On Constructing An Optimal Consensus Clustering from Multiple Clusterings

Piotr Berman*
Department of Computer Science & Engineering
Pennsylvania State University
University Park, PA 16802
Email: berman@cse.psu.edu

Bhaskar DasGupta†
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: dasgupta@cs.uic.edu

Ming-Yang Kao‡
Department of Electrical Engineering & Computer Science
Northwestern University
Evanston, IL 60208
Email: kao@cs.northwestern.edu

Jie Wang§
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
Email: wang@cs.uml.edu

May 29, 2007

Abstract

Computing a suitable measure of consensus among several clusterings on the same data is an important problem that arises in several areas such as computational biology and data mining. In this paper, we formalize a set-theoretic model for computing such a similarity measure. Roughly speaking, in this model we have $k > 1$ partitions (clusters) of the same data set each containing the same number of sets and the goal is to align the sets in each partition to minimize a similarity measure. For $k = 2$, a polynomial-time solution was proposed by Gusfield (*Information Processing Letters*, 82, pp. 159-164, 2002). In this paper, we show that the problem is MAX-SNP-hard for $k = 3$ even if each partition in each cluster contains no more than 2 elements and provide a $2 - \frac{2}{k}$-approximation algorithm for the problem for any k.

Keywords: computational complexity, approximation algorithms, consensus clustering.

1 Introduction

Many applications in data mining and computational biology produce clusterings that are *partitions* of a set of elements. Quite often, different algorithms for the same application are used, thereby

*Supported by NSF grant CCR-0208821.
†Supported in part by NSF grants IIS-0346973, IIS-0612044 and DBI-0543365.
‡Supported in part by NSF grant EIA-0112934.
§Supported in part by NSF under grant CCR-0296037 and CCF-04080261, and by NSF of China under grant 60273062.

generating different clusterings (partitions) of the same set of elements. It is thus of interest to compare these partitions over the same universe to find their combined similarity or distance to check for discrepancies in individual partitions and to determine if a consensus of these different clusterings provides any meaningful interpretation of the given data [5, 8, 9]. See [1] for one such application of computing the consensus in bioinformatics in the context of determining the family structure of individuals based on genetic data. The set-theoretic model that we formalize for comparing partitions is summarized next.

Let $\Delta(S,T) = |\,(S \setminus T) \cup (T \setminus S)\,|$ for two sets S and T. A precise description of the set-theoretic formulation that computes a distance measure between the clusters is captured by the following problem.

> **Problem name:** k-partition clustering (PC_k)
>
> **Instance:** A universe Σ, a collection of k partitions $P_1, P_2, \ldots, P_k$ of Σ with each partition $P_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,q}\}$ containing exactly the same number q of sets.
>
> **Valid Solutions:** a sequence of k permutations $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ of $\{1, 2, \ldots, q\}$ that "aligns" the partitions.
>
> **Notation:** For any permutation ρ of $\{1, 2, \ldots, q\}$, $\rho(i)$ is the i^{th} element of ρ for $1 \leq i \leq q$.
>
> **Objective:** *minimize* $f(\sigma) = \sum_{i=1}^{q} \sum_{1 \leq j < r \leq k} \Delta(S_{j,\sigma_j(i)}, S_{r,\sigma_r(i)})$.

The most recent prior work in comparing partitions is by Gusfield [6] where the distance measure used is the minimum number of elements that need to be deleted such that the partitions become identical. Gusfield provides a polynomial-time algorithm for two partitions based on matching algorithms and observed that the problem becomes NP-hard for 3 or more clusters. The problem PC_2 can be seen to be identical to the distance measure between two partitions introduced in [6] and thus can be solved in polynomial time; the problem PC_k is one way to generalize such a distance measure between more than two partitions.

1.1 Basic Definitions

Recall that a γ-approximation of a minimization problem is a solution obtained in polynomial time with an objective value no larger than γ times the value of the optimum. In [7] Papadimitriou and Yannakakis defined the class of *MAX-SNP-hard* optimization problems and a special approximation-preserving reduction, the so-called *L-reduction*, that can be used to show MAX-SNP-hardness of an optimization problem. The definition of an L-reduction is as follows.

Definition 1 [3, 7] *Given two optimization problems* Π *and* Π'*, we say that* Π L-reduces *to* Π' *if there are three polynomial-time procedures* T_1,T_2, T_3 *and two constants* a *and* $b > 0$ *such that the following two conditions are satisfied:*

1. *For any instance* I *of* Π*, algorithm* T_1 *produces an instance* $I' = f(I)$ *of* Π' *such that the optima of* I *and* I'*,* $OPT(I)$ *and* $OPT(I')$*, respectively, satisfy* $OPT(I') \leq a \cdot OPT(I)$.

