Clustering and the Biclique Partition Problem

D. Bein
Department of Computer Science
University of Texas at Dallas
Dallas, TX 75080, U.S.A
siona@utdallas.edu

W. Bein[†]
Center for the Advanced Study of Algorithms
School of Computer Science
University of Nevada
Las Vegas, NV 89154, USA
bein@cs.unlv.edu

Z. Meng
Department of Computer Science
University of Texas at Dallas
Dallas, TX 75080, U.S.A
zhaobmeng@yahoo.com

L. Morales[*]
Department of Computer Science and Information Systems
Texas A&M University-Commerce
Commerce, TX 75429, U.S.A
Linda_Morales@tamu-commerce.edu

C. O. Shields, Jr.
Department of Computer Science
University of Texas at Dallas
Dallas, TX 75080, U.S.A
cshields@utdallas.edu

I. H. Sudborough
Department of Computer Science
University of Texas at Dallas
Dallas, TX 75080, U.S.A
hal@utdallas.edu

Abstract

A technique for clustering data by common attribute values involves grouping rows and columns of a binary matrix to make the minimum number of submatrices all 1's. As binary matrices can be viewed as adjacency matrices of bipartite graphs, the problem is equivalent to partitioning a bipartite graph into the smallest number of complete bipartite sub-graphs (commonly called "bicliques"). We show that the Biclique Partition Problem (BPP) does not have a polynomial time α-approximation algorithm, for any $\alpha \geq 1$, unless P=NP. We also show that the Biclique Partition Problem, restricted to whether at most k bicliques are sufficient (i.e. BPP(k)) for each positive integer k, has a polynomial time 2-approximation algorithm. In addition, we give an O(VE) time algorithm and BPP(2), and an O(V) algorithm to find an optimum biclique partition of trees.

1. Introduction

We consider the problem of clustering data based on attributes of the objects as expressed in a binary matrix. That is, the objects are represented by rows and the attributes by columns, with a "1" ("0") in row *i* and column *j* if object *i* has (does not have) attribute *j*.

Binary matrices arise in many applications, including biology, market analysis, information retrieval, data mining, and network security analysis [5,10,11,12,15]. Determining the underlying structure inherent in these matrices is important. Specifically, we want algorithms that group rows and columns such that all common values in the corresponding row groups and column groups are 1's, and one obtains a minimum number of groups.

In market basket analysis, for example, the rows of the matrix represent customers, the columns represent products, a "1" in position (*i*,*j*) of the matrix indicates that customer *i* purchased product *j*, and a "0" indicates that the customer did not. Then, a cluster of rows and columns whose values are all 1's could reveal groups with common purchasing preferences and trends, an important part of market research [2]. In the case of attack graphs, a component of network security analysis, such clusters might identify particular vulnerabilities in the network, bottlenecks, or densely connected sub-networks [11]. An informal description of applications in other areas can be found in [16].

* Research supported in part by NSF award no. DUE-0416901
† Research conducted while on sabbatical from the University of Nevada, Las Vegas. Sabbatical support from UNLV is acknowledged.

Of particular interest are the clusters of 1's formed by the association of rows and columns. A *cross association*, for the purpose of this paper, is a partition of the rows and columns into disjoint groups, where each column group is paired with a specific row group, and the values of the sub-matrices thus formed are all 1's [2,6].

The problem of finding cross associations in a binary matrix is equivalent to the problem of finding bicliques in a bipartite graph. A *bipartite graph* is a graph whose vertices can be divided into two disjoint sets such that all edges connect vertices in one set with a vertex in the other set. A *biclique* is a complete bipartite subgraph. That is, given a bipartite graph $G=(X,Y,E)$ and a subset $A \subseteq (X \cup Y)$, the subgraph of G formed by A is a biclique if, for all $x \in (A \cap X)$ and $y \in (A \cap Y)$, $\{x,y\} \in E$.

A binary matrix can be viewed as an adjacency matrix for a bipartite graph in the following way: The rows of the binary matrix correspond to the first set of nodes X in the bipartite graph, the columns correspond to the second set of nodes Y, a "1" in row i and column j indicates that an edge exists between node i and node j in G, while a "0" indicates that there is no edge. Paired groups of rows and columns whose values are all "1" in the matrix then directly correspond to subsets of vertices in the graph that induce bicliques. Submatrices of 1's, therefore, represent complete bipartite sub-graphs, i.e. *bicliques*.

Bicliques have been studied extensively. Peeters [13] has shown that the problem of finding a biclique in G with the maximum number of edges is NP-complete. Hochbaum [8] describes approximation algorithms for several problems involving bicliques, including the problem of deleting a minimum weight collection of edges so that the residual graph is a biclique. Mishra *et al*. [9] consider the problem of finding a maximal subgraph that is almost a biclique, in the sense that every vertex in A is adjacent to at least (1-ε) vertices in B. The biclique edge covering problem asks for the minimum number of bicliques to cover the edges of G. Results for a special case of the biclique edge covering problem are described by Bezrukov *et al*. [1]. Szeider [14] defines a total biclique cover as a collection of disjoint bicliques such that every vertex in the set X is in one of the bicliques. Note that this definition does not require that every vertex in the set Y is in one of the bicliques. Szeider [14] shows that the problem of determining if a bipartite graph has a total biclique cover is NP-complete. This problem is not the same as the Biclique Vertex Partition Problem, as a total biclique cover as defined by Szeider [14] need not cover all vertices.

