minimizes

$$\mathcal{E}[\mathcal{R}] = \text{Boundary}(\mathbf{r}) + \text{Cut}(\tilde{c}_{\mathcal{R}}) + \text{Cut}(\tilde{c}_{\bar{\mathcal{R}}})$$

   from the expressions above using a max-flow/min-cut algorithm.
   b. Assign each centroid, $\tilde{c}_{\mathcal{R}}$ and $\tilde{c}_{\bar{\mathcal{R}}}$, to the mean of the data vectors of all nodes in $\mathcal{R}$ and $\bar{\mathcal{R}}$, respectively.

The $k$-means model was generalized to incorporate spatial information by introducing a spatial regularization term. However, in cases where the spatial arrangement of the nodes is relevant, this generalized $k$-means approach seems limited by the fact that the data is modeled as a single centroid for every node in the cluster (i.e., the centroid value is uniform *everywhere in space* within a cluster). Consequently, this generalized $k$-means model may be further generalized by allowing each node to have its own idealized "centroid" (i.e., to allow the term corresponding to the centroid to vary spatially). To avoid confusion, we term this spatially varying version of the region centroid as a *pseudocentroid*, in which every node in the graph has a both pseudocentroid for $\mathcal{R}$ and a pseudocentroid for $\bar{\mathcal{R}}$. However, to be meaningful, the pseudocentroid cannot be allowed to vary arbitrarily and therefore we can impose

a smoothness penalty on the pseudocentroids for each region. Specifically, for the two-class problem, let

$$\mathcal{E}_{\text{smooth}}[\mathcal{R}] = \sum_{e_{ij} \in \mathcal{E}} r_i \|\tilde{c}_{i\mathcal{R}} - \tilde{c}_{j\mathcal{R}}\|_2^2 + \sum_{e_{ij} \in \mathcal{E}} (1 - r_i) \|\tilde{c}_{i\bar{\mathcal{R}}} - \tilde{c}_{j\bar{\mathcal{R}}}\|_2^2, \tag{6.32}$$

where now $\tilde{c}_{i\mathcal{R}}$ represents the pseudocentroid of node $v_i$ in set $\mathcal{R}$. Therefore, the pseudocentroid at each node should change smoothly between nodes within a region. This model may be called the *piecewise-smooth Mumford–Shah model*, even though the formulation of Mumford and Shah [289] was in a continuous space. This form of the model on a graph appeared in Ref. [163]. By allowing the pseudocentroids to vary with each node, the output of our minimization is both a clustering *and* and idealized form of our data. This idealized form of the data was used in Chap. 5 for filtering noisy data, while the focus of this chapter is on the clustering. This smoothness term could be penalized differently for smoothness changes across different edges by employing any of the affinity edge weighting functions derived from data which were detailed in Chap. 4. However, since this smoothness penalty has not been adjusted for edge weighting in the literature, we will continue the exposition assuming an unweighted (i.e., a unity weighted) edge set.

When we consider each region to be represented by a single centroid, then it is easy to compare the data at every node to a cluster centroid to see how well the node might fit in the cluster. However, when we allow each cluster to be represented by several pseudocentroids (in fact, each node is represented by its own pseudocentroid), then it is less clear how we might compare the data at nodes which are not inside a particular cluster with the pseudocentroids of that cluster, since each cluster is no longer represented by a single centroid. Therefore, the piecewise smooth Mumford–Shah model requires an estimate of the pseudocentroid for each cluster at each node. This estimation was performed by Grady and Alvino [163] by smoothly extending the pseudocentroids outside of each cluster to all nodes by solving a Laplace equation. Figure 6.8 shows an example of these pseudocentroids in the context of image segmentation.

Once a cluster pseudocentroid is generated for each cluster and node, then the optimization of the piecewise smooth Mumford–Shah model is the same as the piecewise constant model above. At each iteration of the model, it is necessary to estimate the cluster pseudocentroid for each pixel and then solve a max-flow/min-cut problem to find the optimal clustering (for the two-class case). See Ref. [163] for more details. This clustering model could also be extended to multiple classes by recursive bisection or by the approximation approaches taken by El-Zehiry et al. [122] and by Bae and Tai [17]. Figure 6.9 shows the clustering and idealized data values obtained for a synthetic image.

Among the class of semi-targeted algorithms, we have so far considered only the $k$-means algorithm and the more generalized Mumford–Shah algorithm, which both operate on the primal graph. Dual semi-targeted algorithms are not considered here because the authors are unaware of any algorithms of this type. However, in the image processing literature there is a vast amount of work on *active contour* methods

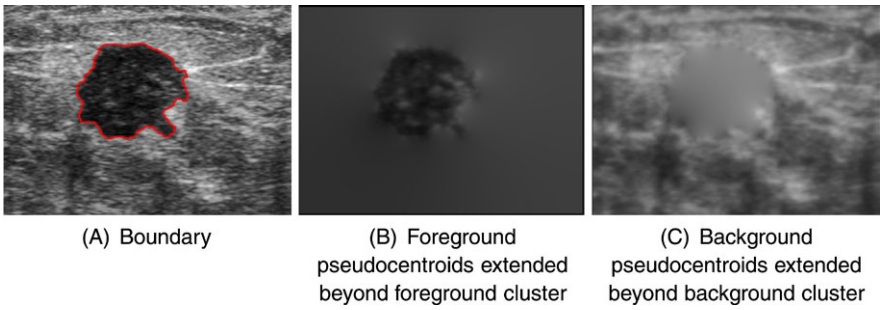


**Fig. 6.8** Optimization of the piecewise smooth Mumford–Shah using max-flow requires specification of values for the cluster pseudocentroid estimation function $\tilde{c}_{i\,\mathcal{R}}$ in the region *outside* the cluster region $\mathcal{R}$. Using a Laplace equation regularizer allows us to estimate the values for the foreground $\tilde{c}_{i\,\mathcal{R}}$ tuples and the background $\tilde{c}_{i\,\bar{\mathcal{R}}}$ tuples for the entire domain (i.e., the graph representing the image), as shown above. See Ref. [163] for more details

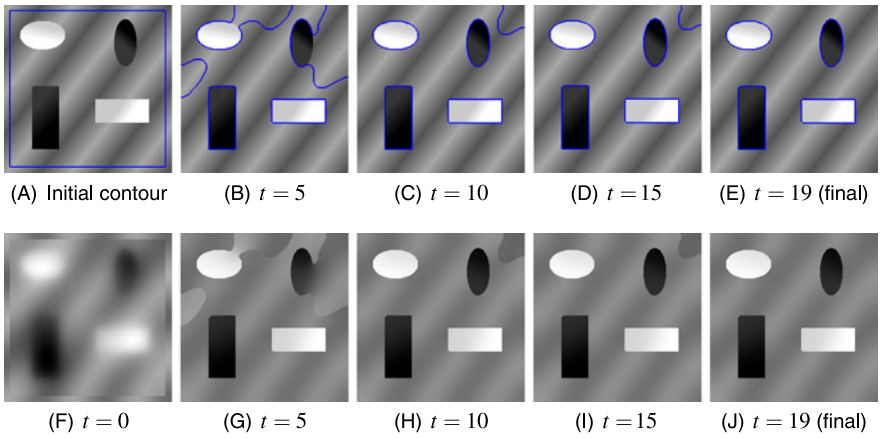


**Fig. 6.9** Several steps of the algorithm to optimize the piecewise smooth Mumford–Shah Model. (**A**–**E**) Partitioning evolution from initialization to stabilization. *Blue contours* indicate boundary location. (**F**–**J**) Corresponding pseudocentroid reconstruction of the piecewise smooth estimate of the image data, given a contour. For each image, the iteration step $t$ is provided below

(e.g., [222, 409]) which are very similar in spirit. These methods parameterize a cluster boundary as a polygon with a series of control points which are evolved to (locally) optimize an objective function. However, these methods do not fit into the scope of the present work since they require an embedding, the control points are allowed to vary anywhere in the embedded space and the methods are not explicitly formulated on a dual complex.