2. *For any solution of* I' *with cost* c'*, algorithm* T_2 *produces another solution with cost* c'' *no worse than* c'*, and algorithm* T_3 *produces a solution of* I *of* Π *with cost* c *(possibly from the solution produced by* T_2*) satisfying* $|c - OPT(I)| \leq b \cdot |c'' - OPT(I')|$.

An optimization problem is MAX-SNP-*hard* if any problem in MAX-SNP L-reduces to that problem. The importance of proving MAX-SNP-hardness results comes from the fact that Arora et al. [2] showed that, assuming P≠NP, for every MAX-SNP-hard problem there exists a constant $\varepsilon > 0$ such that no polynomial time algorithm achieves an approximation ratio better than $1 + \varepsilon$.

1.2 Brief Summary of Our Results

We show the following:

- PC_k is MAX-SNP-hard even for $k = 3$ even when each set has exactly 2 elements;
- there is a $\left(2 - \frac{2}{k}\right)$-approximation algorithm for PC_k for any k.

2 MAX-SNP-hardness of PC_3

In the sequel, for two edges $e = \{u, v\}$ and $e' = \{u', v'\}$, the notation $e \oplus e'$ denotes the set $(e \setminus e') \cup (e' \setminus e)$. We consider the following special case of PC_3:

> **Problem name:** Aligned Matching (AlMa).
>
> **Instance:** A cubic graph $(V, A \cup B \cup C)$ where A, B, C are three disjoint perfect matchings; thus for some n we have $|V| = 2n$ and $|A| = |B| = |C| = n$.
>
> **Valid Solutions:** an ordering of each matching into an ordered sequence of edges; matching A (resp. B, C) is ordered as $a_1, \ldots, a_n$ (resp. $b_1, \ldots, b_n$, $c_1, \ldots, c_n$)
>
> **Definitions and notations:** A triple (a_i, b_i, c_i) is a *column* of the alignment; the *cost of such a column* is $cost(i) = \frac{1}{2} \cdot [\, |a_i \oplus b_i| + |b_i \oplus c_i| + |c_i \oplus a_i| \,]$ and the *cost of the entire alignment* is $\sum_{i=1}^{n} cost(i)$.
>
> **Objective:** find an alignment with the minimum cost.

Obviously, AlMa is a special case of PC_3 since the cost function differs only by a multiplicative factor 2; moreover, in AlMa every set contains exactly two elements. We prove the following result.

Theorem 2 AlMa *is* MAX-SNP-hard.

The rest of the section discusses the proof of the above theorem. Let 3-MaxCut be the MaxCut problem restricted to cubic graphs, namely the following problem:

> **Problem name:** 3-MaxCut
>
> **Instance:** A cubic graph (V, E), *i.e.*, a graph with every vertex of degree 3 (and, thus, $|E| = \frac{3}{2}|V|$).
>
> **Valid Solutions:** a partition of V into two sets V_1 and V_2.
>
> **Objective:** *maximize* the number of "cut" edges, *i.e.*,
> maximize $|\{\, \{u, v\} \in E \mid u \in V_1 \text{ and } v \in V_2 \,\}|$.

It is known that 3-MaxCut is MAXSNP-hard [4]. We will reduce 3-MaxCut to AlMa. Here is an overview of the entire reduction. Given an instance I of 3-MaxCut with $2n$ vertices and $3n$ edges, we will create in polynomial time an instance I' of AlMa with $n' = 2n \times 96 = 192n$ nodes,

$m' = \frac{3}{2}n' = 288n$ edges and with alignments having $96n$ columns[1]. The reduction will satisfy the properties of L-reduction in the following manner:

(i) A solution of instance I of 3-MaxCut with $3n - a$ cut edges (hence, *a non-cut* edges) leads to a solution of instance I' of AlMa with cost $288n + 6a$.

(ii) From a solution of instance I' of AlMa of cost $288n + 6a'$ we can construct in polynomial time a solution of an instance I of 3-MaxCut with $3n - a'$ cut edges.

Since obviously $2n \leq \text{OPT}(I) \leq 3n$, it is easy to see that the above properties guarantee that conditions **(1)** and **(2)** of Definition 1 for an L-reduction are satisfied.

Now, we describe the reduction in more details. Given an instance (V, E) of 3-MaxCut we create an instance of AlMa as follows. Each node u of V is replaced with gadget Γ_u that consists of 96 nodes (see Figure 1) arranged in the form of four concentric rings of hexagons. From each node three edges extend in three directions:

- a horizontal edge from matching A `(striped)`;
- a diagonal edge from matching B `(white)` and
- an anti-diagonal edge from matching C `(black)`.