In previous work [6], we described the cross association problem as the problem of finding a set of disjoint bicliques that includes all vertices in the graph. A *biclique vertex partition* (or a *biclique partition*) of a bipartite graph G is a collection $\{B_i\}$ of disjoint sets of vertices, such that each B_i induces a biclique, and each vertex is in some B_i. The associated decision problem is:

Biclique Vertex Partition Problem (BPP)
Instance: A finite bipartite graph $G=(X,Y,E)$ and a positive integer k.
Question: Is there a biclique vertex partition of G consisting of at most k subsets?

We showed that the problem is NP-complete, and provided a heuristic to find a biclique cover of an arbitrary bipartite graph in polynomial time [6]. In addition, we showed that a bipartite graph in which every vertex is incident to at least one edge has a biclique vertex partition.

In this paper, we extend our work on BPP in several ways. We show that BPP has no constant factor polynomial time approximation algorithm (unless P=NP). We also consider the fixed parameter variant of BPP, called BPP(k):

Biclique Vertex Partition Problem (k) (BPP(k))
Instance: A finite bipartite graph $G=(X,Y,E)$.
Question: Is there a biclique vertex partition of G consisting of at most k subsets?

BPP(k) defines a family of problems, for each $k \geq 0$ where the value of k is fixed. We give an $O(VE)$ algorithm for BPP(2), and a polynomial time 2-approximation algorithm for BPP(k) when $k \geq 3$. This is not a contradiction to our proof of the inapproximabililty of the Biclique Vertex Partition Problem, described in the next section, because the polynomial run time for the 2-approximation algorithm for BPP(k) grows with k and hence is not polynomial when k is variable.

We also give an $O(V)$ algorithm to compute an optimum solution for BPP when restricted to trees. Finally, we conclude with open problems and observations.

2. Inapproximability of the Biclique Partition Problem

We now show that BPP cannot be approximated by a polynomial time k-approximation algorithm for any constant k. This result depends on previously established results for the Dominating Set problem.

Given a graph $G=(V,E)$, a *dominating set* for G is a subset V' of V such that every vertex in V is either in V' or is adjacent to a vertex in V'. The associated decision problem is defined as follows:

Dominating Set
Instance: A finite graph $G=(V,E)$ and a positive integer k.
Question: Is there a dominating set in G of size k?

The Dominating Set problem is a well known NP-complete problem [4]. In addition, it is known that Dominating Set is not k-approximable for any constant k (unless P=NP), see [7]. Our proof of the inapproximability of BPP, given in Theorem 2, makes use of this fact.

We state Lemma 1 first; a proof is provided after Theorem 2.

Lemma 1. Given an arbitrary graph G, a graph G' can be created from G that has the following properties:
(a) If there is a dominating set of size k in G, then there is a biclique partition of size $2k$ in G'.
(b) If there is a biclique partition of size k in G', then there is a dominating set of size k in G.

We now show that:

Theorem 2. There is no polynomial time k-approximation algorithm for BPP for any constant k, unless P=NP.

Proof: For a contradiction, assume that there is a polynomial time constant factor approximation algorithm for BPP. We show, as a result, that a polynomial time constant factor approximation algorithm for Dominating Set must also exist. Since it is known that such an algorithm does not exist unless P=NP, this is a contradiction.

Let $G=(V,E)$ be a graph with a dominating set of size k. By Lemma 1, there is a graph G' that has a biclique partition of size at most $2k$. If a polynomial time α-approximation algorithm existed for BPP, for some constant α, we would obtain, by Lemma 1(a), a biclique cover of size at most $2\alpha k$ for G'. By Lemma 1(b), we would then obtain a dominating set for G of size $2\alpha k$.

Since α is a constant, this yields a polynomial time 2α-approximation algorithm for Dominating Set. This is known to be impossible.

We now prove Lemma 1.

Proof of Lemma 1: Let $G=(V,E)$ be an arbitrary graph and $D=\{v_1, v_2, ..., v_k\}$ be a dominating set for G. We construct a bipartite graph $G' = (V,V',E')$ from G as follows.

Create a set of vertices, V', that duplicate the vertices in V. That is, $V' = \{v' \mid v \in V\}$. For any $v \in V$, we call $v' \in V'$ the *sibling* of v, and vice versa. We can use the function SIB( ) to determine the sibling of a vertex in G'.

Let $E' = \{ \{x,y'\}, \{x', y\} \mid \text{for all } \{x,y\} \in E \}$. That is, for every edge $\{x,y\} \in E$, we add two edges to G'. Note that G' is a bipartite graph, since all edges connect a vertex in V with some vertex in V'.

Let $D^{\#} = \{ v, v' \mid v \in D\}$. That is, $D^{\#} = D \cup \text{SIB}(D)$.

