Robust Graph Mode Seeking by Graph Shift

Hairong Liu LHRBSS@GMAIL.COM
Shuicheng Yan ELEYANS@NUS.EDU.SG
Department of Electrical and Computer Engineering, National University of Singapore, Singapore

Abstract

In this paper, we study how to robustly compute the modes of a graph, namely the dense subgraphs, which characterize the underlying compact patterns and are thus useful for many applications. We first define the modes based on graph density function, then propose the graph shift algorithm, which starts from each vertex and iteratively shifts towards the nearest mode of the graph along a certain trajectory. Both theoretic analysis and experiments show that graph shift algorithm is very efficient and robust, especially when there exist large amount of noises and outliers.

1. Introduction

Graph is an important representation approach, especially for data which cannot be represented in vectorial form. Even for data with vectorial form, many algorithms are essentially founded on graph representation, such as graph based image segmentation (Shi & Malik, 2000) and graph based data clustering (Ng et al., 2002).

A dense subgraph refers to a coherent subset of vertices in a graph and such cohesiveness is not easy to be disturbed by noises and outliers, thus the dense subgraphs can robustly indicate key patterns. For example, in World Wide Web, dense subgraphs might be communities or link spam; in telephone call graph, dense subgraphs might be groups of friends or families. In these situations, the graphs are usually very sparse in global, but have many dense subgraphs of different sizes, these dense subgraphs are the natural focal points for studying graph structure and extracting the underlying meaningful patterns.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

In computer science, the pursue of maximal cliques (cliques that cannot be enlarged) (Ouyang et al., 1997) is a fundamental problem and has been widely studied for decades. The Motzkin-Straus theorem (Motzkin & Straus, 1983) has proven that solving maximal clique problem is equivalent to finding the maxima of a quadratic function, namely the graph density function used in this work. Thus, the maximal cliques correspond to the modes of the graph and the maximal clique problem is actually the mode seeking problem on graph. The weighted counterpart of maximal clique, dominant set, also corresponds to the mode of the graph, and has ever been used for pairwise clustering (Pavan & Pelillo, 2007).

In machine learning literature, the one-class clustering/classification problem (Gupta & Ghosh, 2005; Crammer et al., 2008), which finds a small and coherent subset of points within a given data set, rises naturally in a wide range of applications, from finding gene-modules to extracting documents' topics, where many data points are irrelevant to the task at hand, or in applications where only positive examples are available. Such coherent subset of points forms dense subgraphs, thus, one-class clustering/classification problem is closely related to the mode seeking problem on graph.

Owing to the commonality of the dense subgraphs in many applications, many algorithms have been proposed (Pavan & Pelillo, 2007; Ouyang et al., 1997) for computing such subgraphs. These algorithms are however usually heuristic and can only find partial dense subgraphs, and at the same time, these algorithms are usually very demanding in both computational cost and memory requirement.

In this paper, we propose the graph shift algorithm, which can find all significant dense subgraphs, with low time and memory complexity. The graph shift algorithm is very similar with mean shift algorithm (Comaniciu & Meer, 2002), a well-known non-parametric feature space analysis technique. Both algorithms can start from any start point, shift along a certain tra-

jectory, finally reach the nearest mode. Thus they can be considered as evolutionary strategies that perform multi-start global optimization; however, mean shift operates directly on the feature space, while our graph shift operates on the affinity graph. In many situations, we can only obtain the affinity graphs, some even with no corresponding vectorial representation, thus, our method can be considered to be a complement method of mean shift. The same as mean shift, graph shift can be used for robust model seeking and clustering analysis.

The main contributions of this work are two-fold. 1) We define the modes of a graph and analyze its property. Although the mode of a graph is widely used in many areas, it has not been well defined and systematically analyzed. 2) We propose the graph shift algorithm, which can efficiently approach the nearest mode of a graph from any start point. This algorithm provides a robust tool for mode seeking and cluster analysis on graph.

The rest of the paper is organized as follows. We define the modes of a graph and analyze its properties in Section 2, then we present the graph shift algorithm in Section 3. The experimental evaluation of our algorithm for robust mode seeking and clustering is performed in Section 4, and we conclude this work in Section 5.

2. Modes of Graph

In this section, we first define graph density, then define the modes of a graph. Finally we analyze the properties of graph mode, which shall guide the inference of graph shift algorithm presented in the next section.

