The Clique Partition Problem with Minimum Clique Size Requirement [1]

Xiaoyun Ji, John E. Mitchell

Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Abstract

Given a complete graph $K_n = (V, E)$, with edge weight c_e on each edge, we consider the problem of partitioning the vertices of graph K_n into subcliques that each have at least S vertices, so as to minimize the total weight of the edges that have both endpoints in the same subclique. It is an extension of the classic Clique Partition Problem that can be well solved using triangle inequalities, but the additional size requirement here makes it much harder to solve. The previously known inequalities are not good enough to solve even a small size problem within a reasonable amount of time. In this paper, we'll discuss the polyhedral structure and introduce new kinds of cutting planes: pigeon cutting planes and flower cutting planes. We will report the computational results with a branch-and-cut algorithm confirming the strength of these new cutting planes.

Key words: clique partition; branch-and-cut; clustering; microaggregation

1 Introduction

Given a graph $G = (V, E)$ with edge weights c_e, and an integer $S < |V|$, the Clique Partition Problem with Minimum clique size requirement (CPPMIN) requires dividing the vertices V into subsets that each have at least S vertices. The objective is to minimize the total weight of the edges that have both endpoints in the same set.

We define a binary variable x_e, which takes the value 1 if edge e is within one subset and 0 if it is across two subsets. We can formulate CPPMIN as an integer programming problem,

[1] Research supported in part by NSF grant number DMS-0317323.

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(v)} x_e \geq S - 1 \qquad \forall v \in V \qquad (1) \\
& x \text{ is the incidence vector of a partition on } G
\end{aligned}
$$

where $\delta(v)$ denotes the set of edges incident to vertex v. Constraint (1) is called the size constraint. We will use $n = |V|$ to denote the number of vertices. With S given, we can define $k = \lfloor n/S \rfloor$, $r = n \bmod S$, $0 \leq r \leq S-1$. Thus we have $n = kS + r$. We will explain later in section 2.1, what we mean by "x is the incidence vector of a partition on G", and rewrite this above formulation in a more concrete formulation as problem (CPPMIN).

Throughout this paper, we assume G is the complete graph on V. A non-complete graph can easily be converted to a problem on a complete graph by adding the missing edges in with a very big edge weight c_e, or with edge weight 0, depending on the application. We can also just add explicit constraints $x_e = 0$ for the missing edges e in a non-complete graph.

The CPPMIN problem is closely related to the classic partition problem that has no restrictions on the size or number of the partitions – the Clique Partition Problem (CPP). CPP partitions the vertices V into subcliques, so as to minimize the total weight of the edges that have both endpoints in the same subcliques. Grötschel and Wakabayashi [6, 7] have provided a detailed description with several kinds of cutting planes, especially triangle constraints and 2-partition constraints. Their experiments showed that it suffices to use only triangle inequalities and 2-partition inequalities. An LP relaxation consisting of all triangle inequalities and some 2-partition inequalities turns out to be empirically very effective. In fact, only adding cutting planes of triangle inequalities was sufficient to find and prove the optimal solution in most cases.

Variations of CPP have also been studied. One of them is the equipartition problem, which requires dividing the vertices into 2 clusters of equal size. When there is an odd number of vertices, they need to be divided into to 2 clusters with size difference of 1 vertex. Conforti, Rao and Sassano [4] discussed the polyhedral theory for this problem. A further extension is the k-way equipartition problem, which requires dividing the n points into k equally sized clusters. In this problem, the number of the vertices n has to be a multiple of the number of partitions k. This is equivalent to the constraint that each cluster has to be of size n/k. Mitchell [16] discussed the facial structure of the polytope associated with this problem and provided a branch-and-cut algorithm to solve it. He used this method to successfully solve the NFL scheduling problem in [17].

However in many instances the total number of vertices n may not be a multiple of the desired number of partitions k, so a k-way equipartition is not

possible. One way is to solve the k-way partition problem instead, which requires dividing the graph into no more than k clusters. Chopra and Rao [3] investigated this problem on a general graph. They don't assume a complete graph, so they can take advantage of the structure of the graph to be divided. They also discussed the opposite problem of requiring at least k clusters. A special case of the k-way clustering problem is the max cut problem, which has $k = 2$. Various approaches have been studied on this problem. A good survey can be found in Poljak and Tuza [21].

Another way to solve the problem when n is not a multiple of k is to relax the constraint on the size of each cluster and only require that each cluster size is not smaller than $S = \lfloor n/k \rfloor$. This leads to the problem in this paper: CPPMIN, a Clique Partition Problem with Minimum Clique Size requirement. Applications of CPPMIN include micro-aggregation of statistical data, see Domingo-Ferrer [5], sports team alignment, see Mitchell [17], Ji and Mitchell [11], telecommunication clustering, see Lisser and Rendl [14], locating communication centers, see Guttmann-Beck and Rubinstein [8].

It is natural to ask about the opposite problem : Clique Partition Problem with an upper bound on the cluster size. In fact, this can be converted into a k-way equipartition problem by adding dummy vertices. So the k-way equipartition problem and CPPMIN have covered all three kinds of constraints on cluster size for a clique partition problem.

Mehrotra and Trick [15] did research on clustering problem with Knapsack capacity constraints. In their problem, each vertex has a weight. The total weight of the vertices in the same cluster cannot exceed a capacity. This is a generalized version of the problem requiring each cluster has to be smaller than a certain size. They used a branch and price method to solve the problem. Similarly, we can consider the opposite problem of requiring each cluster to be bigger than a certain weight. It can also be formulated as an integer programming problem in the similar fashion as CPPMIN:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{j \in V} w_j x_{ij} \geq CAP - w_i \qquad \forall i \in V \\ & x \text{ is the incidence vector of a partition on } G \end{aligned} \tag{2}$$

where w_i denotes the the weight of vertex i. CAP is the required minimum capacity. This problem is addressed in a later work [12].

1.1 Notations and Definitions

In this section, we define some useful notations.

Given an undirected graph $G = (V, E)$, let V_1 and V_2 be two disjoint subsets of V, we define the following notation.

$$\begin{aligned}
\delta(V_1) &= \{uv \in E | u \in V_1, v \notin V_1\} \\
\delta(V_1, V_2) &= \{uv \in E | u \in V_1, v \in V_2\} \\
\delta(V_1, V_2, \ldots, V_k) &= \{uv \in E | u \in V_i, v \in V_j, i, j = 1, ..., k, i \neq j\} \\
E(V_1) &= \{uv \in E | u, v \in V_1\} \\
E(V_1, V_2) &= E(V_1) \cup E_(V_2) \\
E(V_1, V_2, \ldots, V_k) &= \cup_{i=1...k} E(V_i)
\end{aligned}$$

A **clustering** of G is a division of V into $\Pi = \{V_1, V_2, \ldots, V_k\}$, where $V = \cup_{i=1..k} V_i$ and $V_i \cap V_j = \emptyset, \forall 1 \leq i < j \leq k$. $V_1, V_2, \ldots, V_k$ are the **components** in the clustering. When G is a complete graph, we also call these components subcliques. The edge set $E(\Pi) = E(V_1, V_2, \ldots, V_k)$ is called the **partition set** or simply **partition**, while $\delta(\Pi) = \delta(V_1, V_2, \ldots, V_k)$ is called the **multi-cut set** or simply **multi-cut**. Notice that $E(G) = \delta(\Pi) \cup E(\Pi)$, $\delta(\Pi) \cap E(\Pi) = \emptyset$. The partition and the multi-cut are just two ways of representing the same clustering problem. They are both used often in the literature. Next, we define the corresponding incidence vectors.

Definition 1 *Given a graph $G = (V, E)$ and a clustering $\Pi = (V_1, V_2, \ldots, V_k)$ on G. Let $|E| = m$, $x_\Pi \in \{0, 1\}^m$ is called the* ***incidence vector of partition*** *Π if the entries in x_Π satisfy*

$$x_e = \begin{cases} 1 & e \in E(\Pi) \\ 0 & otherwise \end{cases}$$

$y_\Pi \in \{0, 1\}^m$ is called the ***incidence vector of multi-cut*** *Π if the entries in y_Π satisfy*

$$y_e = \begin{cases} 1 & e \in \delta(\Pi) \\ 0 & otherwise \end{cases}$$

For any $U \subseteq V$, define $\mathbf{x}(\mathbf{U}) = \sum_{i,j \in U} x_{ij}$, $\mathbf{y}(\mathbf{U}) = \sum_{i,j \in U} y_{ij}$.

According to the above definition $x_\Pi + y_\Pi = \mathbf{1}$, where $\mathbf{1}$ denotes the vector of all coefficients one.

In the following, we repeat a few definitions in integer programming. They will be used in the theorems and proofs in the later sections. For more details on these definitions, see Nemhauser and Wolsey [20].

Definition 2 *A set of points $x^1, \ldots, x^k \in I\!R^n$ is **affinely independent** if the unique solution to $\sum_{i=1}^{k} \lambda_i x^i = 0, \sum_{i=1}^{k} \lambda_i = 0$, is $\lambda_i = 0$ for all $i \in \{1, \ldots, k\}$.*

Proposition 3 *A set of points $x^1, \ldots, x^k \in I\!R^n$ is affinely independent if and only if the vectors $x^2 - x^1, x^3 - x^1, \ldots, x^k - x^1$ are linearly independent.*

Definition 4 *Let P be a polyhedron of dimension d. A face of dimension $d-1$ is a **facet**.*

Definition 5 *Given an inequality $\pi^T x \leq \pi_0$, the **support** of this inequality is defined as the index set $J = \{j \in \{1, \ldots, n\} : \pi_j \neq 0\}$. If the x variables correspond to the edges of a graph G, the **support graph** of this inequality is the subgraph of G, spanned by the edges indexed by the elements in the support J.*

2 Polyhedral Theories

We define the corresponding polytope $R(G, S)$ and $\bar{R}(G, S)$ for problem (1) as follows:

$$\begin{aligned} R(G,S) := conv\{x \in \{0,1\}^n : & \sum_{e \in \delta(v)} x_e \geq S - 1 \quad \forall v \in V \\ & \text{x is the incidence vector of a partition on G}\} \\ \bar{R}(G,S) := \{x \in [0,1]^n : & \sum_{e \in \delta(v)} x_e \geq S - 1 \quad \forall v \in V\} \end{aligned}$$

where $conv\{X\}$ represents the convex hull of a set of points X. Obviously $R(G, S) \subseteq \bar{R}(G, S)$. Once we solve the problem on $R(G, S)$, we are done, but since we don't know a concrete description for $R(G, S)$, we will start from $\bar{R}(G, S)$.