For each $v \in D$, let $N(v)$ be a set of neighbors of v in G, not including v. Note that the sets $N(v)$ can be adjusted so that (a) V is partitioned by the sets $N(v) \cup \{v\}$ for all $v \in D$, and (b) $|N(v)| \geq 1$ for all $v \in D$.

For each $v \in D^{\#}$, we define $N'(v)$ as follows:
- $N'(v) = \text{SIB}(N(v))$, if $v \in V$
- $N'(v) = N(\text{SIB}(v))$, if $v \in V'$

Clearly, $D^{\#}$ is a dominating set for G'.

We now prove (a): If there is a dominating set of size k in G, then there is a biclique partition of size $2k$ in G'.

Consider the sets $\{v\} \cup N'(v)$ for all $v \in D^{\#}$. We show that these sets form a biclique partition of G' of size $2k$.

By the construction of G', v is connected to every vertex in $N'(v)$, for all $v \in V \cup V'$. Therefore, the sets $\{v\} \cup N'(v)$ form bicliques. Furthermore, the sets $N'(v)$ for $v \in V'$ are disjoint since the sets $N(x)$ are disjoint for $x \in V$. And finally, every vertex in G' will appear in one of these sets, since $D^{\#}$ is a dominating set for G'.

Since $|D^{\#}| = 2k$, the collection of sets $\{v\} \cup N'(v)$ for all $v \in D^{\#}$ form a biclique partition of G' of size $2k$.

We now prove (b): if there is a biclique partition of size k in G', then there is a dominating set of size k in G.

Let $W_1, W_2, ..., W_k$ be a partition of the vertex set $V \cup V'$ of G' such that each set W_i induces a biclique in G'. From this partition we create a set D by choosing an element $x_i \in V \cap W_i$ for each W_i and placing it in D. It is clear that $|D| = k$. We show that D dominates G.

Let y be any vertex in G that is not in D. As $W_1, W_2, \ldots, W_k$ is a biclique partition of G', the vertex SIB(y) must be in one of these sets, say W_i. Let x_i be the vertex put into D from the set W_i. There must be an edge connecting x_i with SIB(y) in G', as W_i is a biclique of G'. Therefore, by the construction of G', y is a neighbor of x_i in G. Since all vertices are either in D or are neighbors of vertices in D, D is a dominating set.

3. An O(*VE*) algorithm for BPP(2)

We now give an iterative polynomial time algorithm that determines if a bipartite graph has a biclique partition of size 2. The input to algorithm TWO-PARTITION is a bipartite graph $G=(X,Y,E)$. The algorithm answers "yes" if and only if G has a biclique partition of size 2. If the answer is "yes", a biclique partition is defined by the sets A and B. The algorithm iterates by processing the vertices in Y and the vertices in X in an alternating fashion until no more vertices can be added to the sets A and B.

```
TWO-PARTITION(G)
  B←∅
  Let s be an arbitrary vertex in X that is adjacent
  to some but not all of the vertices in Y
  A←{s}
  repeat
      for all vertices y∈Y such that y∉B
          if there exists a vertex a in A that is not
          adjacent to y
                then B←B∪{y}
      for all vertices x∈X such that x∉A
          if there exists a vertex b in B that is not
          adjacent to x
                then A←A∪{x}
  until no more vertices can be added to either A
  or B

  if A=X or B=Y
      then return "no"
  else
      A←A∪(Y\B)
      B←B∪(X\A)
      return "yes"
```

Theorem 4. Algorithm TWO-PARTITION returns "yes" if and only if the bipartite graph G has a biclique partition of size 2. If the answer is "yes", a biclique partition is defined by the sets A and B.

Proof: To see that TWO-PARTITION is correct, first consider the initial iteration. The vertex s is put into the set A and the set of all vertices not adjacent to s are put into B. Clearly, the vertices in set B cannot be in the same biclique as s. Now we consider the sets A and B iteratively created in the loop.

At the start of an arbitrary iteration of the loop, the set $B \subset Y$ consists of vertices in Y, each of which is not adjacent to at least one vertex in A. However, there may be other vertices in $Y\backslash B$ that are not adjacent to every vertex in A, as the set A has changed from the previous iteration. The current iteration adds to B all vertices in $Y\backslash B$ that are not adjacent to at least one vertex in A, i.e. the vertices in $Y\backslash B$ which cannot be part of a biclique that includes the set A. Thus, when the loop terminates, the set $Y\backslash B$ consists of all vertices in Y that are adjacent to every vertex in A. That is, the set $A\cup(Y\backslash B)$ induces a biclique.

A symmetric argument applies to the set $A\subset X$. Hence, the set $B\cup(X\backslash A)$ also induces a biclique.

Finally, note that the sets $A\cup(Y\backslash B)$ and $B\cup(X\backslash A)$ are indeed a biclique partition since

$$[A\cup(Y\backslash B)] \cup[B\cup(X\backslash A)]=[A\cup(X\backslash A)] \cup[B\cup(Y\backslash B)] = X\cup Y.$$

Now let $|V|=|X|+|Y|$ be the total number of vertices in G. Observe that each iteration of the loop adds at least one vertex to A or B, so the maximum number of iterations is $|V|$. During each iteration, at most $|E|$ edges need to be considered. Hence, the running time of TWO-PARTITION is O(*VE*).

