Telecommunication Node Clustering with Node Compatibility and Network Survivability Requirements

Kyungchul Park

Telecommunication Network Research Lab., Korea Telecom, Wha-am-dong, Yusong-gu, Taejon, 305-348, Korea

Kyungsik Lee

Electronics and Telecommunication Research Institute, 161 Kajong-dong, Yusong-gu, Taejon, 305-350, Korea

Sungsoo Park

Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Gusong-dong, Yusong-gu, Taejon, 305-701, Korea

Heesang Lee

Department of Industrial Engineering, Hankuk University of Foreign Studies, 89 Mohyun-ri, Wangsan-myun Yongin-gun, Kyunggi-do 449-791, Korea

December 1995
Revised February 1997, June 1998, September 1999

* Corresponding Author
Telecommunication Node Clustering with Node Compatibility and Network Survivability Requirements

ABSTRACT

We consider the node clustering problem which arises in designing a survivable two level telecommunication network. The problem simultaneously determines an optimal partitioning of the whole network into clusters (local networks) and hub locations in each cluster. Inter-cluster traffic minimization is chosen as the clustering criterion to improve the service quality. Various constraints on the clustering are considered which reflect both the physical structures of local networks such as the connectivity requirement and the node compatibility relations such as community of interest or policy. Additional constraints may be imposed on the hub selection to ensure network survivability. We propose an integer programming formulation of the problem by decomposing the entire problem into a master problem and a number of column generation problems. The master problem is solved by column generation and the column generation problems by branch-and-cut. We develop and use strong cutting-planes for the cluster generation subproblems. Computational results using real data are reported.

Key Words: Network Design, Clustering, Column Generation, Branch-and-Cut
1. Introduction

Large-scale public telecommunication networks usually have a hierarchical structure for economic construction and operation. There can be many levels in the hierarchy. However, in this paper, we concentrate on the two-level hierarchy since it is the most popular structure and can be served as a building block for more general cases. In a two-level network, the higher level is called the backbone network. It is composed of hub nodes, each of which serves as a traffic concentration point. The lower level consists of local networks, each of which is composed of one or two hub nodes and a set of user nodes. Telecommunication traffic between two user nodes in different local networks is transported via hub nodes, that is, through the backbone network. In this study, we consider mesh topology for the backbone network. Ring topologies and the hubbed networks are considered for the local networks. These types of network topologies can be found in many public telecommunication networks using SDH/SONET (Synchronous Digital Hierarchy/Synchronous Optical NETwork) transport technologies, see Wu (1992) for more details. An example of the two-level network considered in this paper is illustrated in Figure 1. There are four local networks in the figure. One has two ring topology, another has both the ring topology and the hubbed network structure, and the others have the hubbed network structures. The figure also shows the types of the facilities used, see Wu (1992) for more details.

>> Insert Figure 1 Here <<
In this paper we consider the node clustering problem for a two-level network that determines simultaneously (1) which user nodes to connect together to form a local network, and (2) which of the hub nodes to connect to each local network. As an input to the problem, we are given a set of user nodes, traffic between each pair of user nodes, and candidate hub nodes which are chosen among the given user nodes. We are also given a physical network whose nodes and links represent the given user nodes and available conduits, respectively.

Among possible clustering criteria, we chose to minimize the inter-cluster traffic. The reasons are twofold. Firstly, the local network is faster and more reliable than the backbone network. It is thus desirable to have the largest possible share of the traffic restricted to the local network. Secondly, the inter-cluster traffic necessarily transits through both local and backbone networks in a two-level network. So, the large amount of the inter-cluster traffic needs much investment in both local and backbone networks. Therefore, the proposed criterion of inter-cluster traffic minimization may also result in a smaller total network cost, see Flanagan (1990).

The clustering must meet three types of constraints. The first one, called connectivity constraint, concerns the target local network. Its purpose is to ensure survivability of the local network after any single link failure. This is very important when designing a fiber-optic based telecommunication network, see Wu (1992). The connectivity constraint is the connectivity requirement imposed on each cluster. It is defined on the given physical network representing the conduits available. When the target local network topology and survivability requirements only need the selected cluster to induce
a connected subgraph in the physical network, it is sufficient to impose the simple connectivity constraint. However, when the local network topology is a ring and/or the link survivability requirement is important, the 2-edge-connectivity (or 2-node-connectivity) constraint should be imposed, which already implies the simple connectivity.