## 6.4 Clustering Higher-Order Cells

In the previous sections we described variational models for targeted and untargeted clustering of the nodes in a graph. We end this chapter by briefly recasting these models to apply to the clustering of higher-order cells. We will focus on recasting these models in terms of edge clustering with the understanding that the same methods could be further applied to the clustering of $p$-cells for any $p$ (as in Chap. 5).

### *6.4.1 Clustering Edges*

We will consider targeted, untargeted and initialized primal clustering formulations. Formally, the clustering problem on edges may be formulated by assigning labels $b$ of a label set $\mathcal{L}$, with $b \in \mathcal{L}$, to the set of edges, $\mathcal{E}$. As in the case of nodes, the goal of a clustering algorithm is to produce a segmentation function $\sigma : \mathcal{E} \to \mathcal{L}$. In the following discussion, we assume that a cycle set has been identified and this cycle set constitutes a basis for the cycle space, i.e., that the edge Laplacian is full rank. Additionally, we assume that $\mathbf{G}_1 = \mathbf{I}$.

#### 6.4.1.1 Targeted Edge Clustering

The basic components of a targeted edge clustering algorithm are the same as those components of a targeted node clustering algorithm. Specifically, additional information is required that associates some or all edges to particular labels. Examples of additional information in a targeted edge clustering algorithm could include:

1. A method for assigning probabilities to a subset of the edges that these edges belong to each label.
2. Known labels for a subset of the edges.

In all of the subsequent discussion on targeted edge clustering algorithms, we can formulate our goal as solving for a membership probability $y_{i,b}$ that edge $e_i$ belongs to label $b$. Unless otherwise noted, the segmentation function $\sigma(e_i) = b$ is obtained via $\sigma(e_i) = \text{argmax}_b\{y_{i,b}\}$, similar to the case with segmenting nodes. Consequently, our focus will be on the calculation of $y_{i,b}$ for each edge and label.

We begin by rewriting the weighted version of the Basic Energy Model in (6.5) to apply to clustering edges. As in the filtering case treated in Chap. 5, we replace the gradient of the node data by both the curl and the divergence of the flow data along each edge, since this replacement preserves the character of the operator as penalizing high-frequency functions (with respect to the edge Laplacian). Specifically, we may rewrite (6.5) for edges as

$$\mathcal{E}_{\text{BEM}}[\mathbf{y}_b] = \mathbf{1}^{\mathsf{T}} \left| \mathbf{G}_2^{-1} \mathbf{B} \mathbf{y}_b \right|^p + \mathbf{1}^{\mathsf{T}} \left| \mathbf{G}_0 \mathbf{A}^{\mathsf{T}} \mathbf{y}_b \right|^p. \tag{6.33}$$

Once again, we can avoid a trivial minimum of this energy by incorporating a prespecified set of labels for some edges (similar to the use of *seeds* in the node clustering case). Formally, define the set of seeded or "marked" edges $\mathcal{E}_M \subset \mathcal{E}$ for which $\sigma(\mathcal{E}_M)$ is known. These labels are assumed to have been obtained via a different process such as user interaction. Using this information, we can fix $y_{i,b}$ for all edges $e_i \in \mathcal{E}_M$ via the assignment

$$y_{i,b} = \begin{cases} 1 & \text{if } \sigma(e_i) = b, \\ 0 & \text{if } \sigma(e_i) \neq b. \end{cases} \tag{6.34}$$

These fixed values allow us to solve for a nontrivial minimum of the edge-focused Basic Energy Model in (6.33), which can be computed via the optimization methods in Appendix B.

As observed for node clustering, the trivial minimum of the Basic Energy Model could also be avoided by adding a term encoding the prior of each edge to belong to each label. We may continue to apply the targeted clustering models of Sect. 6.1 to flow data. When considering the Basic Energy Model expressed in (6.33) we treated the prior probabilities as initialized with an initial condition then iteratively updated to minimize the target energy. However, unless pre-labeled seeds are known, an optimal solution of (6.33) is trivially zero. Therefore, we may take a different approach to incorporating the prior probabilities by separating the smoothness constraints and priors into two different terms whose influence on the solution is controlled with a parameter. This edge-focused Extended Basic Energy Model is expressed by

$$\mathcal{E}_{\text{EBEM}}[\mathbf{y}_b] = \mathbf{1}^{\mathsf{T}} \left| \mathbf{G}_2^{-1} \mathbf{B} \mathbf{y}_b \right|^p + \mathbf{1}^{\mathsf{T}} \left| \mathbf{G}_0 \mathbf{A}^{\mathsf{T}} \mathbf{y}_b \right|^p + \lambda \sum_a \mathbf{s}_a^{\mathsf{T}} \left| \mathbf{y}_b - \delta(a-b) \right|^p, \tag{6.35}$$

where $\mathbf{s}_a$ represents the prior likelihoods that each edge belongs to label $a$, $\delta(a-b) = 1$ if $a = b$ and $\delta(a-b) = 0$ otherwise. At the cost of introducing a free parameter $\lambda$ we now have a model that produces a nontrivial minimum. Additionally, we may also introduce a set of seeds in the same manner as before, which allows us to find a nontrivial solution even when $\lambda = 0$ (i.e., prior probabilities are unknown).

### 6.4.1.2 Untargeted Edge Clustering

Building on the untargeted Basic Energy Model of (6.17), we now consider the edge-focused untargeted Basic Energy Model in the context of untargeted bipartitioning of an edge set. Specifically, consider the energy

$$\mathcal{E}_{\text{UBEM}}[\mathbf{y}] = \mathbf{1}^{\mathsf{T}} \mathbf{G}_2^{-1} |\mathbf{B}\mathbf{y}|^p + \mathbf{1}^{\mathsf{T}} \mathbf{G}_0 |\mathbf{A}^{\mathsf{T}} \mathbf{y}|^p - \lambda \mathbf{1}^{\mathsf{T}} |\mathbf{y}|^q, \tag{6.36}$$

for some $\lambda > 0$. As in the node case, the solution $\mathbf{y}$ may be thresholded to produce a bipartitioning of the edge set. This energy takes a minimum for $p > 1$ when $\mathbf{y}$ satisfies

$$\mathbf{B}^{\mathsf{T}} \mathbf{G}_2^{-1} |\mathbf{B}\mathbf{y}|^{p-1} + \mathbf{A} \mathbf{G}_0 |\mathbf{A}^{\mathsf{T}} \mathbf{y}|^{p-1} = \lambda |\mathbf{y}|^{q-1}. \tag{6.37}$$

We briefly consider some combinations of the norm parameters $p$ and $q$ values with a focus on a contrast with the node clustering case.