2.1. Notations of Graph

A graph G is represented as $G = (V, E, w)$, where $V = \{v_1, \cdots, v_n\}$ is the vertex set, n is the number of vertices, $E \subseteq V \times V$ is the edge set, and $w : E \to \mathrm{IR}_+^*$ is the (nonnegative) weight function. Vertices in G correspond to data points, edges represent neighborhood relationships, and edge-weight reflects similarity between a pair of linked vertices. As is customary, we represent the graph G with the corresponding weighted adjacency (or similarity) matrix, more specifically, an $n \times n$ symmetric matrix $A = (a_{ij})$, where $a_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in E$, and $a_{ij} = 0$ otherwise. Clearly, if there are no self-loops, all the diagonal elements of A are zeros. In this paper, we only consider graphs with no self-loops.

Let $S = \{1, \cdots, n\}$ be the index set of the vertex set V, for any subset $T \subseteq S$, a subgraph G_T of G with vertex set $V_T = \{v_i | i \in T\}$ is introduced and the corresponding edge set is $E_T = \{(v_i, v_j) | (v_i, v_j) \in E, i \in T, j \in T\}$.

2.2. Probabilistic Coordinate On Graph

The probabilistic coordinate on graph G is defined as a mapping $X : V \to \Delta^n$, where $\Delta^n = \{x \in \mathrm{R}^n : x \geq 0 \text{ and } |x|_1 = 1\}$, that is, the mapping from the vertex set V to the standard simplex of R^n. Each point $x \in \Delta^n$ represents a probabilistic combination of vertices, called *probabilistic cluster*, and x_i, the i-th component of x, represents the probability of this probabilistic cluster contains vertex v_i. Since under the probabilistic coordinate, a point x uniquely corresponds to a probabilistic cluster and vice versa, we will refer to them interchangeably. For a point x, $x_i = 0$ means that this probabilistic cluster does not contain vertex v_i. The indices of all nonzero components of x constitute its *support*, denoted as $\sigma(x) = \{i | x_i \neq 0\}$, and it corresponds to a subgraph $G_{\sigma(x)}$. Particularly, the coordinate of the probabilistic cluster containing only vertex v_i is I_i, whose i-th component is 1, and other components are 0.

Since a_{ij} represents the affinity value between vertex v_i and vertex v_j, the affinity value between point x and point y can be defined as follows:

$$a(x, y) = \sum_{i,j} a_{ij} x_i y_j = x^T A y \tag{1}$$

Note that $a(I_i, I_j) = a_{ij}$, which is consistent with the definition for the weights of edges.

2.3. Graph Density and Modes

The affinity value between a point x and itself is $a(x, x) = x^T A x$, abbreviated by $g(x)$. As a good cluster should be the one in which strongly associated vertices should have edges with large affinity values connecting each other in the graph, $g(x)$ is a natural measure of the cohesiveness (dense) of the probabilistic cluster x, which is referred to as graph density in this work.

We may reveal the meaning of the graph density from the graph constructed from a data set $D = \{d_i | i = 1, \cdots, n\}$ in feature space, where vertex v_i corresponds to data d_i, $w(v_i, v_j) = K(d_i, d_j), i \neq j$ and $w(v_i, v_i) = 0$. K is a kernel function defined on feature space. The probabilistic coordinate x can be considered to be a distribution, namely the probability of choosing vertex v_i (thus, data d_i) is x_i. Suppose we sample this distribution N ($N \to \infty$) times, then the number of data d_i is Nx_i. At point d_i, the density is $f(d_i) =$

$\frac{\sum_j Nx_jK(d_i,d_j)}{N}$, then the average density of these N points are:

$$f_{av} = \frac{\sum_i Nx_i f(d_i)}{N} = \sum_{i,j} x_i K(d_i, d_j) x_j \qquad (2)$$

Since $K(x,x) \geq K(x,y)$, the average density will reach maxima when these points are identical to one point d_i, that is, when $x = I_i, i \in S$. However, if we only consider the contribution of a point to other points, not considering the contribution of the point to itself, that is, we set $K(x,x) = 0$, then the average density of these N points are:

$$f_{av} = \sum_{i \neq j} x_i K(d_i, d_j) x_j = x^T A x = g(x) \qquad (3)$$

where A is the adjacency matrix of graph G. It means that the graph density is the limit of average density when $N \to \infty$ and not considering self contribution.

Note the differences between graph density and density in feature space: graph density considers all mutual contribution within a probabilistic cluster, not considering self contribution and the contribution of points outside this probabilistic cluster; while the density in feature space consider the contribution of all other points to one point. Owing to not considering the contribution of the points outside the probabilistic cluster, graph density is not sensitive to outlier; also because the graph density considers all mutual contribution within the probabilistic cluster, it is more robust to noises.

Definition 1. The modes of a graph G are local maximizers of graph density $g(x) = x^T Ax$.