The second constraint for the node clustering problem is called the *compatibility constraint*, and takes care of compatibility relations between user nodes. Two user nodes are defined to be compatible if they can be included in the same cluster. Compatibility between user nodes can be determined by several factors, for example, the distance between them, geographic restrictions, community of interest, and policy. Compatibility relations can be represented by a graph that is called the *compatibility graph*. The nodes of the graph represent the user nodes and an edge between two nodes exists if and only if they are compatible. So a feasible cluster should induce a clique in the compatibility graph.

The final constraint is the *hub survivability constraint*. Hub survivability means the ability of the network to restore services upon a single hub failure. Since inter-cluster traffic should be routed through hubs in the cluster, all of it is lost if only one hub exists and it fails. So when the hub survivability is important, we use the double hub constraint, which requires two hubs are located in each cluster. Otherwise, we need the single hub constraint, which requires one hub in each cluster.

The clustering problem, as in graph theory and mathematical programming, consists of finding a node partition into clusters, so as to maximize (minimize) the total cost of the edges inside the clusters. In the literature, many variations of the clustering problem such
as min-cut clustering, max-cut clustering, mixed-cut clustering, and equipartition problems have been studied, see Conforti et al. (1990a, 1990b) and Johnson et al. (1993).

Recently, the hierarchical network design problem (HNDP) has received an increasing attention. In the literature on (HNDP) one usually assumes a special type of backbone and local network structures, for example, star-star network (Gavish and Pirkul 1986, Mirzaian 1985), tree-star network (Kim and Tcha 1992, Lee et al. 1996b), mesh-star network (Chung et al. 1992, Helme and Magnanti 1989), and ring-star network (Lee et al. 1993). Dual-based heuristics or Lagrangian relaxations were proposed to solve the problems. The main difference between (HNDP) and the node clustering problem considered in this paper is that the physical connection between the nodes is not determined in the latter problem. Rather, various constraints that ensure the desired topology, such as connectivity and compatibility constraints, are considered. From the application viewpoint, the (HNDP) usually deals with the design of a special-purpose network over a relatively small geographic area, for example a CATV network. However, our node clustering problem addresses the design of a public network infrastructure, and incorporates more generalized constraints to allow the network planner a higher degree of freedom in choosing the topology based on local needs.

We develop a mixed integer program (MIP) model for our node clustering problem. The (MIP) problem is decomposed into a master problem and a number of column generation problems. We solve the master problem by a column generation algorithm and the column generation problems by a polyhedral cutting plane algorithm. A similar scheme was applied to the min-cut clustering problem, see Johnson et al. (1993).
Laguna (1994) has studied a clustering problem for the design of SONET rings. However, the problem considered in this paper assumes a different network structure (the two-level network structure) and considers more complicated constraints (node compatibility and survivability constraints).

The rest of this paper is organized as follows. In section 2, we present the formulation of the problem. The preprocessing and polyhedral results on the cluster generation problem are given in section 3. In section 4, we give a description of the overall algorithm. In section 5, computational results are given and finally, in section 6, concluding remarks are provided.

2. Formulation

In this section, we present the formulation of the node clustering problem and the column generation procedure to solve its LP relaxation. For modeling purposes, we consider two types of graphs, namely, the physical graph $G = (V, E)$ and the compatibility graph $H = (V, F)$. The set of nodes $V$ in both graphs represents the user nodes. It is divided into two subsets. The first one consists of cluster nodes, which are the real concern in the clustering process. The other is the set of transit nodes, which are introduced to represent the topological structure of the conduit network. The transit nodes represent the junction points in the conduit network, but no facilities are located at them. They are needed only to model the conduit network as a graph, but they are not
direct objects of the clustering since no traffic demands are associated with them. Therefore, a transit node may or may not be included in a feasible clustering of the nodes.