We also need to define some other related polytopes:

$$
\begin{aligned}
P(G) &:= conv\{x \in \{0,1\}^n : x \text{ is the incidence vector of a clique partitioning of graph } G\} \\
P_{BC}(G) &:= conv\{y \in \{0,1\}^n : y \text{ is the incidence vector of a 2-way cut of graph } G\} \\
P_{EC}(G) &:= conv\{y \in \{0,1\}^n : y \text{ is the incidence vector of an equicut of graph } G\} \\
P_{EP}(G) &:= conv\{x \in \{0,1\}^n : x \text{ is the incidence vector of an equipartition of graph } G\}
\end{aligned}
$$

2.1 Polyhedral Theory for $P(G)$, $P_{BC}(G)$, $P_{EC}(G)$ and $P_{EP}(G)$

We are going to summarize some results for the above polytopes. Since all other variations base on the basic Clique Partition Problem (CPP), we start with its corresponding polytope $P(G)$. Grötschel and Wakabayashi [6, 7] described a simplex-based cutting plane algorithm for CPP. The most important facets they used there are Triangle Inequalities and 2-partition equalities.

Theorem 6 *([7]) $P(G)$ is full dimensional, i.e., $dim(P(G)) = |E|$.*

Theorem 7 *([7], Theorem 3.1) Each* ***Triangle Inequality***

$$x_{ij} + x_{il} - x_{jl} \leq 1 \quad \forall 0 \leq i \neq j \neq l \leq |V| \tag{3}$$

defines a facet of the clique partitioning polytope $P(G)$.

In fact the triangle inequalities with binary constraints completely describe the incidence vector of the clique partition problem of a complete graph. In other words, the solution of (4) are exactly the incidence vectors for the CPP of a complete graph.

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e & & \\
\text{s.t.} \quad & x_{ij} + x_{il} - x_{jl} \leq 1 & & \forall i,j,l \quad 0 \leq i \neq j \neq l \leq n \\
& x_{ij} \in \{0,1\} & & \forall i,j \quad 1 \leq i < j \leq n
\end{aligned} \tag{4}
$$

Similarly, we can rewrite our initial formulation for CPPMIN, i.e., problem (1), in a more concrete form as the following:

$$
\begin{aligned}
\min \quad & \sum_{i,j\in E, i\neq j} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j\in E, j\neq i} x_{ij} \geq S-1 \qquad \forall i \in V \\
& x_{ij} + x_{il} - x_{jl} \leq 1 \qquad \forall i,j,l, \quad 0 \leq i \neq j \neq l \leq n \\
& x_{ij} \in \{0,1\} \qquad \forall i,j, \quad 1 \leq i < j \leq n
\end{aligned}
\tag{CPPMIN}
$$

Theorem 8 *([7], Theorem 4.1)For every nonempty disjoint subsets $U, W \subseteq V$, the **2-partition inequality***

$$x(U,W) - x(U) - x(W) \leq min\{|U|, |W|\} \tag{5}$$

defines a facet of the clique partitioning polytope $P(G)$, provided $|U| \neq |W|$.

Their experiments showed that these two kinds of constraints, especially the triangle constraints give a good approximation for the polytope.

The results below for $P_{BC}(G)$ and $P_{EC}(G)$ are from Barahona and Mahjoub [2], Barahona, Grotschel and Mahjoub [1], Conforti, Rao and Sassano[4]. We will use them in our later proofs.

Theorem 9 *([1]) $P_{BC}(G)$ is full dimensional, i.e., $dim(P_{BC}(G)) = |E|$.*

Theorem 10 *([4], Lemma 3.4) The dimension of $P_{EC}(K_{2p+1})$ is $\binom{2p+1}{2} - 1$.*

Theorem 11 *([4], Lemma 3.5) The dimension of $P_{EC}(K_{2p})$ is $\binom{2p}{2} - 2p$.*

Corollary 12 *The dimension of equipartition polytope $P_{EP}(K_{2p+1})$ is $\binom{2p+1}{2} - 1$. The dimension of equipartition polytope $P_{EP}(K_{2p})$ is $\binom{2p}{2} - 2p$.*

PROOF. $\forall y \in P_{EC}$, let $x = e - y$, then $x \in P_{EP}$. So every set of affinely independent vectors in P_{EC} corresponds to a set of affinely independent vectors in P_{EP}, thus the corollary follows.

2.2 Dimension for $R(G,S)$

Theorem 13 *The dimension of CPPMIN polytope $R(G,S)$ is*

(i) If $S < \frac{n}{2}$, then $dim(R(G,S)) = \binom{n}{2}$.
(ii) If $S = \frac{n}{2}$, then $dim(R(G,S)) = \binom{n}{2} - n$.

(iii) If $S > \frac{n}{2}$, then $dim(R(G,S)) = 0$.

PROOF. We are going to prove the theorem according to three cases.

(i) For $S < \frac{n}{2}$ case, consider:

(a) $n = kS + r$ **is odd**. Suppose $n = kS + r = 2p+1$, consider the hyperplane corresponding to an equipartition $H_{ep} = \{x \in R^{\binom{2p+1}{2}} : x(G) = \binom{2p+1}{2} - p(p+1)\}$. From Corollary 12, we have $dim(H_{ep}) = \binom{2p+1}{2} - 1$. Since $H_{ep} \subset R(G,S)$, $e \in R(G,S)$, $e \notin H_{ep}$, we get $dim(R(G,S)) = \binom{2p+1}{2} = \binom{n}{2}$.

(b) $n = kS + r$ **is even**. Suppose $kS + r = 2p$, we will consider the case when $S = p - 1$. We can embed the graph $G = K_{2p}$ into a complete graph K_{2p+1} by adding one more vertex v_{2p+1}. Let $m = \binom{2p}{2}$, $m' = \binom{2p+1}{2}$. From (a), we know $dim(R(K_{2p+1}, p)) = \binom{2p+1}{2} = m'$, so there exist $m'+1$ affinely independent incidence vectors $x^1, \ldots, x^{m'}, x^{m'+1} \in R(K_{2p+1}, p)$. They can be represented as the column vectors in a $m' \times (m'+1)$ matrix A'. According to the definition of affine independence, system $\begin{bmatrix} e^T \\ A' \end{bmatrix} \lambda = 0$ has a unique solution $\lambda = 0$. So $rank(\begin{bmatrix} e^T \\ A' \end{bmatrix}) = m' + 1$. Now we remove the $2p$ rows in A' that corresponds to the $2p$ edges incident to vertex v_{2p+1}. The remaining matrix $\begin{bmatrix} e^T \\ A \end{bmatrix}$ has full row rank $\binom{2p+1}{2} + 1 - 2p = \binom{2p}{2} + 1$. We can pick $\binom{2p}{2}+1$ linearly independent column vectors from $\begin{bmatrix} e^T \\ A \end{bmatrix}$ to get $\begin{bmatrix} e^T \\ B \end{bmatrix}$, now $\begin{bmatrix} e^T \\ B \end{bmatrix} \mu = 0$ implies $\mu = 0$, so the $\binom{2p}{2} + 1$ column vectors in B are affinely independent, and they correspond to the incidence vectors in $R(K_{2p}, p-1)$, so we have $dim(R(K_{2p}, p-1)) \geq \binom{2p}{2}$. Since $\binom{2p}{2}$ is already full dimension, we have $dim(R(K_{2p}, p-1)) = \binom{2p}{2} = \binom{n}{2}$.

In the case that $S \leq p-1$, $R(G,S) \supseteq R(G, p-1)$, so $dim(R(G,S)) = \binom{n}{2}$.

(ii) For $S = \frac{n}{2}$ case, it becomes an equipartition problem, so $dim(R(G,S)) = \binom{2S}{2} - 2S$ from Conforti and Rao[4],

(iii) For $S > \frac{n}{2}$ case, the only possible solution is to put all vertices in the same subset. Then polytope $R(G,S)$ degenerates to a single point, so $dim(R(G,S)) = 0$.

3 Cutting Planes

3.1 Bound Constraints

Before introducing the theorems to establish facets for CPPMIN, we introduce two lemmas first. Lemma 14 is almost exactly the same as lemma 3.2 from Chopra and Rao [3], except that we need to make sure here that every component in the partition satisfies the minimum size requirement. The proof, which we will skip here, follows the same as in Chopra and Rao [3].

Lemma 14 *Let $a^T x \geq b$ be any valid inequality with respect to $R(G, S)$, let j be any vertex in G. Consider the following partitions of G,*

$$\Pi_1 = N_1, N_2 \cup j, N_3, \ldots, N_r,$$
$$\Pi_2 = N_1 \cup j, N_2, N_3, \ldots, N_r.$$

If $\Pi_1, \Pi_2 \in R(G, S)$ and the incidence vector for Π_1 and Π_2 both satisfy the inequality $a^T x \geq b$ at equality, then we have $a(N_1, j) = a(N_2, j)$, where $a(N, j) = \sum_{v \in N} a(v, j)$.

CPPMIN polytope $R(G, S)$ is full dimension implies that every facet-defining inequality is unique up to multiplication by a constant (cf. [20] Theorem 3.5). We summarize in the next lemma another result on the relationship between the constraints of CPPMIN.

Lemma 15 *Suppose $a^T x \leq b$ and $g^T x \leq h$ are both valid constraints for CPPMIN polytope $R(G, S)$, s.t. $\{x | a^T x = b, x \in R(G, S)\} \subseteq \{x | g^T x = h, x \in R(G, S)\}$, then $a(i, j) = 0$ implies $g(i, j) = 0$ under the following condition: there exist a feasible partition incidence vector $\bar{x}$ s.t.*

- $a^T \bar{x} = b$,
- $\bar{x}_{ij} = 1$,
- *The cluster U containing vertex i and j in partition $\bar{x}$ satisfies $|U| \geq 2S+1$, $E(U) \subseteq E^c := \{e \in E : a_e = 0\}$.*

PROOF. $\forall v, u, w \in U$, $N_1 \subseteq U$, $N_2 \subseteq U$ s.t. $|N_1| \geq S$, $|N_2| \geq S$, $N_1 \cap N_2 = \emptyset$, $v \in U - N_1 - N_2$, $u \in N_1$, $w \in N_2$. Without loss of generality we can rearrange $\bar{x}$ s.t. $a^T \bar{x} = b$, $\bar{x}_e = 1, \forall e \in E(N_1 + v)$; $\bar{x}_e = 1, \forall e \in E(N_2)$; $\bar{x}_e = 0, \forall e \in \delta(N_1 + v, N_2)$. We can move v from the cluster of N_1 to the cluster of N_2 to construct another solution $\tilde{x}$ s.t. $a^T \tilde{x} = b$, and $\tilde{x}_e = 1, \forall e \in E(N_1)$; $\tilde{x}_e = 1, \forall e \in E(N_2 + v)$; $\tilde{x}_e = 0, \forall e \in \delta(N_1, N_2 + v)$.

Since $E(U) \subseteq E^c$, both $\bar{x}$ and $\tilde{x}$ satisfy $a^T x = b$, therefore both satisfy $g^T x = h$ too.