The $p = 2$ and $q = 2$ case also yields an eigenvector problem

$$\left(\mathbf{B}^{\mathsf{T}}\mathbf{G}_2^{-1}\mathbf{B} + \mathbf{A}\mathbf{G}_0\mathbf{A}^{\mathsf{T}}\right)\mathbf{y} = \mathbf{L}_1\mathbf{y} = \lambda\mathbf{y}, \tag{6.38}$$

in the edge Laplacian matrix. The eigenvectors of the edge Laplacian matrix have been lightly treated in the literature to date. These early investigations [1, 80, 288] reveal that the eigenvectors corresponding to low eigenvalues of the edge Laplacian roughly correspond to *circulations* around the complex. When the complex has a higher genus (i.e., there is one or more "handles" in the domain), a number of eigenvalues equal to the genus reach zero exactly and the corresponding eigenvectors represent flows that encircle these handles. From the standpoint of clustering, such circulations may indeed be meaningful clusters, since each circulation is a set of flows with minimal curl and divergence. Figure 6.10 shows the result of this model for clustering the flows in a graph.

Similarly, the $p = 2$ and $q = 1$ case yields a minimum to (6.37) when

$$\left(\mathbf{B}^{\mathsf{T}}\mathbf{G}_2^{-1}\mathbf{B} + \mathbf{A}\mathbf{G}_0\mathbf{A}^{\mathsf{T}}\right)\mathbf{y} = \mathbf{L}_1\mathbf{y} = \mathbf{1}. \tag{6.39}$$

This expression represents the flow version of the isoperimetric algorithm of Ref. [169]. The isoperimetric algorithm on nodes required an additional step of "grounding" a node (in accordance with the electrical circuit analogy) at $x_g = 0$ to break the singularity of the node Laplacian. Although we assumed that we have a complete cycle basis (i.e., $\mathbf{L}_1$ is full rank), (6.39) still has a trivial solution unless some edge is chosen as a fixed "ground" where $y_g = 0$. Figure 6.10 gives an example of this model for clustering the flows on a graph.

## 6.5 Applications

Clustering has an enormous number of applications across many fields. In this section, we give several applications of these clustering techniques with the goal of demonstrating the variety of applications of these methods.

### *6.5.1 Image Segmentation*

Clustering image data is a core problem in image processing in which the clustering procedure is known as *image segmentation*. Image segmentation is an essential part of many practical applications and graph-theoretic algorithms have historically played an important, and increasing, role in image segmentation [345, 407, 416]. Example applications include the quantification of objects in medical/scientific images, photo editing, and enhanced visualization of image contents. Additionally,

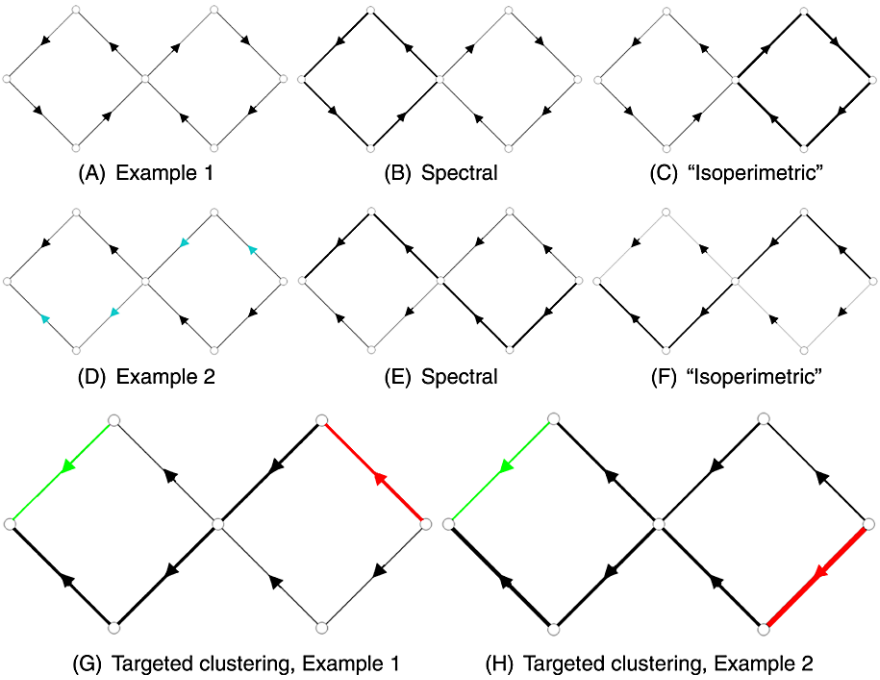


**Fig. 6.10** Clustering flows. In all cases, the thickness of the line represents the strength of the **y** vector used to determine the partitioning (by thresholding). *Top row*: Example 1, showing two circulations around the cycles cluster easily by both the spectral edge clustering method and the "isoperimetric" edge clustering method. The clusters are obtained by thresholding the resulting flows. *Middle row*: Example 2, in which the orientation of four edges are flipped (marked in *blue*). A leftward flow is partitioned into two streams by both partitioning methods. *Bottom row*: The "random walker" model applied to targeted clustering. The labeled edges were the upper left edge (*green*) and the second labeled edge was in the lower right or upper right (*red*). In the first case, the two diagonal flows are clustered together while in the second case, the upper and lower flows are clustered together

image segmentation can be used as a preprocessing step before many other image analysis tasks, such as object recognition, image registration or image compression. Traditional two-dimensional image data consists of "picture elements" (pixels) typically arranged in a square grid and traditional three-dimensional image data consists of "volume elements" (voxels) arranged in a rectilinear grid. In our applications, we associate each pixel or voxel with a node and use a 4-connected lattice as the edge set (6-connected in three dimensions). In all of our examples, the edges were weighted using the Welsch function of the difference of image intensities (see Chap. 4).

Our goal in this example is to demonstrate the varieties of clustering algorithm that were discussed previously. Specifically, image processing applications permit both primal *and* dual segmentation algorithms, since the image content has a geometric arrangement that naturally admits description as a cell complex with a dual. Consequently, the segmentation problem may be cast as having the goal of labeling