A node gadget is bipartite and we can color its vertices with white and black (see Figure 1). If $\{u, v\} \in E$, we connect the gadgets of u and v with 6 edges as shown in Figure 1. Note that among the 6 connections, there are two in every of the three matchings/directions, one connecting two white nodes and one connecting two black nodes. We use the following convention in the sequel:

If $x, y \in \{\text{white,black}\}$, e is an x-y edge if it connects vertices with colors x and y.

Thus we defined an instance of AlMa (V', A, B, C), where node set V' is the union of nodes of Γ_u gadgets (thus, $|V'| = 96 \cdot |V| = 192n$) and A, B and C are the three matchings that form the set of edges (thus, $|A| = |B| = |C| = \frac{|V'|}{2} = 96n$ and $m' = |A| + |B| + |C| = 288n$).

Note that since in AlMa sets have exactly two elements, the cost of a column of an alignment may be 3, 4, 5 or 6. Because (V', A, B, C) has no triangles, we observe the following.

- A column with cost 3 consists of three edges incident to a single node. We call it a `star`; the `center` of the star is their common node. Obviously, a `star` is connected [2]
- A column with cost 4 consists of three edges that form a path. We call it a `crescent` (because each edge in a crescent must be of different color, each edge must follow a different direction, hence the three edges form a crescent rather than a zig-zag). Obviously, a `crescent` has one connected component. Note that the cost of the column is 3 plus an extra 1 corresponding to this one component.
- A column with cost 5 consists of a path of two edges (called a `pair`) and another edge (called a `single`). We call it a `pair-with-a-single`. Obviously, a `pair-with-a-single` has two connected components. Note that the cost of such a column is 3 plus an extra 1 for each connected component.

[1] A smaller construction is possible, but seems to require a substantially longer proof.

[2] A set of edges is *connected* if it is connected in the line-dual graph, where vertices are (former) edges and edges indicate having a common (former) node. In this framework we can have *connected components* of a set of edges.

- A column with cost 6 consists of 3 `single`s (non-adjacent edges). Obviously, such a column has connected components corresponding to each `single`. Note that the cost of such a column is 3 plus an extra 1 for each connected component.

Thus, observe that an alignment (of $96n = \frac{n'}{2}$ columns) partitions $A \cup B \cup C$ into connected sets with at most one edge from each matching in each column, and if besides stars this partition contains b crescents, c pairs and d singles then the cost is $\frac{3}{2}n' + b + c + d$.

Lemma 3 (Proof of Property (i)) *If (V, E) has a cut (V_1, V_2) with $3n-a$ cut edges, (V', A, B, C) has an alignment with cost $288n + 6a$.*

Proof. We construct the alignment as follows: if $u \in V_1$, for every white node w in Γ_u we create star column with center w, and if $u \in V_2$, we do the same for every black node in Γ_u. Because each node gadget contains 48 nodes of each color, a first glance would seem to indicate that we have created an alignment with a cost of $3 \times 48|V| = 144|V| = 288n$.

However, the above calculation is not quite correct: contact edges may be allocated to two stars or to none. The following two cases arise:

Case 1: $\{u, v\} \in E$ belongs to the cut. Assume without loss of generality that $u \in V_1$ and $v \in V_2$. Then such a white-white contact edge belongs to a star with the center in Γ_u, and such a black-black contact edge belongs to a star with the center in Γ_v.

Case 2: $\{u, v\} \in E$ does not belong to the cut. Assume without loss of generality that $\{u, v\} \subset V_1$. Consider a pair of contact edges of Γ_u and Γ_v that belong to the same matching, say A. Then the black-black edge is not taken by any star, and thus it forms a single from A. The white-white edge is "taken" by two stars, so we remove it from one of them, and what is left is a pair of edges from matchings B and C. Thus we can combine the single and the pair and we have a pair-with-a-single column; this increases the cost by 2 (from 3 to 5), and we do it for each of the three matchings, so the total cost increase is $3 \times 2 = 6$. As we do it for a edges, we obtain a solution with cost $288n + 6a$. ∎

Now, we turn to proving Property **(ii)** of our reduction. We need to show that for every alignment in (V', A, B, C) there exists another alignment with a cost that is not larger and which is derived from a cut of (V, E) as in the proof of Lemma 3. We will *normalize* the alignment in stages.