For a point x, the subgraph corresponds to x is $G_{\sigma(x)}$, composed by all vertices whose indices are in $\sigma(x)$. If x^* is a local maximizer (mode) of $g(x)$, then $G_{\sigma(x^*)}$ is a dense subgraph. Such dense subgraphs are very important in many applications. For example, 1) they are maximal cliques in graph analysis; 2) they represent the core of a cluster in cluster analysis; and 3) they represent common patterns in common pattern detection problem.

2.4. Properties of Modes

Since the modes are local maximizers of $g(x)$, to find these modes, we need to solve the standard quadratic optimization problem (StQPs) (Bomze, 2002):

$$\begin{cases} \text{maximize} & g(x) = x^T Ax \\ \text{subject to} & x \in \Delta^n \end{cases} \qquad (4)$$

It is a constrained optimization problem, and a local maximizer $x^* \in \Delta^n$ must satisfy the Karush-Kuhn-Tucker (KKT) condition for problem (4), i.e., the first-order necessary conditions for local optimality. That is, there exist $n+1$ real constants (Lagrange multipliers) $\mu_1, \cdots, \mu_n$ and λ, with $\mu_i \geq 0$ for all $i = 1, \cdots, n$, such that:

$$(Ax^*)_i - \lambda + \mu_i = 0 \qquad (5)$$

for all $i = 1, \cdots, n$, and $\sum_{i=1}^n x_i^* \mu_i = 0$.

Since both x_i^* and μ_i are nonnegative for all $i = 1, \cdots, n$, the latter condition is equivalent to saying that $i \in \sigma(x^*)$ implies $\mu_i = 0$. Hence, the KKT conditions can be rewritten as:

$$(Ax^*)_i \begin{cases} = \lambda, & i \in \sigma(x^*); \\ \leq \lambda, & i \notin \sigma(x^*). \end{cases} \qquad (6)$$

Note that $(Ax^*)_i = a(x^*, I_i)$, the affinity value between cluster x^* and vertex i, thus (6) has an obvious geometric meaning, which is summarized in the following theorem.

Theorem 1. If x^* is the mode of graph G, then 1) the affinity values between x^* and all the vertices in the subgraph $G_{\sigma(x)}$ are identical to $g(x^*)$; 2)the affinity values between x^* and other vertices of graph G are not larger than $g(x^*)$. At the same time, if x^* satisfies 1) and 2), then it is the mode of graph G.

Proof: According to (6), $x^{*T} Ax^* = \sum_i x_i^* (Ax^*)_i = \sum_i x_i^* \lambda = \lambda$. Since KKT is a necessary condition, then if x^* is the mode, it must satisfies 1) and 2). At the same time, when equality constraints are affine functions, inequality constraints and the objective function are continuously differentiable invex functions, KKT

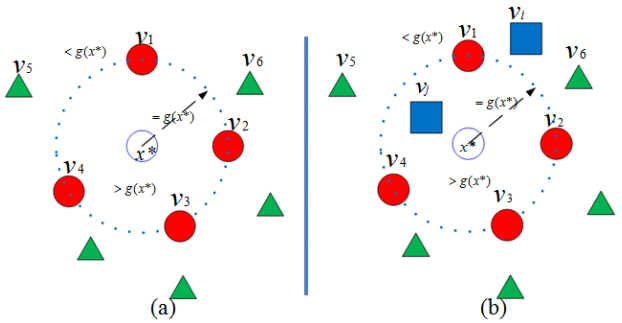

Figure 1. (a) Geometric explanation of graph mode. x^* is the mode of graph G, all the vertices (red points) belonging to $G_{\sigma(x^*)}$ are on the sphere $\{y|a(x^*,y) = g(x^*)\}$ and all other vertices are within the space $\{y|a(x^*,y) \leq g(x^*)\}$. (b) The relation between the mode of graph and the mode of its subgraph. x^* is the mode of a subgraph, whether it is the mode of graph G depends on whether there is no vertex (blue point) within the space $\{y|a(x^*,y) > g(x^*)\}$. For better viewing, please see color pdf.

condition is also sufficient, thus if x^* satisfy 1) and 2), it is the mode of graph G.

Theorem 1 tells that if x^* is a mode of graph G, then all the vertices belonging to subgraph $G_{\sigma(x)}$ are on the sphere $\{y|a(x^*,y) = g(x^*)\}$, and all the vertices not belonging to subgraph $G_{\sigma(x)}$ are in the space $\{y|a(x^*,y) \leq g(x^*)\}$. Figure 1(a) illustrate such scenario, x^* is a mode, $\sigma(x^*) = \{v_1, v_2, v_3, v_4\}$, they all lie on the sphere $\{y|a(x^*,y) = g(x^*)\}$.