Let $C$ be the set of cluster nodes and $T$ be the set of transit nodes. Also let $D \subseteq C$ be the set of candidate hub nodes. Then $G = (V, E)$ denotes the physical graph, where $V = C \cup T$ and $e = (i, j) \in E$ if and only if there exists a conduit connecting nodes $i$ and $j$. The compatibility graph $H = (V, F)$ can be defined similarly, where $e = (i, j) \in F$ if and only if nodes $i$ and $j$ are compatible, that is, they can be included in the same cluster. For an edge $e = (i, j) \in F$, $t_{ij}$ is the amount of traffic between nodes $i$ and $j$. Note that $t_{ij}$, for $(i, j) \notin F$, can also be defined, but it is not needed here. Also note that $t_{ij} = 0$, if at least one of the nodes $i$ and $j$ is a transit node. A positive integer $q$ is given, which represents the upper bound on the number of clusters being constructed. $L$ and $U$ denote the lower and upper bounds on the number of cluster nodes to be included in each cluster, respectively. Let $Q$ be the set of feasible clusters. Furthermore, let $w_k$ be the amount of intra-cluster traffic for the cluster $k$, for all $k \in Q$. Note that $w_k = \sum_{e \in F(k)} t_e$, where $F(k) \subset F$ is the set of edges in the subgraph of $(V, F)$ induced by the cluster $k \in Q$.

A feasible cluster $k \in Q$ is conveniently described by a binary vector $\delta_k$, with component $\delta_k^v = 1$ if node $v$ is contained in the cluster, and 0 otherwise. Using the above notation, the node clustering problem can be formulated as follows:

$$(\text{CSP}) \quad \text{max} \sum_{k \in Q} w_k z_k \quad (2.1)$$
\[
\begin{align*}
\text{s.t.} \quad & \sum_{k \in Q} \delta^v_k z_k = 1 \quad \text{for all } v \in C \quad (2.2) \\
& \sum_{k \in Q} \delta^v_k z_k \leq 1 \quad \text{for all } v \in T \quad (2.3) \\
& \sum_{k \in Q} z_k \leq q \quad (2.4) \\
& z_k \in \{0,1\} \text{ for all } k \in Q,
\end{align*}
\]

In the above formulation, (CSP) stands for the cluster selection problem. Decision variable \( z_k \) equals 1 if and only if the cluster \( k \) is chosen to be included in the clustering, for all \( k \in Q \). The objective function (2.1) maximizes intra-cluster traffic, which is equivalent to minimization of the inter-cluster traffic. Constraints (2.2) ensure that each cluster node should be contained in exactly one cluster. Constraints (2.3) imply that each transit node can be contained in at most one cluster. Finally, the constraint (2.4) is the upper bound constraint on the number of clusters selected.

Note that (CSP) may have exponentially many variables. However, its LP relaxation can be solved efficiently by using the column generation technique of Gilmore and Gomory (1961).

Let (CSPL) be the LP relaxation of (CSP). As it is customary in column generation techniques, we assume that a subset \( Q' \subset Q \) of feasible clusters is given. Replacing \( Q \) by \( Q' \) in (CSPL) yields the restricted linear program (CSPL’), whose solutions are suboptimal to (CSPL). Let \( \alpha_v \) be the dual variable associated with each constraint (2.2) and (2.3), and let \( \mu \) be the scalar dual variable associated with (2.4). The
constraints in the dual of (CSPL’) are

\[ \sum_{v \in C \cup T} \delta^y_k \alpha_v + \mu \geq w_k, \text{ for all } k \in Q', \]

\[ \alpha_v \geq 0, \text{ for all } v \in T. \]

Let \((\overline{\alpha}, \overline{\mu})\) be an optimal solution to the dual of (CSPL’). Then, it is also optimal to the dual of (CSPL) if and only if

\[ \sum_{v \in C \cup T} \delta^y_k \overline{\alpha}_v + \overline{\mu} \geq w_k, \text{ for all } k \in Q \setminus Q'. \]

Recall that \(w_k = \sum_{e \in F(k)} t_e\). Using further \(\delta^y_k = 1\) if \(v \in V(k)\) and 0 otherwise, we may write the optimality condition for (CSPL):

\[ \max \{ \sum_{k \in Q} t_e - \sum_{v \in V(k)} \overline{\alpha}_v \leq \overline{\mu}, \] (2.5)

where \(F(k)\) and \(V(k)\) are the sets of edges and nodes in the subgraph of \(H\) induced by cluster \(k\), respectively.