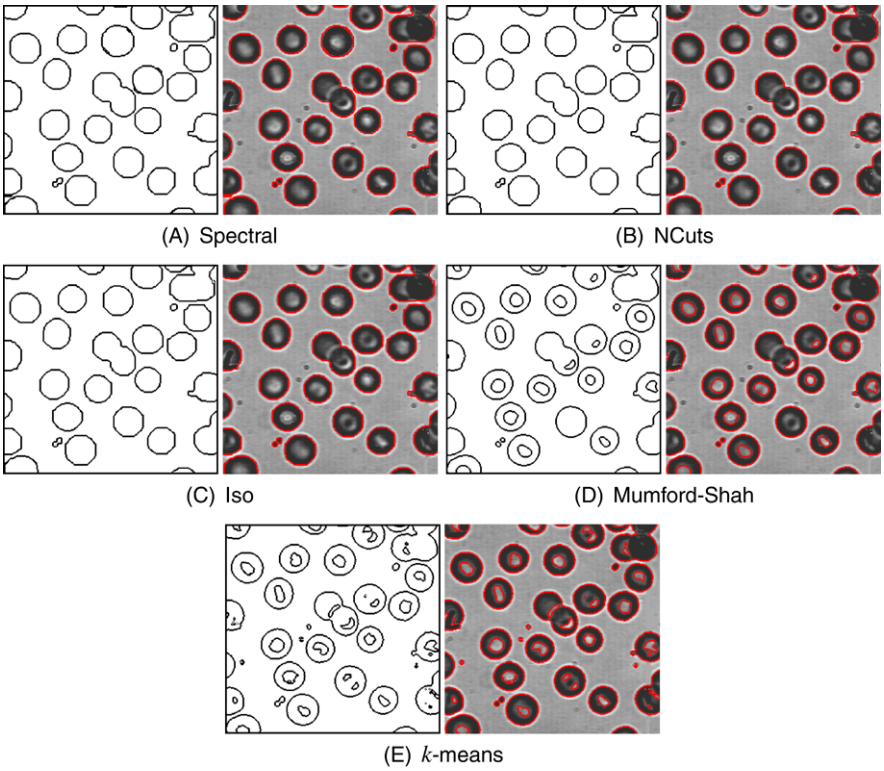


**Fig. 6.11** Untargeted image segmentation of blood cells using spectral partitioning, Normalized Cuts (NCuts), isoperimetric clustering (Iso), $k$-means and the piecewise constant Mumford–Shah. The $k$-means algorithm employs image intensity and the Mumford–Shah algorithm employs both image intensity and spatial regularity causing them both to associate the bright center of each cell with the bright background. In contrast, the purely spatial isoperimetric, spectral and normalized cuts algorithms all associated this inner part with the cells themselves. For each algorithm, the outline of the computed cluster boundaries is presented on the *left*, and the same cluster boundaries are superimposed on the image on the *right*, indicating the final segmentation

each pixel (in the primal case) or of labeling the boundary of the pixels (in the dual case).

We begin by demonstrating the primal untargeted algorithms on the application of cell counting in an image of blood cells. Figure 6.11 displays the results of each algorithm on the task of separating the cells from the background. Note that no attempt was made to further subdivide the cell clusters. Due to the clarity of this image, all of the algorithms produced a reasonable clustering. Since the Mumford–Shah energy balances image intensity clustering with spatial clustering, and $k$-means removes the spatial aspect, these two algorithms associated the inner, brighter, part of the cells with the background. In contrast, the purely spatial isoperimetric, spectral and Normalized Cuts algorithms all associated this inner part with the cells themselves.

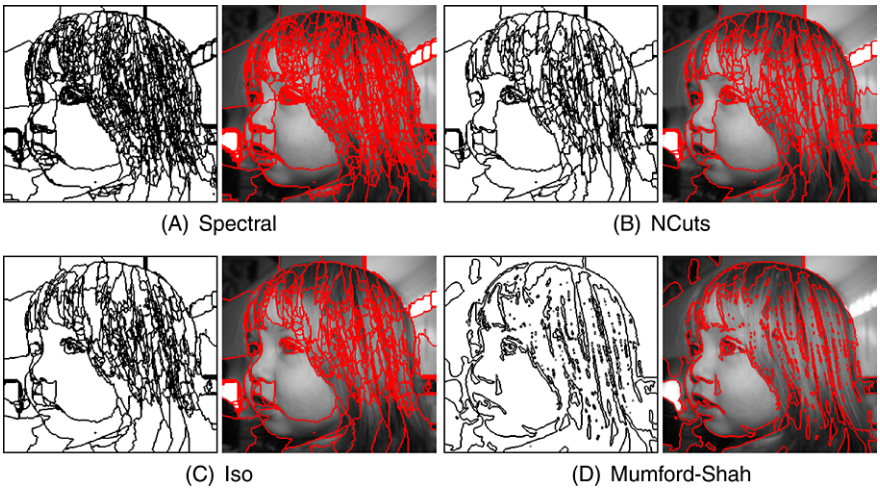


**Fig. 6.12** Untargeted segmentation algorithms applied to a grayscale photograph of a girl. The algorithms used only image intensity to define edge weights and unary terms rather than more complicated models such as using texture boundaries to set weights. Consequently, the algorithms used in this way try to isolate each strand of hair as a separate object rather than as a unit

The blood cells image contained pixels belonging to two types, cell and background, which were generally separated by different intensity profiles. Images that contain more classes of objects are more difficult for an untargeted image segmentation method to parse, especially if the object classes have a differing intensity/texture profile. To illustrate this issue, the same set of untargeted algorithms were applied to find a segmentation of the photograph of a girl given in Fig. 6.12. As before, edge weights were derived from a Welsch weighting function of the image intensities of the neighboring pixels. The models could have been made more appropriate to the segmentation of this image by deriving weights from texture boundaries and using such information as features for the Mumford–Shah model. By basing the models on image intensity, the algorithms attempt to segment each strand of hair as a separate object, rather than segmenting all of the hair as a single unit. Since these algorithms all explicitly optimize a separation of two objects, the algorithms were used to generate multiple labels by applying the algorithms recursively and determining when to stop the recursion based on the energy obtained by the minimum produced by the segmentation.

An untargeted image segmentation algorithm tells us what the best clusters are. If we instead wanted to find a *particular* object, then we need to target that object through a targeted image segmentation approach. We may isolate the girl's head in this photograph by supplying some labeled pixels which identify the object to be isolated. In this example, these labeled pixels were supplied interactively by the authors and each of the targeted segmentation algorithms were run to produce a segmentation separating the image into foreground and background. Results of this experiment are displayed in Fig. 6.13. Although each of these algorithms also used edge weights derived from intensity differences, the targeting of a particular object

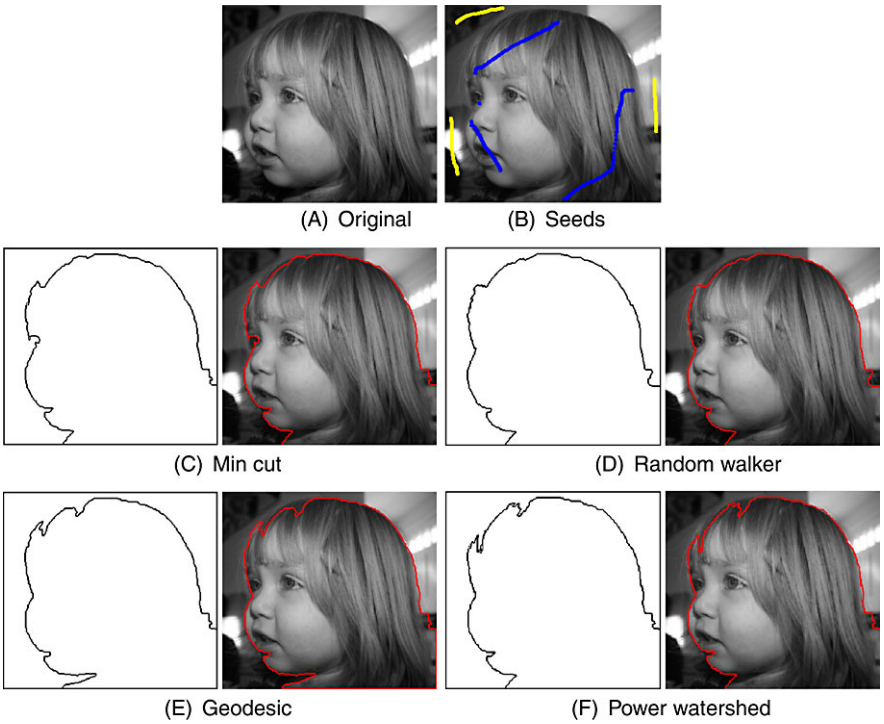


**Fig. 6.13** Example of targeted image segmentation algorithms. These algorithms were used to isolate the girl's head by interactively supplying some labeled pixels. Although these algorithms also used the same intensity-based edge weighting as before, the act of targeting a particular object allows the algorithms to know which objects should be grouped together (e.g., each strand of hair belongs to the foreground label)

was sufficient for these algorithms to group together the parts of the objects (e.g., every strand of hair is grouped together in the foreground).