2.5. Modes of Subgraph

In many situations, graph G is very large, that is, it has many vertices and edges, it is very inefficient to deal with them as a whole. Note that $\sigma(x^*)$, the support of the mode x^*, usually contains very limited number of vertices and $\tilde{x}^* = \{x_i^*|x_i^* > 0\}$, which retains all nonzero components of x^*, is also the mode of the subgraph $G_{\sigma(x^*)}$. Subgraph $G_{\sigma(x^*)}$ contains only $m = |\sigma(x^*)|$ vertices, thus much easier to deal with. This phenomenon inspires us to search for the modes of G through the modes of its subgraphs.

For a subgraph G_T, suppose one of its mode is x_T^*, according to Theorem 1.

$$(Ax_T^*)_i \begin{cases} = \lambda, & i \in \sigma(x_T^*); \\ \leq \lambda, & i \notin \sigma(x_T^*). \end{cases} \tag{7}$$

where $\sigma(x_T^*) \subseteq T \subseteq S$.

By adding zeros to the components whose index are in the set $S - T$, we can expand the m dimensional vector x_T^* to n dimensional vector x^*. The problem is whether x^* is also the mode of graph G. The following theorem answers this question.

Theorem 2. A mode x_T^* of the subgraph G_T is also the mode of graph G if and only if for all vertex v_i, $a(x^*, I_i) \leq g(x^*) = g_T(x_T^*)$, $i \in S - T$, where x^* is obtained from x_T^* by adding zeros to the components whose indices are in the set $S - T$.

Proof: Since x^* is obtained from x_T^* by adding zeros to the components whose indices are in the set $S - T$, then $\sigma(x^*) = \sigma(x_T^*)$, $g(x^*) = g_T(x_T^*)$, and

$$(Ax^*)_i \begin{cases} = \lambda, & i \in \sigma(x^*); \\ \leq \lambda, & i \in T, i \notin \sigma(x^*); \\ = a(x^*, I_i), & i \in S - T. \end{cases} \tag{8}$$

where $\lambda = g(x^*) = g(x_T^*)$.

If $a(x^*, I_i) \leq g(x^*) = \lambda$ for all $i \in S - T$, according to Theorem 1, x^* is also the mode of graph G. If $a(x^*, I_i) > g(x^*) = \lambda$ for some $i \in S - T$, then x^* does not satisfy the KKT condition, according to Theorem 1, thus it is not the mode of graph G.

Theorem 2 has an intuitional geometric meaning: Since the affinity values between a mode x^* of graph G and the vertices in subgraph $G_{\sigma(x^*)}$ are identical to a constant value λ, we can regard the sphere $\{y|a(x^*,y) = \lambda\}$ to be a separating surface, if no newly added points fall into the space $\{y|a(x^*,y) > \lambda\}$, then x^* is still the mode of the expanded graph; otherwise, x^* is not a mode of the expanded graph. Figure 1(b) illustrates this point, if the newly added point is v_i, then x^* is still the mode of the expanded graph; however, if the newly added point is v_j, x^* is not the mode of the expanded graph.

3. Graph Shift Algorithm

There are many algorithms to obtain a local maxima of StQP (4), from any initialization $x(0)$. In this section, we first review the most popular method, replicator dynamics (Weibull, 1997). Then we will present the neighborhood expansion procedure, which can expand the support of the mode of a subgraph to its neighborhood. The combination of these two steps forms our graph shift algorithm.

3.1. Mode Seeking by Replicator Dynamics

Replicator dynamics, which arises in evolutionary game theory, is the most popular method to find the local maxima of StQP (4). Given an initialization $x(0)$, corresponding local solution x^* of StQP (4) can be efficiently computed by the discrete-time version of first-order replicator equation, which has the following form:

$$x_i(t+1) = x_i(t)\frac{(Ax(t))_i}{x(t)^T Ax(t)}, \ i = 1, \cdots, n. \tag{9}$$

It can be observed that the simplex Δ^n is invariant under these dynamics, which means that every trajectory starting in Δ^n will remain in Δ^n. Moreover, it has been proven in (Weibull, 1997) that, when A is symmetric and with nonnegative entries, the objective function $g(x) = x^T Ax$ strictly increases along any nonconstant trajectory of (9), and its asymptotically stable points are in one-to-one correspondence with strict local solutions of StQP (4).