Using (2.5), we can derive the formulation of the column generation problem, which is called the cluster generation problem (CGP). The formulation of (CGP) depends on the connectivity and hub survivability constraints chosen. In the following, for simplicity of presentation, we assume that the double hub and 2-edge-connectivity constraints are chosen.
Cluster Generation Problem

Now we consider the cluster (column) generation problem. The set of feasible clusters \( Q \) can be decomposed into disjoint subsets, where each subset is composed of clusters which have the same hub nodes. Therefore, we can solve the cluster generation problem for each subset by fixing a pair of compatible hub nodes. By using this approach, we can add several columns simultaneously. Furthermore, the resulting problems usually have much smaller number of variables and constraints than those without fixing the hubs. Moreover, this approach enables the network planners only to consider the hub pairs which are likely to be used in the clustering by specifying possible hub pairs as an input.

Let us define \( R = \{(s,t) \mid \exists D, (s,t) \in F\} \). Note that \( R \) is the set of compatible hub pairs. In practice, \( R \) can be given as an input by network planners. Otherwise, \( R \) can be generated by considering all compatible hub pairs. Let \( Q_{st} \subseteq Q \) be the set of clusters which have the nodes \( s \) and \( t \) as two hub nodes, where \( (s,t) \in R \). Let \( \text{CGP}(s,t) \) be the cluster generation problem restricted to the clusters in \( Q_{st} \). Let \( C_{st} \) be the set of cluster nodes that are compatible with \( s \) and \( t \), that is,

\[
C_{st} = \{i \in C \mid (s,i),(t,i) \in F\} \cup \{s,t\}.
\]

\( T_{st} \) can be defined similarly. Let \( V_{st} = C_{st} \cup T_{st} \) and \( E_{st}(F_{st}) \) be the subset of edges in the graph \( G = (V,E) \) \( (H = (V,F)) \) with their two end nodes in \( V_{st} \). Therefore, we
define a subgraph \( G_{st} = (V_{st}, E_{st}) \) \((H_{st} = (V_{st}, F_{st})\)) of \( G(H) \) which is induced by the node set \( V_{st} \). Without loss of generality, we can assume \( E_{st} \subseteq F_{st} \).

The following is a 0-1 integer programming formulation of CGP(s, t).

\[
\text{CGP}(s, t) \quad \max \sum_{e \in F_{st}} t_{e} y_{e} - \sum_{v \in V_{st}} \alpha_{v} x_{v} \\
\text{s.t.} \quad L \leq \sum_{v \in C_{st}} x_{v} \leq U \quad (2.6) \\
x_{i} + x_{j} \leq 1 \quad \text{for all } (i, j) \notin F_{st} \quad (2.7) \\
y_{ij} \leq x_{i}, y_{ij} \leq x_{j}, x_{i} + x_{j} - y_{ij} \leq 1 \quad \text{for all } (i, j) \in F_{st} \quad (2.8) \\
2 y_{ij} \leq \sum_{e \in \delta(S)} y_{e} \quad \text{for all } (i, j) \in F_{st} \text{ and } i \in S \subseteq V_{st} \setminus \{j\} \quad (2.9) \\
\sum_{e \in F_{st}} x_{e} = x_{i} = 1 \quad (2.10) \\
x_{v} \in \{0,1\} \text{ for all } v \in V_{st}, \quad y_{e} \in \{0,1\} \text{ for all } e \in F_{st},
\]

where \( \delta(S) = \{e = (u, v) \in E_{st} \mid u \in S, v \in V \setminus S\} \).

In the above formulation, \( x_{v} \) equals 1 if and only if \( v \) is chosen to be contained in the cluster, for all \( v \in V_{st} \). Similarly, \( y_{e} \) has the value 1 if and only if the edge \( e \) is chosen to be contained in the cluster, for all \( e \in F_{st} \). Constraints (2.7) represent the compatibility constraints imposed on the pairs of nodes. Constraints (2.8) together with constraints (2.7) guarantee that the selected nodes and edges form a clique in \( H \). Constraints (2.9) are the 2-edge-connectivity constraints defined on the physical graph \( G \).
If the optimal value of CGP(s, t) is greater than $\bar{\mu}$, then the cluster corresponding to the optimal solution can be added to (CSP), and otherwise, no cluster with $s$ and $t$ as hub nodes is generated.