As observed before, an alignment partitions $A \cup B \cup C$ into connected sets with at most one edge from each matching in each column, and if besides stars this partition contains x crescents, y pairs and z singles then the cost is $1.5n' + x + y + z$. We say that crescents, pairs and singles are *irregular sets*. For ease of describing the normalization, we now introduce the following problem:

> **Problem name:** StEP (strange edge partition)
>
> **Instance:** same input as in AlMa.
>
> **Valid Solutions:** a partition of the set of edges such that each member of the partition is a star (a set of three edges), a crescent (a set of three edges), a path (a set of two edges) or a single (one edge) with at most one edge from any matching in a partition.
>
> **Objective:** *minimize* the number of irregular sets, *i. e.* the number of crescents, pairs and singles.

Note that a solution of AlMa is also a solution of StEP, but a solution of StEP is not necessarily a solution of AlMa since we do not mention in a solution of StEP how to produce the columns of the alignments.

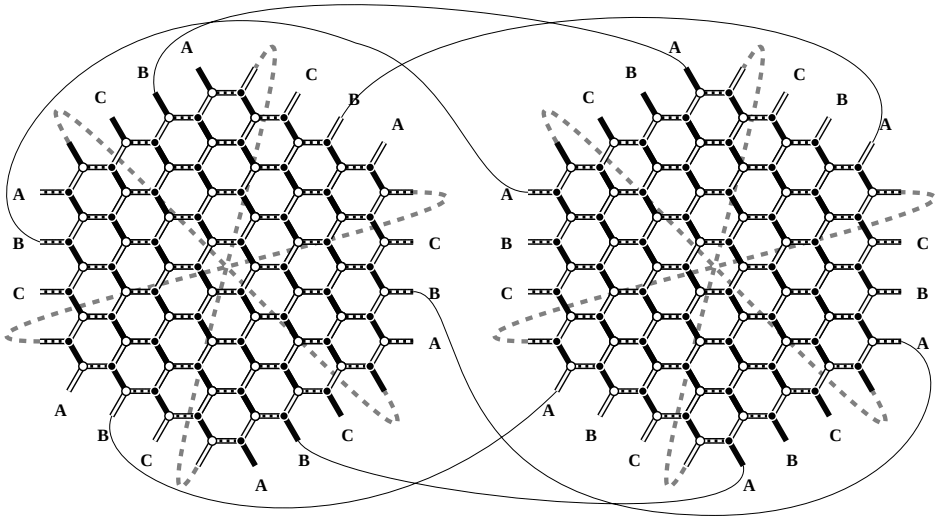

Figure 1: Two node gadgets with a connection. A gadget has 18 outgoing edges, so that for each of the two colors of a node and each of the three colors of an edge and each of the three neighbors there is an outgoing edge. Two gadgets are connected by identifying six pairs of outgoing edges in such a way that in a pair both edges are of the same kind: the same *edge* color and the same *node* color. Here the left gadget is an A-neighbor of the right gadget and the right gadget is a B-neighbor of the left one. A gadget also has 3 "self"-edges (shown by dashed lines), one for each color (striped, white or black), that are created by identifying pairs of the "unnamed edges" of the same color. Note that each such self-edge connects a black vertex and a white vertex.

Clearly, a solution to AlMa instance (V', A, B, C) with cost $1.5|V'| + f$ yields a solution of StEP with cost f. An overview of our normalization process is as follows:

- We will start with a solution of AlMa on (V', A, B, C) of cost $1.5|V'| + c$. This implies a solution of StEP on (V', A, B, C) of cost c.
- The normalization process has polynomially many steps. In each step we obtain a new solution, with a cost that is the same or smaller, so the final solution has cost $c' \leq c$.
- From the final normalized solution of cost c' we create a solution of 3-MaxCut on (V, E) that has at least $1.5|V| - c'/6 \geq 1.5|V| - c/6$ cut edges. This will be obtained by observing that the final solution corresponds to a solution of 3-MaxCut on (V, E) in a manner as shown in Lemma 3.

Intuitively, the reason we introduce the problem StEP is that at every step of the normalization the cost of the solution of StEP will be unchanged or it will become smaller, but we will *not* need to ensure that this solution of StEP is also a solution of AlMa.

We will use the following notations and terminologies for convenience:

- The term *solution* refers to a solution of StEP on the instance (V', A, B, C). The cost of such a solution is therefore defined in terms of StEP.
- `sstar` (resp. `sirset`) is a star (resp. an irregular set) that belongs to the current *solution*.

We now state a series of Lemma describing successive steps of the normalization process. Each lemma assumes that the alterations in the solutions as necessary in the preceding lemmas have been applied.

Lemma 4 *A solution can be altered, without increasing its cost, so that every node u is a center of a* `sstar`*, or is adjacent to such a center.*

Proof. Otherwise remove each edge adjacent to u from its `sirset` and create a new `sstar` with these edges. This obviously does not increase the number of irregular sets. ∎

Thus, from now on, assume that all appropriate alterations have been made to satisfy Lemma 4.