From lemma 14, we get $g(v, N_1) = g(v, N_2)$, which can be written equivalently as

$$g(v, N_1 - u) + g(v, u) = g(v, N_2 - w) + g(v, w). \tag{6}$$

Switch u and w between N_1 and N_2, repeat the above process, we get

$$g(v, N_1 - u) + g(v, w) = g(v, N_2 - w) + g(v, u). \tag{7}$$

(6)-(7) gives us

$$g(v, w) = g(v, u) =: \alpha \qquad \forall u, v, w \in U \tag{8}$$

Now consider another incidence vector $\hat{x}$ with $N = N_1 + N_2 + v$ all in one cluster, i.e. $\hat{x}(e) = 1, \forall e \in E(N)$. Comparing with incidence vector $\bar{x}$, we get

$$\begin{aligned} & \sum_{e \in \delta(N_1 + v, N_2)} g_e = 0 \\ \Rightarrow \quad & (|N_1| + 1)|N_2|\alpha = 0 \\ \Rightarrow \quad & \alpha = 0 \\ \Rightarrow \quad & g(u, v) = 0 \qquad \forall u, v \in U. \end{aligned}$$

In particular, $g(i, j) = 0$.

Theorem 16 *$x_{kl} \geq 0$ is a facet for $R(G, S)$ when $n \geq 3S + 1$.*

PROOF. This follows easily from Lemma 15 as follows. For every edge $(i, j) \neq (k, l)$, there exist a feasible solution with $x_{kl} = 0$ and $x_{ij} = 1$. Taking $U = V - k$ satisfies the other conditions in the lemma. Thus any constraint $g^T x \geq h$ that implies the constraint $x_{kl} \geq 0$ must have $g_{ij} = 0, \forall i, j \neq k, l$, therefore $g^T x \geq h$ becomes the same as $x_{kl} \geq 0$.

Note that $x_{ij} \leq 1$ is not a facet because it is implied by summing up 2 triangle constraints: $x_{ij} + x_{ik} - x_{jk} \leq 1$, $x_{ij} + x_{jk} - x_{ik} \leq 1$.

Triangle Inequalities and 2-partition Inequalities are inherited directly from CPP. Since CPPMIN is just a special case of CPP, these two kind of constraints are also valid for CPPMIN. Using lemma 15, we can show the following result for $R(G,S)$.

Theorem 17 *2-partition constraint* (5)*:* $x(U,W)-x(U)-x(W) \leq min\{|U|,|W|\}$ *is a facet for* $|U|>|W|$*, provided* $|V|>S|U|+2S+1$.

PROOF. Suppose $|U|=p, |W|=q$, we can write $U=\{u_l, l=1..p\}$, $W=\{w_l, l=1..q\}$. Consider any constraint $g^Tx=h$ such that $\{x|x$ satisfy (5) at equality, $x\in R(G,S)\} \subseteq \{x|g^Tx=h, x\in R(G,S)\}$,

(i) First, we would show that $g(i,j)=0, \forall (i,j)\in E-E(U\cup W)$.

Consider the partition $\Pi=\{N_1,N_2,...,N_q,...,N_p,N_{p+1}\}$ s.t. $u_l\in N_l$ for $l=1..p$, $w_l\in N_l$ for $l=1..q$, $N_{p+1}=V-\cup_{l=1}^{p}N_l$. Partition Π satisfies (5) at equality. We use x to denote its corresponding incidence vector.

$\forall i\in U$, $j\in V-U-W$, we can rearrange Π to also satisfy $j\in N_{\bar{l}}, |N_{\bar{l}}|\geq 2S+1$ for $\bar{l}$ s.t. $q<\bar{l}\leq p$ and $i\in N_{\bar{l}}$. Now the three conditions in Lemma 15 are satisfied by Π, namely, $a^Tx=b$, $x_{ij}=1$, $N_{\bar{l}}\subseteq E^c$, and $|N_{\bar{l}}|>2S+1$. So $g(i,j)=0$.

$\forall i\in V-U-W, j\in V-U-W$, we can rearrange Π to also satisfy $i\in N_{p+1}, j\in N_{p+1}, |N_{p+1}|\geq 2S+1$. Again Lemma 15 is satisfied on this partition. So $g(i,j)=0$.

Now we only need to show $g(i,j)=0, \forall i\in W, j\in V-U-W$. We can rearrange Π to satisfy $j\in N_{\bar{l}}, |N_{\bar{l}}|\geq S+1$ for $\bar{l}$ s.t. $i\in N_{\bar{l}}$. We call this partition $\bar{\Pi}$. Since $1\leq\bar{l}\leq q$, $N_{\bar{l}}\cap U\neq\emptyset$. Let $v=N_{\bar{l}}\cap U$.

We can get another partition $\tilde{\Pi}$ by switching vertex j from $N_{\bar{l}}$ to N_{p+1} in $\bar{\Pi}$. Both $\bar{\Pi}$ and $\tilde{\Pi}$ satisfy constraint (5) at equality, so we have

$$\begin{aligned} & g(j, N_{\bar{l}}-j)=g(j,N_{p+1}) \\ \Rightarrow \quad & g(i,j)+g(v,j)+g(j,N_{\bar{l}}-j-i-v)=g(j,N_{p+1}). \end{aligned}$$

Along with

$$\begin{aligned} g(v,j)=0 \qquad & \text{since } v\in U, \\ g(j,N_{\bar{l}}-j-i-v)=0 \qquad & \text{since } N_{\bar{l}}-j-i-v\subseteq V-U-W, \\ g(j,N_{p+1})=0 \qquad & \text{since } N_{p+1}\subseteq V-U-W, \end{aligned}$$

we get $g(i,j)=0, \forall i\in W, j\in V-U-W$.

Thus we have showed that $g(i,j)=0$, $\forall (i,j)\in E-E(U\cup W)$.

(ii) In this part, we are going to show that $g(u,v)=g(w,w')=-g(u,w)=-\alpha$, $\forall u,v\in U, w,w'\in W$.

$\forall u, v \in U, w, w' \in W$, without loss of generality, we can assume $u = u_1, v = u_{q+1}, w = w_1, w' = w_2$. Again we can construct a partition Π in the same way as in (i), $\Pi = \{N_1, N_2, ..., N_q, ..., N_p, N_{p+1}\}$ s.t. $u_l \in N_l$ for $l = 1..p$, $w_l \in N_l$ for $l = 1..q$, $N_{p+1} = V - \sum_{l=1}^{p} N_l$. Partition Π satisfies (5) at equality. With x denoting the incidence vector of Π, we have $g^T x = h$.

Now switch $u = u_1$ and $v = u_{q+1}$ between N_1 and N_{q+1}, we get partition $\bar{\Pi}$, represented by incidence vector $\bar{x}$, which also satisfies constraint (5), thus satisfying $g^T \bar{x} = h$. Comparing $g^T x = h$ and $g^T \bar{x} = h$, along with the result from (i), we get $g(u, w) = g(v, w)$. Since u, v, w are picked randomly from U and W, we have that for any fixed $w \in W$, $g(u, w) = \alpha_w$ is a constant $\forall u \in U$.

Modify partition Π by combining two clusters N_1 and N_{q+1}, we get partition $\tilde{\Pi}$, represented by incidence vector $\tilde{x}$, which also satisfies (5) at equality, so $g^T \tilde{x} = h$. Comparing $g^T x = h$ and $g^T \tilde{x} = h$, we get $g(u, v) = -g(v, w) = -\alpha_w$.

Since w is picked randomly from W, we have $g(u, v) = -\alpha, g(u, w) = \alpha, \forall u, v \in U, w \in W$.

Finally, we need to show $g(w, w') = -\alpha$. Since $w = w_1, w' = w_2$, combining clusters N_1 and N_2 in partition Π, we get partition $\hat{\Pi}$, represented by incidence vector $\hat{x}$. $\hat{x}$ satisfies constraint (5), thus satisfying $g^T \hat{x} = h$. Comparing with $g^T x = h$, we have $g(u, v) + g(w, w') + g(u, w') + g(v, w) = 0$. Since $g(u, v) = -\alpha, g(u, w') = g(v, w) = \alpha$, we have $g(w, w') = -\alpha \ \forall w, w' \in W$.

Summarize the above results, we get that $g(i, j) = 0, \forall (i, j) \in E - E(U \cup W)$; $g(u, v) = g(w, w') = -g(u, w) = -\alpha, \ \forall u, v \in U, w, w' \in W$. So we conclude that $g^T x = h$ is equivalent to the constraint (5). Therefore we have proved that (5) is a facet.

Since triangle constraints are just special cases of 2-partition constraints when $min\{|U|, |W|\} = 1$, the above theorem also applies to triangle constraints.

3.3 Pigeon Inequalities

From now on, we only consider the case when $r > 0$, i.e. when n is not a multiple of S. When $r = 0$, the new constraints we are going to introduce are still valid, but they may not be facets under the conditions given.

3.3.1 Pigeon Inequalities for k-way Partition $P1(G, k)$

To consider the pigeon constraint for CPPMIN problem, we would like to introduce the pigeon constraint for the k-way partition problem first. Given a

connected graph $G = (V, E)$ with edge weights c_e, the k-way partition problem is to partition the vertices V into no more than k nonempty subsets so as to minimize the total weight of the edges with end points in two different subsets. Chopra and Rao [3] investigated this problem, and gave the following integer programming formulation. Let $\Pi = (N_i, i = 1, 2, \ldots m)$ denote an $m-$partition of the vertices V, i.e., a partition that has m clusters. Each subset N_i in Π is assigned an index $t_i, 1 \leq t_i \leq m$ to identify the subset N_i. For partition Π, they defined an incidence vector (x, z) where

$$x_e = \begin{cases} 1 \text{ if edge } e \in E_\Pi \\ 0 \text{ otherwise} \end{cases} \quad z_{it} = \begin{cases} 1 \text{ if node } i \in N_t \\ 0 \text{ otherwise} \end{cases}$$

Notice that z variables are introduced here to differentiate different cliques, so as to take advantage of the sparsness of graph G, since G is not assumed to be complete in [3].

The corresponding polytope is defined as

$P1(G, k) = conv\{(x, z) | (x, z) \text{ is the incidence vector of an } m\text{-partition for } m \leq k\}$

Theorem 18 *([3], Theorem 3.1.1) Let $Q = (V(Q), E(Q)) \subseteq G$ be a clique of size $p = tk + q$ where $t \geq 1$ and $0 \leq q < k$. The pigeon inequality*

$$\sum_{e \in E(Q)} X_e \geq \frac{1}{2} t(t-1)(k-q) + \frac{1}{2} t(t+1)q \tag{9}$$

is facet defining for P1 if and only if $1 \leq q < k$, i.e. p is not an integer multiple of k.

3.3.2 Pigeon Inequality for $R(G, S)$

Now consider CPPMIN: S is the minimum size for each cluster, so the graph can be divided into no more than $k = \lfloor \frac{n}{S} \rfloor$ clusters. Thus any solution to our problem can be represented as a valid solution to Chopra and Rao's k-way partition problem, so the pigeon constraints are valid for our problem too. We will give a necessary and sufficient condition for pigeon constraint (9) to be a facet for $R(G, S)$.