Note that Equation (9) has a property: if $x_i(t) = 0$, then $x_i(t+1) = 0$ and $x_i(t)$ does not affect the computation of $x_j(t), j \neq i$, which means that during the evolution procedure, replicator equation (9) can drop vertices, but it cannot automatically expand the vertices. Thus, from a subgraph G_T, it can find the mode of G_T, but this mode may not be the mode of graph G. In the following subsection, we will present the method of expanding vertices from the modes of subgraphs.

3.2. Neighborhood Expansion from Modes of Subgraphs

From a mode x_T^* of the subgraph G_T, according to Theorem 2, we can judge whether it is also the mode of graph G. If yes, then no further step is required; if not, we need to find an update vector Δx, $g(x^*+\Delta x) > g(x^*)$, where x^* is obtained from x_T^* by adding zeros to the components whose indices are in the set $S-T$.

Since x^* is not a mode of $g(x)$, according to Theorem 2, there are some vertices v_i, $a(x^*, I_i) > g(x^*)$, $i \in S-T$. We define a vector v with

$$v_i = \begin{cases} 0, & i \in \sigma(x^*) \\ \max(a(x^*, I_i) - g(x^*), 0), & i \notin \sigma(x^*) \end{cases} . \quad (10)$$

Suppose $s = \sum_i v_i$, $\zeta = \sum_i v_i^2$ and $\omega = \sum_{i,j} v_i a_{ij} v_j$, then $s > 0$ and $\zeta > 0$. We update x^* in direction $b = \begin{cases} -x_i^* s, & i \in \sigma(x^*) \\ v_i, & i \notin \sigma(x^*) \end{cases}$. That is, decreases the possibility of vertices belonging to current mode and increases the possibility of vertices with large rewards.

Suppose $g(x^*) = \tilde{\lambda}$, then $(Ax^*)_i = \tilde{\lambda}, i \in \sigma(x^*)$,

$$\begin{aligned} & g(x^* + tb) - g(x^*) \\ = \; & 2t(1-ts)(\zeta + \tilde{\lambda}s) - ts(2-ts)\tilde{\lambda} + \omega t^2 \\ = \; & -(\tilde{\lambda}s^2 + 2s\zeta - \omega)t^2 + 2\zeta t \end{aligned} \quad (11)$$

According to $x_i^* \geq 0$, $i \in \sigma(x^*)$, $x_i^* - t x_i^* s \geq 0$, then $t \leq \frac{1}{s}$. When $\tilde{\lambda}s^2 + 2s\zeta - \omega \leq 0$, the increase from $g(x^*)$ to $g(x^* + tb)$ will reach maximum at $t^* = \frac{1}{s}$; When $\tilde{\lambda}s^2 + 2s\zeta - \omega > 0$, the increase from $g(x^*)$ to $g(x^* + tb)$ will reach maximum at $t^* = \min\{\frac{1}{s}, \frac{\zeta}{\tilde{\lambda}s^2 + 2s\zeta - \omega}\}$, and the update vector is:

$$\Delta x = t^* b, \quad (12)$$

which is called *neighborhood expansion vector*.

The update from x^* to $x^* + tb$ not only increases the value of $g(x)$, but also expands the support $\sigma(x^*)$ to its neighborhood, which is the desirable property.

3.3. Graph Shift Procedure

The replicator dynamics and the neighborhood expansion procedure have complementary properties: 1) replicator dynamics can efficiently drop vertices, but neighborhood expansion cannot; 2) neighborhood expansion can expand the support, but replicator dynamics cannot. Their combination leads to the graph shift algorithm, which is summarized in Algorithm 1.

Algorithm 1 is an EM-style procedure, the neighborhood expansion procedure expands current subgraph to its neighborhood, thus provides a much larger lower bound of $g(x)$, which corresponds to the mode of current subgraph; while replicator dynamics procedure evolves towards this lower bound, and guarantees to reach this lower bound. These two steps iterate until a local maxima is reached. In the neighborhood expansion procedure, only nearest vertices are added into current subgraph, and in the replicator dynamics procedure, most of vertices are dropped, and only a very compact cluster of vertices are retained. Thus, our graph shift algorithm always operates on small subgraphs, which is very efficient, both in time and memory.

Algorithm 1 Graph Shift Algorithm

Input: The affinity matrix A of graph G, the start point x (a vertex, or a cluster of vertices)
repeat
  Evolve x towards the mode of subgraph $G_{\sigma(x)}$ by replicator dynamics (9)
  if x is not the mode of graph G **then**
    Update x by neighborhood expansion vector
  end if
until x is the mode of graph G

The main computation load is the replicator dynamics procedure, which evolves toward the mode of current subgraph. Suppose the average number of edges in the subgraph is h, and the average number of iterations for the replicator equation is t, then the time complexity of the replicator dynamics procedure is $O(th)$, and the space complexity is $O(h)$. The total time complexity of graph shift procedure is then $O(lth)$, where l is the number of iterations for the shrink and expansion phases.