The previous example applied targeted segmentation algorithms to isolate the image foreground. However, all of these algorithms operated on the *primal* graph in the sense that the final output was a labeling of each pixel and the boundary was implied by the change in label. We discussed in the text how a *dual* algorithm could also be applied to image segmentation by finding a set of edges in the dual graph which separated the nodes into two pieces. Dual clustering algorithms are available for two-dimensional image processing because the planar graph defined by the 4-connected lattice has a well-defined 2-dual. The most popular dual segmentation algorithm in the computer vision literature is the intelligent scissors/live wire algorithm [129, 287], which inputs boundary points from a user which are then connected via the shortest path in the dual graph. Figure 6.14 gives an example of using this algorithm to target the same object in the photograph.

One advantage of a primal algorithm is that it easily extends to alternate embeddings (or no embedding). However, since the dual graph (or lack thereof) depends on the availability of an embedding, these algorithms are less portable between prob-

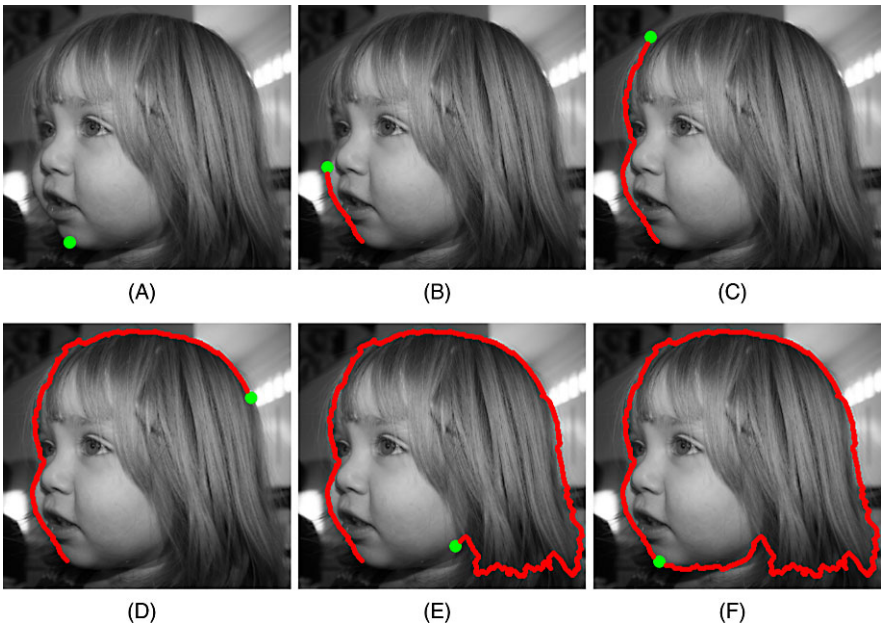


**Fig. 6.14** Targeted segmentation using a dual algorithm. The dual algorithm operates on the dual graph to find a set of edges in the dual graph comprising the boundary of the nodes in the primal. We applied the intelligent scissors/live wire algorithm [129, 287] to segment the girl's head in this image. At each step of the algorithm the user determines a point on the boundary (shown by the *bright circle*) and a shortest path is calculated between the new point and the previous point (shown by the *thick line*). The process is continued until the boundary is completed. In this example, six points total were manually chosen

lems. An example of this change in embedding is the application of a segmentation algorithm in a two-dimensional image compared to a three-dimensional image. A primal segmentation algorithm can just as easily be applied in two or three dimensions, since the output is a labeling of the pixels in two dimensions or the voxels in three dimensions. However, a dual algorithm that finds a one-dimensional boundary in two dimensions that isolates two-dimensional regions of an image will not subdivide the three-dimensional image. Consequently, such an algorithm must be reworked to apply to find the boundary of regions in three dimensions, i.e., to find two-dimensional boundary surfaces. This topic was addressed earlier in Sect. 6.1.2.1 in which we showed how the intelligent scissors/live wire algorithm could be extended to three-dimensional segmentation by computing a minimal surface in the dual complex. Instead of inputing points and computing a shortest path between them, the three-dimensional algorithm inputs closed contours and computes a minimal surface between them. Figure 6.15 gives three examples from medical imaging of this dual algorithm in three dimensions. Note the bottom image of the aorta which splits into two pieces at the iliac branch. Although one contour was input above the split and two contours were input below the split, the resulting segmentation naturally merges these contours into a single surface.

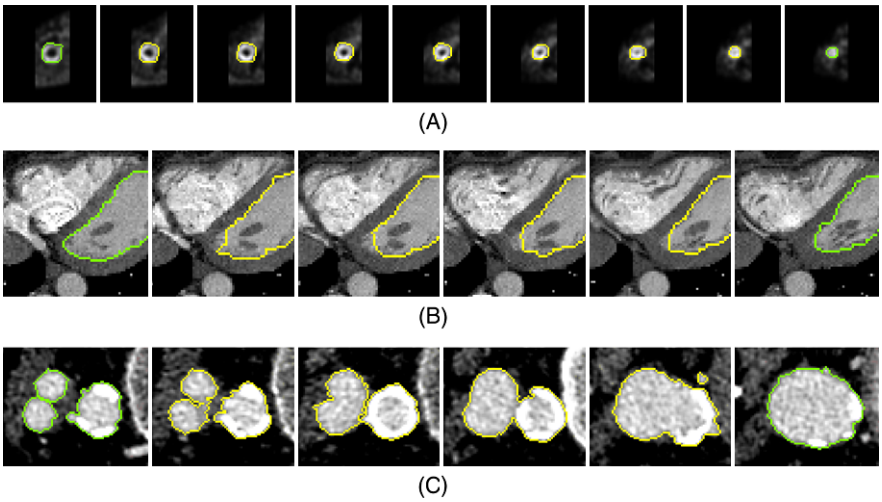


**Fig. 6.15** Dual segmentation in three dimensions applied to the segmentation of three-dimensional medical data. Several slices of three-dimensional volumetric data are shown. In each example, the *green contours* were placed on the bounding slices and the intermediate *yellow contours* represent the minimum-weight surface between these contours. (**A**) SPECT cardiac data. (**B**) CT cardiac data. (**C**) CT aorta near iliac branch. Multiple closed contours may be placed and a single surface may be found that splits accordingly to accommodate the prescribed boundaries. Note that not all slices between the closed contours are displayed

### *6.5.2 Social Networks*

Clustering is a common task in the study of social networks, where it is often known as *community finding*. In a social network, each node typically represents a person (or group) and the edges represent a social connection (e.g., friendship). If the network is weighted, the edge weight indicates the strength of the social connection. Since a larger weight typically represents a stronger social bond, we will treat these edge weights as *affinity weights* (similar to conductances in the circuit theory analogy).