Figure 2 shows an example of the node clustering problem with the 2-edge-connectivity and double hub constraints. The dashed nodes are candidate hub nodes and nodes 23 and 24 are transit nodes. Figure 3 shows the complement of the compatibility graph. The corresponding compatibility graph consists of 240 edges, which is very dense, so we give the complement of it for ease of presentation.

>> Insert Figures 2 and 3 Here <<

We mention that CGP(s, t) is a very hard integer programming problem, since the feasibility checking problem associated with it is already NP-complete (by transforming MINIMUM 2-EDGE CONNECTED SUBGRAPH problem into it, Garey and Johnson 1979 pp. 198).

3. Analysis of the Cluster Generation Problem

Since CGP(s, t) is a very hard problem, we investigate in this section the polyhedral structure of the problem and derive useful results that will be used in devising a branch-and-cut algorithm for the problem.
3.1. Preprocessing and Tightening the Connectivity Constraints

When constructing CGP(s, t) for a specified pair of hub nodes s and t, we can eliminate some variables from the problem. First, note that there must exist at least two edge-disjoint paths between s and t in $G_{st}$ for a cluster to be feasible. For a node $i \in V_{st} \setminus \{s, t\}$, we further define a subgraph $G_{st}(i)$ of $G_{st}$, which is induced by node i and the nodes that are compatible with it. Then, if there do not exist two edge-disjoint paths between i and s (or i and t) in $G_{st}(i)$, node i cannot be included in a feasible cluster since otherwise the 2-edge-connectivity constraint will be violated. Note that there exist two edge-disjoint paths between two nodes in $G_{st}(i)$ if and only if the maximal flow value between the two nodes in $G_{st}(i)$ with the capacity of each edge set to 1 is at least 2. This can be easily checked by using a maximal flow algorithm.

The connectivity constraint (2.9) corresponding to hub nodes s and t can be rewritten as follows:

$$2 \leq \sum_{e \in \delta(S)} y_e , \quad \text{for all } S \subseteq V_{st} \setminus \{t\} \text{ with } s \in S . \quad (3.1)$$

Also, note that the constraint (2.9) for a $(s, i)$-cut, where $i \notin \{s,t\}$, can be strengthened as follows:

$$2x_i \leq \sum_{e \in \delta(S)} y_e , \quad \text{for all } S \subseteq V_{st} \setminus \{i\} \text{ with } s \in S . \quad (3.2)$$

(similar result holds for a $(t, i)$-cut).
It can be proved that constraints (3.1) and (3.2) can replace the constraint (2.9) by noting that inequalities $y_{ij} \leq x_i$ and $y_{ij} \leq x_j$ hold and a $(i, j)$-cut should be either a $(i, s)$-cut or $(j, s)$-cut. That is, the constraints (2.9) corresponding to pairs of two non-hub nodes can be removed, which is summarized in the following proposition.

**Proposition 1.** Constraints (3.1) and (3.2) imply the constraint (2.9).

### 3.2. Development of Cutting-planes

To further strengthen the IP formulation of CGP($s, t$), several classes of cutting-planes are derived. They are obtained by considering submodels of the problem. Note that since the feasibility problem associated with CGP($s, t$) is NP-complete, even the dimension of the whole polytope might not be determined easily.

In the following, we present some classes of cutting-planes to CGP($s, t$) that can be used to devise a branch-and-cut algorithm for the problem. For simplicity of presentation, we suppress the subscripts from the notation such as $V_{st}$ and simply write it as $V$.

**Cutting-planes from the extended node packing polytope**

If we consider only the constraints (2.7) and (2.8), a feasible solution induces a clique subgraph of the graph $H$. Note that the clique subgraph corresponds to a set of
compatible nodes. Let \( P(H) \) be the convex hull of the integer solutions satisfying them. Then any valid inequality of the node packing polytope on the graph \( \overline{H} \) (the complement graph of \( H \)) is also valid for \( P(H) \). However, we can show many of them do not define facets of \( P(H) \), although they are facet-defining for the node packing polytope on the graph \( \overline{H} \), see Park et al. (1994).