As shown in Figure 1, a partition Π holding at equality for equation (9) would have the following structure: there are totally k components in the partition, $\Pi = (N_1, N_2, \ldots, N_k)$, $k - q$ of them each has exactly t vertices from Q, i.e., $|N_l \cap V(Q)| = t$ for $l = 1, 2, \ldots, k - q$. The other q components each has exactly $t + 1$ vertices from Q, i.e., $|N_l \cap V(Q)| = t + 1$ for $l = k - q + 1, \ldots, k$.

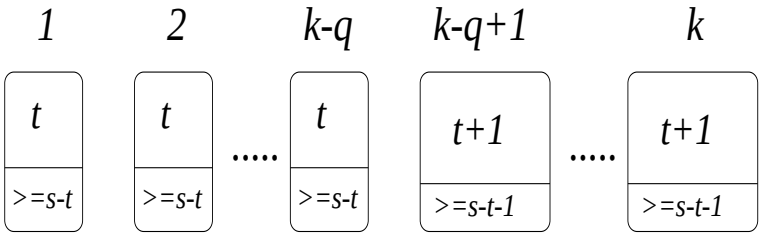

Fig. 1. Illustration for Pigeon Inequalities

Theorem 19 *Given a graph $G = (V, E)$ and an integer S, let $k = \lfloor \frac{n}{S} \rfloor$, $r = n \mod S$, i.e. $n = kS + r$. Let $Q = (V(Q), E(Q)) \subseteq G$ be a clique of size $p = tk + q$ where $t \geq 1$ and $1 \leq q < k$. Consider the pigeon constraint* (9)*,*

(i) When $r = 0$, pigeon constraint (9) *is valid for $R(G, S)$, but not facet defining.*
(ii) When $r = 1$, pigeon constraint (9) *is facet defining for $R(G, S)$ if and only if $Q = V$.*
(iii) When $r > 1$, pigeon constraint (9) *is always facet-defining for $R(G, S)$.*

PROOF. The proof goes according to the three cases.

(i) Case I: $r = 0$.

A partition Π satisfying Pigeon constraint (9) at equality implies that there are exactly k clusters, with $r = 0$, each cluster would have exactly S vertices. Thus the size constraints (1) hold at equality. In other words, the pigeon constraints are implied by the size constraints (1). So no pigeon constraints are facets.

(ii) Case II: $r = 1$.

Any feasible partition Π that satisfies constraint (9) at equality for a $p < n$ must also satisfy at equality the constraint (9) for $p = n$, which is written as (10),

$$\sum_{e \in E(V)} x_e \geq (k - r)\binom{S}{2} + r\binom{S+1}{2} \tag{10}$$

because it has to be of the structure with $k - 1$ clusters of size S and 1 cluster of size $S + 1$, i.e., $\Pi = (N_1, N_2, \ldots, N_k)$, where $|N_i| = S$ for $i = 1, \ldots, k - 1$, and $|N_k| = S + 1$. Thus constraint (9) with $p < n$ is not a facet. It will be shown in Part (iii) (b) along with the case when $r > 1$ that Constraint (10) is a facet.

(iii) Case III: $r > 1$.

Since a full-dimensional polyhedron has a unique (to within scalar multiplication) minimal representation by a finite set of linear inequalities (see Nemhauser and Wolsey [20], Theorem 3.5), we have the following property for a full dimensional polytope P: a valid constraint $a^T x \leq \alpha$ is a facet for P if and only if any valid constraint $b^T x \leq \beta$ of P satisfying

$\{x \in P | a^T x = \alpha\} \subseteq \{x \in P | b^T x = \beta\}$ must also satisfy $\frac{a_i}{b_i} = \frac{\alpha}{\beta}, \forall i$. In the following proof, we are going to show that when $r > 1$, pigeon constraints satisfy the above property, therefore they are facets.

Consider any valid inequality $b^T x \leq \beta$ s.t. $\{x \in P|\ x$ satisfies (9) at equality$\} \subseteq \{x | b^T x = \beta\}$. For clarity, we rewrite the biggest pigeon constraint (10) as the following,

$$\sum_{e \in E(Q)} x_e + \sum_{e \in \delta(Q)} x_e + \sum_{e \in E(V-Q)} x_e \geq (k-r)\binom{S}{2} + r\binom{S+1}{2} \tag{11}$$

$\forall v \in V$, we can construct a partition $\Pi = (N_1, N_2, \ldots, N_k)$, where $|N_l| \geq S$, for $l = 1, 2, \ldots, k$; $|N_l \cap V(Q)| = t$, for $l = 1, 2, \ldots, k-q$; $|N_l \cap V(Q)| = t+1$, for $l = k-q+1, \ldots, k$; and $v \in N_i, |N_i| \geq S+1$, $|N_i \cap V(Q)| = t+1$.

(a) By moving vertex v from V_i to V_1, we obtain partition $\Pi^{'} = (N_1^{'}, N_2, N_3, \ldots, N_i^{'}, \ldots, N_k)$, where $N_1^{'} = N_1 \cup \{v\}$, $N_i^{'} = N_i - \{v\}$.

No matter whether $v \in V(Q)$ or not, both Π and $\Pi^{'}$ satisfy pigeon constraint (9) at equality. From lemma 14, we have

$$b(N_1, v) = b(N_i - v, v) \tag{12}$$

(b) Now suppose $v \in V(Q)$, we are going to show $b(i_1, v) = b(i_2, v) =: \delta_v, \forall i_1, i_2, v \in V(Q)$. This is exactly the same as the proof in [3]. For completeness, we put it down here again.

We can assume the partition Π was constructed so that $i_1 \in N_1 \cap V(Q), i_2 \in N_i \cap V(Q)$. Switching i_1 and i_2, we get another partition $\bar{\Pi} = (\bar{N_1}, N_2, \ldots, \bar{N_i}, \ldots, N_k)$ where $\bar{N_1} = N_1 \cup i_2 - i_1$, $\bar{N_i} = N_i \cup i_1 - i_2$. Now repeat the procedure (a) with partition $\bar{\Pi}$, we will get

$$b(\bar{N_1}, v) = b(\bar{N_i} - v, v) \tag{13}$$

In other words, we move v from $\bar{N_i}$ to $\bar{N_1}$ in $\bar{\Pi}$, to obtain partition $\hat{\Pi} = (\bar{N_1} + v, N_2, \ldots, \bar{N_i} - v, \ldots, N_k)$. Since both $\bar{\Pi}$ and $\hat{\Pi}$ satisfy pigeon constraint at equality, from lemma 14, we obtain (13).

Rewrite (12) and (13), we have

$$b(N_1 - i_1, v) + b(i_1, v) = b(N_i - v - i_2, v) + b(i_2, v) \tag{14}$$
$$b(\bar{N_1} - i_2, v) + b(i_2, v) = b(\bar{N_i} - v - i_1, v) + b(i_1, v) \tag{15}$$

(14) − (15) give us

$$b(i_1, v) = b(i_2, v) = \delta_v \qquad \forall i_1, i_2 \in Q \tag{16}$$

Since i_1, i_2, v are chosen arbitrarily, we have

$$b(i, v) = \delta_v \qquad \forall i, v \in V(Q) \tag{17}$$

This actually implies what we have stated in part (ii) of the theorem, i.e., when $Q = V$, the biggest pigeon constraint (9) is a facet.

(c) For the rest of the proof, we assume $v \in V - V(Q)$. Let $i_1 \in N_1 \cap V(Q)$, $i_2 \in N_i \cap V(Q)$.

Repeat the procedure in (b) on partition Π, we get

$$b(i_1, v) = b(i_2, v)$$

Again, since v, i_1, i_2 are chosen arbitrarily, we have

$$b(i, v) = \alpha_v, \forall i \in V(Q), v \in V - V(Q) \tag{18}$$

(18) holds for every $v \in V - V(Q)$.

(d) When $|V - V(Q)| \geq 3$, without loss of generality, let $j_1, j_2 \in V - V(Q)$, $j_1 \in N_1, j_2 \in N_i$. Switching j_1 and j_2, we get $\tilde{\Pi} = (\tilde{N_1}, N_2, \ldots, \tilde{N_i}, \ldots, N_k)$ where $\tilde{N_1} = N_1 - j_1 \cup j_2$, $\tilde{N_i} = N_i - j_2 \cup j_1$.

Repeat the procedure in (a) with $\tilde{\Pi}$, we have

$$b(\tilde{N_1}, v) = b(\tilde{N_i} - v, v) \tag{19}$$

Rewrite these two equations (12) and (19), we have

$$b(N_1 - j_1, v) + b(j_1, v) = b(N_i - r - j_2, v) + b(j_2, v) \tag{20}$$
$$b(\tilde{N_1} - j_2, v) + b(j_2, v) = b(\tilde{N_i} - v - j_1, v) + b(j_1, v) \tag{21}$$

(20) − (21) gives us

$$b(j_1, v) = b(j_2, v)$$

Again since j_1, j_2, v are chosen arbitrarily, we have

$$b(j, v) = \beta_v, \forall j \in V - V(Q), v \in V - V(Q) \tag{22}$$

When $|V - V(Q)| < 3$, (22) is obviously true.

(e) From equation (12) in part (a), we have

$$\begin{aligned} & b(N_1, v) = b(N_i - v, v) \\ \Rightarrow \quad & t\alpha_v + (|N_1| - t)\beta_v = (t+1)\alpha_v + (|N_i| - t - 1)\beta_v \\ \Rightarrow \quad & \alpha_v = (|N_1| - |N_i| + 1)\beta_v, \forall v \in V - V(Q) \end{aligned} \tag{23}$$

since $r > 1, v \notin Q$, we can construct at least two different valid partitions, one with $|N_1| = |N_i|$ and the other with $|N_1| + 1 = |N_i|$. To have equation (23) hold for both of these two partitions, we have to have

$$\alpha_v = \beta_v = 0 \tag{24}$$

(f) Having (17), (18), (22), (24) together, we have proved that any constraint $b^T x \geq \beta$ dominating pigeon constraint (9) has the same coefficient for the edges within Q and 0 as the coefficient for all other edges, i.e., $b^T x \geq \beta$ is of the form

$$\sum_{e \in E(Q)} x_e \geq \beta \tag{25}$$

Thus we have shown that pigeon constraint (9) is a facet when $r > 1$.

We give two examples to illustrate cases (ii) and (iii) in the theorem.

Example I: A pigeon constraint that is not a facet, shown in Figure 2. Consider a CPPMIN on $G = K_7$, with $S = 3$. Here $n = 7, S = 3, k = 2, r = 1$. Let W be any $q = 3$ vertices from V. The pigeon constraint on W: $x(W) \geq 1$ is not a facet because any solution satisfying this inequality at equality will satisfy the biggest pigeon constraint, i.e. the pigeon constraint involving all edges, $X(G) \geq 9$ at equality. The biggest pigeon constraint is a facet. In the figure, $W = \{v_3, v_4, v_7\}$, the dash line is the support graph for the pigeon constraint on it. The black lines gives one feasible solution to the CPPMIN problem.