In this application, we consider the social network studied by Zachary [415], which has appeared many times in the study of social networks (e.g., [148, 292]). Zachary observed a university karate club for two years which consisted of 34 members. The club split when the karate instructor "Mr. Hi" wanted more money and the club president "John A" fired him. Zachary's goal in studying this social network was to determine if it was possible to predict the faction joined by each member based purely on the social structure of the club. The social structure of the club was modeled as a network in which each member was associated with a node and an edge was assigned between two nodes if the two members met in some venue outside the club (e.g., the campus pub, common classes, outside karate tournaments, etc.). Each edge was assigned a weight equal to the number of outside venues that the two members had in common.

Zachary's method of predicting the split was to designate "Mr. Hi" (node 1) as a source node and "John A" (node 34) as a sink node and then determine the minimum cut (maximum flow) between these nodes in the network. This approach may be seen as an example of the *targeted* clustering problem in which the goal was to isolate a particular cluster (Mr. Hi's cluster) from the club (John A's cluster). In fact, the max-flow/min-cut algorithm employed by Zachary corresponds precisely to the targeted clustering approach defined by solving the Basic Energy Model (6.6) with norm $p = 1$.

The max-flow/min-cut approach taken by Zachary correctly predicted the faction split for every member except for node 9. Zachary's algorithm assigns person 9 to John A's faction when, in fact, this person joined Mr. Hi's club. Zachary explains this discrepancy by noting that person 9 was indeed aligned with John A's faction but he would have had to give up his high belt ranking if he had joined with John A (because the new club planned to teach a different style of karate). In addition to Zachary's method, we employed the random walker ($p = 2$) and geodesic ($p = \infty$) algorithms to this targeted clustering problem. Both of these algorithms also fail to correctly predict the faction joined by person 9 after the split. Additionally, the geodesic algorithm incorrectly predicts the faction joined by person 14. Although person 14 is clearly more strongly connected to Mr Hi's faction, person 14 is connected directly to Mr. Hi and John A with the same weight. In this case, the shortest path from person 14 to Mr. Hi takes the same weight as the shortest path from person 14 to John A and the tie was broken in the wrong direction. This situation illustrates the common problem with the geodesic algorithm that it is sensitive to the characteristics of the shortest path without considering the global structure of the network. However, the geodesic algorithm is still mostly correct (with only two incorrect assignments) and very fast compared to the other algorithms (especially on large networks). Figure 6.16 displays Zachary's karate club network with the true split and the targeted clustering obtained by the three algorithms.

### *6.5.3 Machine Learning and Classification*

Machine learning is a vast topic. The goal of a machine learning algorithm is to reduce a phenomenon of interest to a series of quantities that may be used to identify the phenomenon and to generalize determinations made about the phenomenon to other objects with a similar set of quantities. Machine learning techniques are generally divided into *supervised* learning and *unsupervised* learning techniques. A supervised learning technique inputs a small set of labeled training data that is used to build a model that may be applied to label unseen data. In contrast, an unsupervised learning technique inputs unlabeled data with the goal of producing a division of this data into meaningful labels which may be applied to future unseen data. Between supervised and unsupervised techniques lies the body of *semi-supervised* and *transductive learning* techniques which input a small amount of labeled data *and* a larger amount of unlabeled data which is used together to build a model for the labeling of unseen data.

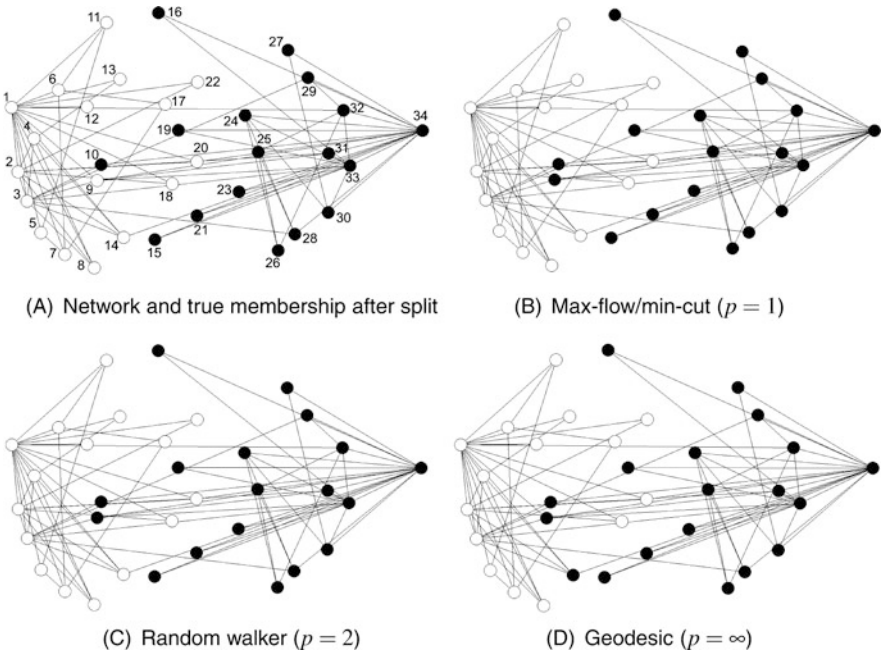


**Fig. 6.16** Zachary's Karate Club network [415]. Zachary tried to predict the actual split of the karate club into two groups based on the social interactions of each member. This problem may be viewed as an instance of the targeted clustering problem in which the leaders of the two factions, "Mr. Hi" (person 1) and "John A" (person 34), are treated as known and Zachary's goal was to predict the membership of each other member

Learning and clustering are intimately related topics. The goal of both learning and clustering algorithms is to assign a labeling to data. Further, untargeted clustering algorithms are similar to unsupervised learning algorithms in the sense that both types of algorithm must find a good method for separating data into different labels. In contrast, the targeted clustering algorithms are similar to supervised learning algorithms (and particularly semi-supervised and transductive learning algorithms) in the sense that these algorithms must take a small amount of labeled training data and determine how to assign labels to a large number of unlabeled data points. The primary difference between learning and clustering is that a clustering algorithm is generally focused on labeling a particular set of data which is available, while a learning algorithm must be able to generalize the clustering to label new (unseen) data. However, transductive learning algorithms are generally defined to label only a certain set of unlabeled data points (provided during training) [73], and therefore the targeted clustering algorithms described in this chapter are more properly viewed as transductive learning algorithms in the context of machine learning. Additional work has been done to further blur the lines between machine learning and clustering by examining how to extend some clustering algorithms to unseen data (see, e.g., Refs. [28, 29]).