4. A 2-approximation algorithm for BPP(*k*)

As shown in [6], BPP is NP-complete. In Section 2 we have further shown that BPP does not have a polynomial time algorithm with constant approximation ratio unless P=NP. The question of whether or not BPP(k) is NP-complete for $k \geq 3$ is open. Recall $G\in BPP(k)$ means that G has a biclique partition of size k, or less.

We now present a polynomial time procedure, LABEL(k), which either produces a biclique partition or size $2k$-2 or less, or "No" if a biclique partition of size k does not exist.

By repeated application of procedure LABEL, the following polynomial time 2-approximation algorithm APPROX_BPP(k) for BPP(k) can be derived:

```
APPROX_BPP(k)
  Input: a bipartite graph G=(X,Y,E)
  Output:
  (a) "No", a partition of size k does not exist, or
  (b) a biclique partition of size p, where p is no
      more than twice the size of the minimum
      size biclique partition.

APPROX_BPP(k)
  if G∈BPP(1)
    then stop with a single biclique partition
  if G∈BPP(2)
    then stop with a 2-biclique partition
  i = 3
  do
    execute LABEL(i)
    i = i + 1
```

until $i > k$ or output is "Yes" or "Inconclusive"
if output is "Yes" or "Inconclusive"
  then stop with a p-biclique partition, $p \leq 2k\text{-}2$
else if $i > k$ **then** stop with "No"

We now describe LABEL(k). Note that in Section 3, instead of using the sets A and B, we could have used two colors to denote the sets. We find it convenient to mark the set of vertices for each biclique with k colors $c_1, c_2, \ldots, c_k$ using one color for each biclique. Thus vertices in the biclique C_i will be colored c_i.

Given the bipartite graph $G=(X,Y,E)$, LABEL(k) arbitrarily chooses k vertex-disjoint edges and colors the two endpoints of each edge with appropriate colors. Such a choice, which we call a "guess", constitutes an initial non-covering biclique of size k. Furthermore, all uncolored vertices receive a label $c_1 \vee c_2 \vee \ldots \vee c_k$ to indicate that, at this stage of the procedure, all k colors are still available for those vertices.

Given a guess, we note that a vertex label at any step of the procedure LABEL(k) can be either:

1. c_i , meaning that this vertex must be part of the biclique C_i, or
2. $c_{i_1} \vee c_{i_2} \vee \ldots \vee c_{i_l}$, where $2 \leq l \leq k$, meaning that colors $c_{i_1}, c_{i_2}, \ldots, c_{i_l}$ have not yet been excluded for that vertex, or
3. $\varnothing$, meaning that there is no color available for this vertex so this guess cannot yield a k partition.

In Case 1, we call the vertex *assigned*, in Case 2 we call it *undetermined* and in Case 3 we call it *empty*.

LABEL(k) iteratively whittles down the possible colors for each *undetermined* vertex by determining which bicliques it cannot be part of. Each iteration updates the labels of all *undetermined* vertices in set X, and then updates the labels of all *undetermined* vertices in set Y.

We now describe how labels are updated for the vertices in X. (For the vertices in Y the procedure is symmetric.) A vertex with a given color in its label must be adjacent to every vertex of that color on the other side. More precisely, for each *assigned* vertex $v \in Y$ we do the following: If v is *assigned* c_j then delete c_j from the label of any vertex in X that is not adjacent to v. If some label now contains a single color c_j then the label is changed to *assigned* c_j. If all colors are deleted for a vertex then its label is changed to $\varnothing$. An iteration stops either when an *empty* vertex is found, or no new vertex is marked *assigned*.

Procedure LABEL(k) processes all guesses and halts with the following:

1. "No", if for all guesses the iteration stops with an *empty* node.
2. "Yes", if there exists a guess which has all vertices labeled *assigned*.
3. "Inconclusive", if there is a guess for which LABEL(k) stops with *assigned* and *undetermined* nodes, and there is no guess that stops with all vertices labeled *assigned*.

We show the following:

Lemma 4. If $G \in BPP(k)$, then there exists a guess such that procedure LABEL(k) outputs either "Yes" or "Inconclusive".

Proof: Suppose $C_1, C_2, \ldots, C_k$ form a biclique partition of $X \cup Y$ of size k. So every vertex $x \in X$ ($x \in Y$) is adjacent to every vertex in some $C_i \cap Y$ (respectively to some $C_i \cap Y$).

So let us choose for the sake of LABEL(k) an edge $e_i \in C_i$ for all $i = 1 \ldots k$.. Since $C_1, C_2, \ldots, C_k$ form a partition of the vertices, $e_1 , e_2 \ldots e_k$ have no common vertices. We show by induction on i that LABEL(k), starting with $D_1=\{e_1\}$, $D_2=\{e_2\}$, …, $D_k=\{e_k\}$, produces after the i^{th} iteration bicliques $D_1, D_2, \ldots D_k$ (containing only assigned nodes) such that $D_i \subseteq C_i$, for all i.

Basis step: (i=0) is obviously true.