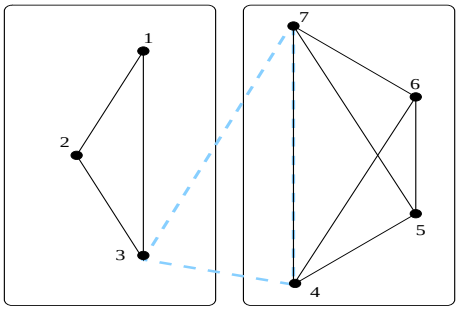

Fig. 2. Example I - Pigeon Constraint on K_7, with $q = 3$

Example II: A pigeon constraint that is a facet, shown in Figure 3. Consider complete graph $G = K_8$, here $n = 8, S = 3, k = 2, r = 2$. Let W be any $q = 4$ vertices from V. The pigeon constraint on W: $x(W) \geq 2$ is in fact a facet. For a problem of this size, it is possible to verify the result by enumerating all the solutions that satisfy this inequality at equality. In Figure 3, $W = \{v_2, v_3, v_7, v_8\}$, the dash line is the support graph for the pigeon constraint on it. The black lines gives one feasible solution to the CPPMIN problem, which satisfies the pigeon constraint on W: $x(W) \geq 2$ at equality, but does not satisfy the pigeon constraint on V, i.e., equation (10) at equality.

3.3.3 How to Find Violated Pigeon Constraints

Take the complete graph G in the CPPMIN problem with the edge weight changed into the corresponding x value in the LP solution. The problem of finding violated pigeon constraints of size p is the same as the problem of finding a clique of size p on this weighted graph that has the total weight smaller than the minimum value required in the pigeon constraint. If we know the minimum weight clique of size p, then we can either find a violated constraint or prove that there is no pigeon constraint of size p violated. However,

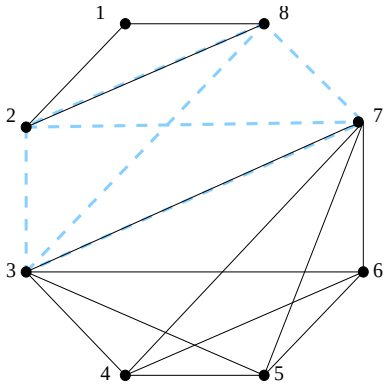

Fig. 3. Example II - Pigeon Constraint on K_8, with $q = 4$

in general, the problem of finding minimum weight clique of size p is a NP-hard problem, except for some specific value of p, such as $p = n$, $p = n - 1$, $p = 1$, $p = 2$. Thus in our experiment, we will look for only 3 kinds of pigeon constraint as follows:

- when $p = n = |V|$, there is only one pigeon constraint. We simply add it into the initial formulation.

- when $p = n - 1$, there are n such pigeon constraints, and they are easy to check. We add the violated ones in the cutting plane step. But in our experiment, this kind of constraint is seldom violated, consequently seldom added.

- When $p = k + 1$, there are $\binom{n}{k+1}$ such pigeon constraints. They are time-consuming to enumerate, and it is hard to find the violated ones. but these constraints are not as dense as the previous ones, and they are more often violated as shown from the experiments.

  We modified a greedy algorithm for maximum dispersion suggested by Hassin [10] to look for the minimum clique of size p. Our modified algorithm is shown in Algorithm 1. Note that one reason that we could adopt this method successfully is based on the assumption that the LP solutions satisfy the triangle inequalities. For this reason and also because pigeon constraints are comparably more dense, we add in this kind of pigeon constraints only when no triangle and 2-way constraints are added in the current iteration. Through the experiments we see that the majority of the added pigeon constraints are of this kind.

3.4 Flower Constraints

3.4.1 Flower Constraints for $R(G, S)$

Lemma 20 *Given G=(V,E), let $W = (V(W), E(W)) \subseteq V$ be a clique of size $w = S + q$, $q < S$, the flower inequality*

Algorithm 1 Find a small weight clique of size $k+1$

Input:

n - the total number of vertices in G

k - number of clusters

lpx - the linear program solution

Output:

support - a small weight clique of size $(k+1)$

sweight - the total weight of *support*

Steps:

Let $G' = (V', E')$ be K_n, a complete graph of size n.

Let $support = \emptyset$.

for $i = 1$ to $(k+1)/2$ **do**

Find an edge (v_i, v_j) s.t. $lpx(v_i, v_j) = min\{lpx(v_k, v_l) | (v_k, v_l) \in G'\}$.

Let $support = support + \{v_i, v_j\}$.

Let $V' = V' - \{v_i, v_j\} - \cup\{v | lpx(v, v_i) > 0.5 \text{ or } lpx(v, v_j) > 0.5\}$.

Let $E' =$ a complete graph on V'.

end for

If k is odd, add to *support* an arbitrary vertex.

Let $sweight = \sum_{v_i, v_j \in support} lpx(v_i, v_j)$.

$$\sum_{e \in E(W)} x_e + \sum_{e \in \delta(W)} x_e \geq \binom{S}{2} + \binom{q}{2} + q(S-q), \qquad \forall W \subset V \tag{26}$$

is a valid constraint for $R(G, S)$.

PROOF. Equation (26) is at equality when S vertices from W are clustered together, and the other q vertices are clustered together. Following from the minimum size constraint on cluster size, each vertex in the set of q vertices has to be connected to $S - q$ vertices outside W. This is the minimum value for the number of edges connected with W, because to replace any 1 edge inside W, we have to add another 2 edges connected to $V \backslash W$ to maintain the size constraint (1); and to shift any vertex from the group of size q to the group of size S would also increase the total number of edges by $(S - q)$.

Figure 4 shows the support graph for an example of constraint (26). Here $w = 7$, $S = 4$, $q = 3$.

$$\sum_{e \in E(W)} x_e + \sum_{e \in \delta(W)} x_e \geq 9 + 3 = 12 \tag{27}$$

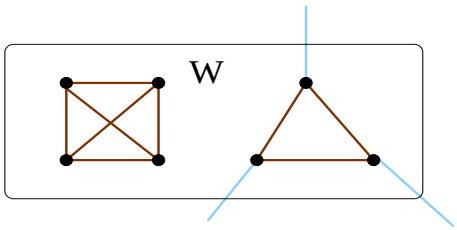

Fig. 4. Illustration for Flower Inequalities

Figure 5 illustrates the two possible ways to tighten up constraint (27), which corresponds to the solid line in the figure. The values of $\sum_{e \in \delta W} x_e$ and $\sum_{e \in E(W)} x_e$ are plotted in the graph as x axis and y axis. The tightenings modify the coefficients of (27) so that other configurations of W also satisfy the constraint at equality. We are interested in two particular configurations: one is to have every vertex in W in one big cluster so that $\sum_{e \in E(W)} x_e = 21$ and $\sum_{e \in \delta(W)} x_e = 0$; the other is to have each vertex in W in a separate cluster so that $\sum_{e \in E(W)} x_e = 0$ and $\sum_{e \in \delta(W)} x_e = 21$. These give the constraints corresponding to the dashed line and the dotted line, respectively. The cutting plane generated from the latter case turns out to be already implied by the degree constraints. So we are left with the first case, tightening up the solid line to the dashed line.

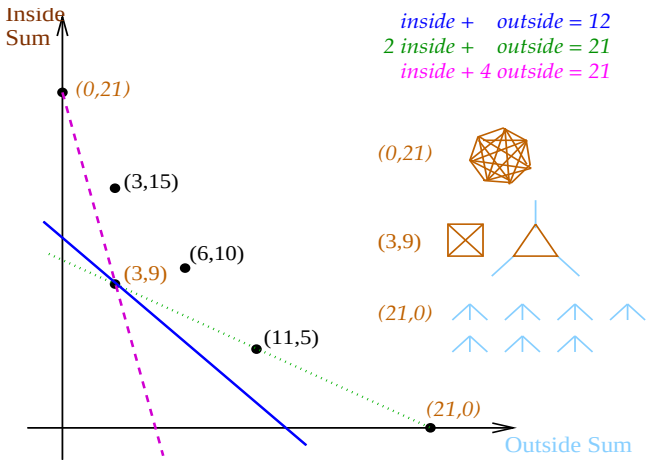

Fig. 5. Lifting Flower Inequalities

As we can see from figure 5, to the right of the intersection point (3,9), the solid line is already tighter than the dashed line. Since solid line is already valid, the dashed line has to be valid. This gives the first part of the proof for the following theorem, which summarizes the above observations.

Theorem 21 *The lifted flower constraint*

$$(S-q)\sum_{e \in E(W)} x_e + S\sum_{e \in \delta(W)} x_e \geq (S-q)\binom{S+q}{2} \tag{28}$$

is a valid constraint for $R(V,S)$, $\forall W \subset V, |W| = S+q, 0 \le q < S$.

PROOF. For any partition Π on W, let x be the incidence vector of Π. We will prove the theorem according to two cases, depending on the value of $\sum_{e\in\delta(W)} x_e$ for Π.

(i) When $\sum_{e\in\delta(W)} x_e > (S-q)q$,

$$\begin{aligned}
&(S-q)\sum_{e\in E(W)} x_e + S\sum_{e\in\delta(W)} x_e \\
&= (S-q)(\sum_{e\in E(W)} x_e + \sum_{e\in\delta(W)} x_e) + q\sum_{e\in\delta(W)} x_e \\
&\ge (S-q)(\frac{S(S-1)}{2} + \frac{q(q-1)}{2} + q(S-q)) + q\sum_{e\in\delta(W)} x_e \\
&\qquad \text{from (26)} \\
&> (S-q)(\frac{S(S-1)}{2} + \frac{q(q-1)}{2} + q(S-q)) + q(q(S-q)) \\
&\qquad \text{from the assumption } \sum_{e\in\delta(W)} x_e > (S-q)q \\
&= (S-q)(\frac{S(S-1)}{2} + \frac{q(q-1)}{2} + qS) \\
&= (S-q)\binom{S+q}{2}
\end{aligned}$$

(ii) When $\sum_{e\in\delta(W)} x_e \le (S-q)q$, partition Π on W has to be of the form $\Pi = \{W_1, W_2\}$, and $S' = |W_1| \ge S$, $r' = |W_2| \le q$, $S'+q' = S+q$. Let $w \in W_2$. Define another two partitions on W, $\hat{\Pi} = \{W_1 + w, W_2 - w\}$ and $\tilde{\Pi} = \{W\}$. Let

$$\begin{aligned}
\widehat{inside}_+ &= |E(\hat{\Pi})| - |E(\Pi)|, \\
\widehat{outside}_- &= |\delta(\Pi)| - |\delta(\hat{\Pi})|, \\
\widetilde{inside}_+ &= |E(\tilde{\Pi})| - |E(\Pi)|, \\
\widetilde{outside}_- &= |\delta(\Pi)| - |\delta(\tilde{\Pi})|
\end{aligned}$$