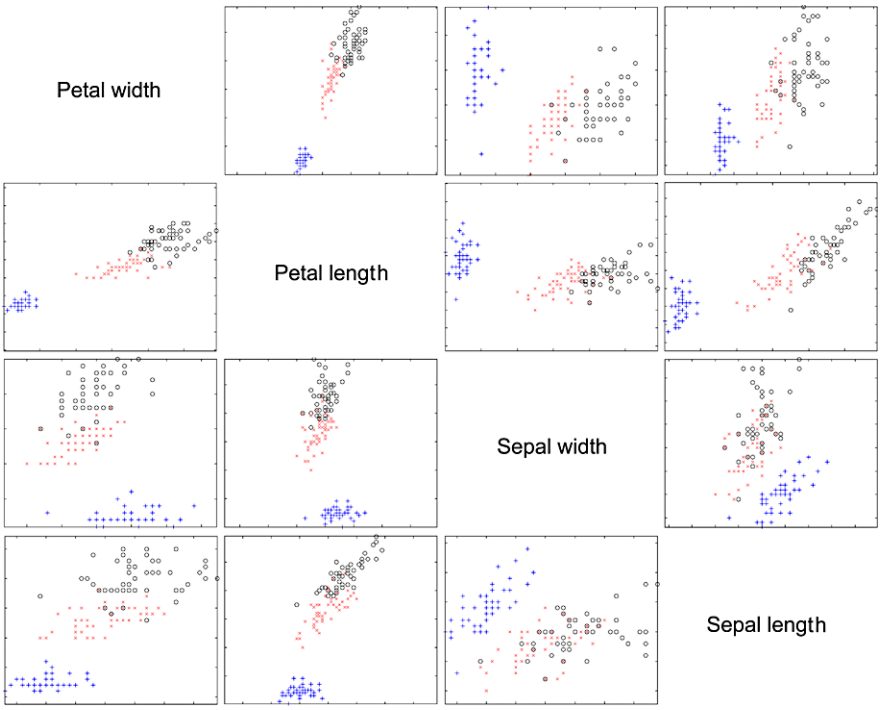


**Fig. 6.17** A scatter plot of the Fisher iris data [138]. Each flower is described by four measurements: petal width, petal length, sepal width and sepal length. The *setosa* are indicated with the marker '+', *Iris versicolor* with the marker 'x', and *Iris virginica* with 'O'. Each row/column represents a variable and the off-diagonal image shows the row variable on the $x$-axis and the column variable on the $y$-axis. The measurements from the *setosa* are well-separated, while the *Iris versicolor* and *Iris virginica* measurements overlap substantially

In this application example, we do not address the issue of extending a clustering algorithm to unseen data, but rather to address the use of clustering to group data used to describe an object that we wish to model. To illustrate these clustering approaches, we use the classical Fisher iris data [138]. Fisher's paper was an early example of the standard approach used now in machine learning. Fisher wanted to determine whether or not it was possible to distinguish three types of irises from a set of measurements of each iris. Fisher's data consisted of 50 samples of each of three different species of iris, *setosa*, *Iris virginica* and *Iris versicolor*. Four measurements were taken from each flower: the petal width, petal length, sepal width and sepal length. Based on these measurements, the data for the *Iris virginica* and *Iris versicolor* are substantially overlapping while the data describing the *setosa* is well-separated from the other two. Figure 6.17 provides a scatter plot illustrating the distribution of the data for each type across measurement dimensions.

We first applied the untargeted clustering algorithms to determine if they could produce the three clusters. The initial step was to generate a graph from the data in which each data point is treated as a node and the nodes are connected with edges

**Table 6.3** The Rand index, as defined in (6.40), for the clusters produced by the various untargeted clustering algorithms when applied to untargeted clustering of the Fisher iris dataset

| | $k$-Means | Mumford–Shah | Isoperimetric | Spectral | Normalized cuts |
|---|---|---|---|---|---|
| Rand index | 0.8623 | 0.8797 | 0.8797 | 0.8797 | 0.8797 |

via a $K$-nearest-neighbor approach (see Chap. 4) in which we set $K = 25$ and all edge weights to unity. Since the algorithms described above partition the graph into only two parts, the algorithms were run recursively to produce three partitions. In each case, the energy value of the partition was used to determine when to stop the recursion with the stop criterion set such that three partitions were produced. The $k$-means algorithm was also included (as a variant of the minimization of the Mumford–Shah energy), although this algorithm did not need to be run recursively in order to produce the prescribed three partitions.

The quality of the clustering produced by minimizing each objective function was evaluated against the true labeling by using the Rand index [315]. The Rand index between two labelings, $x_1$ and $x_2$, is defined as the ratio

$$\text{Rand}(x_1, x_2) = \frac{a+b}{\binom{n}{2}}, \tag{6.40}$$

where $a$ represents the number of pairs of nodes which share the same label in $x_1$ while also sharing the same label in $x_2$, and $b$ represents the number of pairs of nodes which have different labels in $x_1$ while also having different labels in $x_2$. If $x_1$ and $x_2$ represent the same clustering then $\text{Rand}(x_1, x_2) = 1$, while if the two clusterings do not agree on any pair of points then $\text{Rand}(x_1, x_2) = 0$. Table 6.3 gives the Rand index for the clustering obtained by the various primal untargeted clustering algorithms described in the chapter. In this case, all of the algorithms produced a reasonable clustering in which the quality was equal for all algorithms except the $k$-means algorithm which performed slightly worse.

We next tested the primal targeted clustering algorithms on the iris data in a transductive learning context. Samples were chosen randomly from each of the three classes. The number of samples was increased from one to fifty in order to track the performance of each algorithm with respect to the number of training samples. At fifty samples, all of the data is used to train/test, so the labeling will be perfect (thus the Rand index will be equal to unity). For each number of samples, 100 trials were run with randomly generated samples for each algorithm and the Rand index of the resulting labelings were computed with respect to the ground truth. Figure 6.18 displays the results for the targeted clustering algorithms applied to this data using a $K$-nearest neighbor graph in which $K = 5$ and again with $K = 30$. In both cases, the edges were weighted using the Welsch function as a function of the Euclidean distance between data points (in normalized feature space). Figure 6.18 displays plots of the mean Rand index across 100 trials for each number of samples and the standard deviation of the Rand index across trials. These plots allow us to make several observations about the behavior of these algorithms in this scenario. First, all three algorithms behave roughly the same. However, the geodesic

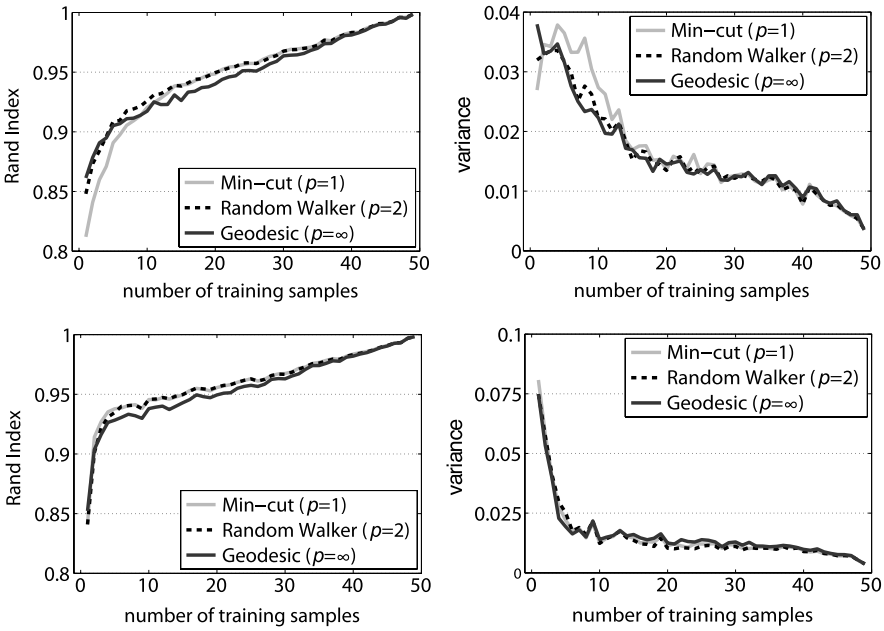


**Fig. 6.18** Targeted clustering using randomly sampled seeds on the Fisher iris dataset. Each trial was run 100 times with an increasing number of samples. The $x$-axis represents the number of training samples and the $y$-axis represents the Rand index. The *left column* shows the mean Rand Index and the *right column* shows the variance of the Rand index. *Top row*: Run on a $K$-Nearest Neighbor graph with $K = 5$. *Bottom row*: Run on a $K$-Nearest Neighbor graph with $K = 30$. We see that the geodesic algorithm consistently has the worst performance for a mid-range number of labeled nodes, but generally the lowest variance in performance of the three algorithms