There exists a very simple method to derive a facet-defining inequality for \( P(H) \) from that of the node packing polytope, which we call rooting. Let us denote \( N_r \) be the set of neighbors of a node \( r \) in the graph \( H \), that is,

\[
N_r = \{ v \in V \mid (r, v) \in F \}.
\]

In addition, let \( NP_r \) be the node packing polytope defined on the complement graph of the subgraph of \( H \) induced by \( N_r \). Note that \( NP_r \) is defined using only \( x \) variables. Then we can obtain the following proposition.

**Proposition 2.** Let \( \pi^T x \leq \pi_0 \) be a nontrivial facet-defining inequality for the polytope \( NP_r \), then the inequality

\[
\sum_{i \in N_r} \pi_i y_{ir} \leq \pi_0 x_r
\]

defines a facet of \( P(H) \).

**Proof.** See Appendix.
As an example of the inequality (3.3), let \( S_r \) be a maximal node packing (clique) in the subgraph of \( H \) (complement of \( H \)) induced by \( N_r \). Then the following packing-star inequality defines a facet of \( P(H) \).

\[
\sum_{i \in S_r} y_{ri} \leq x_r. \tag{3.4}
\]

Further classes of facet-defining inequalities for \( P(H) \) can be obtained by noting that \( P(H) \) is a face of the boolean quadratic polytope, which was studied by Padberg (1989) and others. The classes of (modified) cut and clique inequalities (Padberg 1989) can be shown to define a facet of \( P(H) \) under certain mild conditions, see Park et al. (1994).

**Cutting-planes from the knapsack quadratic polytope**

If we consider the constraints (2.6) and (2.8), we obtain the cardinality-constrained quadratic knapsack polytope. Let \( P_U \) be the convex hull of the integer solutions satisfying the upper bound part of (2.6) and all of (2.8). Then \( P_U \) is a special case of the knapsack quadratic polytope (KQP) studied by Johnson et al. (1993). They provided two classes of strong valid inequalities for KQP and among them, the tree inequalities are very useful for the current problem.

Let \( S \subseteq V \) with \( |S| = U + 1 \). Let \( T \) be a spanning tree of the subgraph of \( H \) induced by \( S \), which is assumed to be connected. In addition, let \( d_i \) be the degree of the node \( i \) in \( T \). Then the tree inequality is defined as follows.

\[
\sum_{e \in T} y_e - \sum_{i \in S} (d_i - 1)x_i \leq 0 \tag{3.5}
\]
Johnson et al. (1993) proved that the inequality (3.5) defines a facet of the subpolytope obtained by restricting the variables \( x_i = y_e = 0 \) for all \( i \in V \setminus S \) and \( e \in F \setminus F(S) \), where \( F(S) \) is the set of edges in \( F \) with both end nodes in \( S \). Moreover, they proved the lifting problem is NP-hard, in general. However, if the cardinality constraint is present, the inequality (3.5) defines a facet of \( P_U \) under a mild condition.

**Proposition 3.** The tree inequality (3.5) defines a facet of \( P_U \) if \( T \) is not a star. When \( T \) is a star, it defines a facet of \( P_U \) if and only if \( U = |V| - 1 \).

**Proof.** See Park et al. (1996b).

When \( T \) is a star and \( U \leq |V| - 1 \), the corresponding inequality can be strengthened as follows.

\[
\sum_{e \in \delta(r)} y_e - (U - 1)x_r \leq 0,
\]

where \( r \in V \) and \( \delta(r) = \{(r, i) \in F_{st} | i \in C_{st}\} \). The above inequality is called the *star inequality* and it defines a facet of \( P_U \) (see Park et al. 1996b).