So the value of $\widehat{inside}_+ / \widehat{outside}_-$ represents the the absolute value of the slope of the line connecting partitions Π and $\hat{\Pi}$, and the value of $\widetilde{inside}_+ / \widetilde{outside}_-$ represents the absolute value of the slope of the line connecting partitions Π and $\tilde{\Pi}$. To compare these two values, we need to compute first

$$\widehat{inside}_+ = S' - (q' - 1),$$
$$\widehat{outside}_- = (S - q') - (q' - 1),$$
$$\widetilde{inside}_+ = S'q',$$
$$\widetilde{outside}_- = (S - q')q'.$$

Therefore we have

$$\frac{\widehat{inside}_+}{\widehat{outside}_-} = \frac{S' - (q' - 1)}{(S - q') - (q' - 1)} = 1 + \frac{S' - S + q'}{(S - q') - (q' - 1)},$$
$$\frac{\widetilde{inside}_+}{\widetilde{outside}_-} = \frac{S'}{S - q'} = 1 + \frac{S' - S + q'}{S - q'}.$$

So we get

$$\frac{\widehat{inside}_+}{\widehat{outside}_-} \geq \frac{\widetilde{inside}_+}{\widetilde{outside}_-}, \tag{29}$$

i.e. the absolute value of the slope of the line connecting Π and $\hat{\Pi}$ is always bigger or equal to the absolute value of slope of the line connecting Π and $\tilde{\Pi}$. This implies that $\hat{\Pi}$ will satisfy equation (28) as long as both $\tilde{\Pi}$ and Π satisfy equation (28). We can easily verify that $\tilde{\Pi}$ satisfy (28), so we only need to find one initial partition Π satisfying equation (28) too. Since partition $\{W_1, W_2\}$, $S' = |W_1| = S$, $q' = |W_2| = q$ satisfies (28), starting with this partition, and from induction we get that all other partitions satisfy (28).

Part (ii) of the proof is also illustrated in figure 5. Notice that a line connecting points $\Pi = (3, 9)$ and $\hat{\Pi} = (3, 15)$ has a bigger slope in absolute value than the dash line that connects $\Pi = (3, 9)$ and $\tilde{\Pi} = (0, 21)$, which also corresponds to constraint (28), so point $\hat{\Pi} = (3, 15)$ is above this dash line, in other words, constraint (28) is valid for the partition $\hat{\Pi}$ corresponding to point (3,15).

From lemma 15, we know any valid constraint $g^T x \leq h$ implying a flower inequality must have $g(i, j) = 0$, $i, j \notin W$. But the flower constraint is not a facet in general. For example, see Figure 6, when $|W| = 5, S = 4, q = 1$, any incidence vector satisfying the following flower constraint at equality

$$3 \sum_{e \in E(W)} x_e + 4 \sum_{e \in \delta(W)} x_e \geq 30 \tag{30}$$

also satisfies the following constraint at equality:

$$-\sum_{e\in cycle(W)} x_e + 4 \sum_{e\in E(W)-cycle(W)} x_e + 2 \sum_{e\in\delta(W)} x_e \geq 15 \tag{31}$$

where $cycle(W)$ is a cycle in W. Figure 6 shows the only two configurations satisfying flower constraint (30) at equality. They also both satisfy constraint (31) on any cycle covering the 5 vertices, for example cycle ABCDEA. The question remains open on how to lift the flower constraint to a facet.

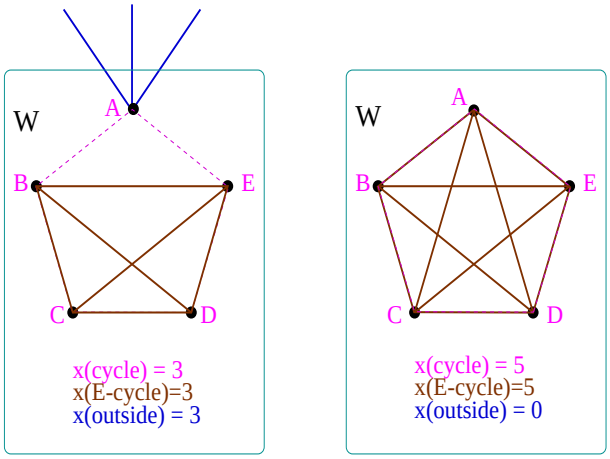

Fig. 6. Two Configurations Satisfying Inequalities (31) at Equality

3.4.2 How to Find Violated Flower Constraints

Since flower constraints are dense constraints, we only look for them when we couldn't find other cutting planes. Again, we use heuristic method to find them. According to the current LP solutions, we divide the vertices into connected components heuristically, then we check if any of these components violate the flower constraint. The intuition behind this strategy is a realization that the flower constraint is more likely to be violated when the x weights are more concentrated on the edges within a component. For example, consider again the example of figure 4, set $x_e = \frac{1}{2}$ for $e \in E(W)$ and $x_e = 0$ for $e \in \delta(W)$. This solution satisfies the degree constrains on W, but violates flower constraint (28).

4 Computational Results

In this part, we are going to show the frame work of our algorithm and the computational results.

4.1 Branch and Cut algorithm

Algorithm 2 shows the framework of a branch and cut algorithm that we use here. When we are concentrating on adding in cutting planes at the root node, the LP relaxations are solved by an interior point method, this is because, initially the cutting planes are deep, the LP solution is only used to generate cutting planes, an absolute optimal solution is not necessary. In fact, a close to optimal solution helps to generate deeper cutting planes. After we start branching, we switch to the simplex method to solve the LP relaxation, since at this time the LP solution are much closer to optimality. A more detailed explanation on the combination of interior point method and simplex method in a branch and cut algorithm can be found in Mitchell and Borchers[18].

The cutting planes that are specific to this problem are added in the separation routine shown in algorithm 3.

Algorithm 2 Branch and Cut Framework

1: Initialize.
2: Approximately solve the current LP relaxation using an interior point method.
3: If the gap between the value of the LP relaxation and the value of the incumbent integer solution is sufficiently small, STOP with optimality.
4: If the duality gap for the current LP is smaller than 10^{-8} or if a given limit on the number of outer iterations have been reached, call MINTO [19] to do a branch-and-cut until an optimal solution is proved or a time limit runs out. To guarantee the solution returned by MINTO is a feasible solution, we add in violated triangle constraints as cutting planes.
5: Try to improve the incumbent solution locally by switching vertices and moving extra vertices around.
6: Use the separation routine Algorithm 3 to find violated cutting planes and return to Step 2.

4.2 A Heuristic Algorithm

Domingo-Ferrer and Mateo-Sanz [5] surveyed heuristic methods for a very similar problem to CPPMIN in the application to micro-aggregation problems. The difference between the problem discussed in [5] and CPPMIN is the objective function. A slightly different objective function is used in [5] to model the homogeneity within a cluster. We implemented one heuristic algorithm for this slightly different problem in [5] to compare with our cutting plane approach. The reason we choose this particular algorithm is because its one-run performance is among the best of the heuristic algorithms mentioned in [5]. Notice computation time is not an issue here, instead the optimality

Algorithm 3 Separation Routine

1: The algorithm first searches for triangle inequalities using complete enumeration. Inequalities are bucket sorted by the size of the violation. Inequalities are added starting with those in the most violated subset, ensuring that no two of these added inequalities share an edge. The violation of the last constraint added is restricted to be no smaller than a multiple of the violation of the first constraint added.
2: If not enough constraints are added so far, add 2-partition inequalities.
3: If not enough constraints are added so far, add pigeon constraints of size $n-1$.
4: If not enough constraints are added so far, add pigeon constraints of size $k+1$.
5: If not enough constraints are added so far, add flower constraints.

of the heuristic solution is important. The heuristic algorithm is illustrated in Algorithm 4.

Algorithm 4 A Heuristic Algorithm for CPPMIN

1: Let $U = V$.
2: Find the two most distant vertices $x_s, x_t \in U$.
3: Form a cluster around x_s with the $S-1$ closet vertices to x_s in U, remove these S vertices from U.
4: Form a cluster around x_t with the $S-1$ closet vertices to x_t in U, remove these S vertices from U.
5: If $|U| \geq 2S$, go back to step 2.
6: If $S \leq |U| < 2S$, form a new cluster using these vertices.
7: If $|U| < S$ assign each vertex to the same cluster its closest neighbor belongs to.
8: Try to improve the solution locally by switching vertices and moving extra vertices around.

4.3 Random Uniform Problems

In this category, we generate 2 types of data.

Type I : Vertices are generated randomly on a unit square following a uniform distribution. The edge weight is the integral part of 100 times the distance between vertices.

Type II : In this case, we don't generate vertices explicitly, instead, edge weights are generated directly as uniformly distributed random variables between 1 and 100. In this case, the edge weights do not satisfy the the triangle inequalities any more, thus the problem turns out to be much harder to solve than Type I problems, because on average, every vertex is equally close to every other vertex.

For each type of data, we consider two choices of S, $S = 4$ or $S = 7$. In the following discussion, we will be concentrating on the case of $S = 4$, but as one can see from the tables, the case of $S = 7$ is very similar, so the analysis for the case of $S = 4$ holds the same as for the case of $S = 7$. In Tables 1, 2 and 4, we consider $S = 4$, we generate 5 groups of problems between 21 and 103 vertices. Each group includes all the 3 cases of remainders, 1, 2 and 3. Our experimental data showed that there is no major difference between these three cases, so we don't show the results separately in the tables. 5 problems of each case were generated, so totally, for each column, 15 instances were generated and solved. Every number reported in the tables is the average performance of 15 instances with the same number of partitions k.

Every problem is tackled in three ways. First, the heuristic method is used to find a good solution, then the cutting plane approach tries to generate a good linear programming approximation to the problem. If this can not prove the optimality of the best feasible solution, we resort to a branch-and-cut code in MINTO.

Accordingly, each table is divided into 3 blocks, corresponding to these three approaches, labelled as "Heuristic Alg", "Cutting Plane" and "MINTO run" respectively. The first block gives the gap and time after running the heuristic algorithm written in Fortran 77. The second block corresponds to the cutting plane stage of the algorithm. At each iteration, the LP relaxation is solved by an interior point method, implemented in Fortran 77. The rows in the table give the total number of instances, the number of problems solved to optimality at this stage, how many cutting plane solutions improved the heuristic solution, the average final gap, the average running time (in seconds), the average number of total number of cuts, pigeon constraints and flower constraints added, the average number of outer iterations, and the average number of interior point iterations. The third block is the result after branching using MINTO 3.0.2. For problems not solved by the cutting plane scheme, we finish the process off with MINTO to try to obtain an optimal solution. It reads in the cutting plane formulation from an MPS file, then uses a branch and cut algorithm with only triangle constraints added in as cutting planes. The LP relaxations are solved by simplex method using CPLEX 6.6. The runtimes are reported with an upper bound of 500 seconds. The results reported include the total number of problems that needs to call MINTO, the number of problems solved to optimality by MINTO before reaching time limit, how many MINTO IP solutions improved best upper bound from the cutting plane stage, the final gap between the best IP solution and LP lower bound, total running time, the number of branch and bound nodes. The heuristic and cutting plane algorithms are coded in FORTRAN. The MINTO branch and bound part is coded in C. The experiments are done on a Sun Ultra 10 Workstation.