Inductive step: Assume $D_i \subseteq C_i$ for all i. Without loss of generality, assume that in the next iteration a vertex x from Y is assigned color c_1, thus x is added to D_1. It follows that x is adjacent to every vertex in $D_1 \cap X$ and x is not adjacent to some vertex in $D_i \cap X$, for all $i > 1$. By the inductive hypothesis $D_i \subseteq C_i$, so it follows that x is not adjacent to some vertex in C_i, for all $i > 1$. Thus x is not in any set C_i, for all $i > 1$. As $\{C_i \mid 1 \leq i \leq k\}$ is a partition of $X \cup Y$, it follows that x must be in the set C_1. So $D_1 \cup \{x\} \subseteq C_1$.

Assume that LABEL(k) outputs "No" after some iteration. This means that, without loss of generality, there exists a vertex x in X such that x is not adjacent to every vertex in each $D_i \cap Y$. So x is not adjacent to very vertex in each $C_i \cap Y$, for all i. Thus x is not part of any C_i. This contradicts the assumption that $\{C_i \mid 1 \leq i \leq k\}$ form a biclique partition of G.

Corollary 5. If LABEL(k) stops with a vertex labeled $\varnothing$ for all guesses, then $G \notin BPP(k)$.

Proof: This is the contrapositive of Lemma 4.

We now show that if there is at least one guess where LABEL(k) finishes with only *assigned* or *undetermined* nodes, then we can construct a biclique partition of size $2k$-2.

We give an example for case k=3. In the example, the guessed edges are colored blue (c_1=b), green

(c_2=g), and red (c_3=r), respectively. The vertices are labeled accordingly (see Figure 1(a)).

We know that a 4-biclique partition is possible (see Figure 1(b)):

- the *undetermined* vertices in X with color blue in their labels are *assigned* blue
- the *undetermined* vertices in X with color green in their labels are *assigned* green
- the *undetermined* vertices in X with color red in their labels are *assigned* red
- the vertices *assigned* blue in Y and the *undetermined* vertices in X with color blue in their labels are *assigned* violet (a new, fourth color).

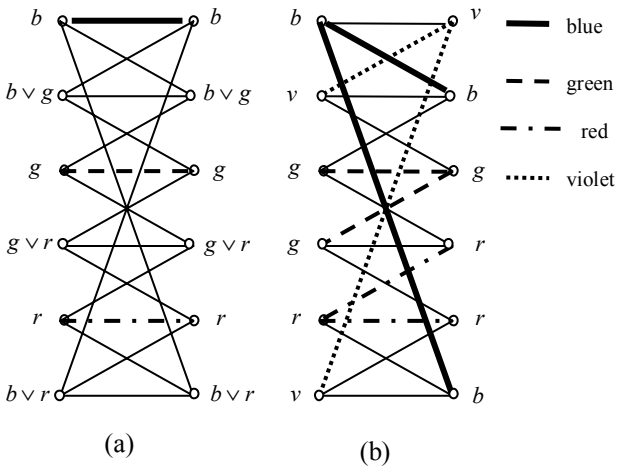

Figure 1. Example of a 4-biclique partition obtained from the "Inconclusive" state of LABEL(3)

We now show how to construct a biclique partition of size 2k-2 for general k, if LABEL(k) outputs "Inconclusive". To this end, consider the following sets: Let A_i be the set of *assigned* vertices in X with label c_i, and let B_i be the set of *assigned* vertices in Y with label c_i. Furthermore, define:

$\overline{A}_1$ = the set of *undetermined* vertices in X with labels containing c_1 and other colors

$\overline{A}_2$ = the set of *undetermined* vertices in X with labels containing c_2 and other colors except c_1

…

$\overline{A}_i$ = the set of *undetermined* vertices in X with labels containing c_i and other colors except $\{c_1, c_2 \ldots c_{i-1}\}$

…

$\overline{A}_{k-1}$ = the set of *undetermined* vertices in X with labels containing c_{k-1} and c_k.

Similarly, define for the vertices in Y the sets $\overline{B}_i$, for all $i = 1 \ldots k$.

We define the following biclique partition (see Figure 2) where sets that are part of the same biclique are connected through an edge:

$A_1 \cup \overline{B}_1$, $\overline{A}_1 \cup B_1$,

…

$A_i \cup \overline{B}_i$, $\overline{A}_i \cup B_i$,

…

$A_{k-2} \cup \overline{B}_{k-2}$, $\overline{A}_{k-2} \cup B_{k-2}$,

$(A_{k-1} \cup \overline{A}_{k-1}) \cup B_{k-1}$, $A_k \cup (\overline{B}_{k-1} \cup B_k)$.

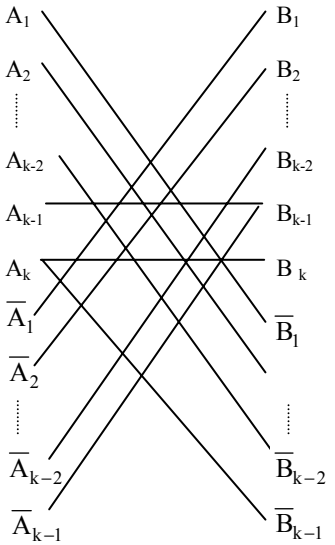

Figure 2. A biclique partition of size 2k-2

We note that there are cases when LABEL(k) outputs "Inconclusive" and a biclique partition of size smaller than 2k-2 can be found. For example, if all vertices on one side of the bipartite graph are labeled *assigned* and all other vertices are either *assigned* or *undetermined* then the guess provides a biclique partition of size k as follows: each *undetermined* vertex is assigned to the first color of its label.