algorithm performs better than the other two with a very small number of samples but lags behind the others as the number of samples increases. This effect is more pronounced for the graph with lower connectivity. Secondly, all of the algorithms gave a better average performance with higher connectivity, but the variance of the quality was greater for a low number of samples.

### *6.5.4 Gene Expression*

Grouping genes by their expression pattern can be useful to deduce the gene regulatory network, classify groups into particular phenotypes (e.g., cancerous or non-cancerous) or to match gene expression with other macroscopic anatomical features such as a physiological atlas [41, 76, 198, 368, 408]. In this example, we follow Bohland et al. [41] to determine whether the grouping of gene expression profiles match the standard anatomical grouping.

Bohland et al. [41] studied the C57BL/6J mouse brain in the Allen Brain Atlas [255] and compared the clustering of locations based on their gene expression pat-

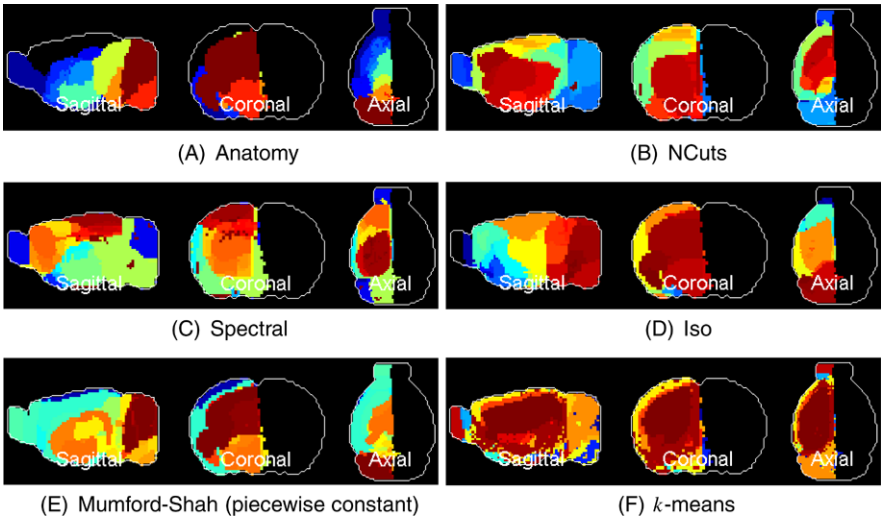


**Fig. 6.19** Do the gene expression patterns match the accepted anatomy? In this example, we follow the Bohland et al. experiment [41] to demonstrate how well a clustering of gene expression patterns match the accepted anatomy. In each experiment, we chose parameters for each algorithm to generate a partitioning with the highest Rand index as compared to the anatomical atlas. The Rand index for each partitioning were, NCuts: 0.881, Spectral: 0.8783, Isoperimetric: 0.8767, Mumford–Shah: 0.8845, $k$-means: 0.8678. Consequently, it could be argued that there is a strong correspondence between anatomy and gene expression profile, but that the relationship is not perfect. Note how the $k$-means clustering is spatially fragmented due to the lack of any geometric regularization, as opposed to the Mumford–Shah result which is effectively $k$-means with geometric regularization

terns with a classically-defined anatomical reference atlas. The goal was to assess the level of correspondence between molecular level and higher-level information about brain organization. Each voxel in the sample was described by 3041 genes which were deemed to be consistent across experimental observations. Bohland et al. reduced these 3041 genes to produce a feature space in which each anatomical location is represented by a tuple consisting of 271-dimensions. They then applied a $k$-means clustering to this data. We replicated the methodology of Bohland et al. [41] using the untargeted clustering algorithms described in this chapter. Figure 6.19 displays views of the clusterings obtained by the various untargeted primal clustering methods. In each case, we weighted the edge weights using an $\ell_\infty$ norm to measure the distance between the (reduced) gene expression data tuples, which was then input into a Welsch function (see Chap. 4).

We can also use the tools of targeted clustering to determine how well a particular anatomical structure matches a gene expression pattern. In order to examine this question, we generated labeled seeds from the anatomical atlas by setting the central portion of the medial axis of each region as seeds. We then targeted the striatum by setting all of its seeds to foreground (‘1’) and seeds from all the other regions as background (‘0’). Figure 6.20 displays the results obtained from applying the targeted clustering algorithms to this data. Each of the algorithms produce a seg-

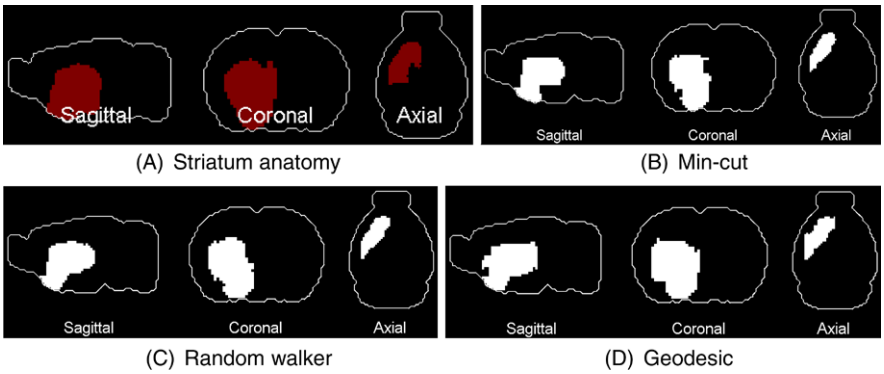


**Fig. 6.20** Targeted clustering of the striatum. By using seeds derived from the anatomy of the striatum, we applied the various targeted clustering algorithms to determine the gene expression pattern associated with this anatomical structure

mentation that matches the anatomy well. Therefore, this analysis suggests that the gene expression profile of striatum is fairly well separated from the gene expression profile of surrounding structures.

## 6.6 Conclusion

In this chapter, we showed how the variational models of Chap. 5 could be applied to the targeted, untargeted and semi-targeted clustering problem on both a primal and dual complex. Essentially, the filtering algorithms modeled the denoised data as having low spatial frequency (possibly with discontinuities). These models could easily be applied to clustering by modeling the *cluster labels* as having a low spatial frequency (possibly with discontinuities). Clustering appears in many applications in image processing, machine learning and complex network analysis. We also showed how to apply these clustering models to the clustering of higher-order cells, such as edges to permit flow clustering.