**4. Overview of the Algorithm**

In this section, we present the algorithm to solve the clustering problem. For simplicity of presentation, we assume that the double hub constraint is chosen. The
algorithm is based on the column generation technique to solve mixed integer programming problems. After initializing the master problem (CSP) with a few columns, we solve the column generation problems CGP(s, t), for all \((s,t) \in R\), and then, add newly generated columns, if any. After the optimality of the LP relaxation of the master problem (CSPL) is attained, and if the current solution to (CSPL) is not integral, we go into the branch-and-bound phase with the current formulation of (CSPL). If the integer solution obtained in the branch-and-bound phase can be shown to be optimal, then we are done. Otherwise, the branch-and-price (see Barnhart et al. 1994) is initiated to prove the optimality of the solution or to further improve the solution. For the applications of the branch-and-price approach, see Park et al. (1996a) and Lee et al. (1996a).

The cluster generation problem CGP(s, t) is solved by a branch-and-cut approach (Pardberg and Rinaldi 1991). Since the number of the connectivity constraints (3.1) and (3.2) is too large, they are generated as cutting-planes. In the following, we explain the details of the algorithm.

To start the column generation approach, we should have an initial feasible solution. In practice, this can be provided by the network planner as an initial clustering solution. When this is not possible, we initialize (CSP) with an identity matrix, that is, a clustering having exactly one cluster node (not transit node) in each cluster. Of course, such a clustering may not be feasible and so, each column is initialized with an objective coefficient of \(-M\), where \(M\) is a sufficiently large positive number. In this case, the
instance is infeasible if the value of \( (\text{CSPL}) \) is less than 0 at the time the column generation is completed.

The initial formulation of CGP\((s,t)\) contains the constraints (2.6), (2.7), (2.8), (2.10) together with the following inequalities:

\[
\sum_{e \in \partial(r)} y_e - (U-1)x_r \leq 0 \quad \text{for all} \quad r \in V_{st} \setminus \{s,t\} \tag{4.1}
\]

The constraints (4.1) are the star inequalities presented in the previous section and they are added to strengthen the initial formulation of CGP\((s,t)\). The connectivity constraints (3.1) and (3.2) are used as cutting-planes for CGP\((s,t)\). Other classes of cutting-planes may be used, but a preliminary test shows that the above choice is sufficiently good.

The separation problem for the connectivity constraints (3.1) and (3.2) can be solved easily by a maximal flow algorithm. We use the shortest augmenting path algorithm (Ahuja et al. 1993) to find the most violated connectivity constraints between two hub nodes and between a hub and a non-hub node. The separation problem is solved only between cluster nodes. Note that the constraint (3.2), where the non-hub node is a transit node, can be ignored.

When the branch-and-price approach is used, the main difficulty arises in the column generation after some subset of the variables is fixed at 0. To prevent the generation of columns that were set to 0, a careful branching rule should be used. For the current problem, we can use the following branching rule, which is similar to the branching rule that is used frequently for solving IP with SOS (special ordered set) constraints. For simplicity of presentation, let us assume that there exist no transit nodes. Let us define

\[
Z_{vh} = 1, \text{ if node } v \text{ is included in a cluster with hubs } h \text{ in a clustering}
\]
0, otherwise,

where \( h \) is one of the possible hub pairs when the double hub constraint is used and it is a hub candidate when the single hub constraint is used. Variables \( Z_{vh} \)'s are considered implicitly for branching purposes only. Note that \( \sum_h Z_{vh} = 1 \) should hold if node \( v \) is a cluster node.

Our branching rule is based on the (newly defined) variables \( Z_{vh} \). For a given optimal solution \( \bar{z} \) to (CSPL) which has fractional components, we first calculate values of \( Z_{vh} \), \( \sum_{k \in K(v,h)} \bar{z}_k \), for all \( v \in V \) and all hubs \( h \), where \( K(v,h) \) is the set of clusters having hubs \( h \) and node \( v \). It can be easily shown that \( \bar{z} \) is integral if and only if \( Z_{vh} \)'s are integral. Then, we select a variable whose value is as close to 0.5 as any other variables whose values are fractional. Let \( Z_{vh} \) be the selected variable on which we perform branching. We make two new nodes in the enumeration tree, one with \( Z_{vh} = 0 \) and the other with \( Z_{vh} = 1 \). Setting \( Z_{vh} \) to 0 is equivalent to setting

\[
\sum_{k \in K(v,h)} z_k = 0.
\]