For each `sirset` we allocate 1 penalty point to gadgets in the following manner: if it contains an edge connecting two gadgets we allocate 0.5 point to each of them, otherwise it is contained in one gadget then we allocate 1 point to this gadget. We now use the following definitions and notations:

- We say that a gadget is `nice` if it gets less than 4 points, otherwise we call it `ugly`.
- We say that a star or a pair with black (resp. white) center is black (resp. white).

Now we will normalize gadgets in stages. In the first stage we normalize the solution within `nice` gadgets. We say that a hexagon of a gadget is *healthy* if if each selected set that covers some of its six edges covers exactly two of them. Otherwise, the hexagon is *sick*, and two `sirset`s cover an odd number of its edges. Observe that a healthy hexagon is covered by three pairs or stars of the *same* color, thus we say that a healthy hexagon is black or white depending on whether the colors of these pairs (or stars) are black of white, respectively.

Lemma 5 (healing of sick hexagons of a specific type within `nice` gadgets) *A solution can be altered, without increasing its cost, so that if a hexagon has two non-consecutive edges shared with healthy hexagons of the same color, it is healthy as well, and with the same color.*

Proof. Let such a hexagon with two adjacent healthy hexagon neighbors be the upper left one in Figure 2. Assume that the healthy neighbors are the lower and the upper right, and that they are both white (other cases are very similar). *We indicate the white pairs/stars implied by this assumption with striped gray background* (indicated by ||||| in gray) *in Figure 2).*

If necessary, we can extend a selected pair to a `sstar` without changing the cost of the solution by taking the third edge of the star from another selected set removes a `sirset` and adds at most one `sirset`. *We indicate the edges we could take in this fashion with solid gray background in Figure 2.*

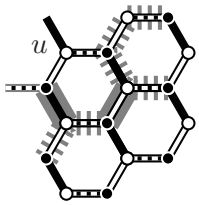

Figure 2: Figure for Lemma 5.

By Lemma 4, node u in Figure 2 is either a center of a `sstar` or adjacent to a center of a `sstar`. If u is a center of a `sstar` then our hexagon is obviously healthy. Otherwise, assume that our hexagon is not healthy with u being adjacent to a center of a `sstar` (and, thus the black edge incident to u belongs to a `sstar`), and white and striped edges adjacent to u belong to different `sirset`s. This means that the striped edge is a single. Now, we can take away black edge incident to u from its `sstar`, the white edge incident on u from its `sirset` and create the `sstar` with center u. Because we removed the single from the ranks of `sirset`s, the number of `sirset`s remains unchanged, while our hexagon is *healed*. ∎

Lemma 6 (healing of all sick hexagons within `nice` gadgets) *A solution can be altered, without increasing its cost, so that in a* `nice` *gadget all hexagons are healthy.*

Proof. A gadget consists of four concentric circles of hexagons. Suppose first that two hexagons, say A and B, are contained in the inner three circles, are not adjacent[3] and are both sick. Then

[3]Two hexagons are adjacent if they share a side (edge).

each of them has two sirsets that cover odd number of its edges. Since our assumption is that the gadget is nice, at least one sirset must be shared between the two hexagons. The only way such a sirset can be shared between the two hexagons is a crescent surrounding a hexagon, say C, with A and B being separated by another hexagon, say D. In this case, by Lemma 4, there are two sstars adjacent to the crescent that separate A from D and D from B; each of the sick hexagons A, D and B has another sirset covering its edges and these sirsets cannot be shared; consequently the gadget has at least four sirsets and it is ugly, a contradiction!

Thus, all the sick hexagons in the inner three circles of hexagons must be mutually adjacent to each other. This implies that the inner three circles of hexagons contain at most 3 sick hexagons (since the line dual graph of inner three circle of hexagons does not have larger cliques than triangles), and then it is easy to see that we can heal them using Lemma 5. Thus we can assume that *all* hexagons in the inner three circles are healthy — and that all of them have the same color, say, white (by an obvious observation that two adjacent healthy hexagons must have the same color). We can also assume that the three inner circle of hexagons are completely covered with sstars (otherwise the solution can be easily normalized). We call these sstars as *inner* sstars.

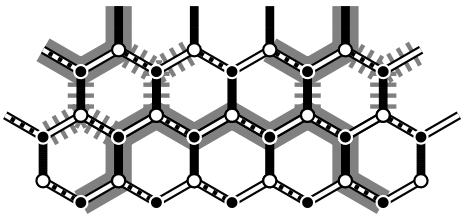

Figure 3: The boundary of the inner hexagon and crescent examples are indicated by solid gray background, and the sstars implied by Lemma 4 are indicated by striped gray.