4. Experiments

We evaluate the proposed graph shift algorithm on two tasks: mode seeking and cluster analysis. Since under real-world scenarios, the graph usually contains considerable noises and outliers, in our experiments, we mainly focus on these scenarios.

4.1. Detecting Common Pattern as Mode Seeking on Graph

A pattern is a set of feature points with fixed relative spatial relation. Two instances of a common pattern not only need to be similar in corresponding feature points, but also need to have similar spatial layout. The common pattern problem is: given two sets of feature points, are there any common patterns between them and where they are? We will show that this

problem is identical to mode seeking on a graph with large amount of noises and outliers. Thus, the common pattern problem is a good test-bed to evaluate the effectiveness and robustness of our graph shift algorithm.

Suppose two sets of feature points are P and Q, with n_P and n_Q feature points, respectively. Each feature point contains local features and coordinates. For each point p in P, according to the local features, we may find some similar points q in Q. Each such pair (p, q) is a possible correspondence and all such pairs form the correspondence set $C = \{(p, q)|p \in P, q \in Q, p \text{ and } q \text{ have similar local features}\}$.

We construct a graph G based on C with each vertex of G representing a correspondence in C. Edge $e = (v_i, v_j)$ connects vertex v_i and v_j, and reflects the relation between correspondences c_i and c_j. For two correspondences $c_i = (p_i, q_i)$ and $c_j = (p_j, q_j)$, suppose the distance between p_i and p_j in the first image, and the distance between q_i and q_j in the second image, are $l_{p_ip_j}$ and $l_{q_iq_j}$, respectively. Obviously, to align these two correspondences, we need to scale the second image by a factor of $l_{p_ip_j}/l_{q_iq_j}$.

Suppose the correct scale factor of a common pattern is s, we can define w_{ij}, the weight of edge $e = (v_i, v_j)$, as follows:

$$w_{ij} = \exp(-\frac{|l_{p_ip_j} - sl_{q_iq_j}|^2}{\varsigma^2}) \qquad (13)$$

Obviously, under such definition, common patterns correspond to dense subgraphs in G, which is illustrated in Figure 2. The common pattern corresponding to the mode x^* can be recovered from the vertices of subgraph $G_{\sigma(x^*)}$, with every vertex corresponding to a correct correspondence.

Graph G has two characteristics: 1) There are large amount of vertices and most of them represents incorrect correspondences. The number of vertices is nearly n_Pn_Q, but only m correct correspondence, where m is the number of points in common pattern. 2) Many edges have large weights. Since the weight only reflects scale relation of two correspondences, many edges, such as the edges between incorrect correspondences, may accidently have large weights. The number of such edges is usually several order of magnitude than the number of edges between correct correspondences. These two characteristics pose a great challenge on mode seeking.

We first conduct an experiment on point sets and compare our method with spectral method in (Leordeanu & Hebert, 2005), which is the state-of-art method to find correct correspondences. We generate a point set T with n_T points, add Gaussian noise $N(0, \eta)$ and rotate it to obtain two version of T. We then add outliers to them by randomly selecting points in the same region and obtain the two point set P and Q. Since the points themselves are not distinctive, the number of vertices is n_Pn_Q. We fix the number of points in the common pattern, $n_T = 15$, and vary the number of outliers. Both algorithms ran on the same data sets over 30 trials and both the mean performance curves and the curves of one standard deviation below the mean are plotted. We score the performances of these two methods by counting how many correspondences agree with the ground truths.

The result is shown in Figure 3. Obviously, the spectral method is sensitive to outliers and its performance curve drops fast; however, our proposed graph shift method works remarkably well. This is because the

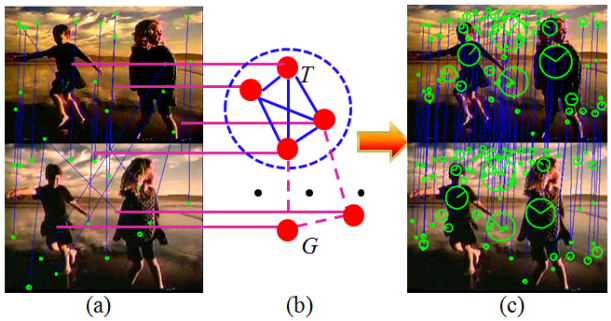

Figure 2. Common pattern detection corresponds to mode seeking on graph G. Find all candidate correspondences shown in (a) by local features (for clarity, only a small subset of the candidate correspondences are shown), and then construct the graph G in (b). The common pattern corresponds to the dense subgraph (mode) T of G.