Consider an example for k=3 (see Figure 3(a)). Let c_1 be blue, c_2 be green, and c_3 be red (abbreviated by b, g, and r, respectively). *Assigned* vertices are labeled appropriately. Edges connecting *assigned* vertices are also colored appropriately. The *undetermined* vertices are *assigned* following the rule mentioned above (see Figure 3(b)).

Also, if some of the combination of colors is missing, it is also possible to use less than 2k-2 colors for biclique partitioning.

For example, for the case k=3, if there are no *undetermined* vertices whose label is either $c_1 \vee c_2$, $c_1 \vee c_3$, $c_2 \vee c_3$ on one side of the graph, then three colors

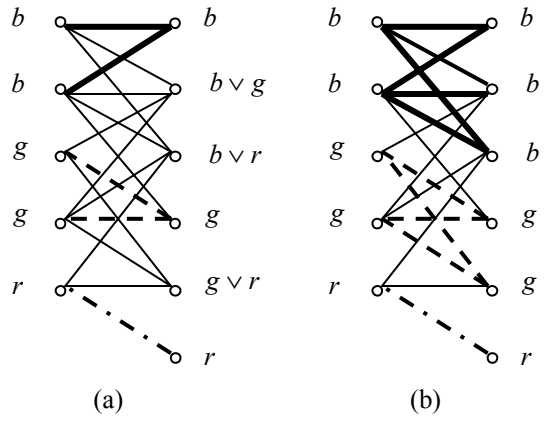

Figure 3. A 3-biclique partition for *k=3*

are enough (see for example, Figure 4, where there are no vertices that are labeled $c_1 \vee c_2$ on the left side). Let A be the set of all vertices assigned c_1, B the set of all vertices assigned c_2, C the set of all vertices assigned c_3, $A \vee B$ be the set of all *undetermined* vertices with label $c_1 \vee c_2$, $A \vee C$ be the set of all *undetermined* vertices with label $c_1 \vee c_3$, $B \vee C$ be the set of all *undetermined* with label $c_2 \vee c_3$, and $A \vee B \vee C$ the set of all *undetermined* with label $c_1 \vee c_2 \vee c_3$.

The union of all of A's vertices in X and all vertices in Y with a label that contains c_1 (i.e set A, $A \vee B$, $A \vee C$, $A \vee B \vee C$) is a biclique.

The union of all of B's vertices in X and all the vertices in Y that are uncovered and have a label that contains c_2 (i.e. set B, $B \vee C$) is another biclique.

The union of all of C's vertices in Y and all the vertices in X that have a label that contains c_3 (i.e. set C, $A \vee C$, $B \vee C$, $A \vee B \vee C$) is another biclique. These three bicliques form a biclique partition.

Theorem 6. For each fixed $k \geq 3$, APPROX_BPP(k) is a polynomial time 2-approximation algorithm for BPP(k).

Proof: The output of the algorithm APPROX_BPP(k) is either "No" or a partition of size $p \leq 2k\text{-}2$ that is no more than twice the minimum size biclique partition. From Lemma 4 it follows that if $G \in BPP(k)$, the output is a biclique partition of size p and p is at most twice the minimum size partition. From Corollary 5 it follows that if the output is "No", then $G \notin BPP(k)$.

An estimate of the run time of LABEL(k) is $O\left(\binom{E}{k} VE\right)$ because there are $O\binom{E}{k}$ many guesses, the number of iterations for a given guess is O(V), and the run time of an iteration is O(E).

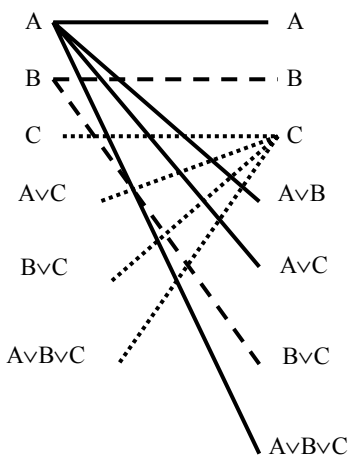

Figure 4. A 3-biclique partition for *k=3* when some labels are missing

An estimate of the run time of APPROX_BPP(k) is:

$$\text{Run time } (G \in \text{BPP}(1)) + \text{Run time } (G \in \text{BPP}(2)) + \sum_{i=3}^{k} \text{Run time (LABEL(i))}$$

$$= O(E) + O(VE) + \sum_{i=3}^{k} \binom{E}{i} VE = O(E^{k+1}V)$$

5. An exact O(V) algorithm for trees

We have seen that finding an optimum biclique partition is hard for arbitrary bipartite graphs. However, for certain classes of bipartite graphs, an optimum biclique partition can be found efficiently.