To demonstrate the strength of the cutting planes, first we would like to mention that feeding all the triangle inequalities to CPLEX to solve the problem directly as an IP is impractical; CPLEX was unable to solve a 25 vertex instance in 1 day. The strength of the pigeon and flower constraints is shown by comparing Table 1 and Table 2. Both tables used the same set of testing data, but in Table 1, we did not try to add in any pigeon or flower constraints. The block labelled "Improvement" compared the difference between the two sets of results after cutting plane code, first by the number of better integer solutions found, then by the improvement on LP lower bound. Considering these improvements with the extra time we have to spent to look for pigeon and flower constraints, which is about 1 or 2 minutes more, we can say that pigeon and flower constraints are important constraints for CPPMIN problems. Of course the performance of the branch and bound part of the problem is improved too in Table 2 since we started with a tighter relaxation here. Figure 7 shows a typical run of the cutting plane algorithm. We show the performance with triangle and 2-way constraints only for Type I data here, but similar results are observed in other kind of data too. Table 3 for the case of $S = 7$ follows the same structure.

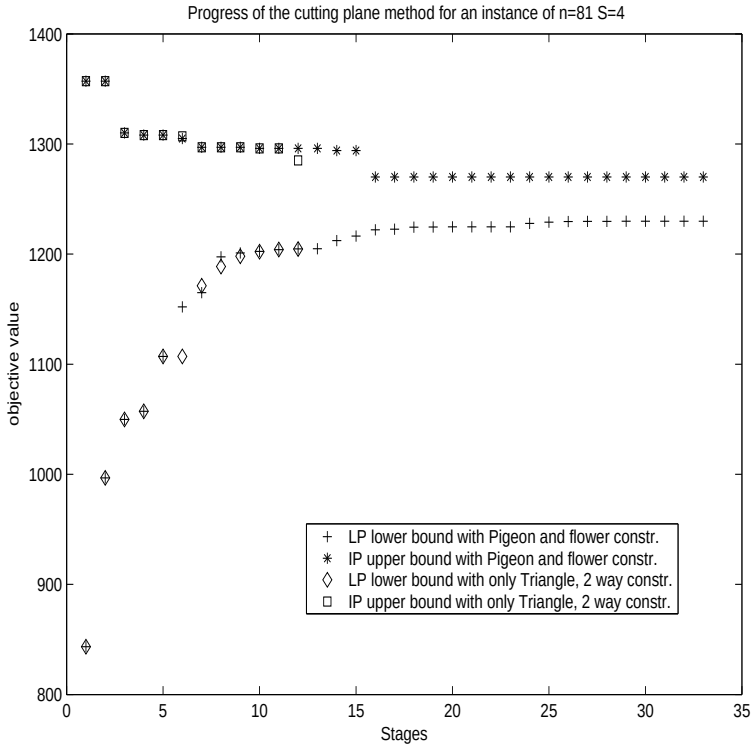

Fig. 7. Progress of the Cutting Plane Algorithm

Table 2 and Table 3 show that the cutting plane algorithm was typically able to solve the Type I problems within 4% of optimality in about 3 minutes for a problem of size around 100. The branching stage basically just check the optimality of the solution. So we only checked and added triangle constraints.

The numerical results show that the branch and bound part does not improve the solution or the gap much, so it is probably not necessary in most of the cases.

Table 4 shows the result for Type II problems. These are harder partition problems, because on average, each vertex is equally close to every other vertex. A lot more violated triangle inequalities are generated since the edge weights do not satisfy the triangle inequality any more. As we can see from Table 4, it takes much more time to solve them. Due to the difficulty of this type of problems, we only show the case for $S = 4$.

The CPPMIN is a partition minimizing problem, it can be equivalently formed as a multi-cut maximization problem, which maximize the sum of inter-cluster distances. In the latter case, the error percentage is far smaller. In our CPPPMIN formulation, suppose the best integer solution found is $\bar{x}$, the optimal solution to the lp relaxation is x. We compute the relative gap as

$$gap_{min} = \frac{\sum_{e \in E} c_e \bar{x_e} - \sum_{e \in E} c_e x_e}{\sum_{e \in E} c_e x_e}$$

In the equivalent problem of maximizing the sum of inter-cluster distances, the corresponding relative gap would be

$$\begin{aligned} gap_{max} &= \frac{\sum_{e \in E} c_e (1 - x_e) - \sum_{e \in E} c_e (1 - \bar{x_e})}{\sum_{e \in E} c_e (1 - x_e)} \\ &= \frac{\sum_{e \in E} c_e \bar{x_e} - \sum_{e \in E} c_e x_e}{\sum_{e \in E} c_e - \sum_{e \in E} c_e x_e} \end{aligned}$$

When the graph G is a complete graph, most of the x_e are zero, therefore, gap_{max} is much smaller than gap_{min}. For example, for a Type I problem of 102 nodes, a gap of $gap_{min} = 3\%$ in our formulation can easily correspond to a gap of around $gap_{max} = 0.02\%$ in this alternative formulation.

4.4 Micro-aggregation Problems

One application of CPPMIN that we are particularly interested in is micro-aggregation problem. Micro-aggregation is a technique to process statistical data for releasing to the public, so that the confidentiality of respondents is protected and the informational content of the data are preserved as much as possible. It is commonly used in economic data where respondent identifiability

n	21-23	41-43	61-63	81-83	101-103
$k = \lfloor \frac{n}{S} \rfloor$	5	10	15	20	25
Heuristic Alg.					
Gap	9.85%	12.32%	13.12%	14.54%	12.68%
Time	0.0060	0.0141	0.0278	0.0441	0.0672
Cutting Plane	15	15	15	15	15
Solved exactly	1	0	0	0	0
Better Solution	12	14	15	15	15
Gap	4.63%	4.65%	5.28%	5.97%	5.37%
Time	0.89	3.80	7.98	19.43	41.05
Cuts added	64	121	168	227	311
Pigeon	-	-	-	-	-
Flower	-	-	-	-	-
Outer Iter	8	12	12	15	18
Inner Iter	55	120	132	195	253
MINTO run	14	15	15	15	15
Solved exactly	14	15	14	4	0
Better Solution	0	2	9	12	12
Gap	0%	0%	0.55%	4.28%	4.47%
time	3.26	41.43	189.18	417.39	500.60
nodes	39	156	361	457	290

Table 1
Branch-and-Cut Results on CPPMIN Type I Problems for $S = 4$ with only Triangle and 2-way constraints

is high. Data are divided into groups with varying size ($\geq$ fixed size), to avoid a large gap within a group. For univariate data, the problem can be solved efficiently using a dynamic programming approach after sorting the data, see [9], but for higher dimensional data, sorting is no longer well defined. Sande [22] introduced the problem and several possible methods for higher dimensional data. Domingo-Ferrer and Mateo-Sanz [5] gives a good summary on the heuristic methods. Micro-aggregation problems in higher dimensions can be described as clustering problems with a variable number of clusters and a minimum cluster size.

Data from micro-aggregation problems usually has the following two important properties one is that cluster size are usually small, i.e $S << k$; the other

n	21-23	41-43	61-63	81-83	101-103
$k = \lfloor \frac{n}{S} \rfloor$	5	10	15	20	25
Heuristic Alg.					
Gap	6.97%	10.27%	11.04%	12.40%	11.19%
Time	0.0066	0.0143	0.0273	0.0438	0.0664
Cutting Plane	15	15	15	15	15
Solved exactly	4	1	0	0	0
Better Solution	12	14	15	15	15
Gap	1.87%	2.67%	3.02%	3.65%	3.77%
Time	6.99	23.08	58.42	110.58	175.63
Cuts added	97	178	261	339	433
Pigeon	12	9	9	8	7
Flower	2	4	7	9	8
Outer Iter	16	25	31	36	37
Inner Iter	121	285	415	523	577
Improvement					
Better than Tri	0	1	0	8	7
LPOBJ Improve	2.57%	1.75%	1.82%	1.87%	1.31%
MINTO run	11	14	15	15	15
Solved exactly	11	14	14	5	0
Better Solution	0	1	3	8	6
Gap	0%	0%	0.30%	2.67%	3.47%
time	3.25	24.07	157.06	393.70	510.13
nodes	11	38	182	240	139

Table 2
Branch-and-Cut Results on CPPMIN Type I Problems for $S = 4$

is that economic data tends to be highly skewed. So in this section, we generate two types of data to simulate economic data for our numerical experiments. The simulation is suggested by Sande [23].

Type III : Vertices are randomly generated on a plane with the axes being bivariate, and each following an exponential distribution independently. The edge weight is the integral part of 100 times the distance between vertices. This case can simulate economic data that tends to be highly skewed with a long

n	36-41	71-76
$k = \lfloor \frac{n}{S} \rfloor$	5	10
Heuristic Alg.		
Gap	4.65%	8.03%
Time	0.0100	0.0272
Cutting Plane	30	30
Solved exactly	0	0
Better Solution	25	35
Gap	2.10%	2.32%
Time	67.93	235.65
Cuts added	418	773
Pigeon	19	15
Flower	1	2
Outer Iter	35	40
Inner Iter	336	546
MINTO run	30	30
Gap	0.58%	2.04%
time	150.60	402.68
nodes	147	76

Table 3
Branch-and-Cut Results on CPPMIN Type I Problems for $S = 7$

tail. An example of the solution on this kind of data is shown in Figure 8. The performance of the algorithm on this kind of data is shown in Table 5 and 6.

Type IV : Vertices are randomly generated on a plane. One axis has an exponential distribution, the other axis has a $uniform \times exponential$ distribution. More precisely, let u be exponential, v be uniform between 0 and 1, then $x = u$ and $y = u \times v$. The edge weight is the integral part of 100 times the distance between vertices. In term of economic data, one can think of u as the size of measure of economic data and v as the fraction of the size which is spent on payrolls. Some businesses are very labor intensive and others are capital intensive with less labor. An example of the solution on this kind of data is shown in Figure 9. The results for this type of problems are shown in Tables 7 and 8.

From the results in Tables 5-8, we can see that these two types of problems

n	21-23	41-43
$k = \lfloor \frac{n}{S} \rfloor$	5	10
Heuristic Alg.		
Gap	13.50%	24.16%
Time	0.0062	0.0124
Cutting Plane	15	15
Solved exactly	0	0
Better Solution	15	15
Gap	4.74%	13.28%
Time	75.69	1657.34
Cuts added	393	1027
Pigeon	11	9
Flower	0	0
Outer Iter	25	34
Inner Iter	192	305
MINTO run	15	15
Solved exactly	15	0
Better Solution	0	6
Gap	0%	8.93%
time	40.83	476.70
nodes	80	169

Table 4
Branch-and-Cut Results on CPPMIN Type II problems for $S = 4$

are harder than the Type I problems, but none the less we can solve them to within 4% of optimality in about 3 minutes. Again it takes quite a long time to use a branch and bound algorithm to try to verify that the solutions we get really are optimal solutions.