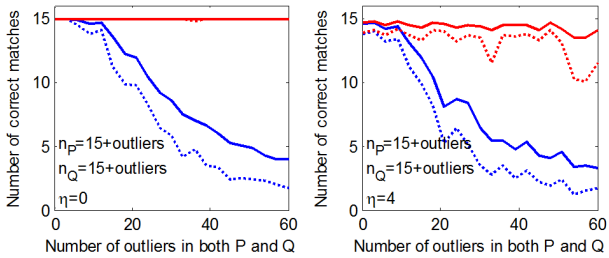

Figure 3. Performance curves for our method vs. the spectral method in (Leordeanu & Hebert, 2005). The mean performance (number of correct matches) is shown as a solid red line (our method) and a solid blue line (spectral method), One *std* below the mean is shown as red dotted lines for our method and blue dotted lines for the spectral method. Left figure: no Gaussian noise ($\eta = 0$). Right figure: added Gaussian noise ($\eta = 4$).

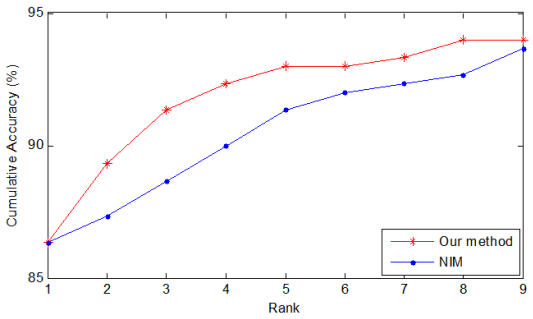

Figure 4. Comparison of cumulative accuracy of near-duplicate image retrieval on Columbia database.

eigenvectors are affected by all weights, especially all the large weights, no matter they are correct or not; however, our graph shift method just try to find a subgraph with all edges have high weights and such weights are usually correct.

We also conduct an experiment on near-duplicate image retrieval, which plays an important role in many multimedia applications. Near-duplicate images usually have a large common pattern, thus judging whether two images are near-duplicate or not is also a mode seeking problem on graph. The experiment is conducted on the Columbia database, which contains 150 near-duplicate pairs and 300 non-duplicate images (600 images in total). For fair comparison, we first rank all images using global features as done in (Zhu et al., 2008), then re-rank the images in top 50 based on the size of detected common patterns. We use SIFT features to find all possible point correspondences and the weight w_{ij} is computed by (13). Since just at correct scale, common patterns correspond to the mode of graph, we search for 11 scales. In Figure 4, the retrieval performance is plotted and compared with the state-of-art method, called NIM (Zhu et al., 2008), which finds common patterns by non-rigid mapping. Obviously, our method gets better cumulative accuracies (ratio between correctly retrieved images in top N images and total number of query images), which verifies that our method can correctly find the modes of the graph.

4.2. Cluster Analysis

Graph shift procedure is a natural clustering tool, and all the vertices shift toward the same mode should belong to a cluster. According to the need, there are two variants of clustering methods based on graph shift: 1) The number of clusters is unknown. In this case, we regard each mode with large density as the core of a cluster, and all the vertices shift towards this mode

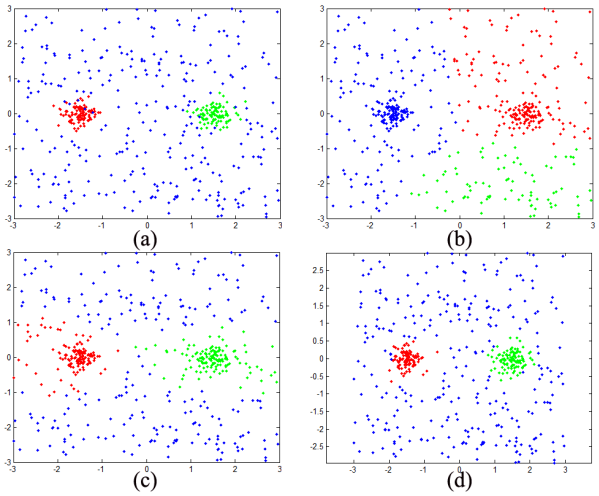

Figure 5. Clustering on data with uniform distributed background points. (a) the data set, (b) clustering result of k-means, (c) clustering result of spectral clustering, (d) clustering result of our method. For better viewing, please see original color pdf.

belong to this cluster, other vertices can be assigned to one cluster according to affinity value or left ungrouped. 2) The number of clusters K is specified. In this case, we just select K modes with the largest densities, and then assign other vertices to these K clusters.