Now if a hexagon A adjacent to a *corner* position (*i.e.*, adjacent to *exactly one* inner hexagon) is healthy, then we can heal two hexagons in the outer circle in each of the two directions from A by application of Lemma 5. Moreover, if two neighbors of a hexagon A in the outer circle are healthy, we can heal A. Thus if there exists a sick hexagon and we cannot heal it, there must be a side of the gadget consisting of 4 hexagons on the outer circle of hexagons, and each is covered in part by two sirsets. If we can collect 0.5 point from each "sirset ownership", we have 4 points and the gadget is ugly, a contradiction.

A sirset may be "giving points" for the following reasons: it contains a contact edge so it has to give 0.5 point to another gadget, or it covers an odd number of edges of some number of hexagon in our "line of four". We have a problem only if there are at least three such commitments. No sirset can share edges with three hexagons that are in a line, but it is possible, for a crescent sirset, to cover a contact edge and one edge in two hexagons.

Such a crescent sirset has 4 nodes, and the middle two must be on the gadget boundary; by Lemma 4 each of the middle nodes must be adjacent to a sstar, one of sstars being white and the other black. In almost all cases both sstars have to contact or overlap the inner sstars, which implies that they have the same color, a contradiction (see left example in Figure 3). The remaining cases look like the right example in Figure 3. Black sstar adjacent to the solid-gray crescent is not adjacent to the inner (white) sstars; as a result, the lower right edge of the corner hexagon has to be covered by a sirset that will deliver 1 point to that hexagon, so we can let the crescent to deliver 0.5 point to another gadget, and 0.5 point to the second hexagon from the right. ∎

Lemma 7 (forcing two adjacent nice gadgets to be of different color) *A solution can be altered, without increasing its cost, so any two adjacent* nice *gadgets are of different colors.*

Proof. Suppose that two adjacent nice gadgets Γ_u and Γ_v have the same color, say black. We will show that we can alter the solution, without increasing its cost, so that one of them is ugly.

By Lemma 6, both gadgets are covered with black `sstars` except that on the boundary they may be covered by black pairs. These gadgets are connected with 6 contact edges; 3 white-white edges are on both ends in contact with black stars and they forms shared `sirsets` in the form of singles, and thus they give 1.5 points to each of Γ_u and Γ_v.

A black-black contact edge is in contact with two selected black pairs; we can combine each such edge with the adjacent black pair in Γ_v such that the contact edge is covered by a black `sstar`, and Γ_u contains a `sirset` black pair. This gives 3 points to Γ_u. Together, Γ_u gets $1.5 + 3 = 4.5$ points and hence it becomes `ugly`. ∎

Lemma 8 (linking `ugly` gadgets to `sirsets`) *Suppose that we have a solution to an instance (V', A, B, C) of* StEP *in which there are a* `ugly` *gadgets of which b are adjacent to* `nice` *gadgets of two different colors. Then this solution contains at least $4a + 2b$* `sirsets`*.*

Proof. Since each `ugly` gadget has at least 4 points, we just need to show that each `ugly` gadget that is adjacent to two `nice` gadgets has 6 points (and, thus, two extra points).

For each `ugly` gadget Γ_v with `nice` neighbors, black Γ_u and white Γ_w, we create 6 paths that ① are included in Γ_v and the contact edges of Γ_v with Γ_u and Γ_w, and ② connect `sstars` that cover Γ_u with `sstars` that cover Γ_v.

If a connecting edge is adjacent to a pair of Γ_u or Γ_w, we create a star that consists of this pair and this edge. We use shortest possible paths; see examples in Figure 4.

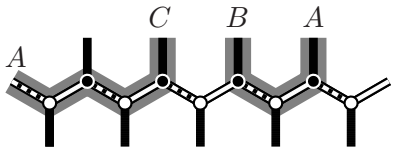

Figure 4: Figure for Lemma 8.

Each of these paths has an odd number of edges. For example, is we connect a white-white contact edge e to Γ_u (the black gadget) with black-black contact edge e' to Γ_w (the white gadget), neither e nor e' is not covered by `sstars` of Γ_u or Γ_w, and the path connecting these two edges has 3 edges, so we have a total of 5 edges. If the connected contact edges are both white-white, then the white gadget contains one of them and the connecting path has 2 edges, so we have the total of 3 edges.