We can also see that the problems of Type IV are a little bit easier than Type III. This is because the y variable is always less than x, thus making the x axis more important, and the problem closer to the easier one dimensional clustering problem.

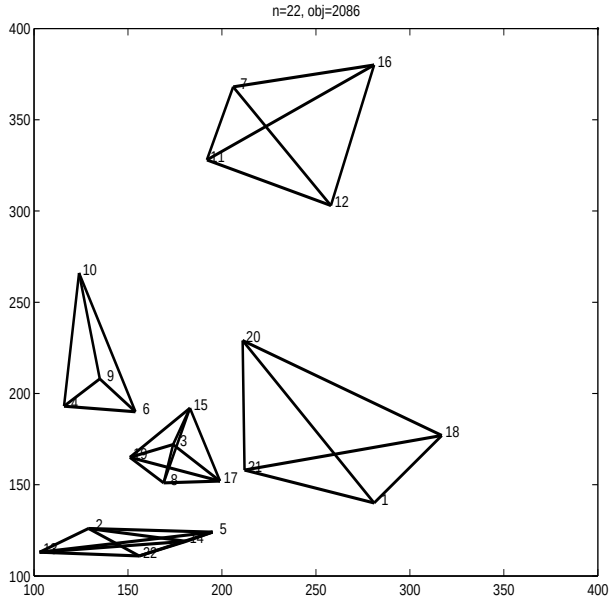

Fig. 8. Solution for a CPPMIN Problem of Type III: $n = 22, S = 4$

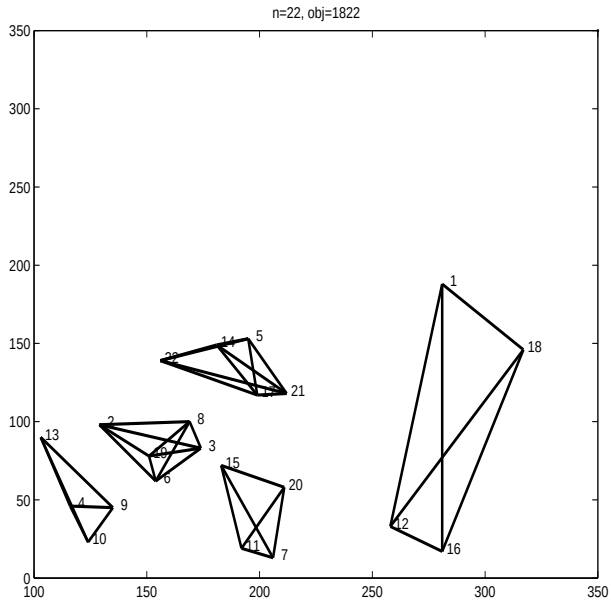

Fig. 9. Solution for a CPPMIN Problem of Type IV: $n = 22, S = 4$

5 Conclusions, Other Applications and Future Work

In summary, we have shown that the Clique Partition Problem with Minimum Size Constraints can be formulated as an IP problem (CPPMIN) and approximated nicely by a LP relaxation by adding cutting planes. We gave the mathematical formulation of the IP problem, discussed the polyhedral structure of the corresponding polytope, gave two kind of specialized cutting planes, and proved the necessary and sufficient condition for the pigeon con-

n	21-23	41-43	61-63	81-83	101-103
$k = \lfloor \frac{n}{S} \rfloor$	5	10	15	20	25
Heuristic Alg.					
Gap	3.12%	6.98%	9.49%	8.54%	9.01%
Time	0.0056	0.0113	0.0207	0.0313	0.0475
Cutting Plane	15	15	15	15	15
Solved exactly	1	0	0	0	0
Better Solution	6	14	15	15	15
Gap	2.14%	2.56%	2.03%	2.60%	3.46%
Time	9.45	20.20	49.94	104.74	181.08
Cuts added	123	184	274	369	438
Pigeon	13	10	8	8	8
Flower	1	4	5	8	11
Outer Iter	24	26	31	36	39
Inner Iter	195	298	434	560	646
MINTO run	14	15	15	15	15
Solved exactly	14	15	13	7	1
Better Solution	0	2	2	7	4
Gap	0%	0%	0.48%	1.76%	3.01%
time	4.39	28.52	155.85	361.73	485.88
nodes	25	81	253	247	125

Table 5
Branch-and-Cut Results on CPPMIN Type III Problems for $S = 4$

straints to be facet defining. Our computational results showed that with these special constraints, a cutting plane scheme leads consistently to high quality solutions in a reasonable amount of time. This proves the effectiveness of these special constraints.

In micro-aggregation problems, different kinds of objective function have also been proposed to characterize the within-group homogeneity. Domingo-Ferrer and Mateo-Sanz [5] introduced a few of them that are all based on the distance between each vertex and its cluster center, which would result in a non-linear objective function. Sande [22] suggested using the total weight of the minimum spanning tree within each clusters. We will address this particular form in Ji and Mitchell [13].

n	36-41	71-76
$k = \lfloor \frac{n}{S} \rfloor$	5	10
Heuristic Alg.		
Gap	3.87%	6.42%
Time	0.0129	0.0285
Cutting Plane	30	30
Solved exactly	0	0
Better Solution	16	29
Gap	2.05%	2.11%
Time	62.12	229.22
Cuts added	390	733
Pigeon	17	14
Flower	1	3
Outer Iter	33	40
Inner Iter	327	547
MINTO run	30	30
Solved exactly	24	0
Better Solution	0	1
Gap	0.59%	1.56%
time	194.75	465.86
nodes	136	71

Table 6
Branch-and-Cut Results on CPPMIN Type III Problems for $S = 7$

References

[1] F. Barahona, M. Grötschel, and A.R. Mahjoub. Facets of the bipartite subgraph polytope. *Mathematics of Operations Research*, 10:340–358, 1985.

[2] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[3] Sunil Chopra and M.R. Rao. The partition problem. *Mathematical Programming*, 59:87–115, 1993.

[4] Michele Conforti and M.R. Rao. The equipartition polytope *i* and *ii*. *Mathematical Programming*, 49:49–70, 1990.

[5] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented mi-

n	21-23	41-43	61-63	81-83	101-103
$k = \lfloor \frac{n}{S} \rfloor$	5	10	15	20	25
Heuristic Alg.					
Gap	6.44%	5.60%	7.43%	8.05%	8.60%
Time	0.0057	0.0115	0.0205	0.0318	0.0482
Cutting Plane	15	15	15	15	15
Solved exactly	3	1	0	0	0
Better Solution	10	12	15	15	15
Gap	3.02%	1.42%	2.41%	2.36%	3.30%
Time	5.12	20.27	43.80	97.17	150.30
Cuts added	93	189	263	355	422
Pigeon	11	12	8	7	7
Flower	1	4	5	8	9
Outer Iter	15	26	28	37	36
Inner Iter	118	313	398	563	593
MINTO run	12	14	15	15	15
Solved exactly	12	14	14	8	1
Better Solution	0	1	3	4	2
Gap	0%	0%	0.10%	1.24%	3.01%
time	6	16.97	152.04	347.16	476.36
nodes	66	38	260	257	177

Table 7
Branch-and-Cut Results on CPPMIN Type IV Problems for $S = 4$

croaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[6] M. Gröschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45:59–96, 1989.

[7] M. Gröschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47:367–387, 1990.

[8] Nili Guttmann-Beck and Shlomi Rubinstein. Approximation algorithms for minimum tree partition. *Discrete Applied Mathematics*, 87(1-3):117–137, 1998.

[9] Stephen Lee Hansen and Sumitra Mukherjee. A polynomial algorithm for optimal microaggregation. *IEEE transactions on Knowledge and Data Engineering*, 15(1):1043–1044, January 2003.

n	36-41	71-76
$k = \lfloor \frac{n}{S} \rfloor$	5	10
Heuristic Alg.		
Gap	4.05%	5.53%
Time	0.0130	0.0360
Cutting Plane	30	30
Solved exactly	1	1
Better Solution	30	29
Gap	1.79%	2.03%
Time	58.86	290.00
Cuts added	386	769
Pigeon	15	15
Flower	1	2
Outer Iter	29	39
Inner Iter	288	533
MINTO run	29	29
Solved exactly	29	3
Better Solution	0	0
Gap	0.56%	1.79%
time	157.59	478.51
nodes	136	38

Table 8
Branch-and-Cut Results on CPPMIN Type IV Problems for $S = 7$

[10] Refael Hassin and Shlomi Rubinstein. Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21:133–137, 1997.

[11] Xiaoyun Ji and John E. Mitchell. Finding optimal realignments in sports leagues using a branch-and-cut-and-price approach. *International Journal of Operational Research*, 1, April 2005.

[12] Xiaoyun Ji and John E. Mitchell. Graph partition problems with minimum size constraints using branch-and-price. Technical report, Rensselaer Polytechnic Institute, 2005. forthcoming.

[13] Xiaoyun Ji and John E. Mitchell. Minimum weight forest problem. Technical report, Rensselaer Polytechnic Institute, 2005. forthcoming.

[14] A. Lisser and F. Rendl. Graph partitioning using linear and semidefinite programming. *Mathematical Programming*, 95(1):91–101, 2003.

[15] Anuj. Mehrotra and Michael. A. Trick. Cliques and clustering: a combinatorial approach. *Operations Research Letters*, 22:1–12, 1998.

[16] J. E. Mitchell. Branch-and-cut for the k-way equipartition problem. Technical report, Rensselaer Polytechnic Institute, 2001.

[17] J. E. Mitchell. Realignment in national football league: Did they do it right. *Naval Research Logistics*, 50(7):683–701, 2003.

[18] J. E. Mitchell and B. Borchers. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In H. L. Frenk *et al.*, editor, *High Performance Optimization*, pages 349–366. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[19] George L. Nemhauser, Martin W. P. Savelsbergh, and Gabriele C. Sigismondi. MINTO, a mixed INTeger optimizer. *Operations Research Letters*, 15:47–58, 1994.

[20] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

[21] Svatopluk Poljak and Zsolt Tuza. Maximum cuts and largest bipartite subgraphs. In William Cook, László Lovász, and Paul Seymour, editors, *Combinatorial Optimization: Papers from the DIMACS Special Year*, volume 20 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. AMS, 1995.

[22] Gordon Sande. Exact and approximate methods for data directed microaggregation in one or more dimensions. *International Journal of Uncertainty,Fuzziness and Knowledge-Base Systems*, 10(5), 2002.

[23] Gordon Sande. Personal communication, 2003.