One such class is trees. Let us examine the biclique structure of an arbitrary tree T to introduce our algorithm. Let u be an interior node of T. Let B_u be the subtree consisting of u and some of the vertices adjacent to u. We call B_u a *starburst centered at u*. Note that B_u is a biclique, and, in fact, every biclique in T must be a starburst.

Now suppose that u has at least one neighbor v that is a leaf. Then an optimum biclique partition for T must include a starburst B_u which contains both u and v, because otherwise v would be isolated. Clearly all leaves adjacent to u must be included in the starburst B_u, but should non-leaf neighbors of u be included as well? The answer is yes, as long as no node is isolated in the process. We prove this in Lemma 7 below, but first, we describe a procedure TREE-PARTITION that uses this strategy to build an optimum biclique partition of T.

TREE-PARTITION takes a tree T as input and returns a collection of sets that induce an optimum biclique

partition of T. TREE-PARTITION is a recursive procedure that discovers the structure of T and assigns nodes to starbursts in a depth-first fashion. The sets B_u created by TREE-PARTITION represent the starbursts.

Initially no node is assigned to a biclique and an arbitrary node is chosen for the root. The procedure traverses the tree recursively, and when a leaf is found, it and all of its leaf siblings are added to the same biclique as their common parent. All non-leaf siblings that have not already been assigned to a biclique are also added to the parent's biclique. This greedy strategy ensures that every node is included in a biclique. Lemma 7 guarantees that bicliques created in this manner are in an optimum biclique partition.

When the recursion terminates, the root node may already be in a biclique. This happens when the root either has at least one leaf, or the root has a child that is the center of a starburst. If neither case holds, then the root must have at least one child, say v, that is in a biclique, say B_x, (*i.e.*, v is not the center of its biclique.) A final biclique is created that includes the root and its child v.

```
TREE-PARTITION(T)
  for each u∈V
      do B_u ← ∅
  Let r be an arbitrary vertex in V
  VISIT(r)
  if r has not been assigned to a biclique
      then let v be a child of r such that v∉B_v,
           and let B_x be the set that contains v
           do
               B_x ← B_x\{v}
               B_r ← {r,v}

VISIT(u)
  for each child v of u that is not a leaf
      VISIT(v)
  if there exists a child v of u such that has not
  been assigned to a biclique
      then do
           B_u ← {u}
           for each child v of u that has not been
           assigned to a biclique
                do B_u ← B_u ∪{v}
  else if there exists a child v of u such that v∈B_v
      then B_v ← B_v ∪{u}
```

Observe that TREE-PARTITION is a modification of the well-known depth-first search algorithm for graphs. Let $|V|$ and $|E|$ be the number of vertices and edges, respectively, in T. In the running of TREE-PARTITION each non-leaf vertex is visited exactly once. When each vertex is visited, its neighbors must be examined, and overall, at most $|E|$ neighbors will be examined during the execution of TREE-PARTITION. So, as for depth-first search, the running time for TREE-PARTITION is $O(V+E)$. Using the fact that $|E| = |V|-1$ for trees, the running time for TREE-PARTITION simplifies to $O(V)$.

We now return to the question of whether a non-leaf neighbor of a node u should be included in the starburst B_u. Observe that when a vertex is added to a starburst, it can be conceptually deleted from the tree. So consider the tree T at the instant when TREE-PARTITION decides whether or not vertex x should be added to a starburst. If x is added to a biclique, the tree $T\backslash\{x\}$ results. If, on the other hand, x is not added, the tree T is unchanged. Lemma 7 shows that is not a bad strategy to add x to B_u, because a biclique partition for the tree $T\backslash\{x\}$ is never larger than an optimum biclique partition for T.

Lemma 7. Let T be a rooted tree and let x be a leaf. Then a biclique partition for the tree $T\backslash\{x\}$ is never larger than an optimum biclique partition for T.

Proof: Let $P=\{B_i\}$ be an optimum biclique partition for T. Let $x\in B_j$ and let y be the parent of x. Assume that y has a parent, because otherwise the proof follows immediately. Let z be the parent of y. There are three cases to consider.

Case 1. B_j contains at least three vertices.
Since x is not the center of B_j, it can be removed without isolating vertices. So let $B_j'=B_j\backslash\{x\}$. Then $(P\backslash\{B_j\})\cup B_j'$ is a biclique partition of $T\backslash\{x\}$ with the same number of bicliques as P.

Case 2. B_j contains at exactly two vertices, x and y, and y's parent z is the center of some starburst B_k.
Let $B_k'=B_k\cup\{y\}$. Then $(P\backslash\{B_j,B_k\})\cup B_k'$ is a biclique partition of $T\backslash\{x\}$ with one less biclique than P. This case is illustrated in Figure 5.

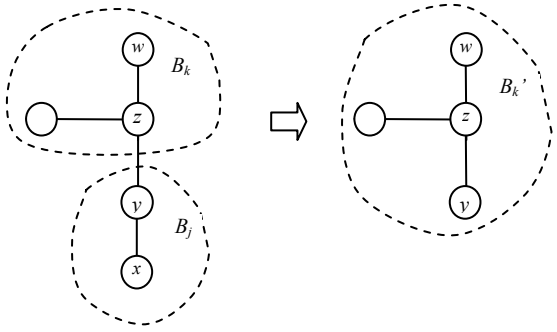

Figure 5. Case 2.