An path of odd length cannot be covered without a `sirset`. One can see that `sirsets` that cover the connecting paths cannot be shared between the paths or with other gadgets, so we have 6 points associated with Γ_u only. Thus for every `ugly` gadget adjacent to two nice gadgets of different colors we have $6 - 4 = 2$ extra points. ∎

Lemma 9 (linking `nice` and `ugly` gadgets to 3-MaxCut**)** *Suppose that we have a solution to the instance (V', A, B, C) of* StEP *in which*

- *every two adjacent* `nice` *gadgets are of different colors;*
- *there are a* `ugly` *gadgets of which b are adjacent to* `nice` *gadgets of two different colors.*

Then, the instance (V, E) of 3-MaxCut *has a solution with at most $\frac{2a+b}{3}$ non-cut edges*[4]*.*

Proof. We transfer our gadget properties to nodes of our 3-MaxCut instance (V, E), so we refer to nice, ugly, white and black nodes. We partition the set on nodes into white and black, so that non-cut edges are either white-white or black-black. Our initial partition does not have any non-cut edges, but we may have to introduce them when we color the ugly nodes.

We give each `ugly` node 2 points plus an extra 1 point of it has nice neighbors of two colors. We will make sure that the `ugly` nodes are colored in such a way that for each non-cut edge (white-white or black-black) we can pay 3 points.

[4]A non-cut edge is an edge that is *not* a cut edge.

We use two strategies: either we convert an ugly node to a nice one without introducing non-cut edges, or we will proceed recursively: delete a node u that has 3 points, color the remaining ugly nodes, then if u has 2 white neighbors, we color it black, and if it has at most 1 white neighbor, we color it white. In either case we introduce at most one non-cut edge.

This strategy can be described as applying the first possible rule from the following list:

① delete an ugly node that has 3 points (always works if a node has a nice neighbors of both colors);

② if an ugly node u has a nice neighbor, we make u white if the nice neighbor(s) is black and vice versa; moreover give each ugly neighbor of u ‘ one of the points that u had (note that since u has a nice neighbor, it has at most two ugly ones);

③ if no ugly node has a nice neighbor, pick one arbitrarily and make it nice by giving it an arbitrary color. ∎

It is now obvious how to finish the proof of the theorem, and more precisely, property (ii) of the constructed instance (V', A, B, C):

- We start with a a solution of cost $\alpha = 1.5|V'| + c = 288n + c$ (for some c) of ALMA on (V', A, B, C) and thus with a solution of cost $c = \alpha - 288n$ of STEP on (V', A, B, C).
- The current solution of STEP is normalized without increasing the cost via Lemma 6 such all nice node gadgets are all-healthy.
- The current solution of STEP is normalized without increasing the cost via Lemma 7 such that the neighboring nice gadgets are of different color.
- For some a and b we now have a `ugly` gadgets of which b are adjacent to `nice` gadgets of two different colors. By Lemma 8, we have some β number of irregular sets in the solution where $4a + 2b \leq \beta \leq \alpha - 288n$.
- By Lemma 9 we can create a solution to the instance (V, E) of 3-MAXCUT instance with at most
$$\frac{2a+b}{3} = \frac{4a+2b}{6} \leq \frac{\alpha - 288n}{6} = \frac{c}{6}$$
non-cut edges. Thus, the number of cut edges is at least $3n - \frac{c}{6}$.

This completes the proof of Theorem 2.

3 An Approximation Algorithm for PC_k

The following additional notations are used for convenience:

- $\sigma^* = (\sigma_1^*, \ldots, \sigma_k^*)$ is a sequence of k permutations that produces an optimal solution of objective value $OPT = \sum_{i=1}^{q} \sum_{1 \leq j < r \leq k} \Delta(S_{j,\sigma^*(i)}, S_{r,\sigma^*(i)})$.
- For any k permutations $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ of $\{1, 2, \ldots, q\}$, $\Delta_\sigma(P_j, P_r) = \sum_{i=1}^{q} \Delta(S_{j,\sigma_j(i)}, S_{r,\sigma_r(i)})$.

Lemma 10 *There is a $\left(2 - \frac{2}{k}\right)$-approximation algorithm for PC_k that runs in $O\left(k^2 \cdot (|\Sigma| + q^3)\right)$ time.*

Proof. The result for PC_2 follows from [6]. Thus we may assume that $k > 2$. It is obvious that Δ satisfies the triangle inequality, that is, *for any three sets X, Y and Z,* $\Delta(X, Z) \leq \Delta(X, Y) + \Delta(Y, Z)$. Our algorithm is simply as follows:

- for each $1 \leq i \leq k$ align P_i optimally individually with each P_j for $j \neq i$ using the algorithm for PC_2 to produce an alignment $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,k})$;
- take the best of all these solutions.