We first consider the problem of extracting dense clusters from cluttered background. Many real world problems belong to this kind, such as image segmentation and perceptual grouping. Because many points should not belong to any clusters, such as the background of an image, the partition methods, such as k-means and spectral clustering method, are not expected to work well, due to their insisting on partitioning all the input data into coherent groups. Our method, on the contrary, appears to be particularly suited for such applications, since it allows one to extract as many clusters as desired, while leaving the remaining points (namely, the clutter) un-grouped.

To illustrate this point, consider the toy data set shown in Figure 5(a), which contains two dense clusters of Gaussian random points surrounded by uniformly distributed clutter points. For comparison, we choose k-means and spectral clustering, two representative partition methods, based on vectorial representation and affinity data, respectively. The clustering results of k-means, spectral clustering and our method are illustrated in (b),(c),(d), respectively. As expected, both k-means and spectral clustering cannot work well; while our method can automatically uncover the two dense clusters and separate them from the background.

Table 1. Clustering results and computational cost for spectral clustering (SC), affinity propagation (AP) and our method on the shape matching affinity data.

	SC	AP	Our Method
Clusters	70	64	70
Precision	73%	76.5%	85.36%
NMI	88.63%	88.27%	91.41%
Time(seconds)	11.6904	514.9922	1.2320

We also conduct an experiment on the affinity data from shape matching. The database is MPEG-7 shape database, there are 70 categories and each category contains 20 shapes. For each two shapes, we calculate their matching score (affinity value) using certain shape matching method, thus obtain a 1400×1400 affinity matrix. Such affinity data has no corresponding vectorial representation; at the same time, it usually contains a large amount of noises, since for different pairs of shapes, their matching scores are computed independently, and the matching method may produce wrong results on some pairs of shapes. We compare our method with the spectral clustering and affinity propagation (Frey & Dueck, 2007), both of which are classical methods based on affinity matrix. The result is shown in Table 1. The performance of clustering is measured by both the precision and normalized mutual information (NMI). Both spectral clustering and our method can specify the number of clusters, but affinity propagation can only approximate the specified number. Obviously, our method outperforms spectral clustering and affinity propagation, and a possible explanation is that the affinity matrix contains many noises and our method is inherently noise-resistent. At the same time, our method spends much less time. Note that AP needs to search an appropriate preference value, thus it runs the clustering algorithm many times and spends very long time.

5. Conclusions and Future Work

In this paper, we define the mode of a graph, which corresponds to a coherent subset of vertices, thus is inherently robust to noises and outliers. We propose the graph shift algorithm to robustly compute all modes. The experimental results show that our algorithm is surprisingly robust to noise and outliers. Future works include more efficient methods on large scale data and applications in one-class clustering/classification problems.

6. Acknowledgement

This work is supported by National Research Foundation/Interactive Digital Media Program, under research Grant NRF2008IDMIDM004-029, Singapore.

References

Bomze, M. Branch-and-bound approaches to standard quadratic optimization problems. *Journal of Global Optimization*, 22:17–37, 2002.

Comaniciu, D. and Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 603–619, 2002.

Crammer, K., Talukdar, P., and Pereira, F. A rate-distortion one-class model and its applications to clustering. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 112–119, 2008.

Frey, J. and Dueck, D. Clustering by passing messages between data points. *Science*, 315:972–974, 2007.

Gupta, G. and Ghosh, J. Robust one-class clustering using hybrid global and local search. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 273–280, 2005.

Leordeanu, M. and Hebert, M. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the International Conference on Computer Vision*, pp. 1482–1489, 2005.

Motzkin, T. and Straus, G. Maxima for graphs and a new proof of a theorem of Turan. *Theodore S. Motzkin: selected papers*, pp. 311–314, 1983.

Ng, Y., Jordan, I., and Weiss, Y. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 2, pp. 849–856, 2002.

Ouyang, Q., Kaplan, D., Liu, S., and Libchaber, A. DNA solution of the maximal clique problem. *Science*, 278:446–448, 1997.

Pavan, M. and Pelillo, M. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:167–172, 2007.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

Weibull, W. *Evolutionary game theory*. The MIT press, 1997.

Zhu, J., Hoi, H., Lyu, R., and Yan, S. Near-duplicate keyframe retrieval by nonrigid image matching. *ACM Multimedia*, pp. 41–50, 2008.