Case 3. B_j contains at exactly two vertices, x and y, and y's parent z is in some starburst B_k, but is not the center.

Case 3a. B_k has only two vertices, say z and w.
Let $B_k'=\{w,y,z\}$. Then $(P\backslash\{B_j,B_k\})\cup B_k'$ is a biclique partition of $T\backslash\{x\}$ with one less biclique than P.

Case 3b. B_k contains at least three vertices.
Let $B_k'=B_k\backslash\{z\}$ and $B_j'=\{y,z\}$. Then $(P\backslash\{B_j,B_k\})\cup\{B_j',B_k'\}$ is a biclique partition of $T\backslash\{x\}$ with the same number of bicliques as P.This case is illustrated in Figure 6.

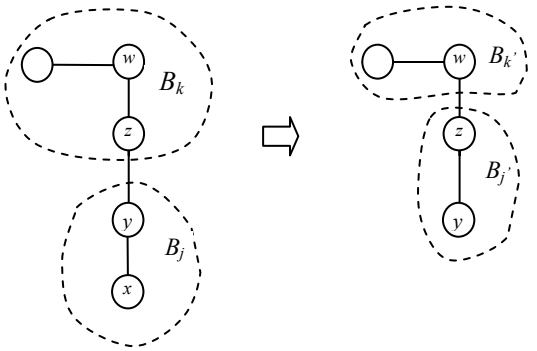

Figure 6. Case 3b.

Theorem 8. An optimum biclique partition for a tree can be found in O(V) time.

6. Conclusions

We have shown that, for any positive integer α, there is no polynomial time α-approximation algorithm for BPP unless P=NP. In contrast, for each positive integer $k\geq 3$, BPP(k) has a polynomial time 2-approximation algorithm and BPP(2) can be solved in *O(VE)* time. It is an interesting open problem whether or not BPP(k) is in P, for all $k \geq 3$, or whether BPP(3), for example, is NP-complete. We have given an optimum BPP algorithm for trees, and have obtained upper bounds on the number of bicliques needed for meshes, tori, and hypercubes. (The bounds for these special families are not given in this paper.) Polynomial time exact algorithms for these and other special graph families are not yet known.

Our algorithm for BPP(k), together with a heuristic given previously in [6], in our opinion provides practical tools for clustering of data. We expect that the running time and performance of these algorithms can be improved. This remains a topic of current research.

7. References

[1] S. Bezrukov, D. Froncek, S. Rosenberg, and P. Kovar, "On Biclique Coverings", *to appear Discrete Mathematics.*

[2] D. Chakrabarti, D. Modha, S. Papadimitriou, and C. Faloutsos, "Fully Automatic Cross-associations", *Proceedings of the 10th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2004)*, Seattle, WA, August 2004, pages 79-88.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms,* second edition, McGraw-Hill Publishing Co., 2001.

[4] M. R. Garey and D. S. Johnson**,** *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman Publishers, San Francisco, 1979.

[5] D. Hand, H. Mannila, P. Smyth (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA,. ISBN 0-262-08290-X.

[6] M. Heydari, L. Morales, C. Shields, I.H. Sudborough, "Computing Cross Associations for Attack Graphs and other Applications", *Hawaii International Conference on System Sciences*, January 3-7, 2007, Computer Society Press, 2007 (10 pages).

[7] D. Hochbaum, "Approximation Algorithms for NP-Hard Problems", *PWS Publishing Co*, 1997

[8] D. Hochbaum, "Approximating Clique and Biclique Problems", *J. Algorithms, 29(1),* Academic Press, Duluth, MN, 1998, pages 174—200.

[9] N. Mishra, D. Ron, and R. Swaminathan**,** "A New Conceptual Clustering Framework", *Machine Learning 56,* Kluwer Academic Publishers, Netherlands, 2004, pages 115-151.

[10] D.M. Mount (2004). *Bioinformatics: Sequence and Genome Analysis* 2nd ed. Cold Spring Harbor Laboratory Press: Cold Spring Harbor, NY.

[11] S. Noel, S. Jajodia, "Understanding Complex Network Attack Graphs through Clustered Adjacency Matrices", *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, Tucson, AZ, December, 2005, pages 160-169.

[12] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs, "Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs", *Proceedings of the 19th Annual Computer Security Applications Conference*, Las Vegas, Nevada, December 2003, pages 86-95.

[13] R. Peeters, "The Maximum Edge Biclique Problem is NP-complete", *Discrete Applied Mathematics, 131,* Elsevier B.V., 2003, pages 651-654.

[14] S. Szeider**,** "Generalizations of Matched CNF Formulas", *Annals of Mathematics and Artificial Intelligence* (*to appear).*

[15] L. Wang, T. Jiang, "On the complexity of multiple sequence alignment". *J Comput Biol* 1:337–348 (1994).

[16] Wikipedia, http://en.wikipedia.org/wiki/Cluster