Suppose that the best solution is achieved by $\sigma_r = (\sigma_{r,1}, \sigma_{r,2}, \ldots, \sigma_{r,k})$. Note that:

- For any i and j, $\Delta_{\sigma_i}(P_i, P_j) = \Delta_{\sigma_j}(P_j, P_i)$.
- For any i and j, $\Delta_{\sigma_i}(P_i, P_j) \leq \Delta_{\sigma^*}(P_i, P_j)$ and thus

$$\sum_{1 \leq i < j \leq k} \Delta_{\sigma_i}(P_i, P_j) \leq \sum_{1 \leq i < j \leq k} \Delta_{\sigma^*}(P_i, P_j) = OPT \tag{1}$$

- For any i, j and r,

$$\Delta_{\sigma_i}(P_j, P_r) = \sum_{i=1}^{q} \Delta(S_{j,\sigma_i}, S_{r,\sigma_i}) \leq \sum_{i=1}^{q} \Delta(S_{i,\sigma_i}, S_{j,\sigma_i}) + \Delta(S_{i,\sigma_i}, S_{r,\sigma_i}) = \Delta_{\sigma_i}(P_i, P_j) + \Delta_{\sigma_i}(P_i, P_r) \tag{2}$$

- For any i,

$$\begin{array}{rcll} f(\sigma_i) & = & \sum_{s \neq i} \Delta_{\sigma_i}(P_i, P_s) + \sum_{j \neq i} \sum_{r \neq i, r > j} \Delta_{\sigma_i}(P_j, P_r) & \\ & \leq & \sum_{s \neq i} \Delta_{\sigma_i}(P_i, P_s) + \sum_{j \neq i} \sum_{r \neq i, r > j} (\Delta_{\sigma_i}(P_i, P_j) + \Delta_{\sigma_i}(P_i, P_r)) & \text{[using inequality (2)]} \\ & = & \sum_{s \neq i} \Delta_{\sigma_i}(P_i, P_s) + (k-2) \cdot \sum_{j \neq i} \Delta_{\sigma_i}(P_i, P_j) & \\ & = & (k-1) \cdot \sum_{j \neq i} \Delta_{\sigma_i}(P_i, P_j) & \end{array}$$

- Thus,

$$\begin{array}{rcll} \sum_{i=1}^{k} f(\sigma_i) & \leq & \sum_{i=1}^{k} [(k-1) \cdot \sum_{j \neq i} \Delta_{\sigma_i}(P_i, P_j)] & \\ & = & (k-1) \cdot \sum_{i=1}^{k} \sum_{j \neq i} \Delta_{\sigma_i}(P_i, P_j) & \\ & = & 2 \cdot (k-1) \cdot \sum_{1 \leq i < j \leq k} \Delta_{\sigma_i}(P_i, P_j) & \text{[since } \Delta_{\sigma_i}(P_i, P_j) = \Delta_{\sigma_j}(P_j, P_i)\text{]} \\ & \leq & (2k-2) \cdot OPT & \text{[using inequality (1)]} \end{array}$$

and $f(\sigma_r) \leq \frac{1}{k} \sum_{i=1}^{k} f(\sigma_i) \leq \left(2 - \frac{2}{k}\right) \cdot OPT$.

∎

Acknowledgments

We would like to thank the anonymous reviewer whose suggestions led to significant improvements to the presentation of the paper.

References

[1] A. Almudevar and C. Field. *Estimation of single generation sibling relationships based on DNA markers*, J. Agricultural, Biological Environment. Statist., 4, pp. 136-165, 1999.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. *Proof verification and hardness of approximation problems*, Journal of the ACM, 45 (3), 1998, pp. 501-555.

[3] P. Berman and G. Schnitger. *On the complexity of approximating the independent set problem*, Information and Computation, 96, 1992, pp. 77-94.

[4] P. Berman and M. Karpinski. *On some tighter inapproximability results*, 26th Int. Coll. on Automota, Languages, and Programming, pp. 200-209, 1999.

[5] S. Datta. *Comparisons and validation of statistical clustering techniques for microarray gene expression data*, Bioinformatics, 19, 2003, pp. 459-466.

[6] D. Gusfield. *Partition-distance: A Problem and class of perfect graphs arising in clustering*, Information Processing Letters, 82, pp. 159-164, 2002.

[7] C. H. Papadimitriou and M. Yannakakis. *Optimization, approximation, and complexity classes*, Journal of Computer and System Sciences, 43 (3), 1991, pp. 425-440.

[8] S. Swift, A. Tucker, V. Vinciotti, N. Martin, C. Orengo, X. Liu and P. Kellam. *Consensus clustering and functional interpretation of gene-expression data*, Genome Biology, 5 (11), November 2004.

[9] K. Y. Yeung, D. R. Haynor and W. L. Ruzz. *Validating clustering for gene expression data*, Bioinformatics, 17, 2001, pp. 309-318.