In this case, we set \( x_v = 0 \) when solving CGP(\( h \)) and set \( z_k = 0 \) for all \( k \in K(v,h) \) when solving (CSP). On the other hand, setting \( Z_{vh} \) to 1 is equivalent to setting

\[
\sum_{k \in \overline{K}(v,h)} z_k = 0,
\]

where \( \overline{K}(v,h) \) is the set of clusters having hubs \( h \) and not containing \( v \). In this case, we set \( x_v = 1 \) when solving CGP(\( h \)) and set \( z_k = 0 \) for all \( k \in \overline{K}(v,h) \) when solving (CSP).
We used CPLEX 3.0 callable library routines as an LP solver. When the branch-and-price phase is needed to solve (CSP), we use the best bound rule for node selection. We also use the best bound rule for node selection in the branch-and-cut phase to solve the cluster generation problems.

5. Computational Results

5.1. Problem Characteristics

We have tested the proposed algorithm on the testbed problems provided by Korea Telecom. Table 1 shows the summary of the characteristics of the problems. The column CON refers to the connectivity constraints used. The column HUB refers to the hub survivability constraints used, where D stands for the double hub and S stands for the single hub constraint, respectively. The column #Hub refers to the number of the hubs (hub pairs) considered in the case of the single hub constraint (double hub constraint, respectively) and so, it is equal to the number of the cluster generation problems.

>> Insert Table 1 Here <<

The set of test problems can be divided into four classes based on the underlying physical graphs used. The physical graph of the first class is the national backbone network. Those of the second and third are the networks in the metropolitan areas.
These problems reflect the typical characteristics of the problem solved in practice. The physical graph of the fourth class is made from the existing network by adding nodes and edges to test the performance of the algorithm on large problem instances.

Several test problems are provided in each class by varying the optional constraints and parameters according to alternative network construction scenarios. We have used traffic data which are projected over the next 5 years. When possible, the compatibility graphs are provided by the planners. In other cases, they are generated by specifying the distance limits. In these cases, the shortest path between each pair of nodes is found and if its length is less than or equal to the distance limit, they are declared to be compatible.

5.2. Computational Results

The test problems are solved on a HP9000/715 (50MHz) workstation. Table 2 summarizes the computational results. The column #LP shows the total number of LP's solved in (CSP). The column #CLUSTER refers to the number of clusters generated excluding initial columns and Gap refers to the percentage gap between the optimal LP solution of (CSPL) and optimal integer solution of (CSP). Finally, the column TIME refers to the CPU time needed to solve the problem. In each column generation phase, several columns are added simultaneously by solving the cluster generation problems for all hub pairs (hub candidates) when the double (single) hub constraint is chosen.

>> Insert Table 2 Here <<
Only the problems SL1 and SL3 require the branch-and-price phase, but only two branch-and-price nodes are needed in each case. In other problems, the final solutions of (CSPL) are integral and so, they are optimal to (CSP). Though the time is not a critical factor in a usual network design process, the CPU times do not exceed 5 minutes except a few problem instances. Hence planners can use the algorithm iteratively for evaluating various design alternatives or finding suitable design parameters. The CPU time increases as #Hub increases. Problems KN and SE represent typical networks that occur in the design of networks in Korea Telecom. The results show that the proposed algorithm performs very well on the problems of those ranges. Moreover, it also produces very good results on the large problem instances (SL). Hence it will be possible to use the algorithm for the future expanded networks.

Figure 4 shows the clustering result for the problem KN11D2 where the dark nodes represent the hub nodes. We note that node 23 and 24 are transit nodes which are not included in any cluster.

>> Insert Figure 4 Here <<

Table 3 shows typical characteristics observed when we solve the cluster generation problems for KN11 networks. In the table, #Var and #Row refer to the numbers of variables and constraints in the initial formulation of the corresponding problem, respectively. We give an explanation of the NO column using an example. In (1) of table 3, NO = 3 means that the thirdly generated column when solving (CSP) is from
Subproblem 1. Subproblem 1 refers to the cluster generation problem corresponding to a candidate hub node of the problem KN11S1. The corresponding row shows the results of subproblem1 at the time that column was generated. The columns #CUT and #B&C refer to the number of cuts added and the number of branch-and-cut nodes generated, respectively. Finally, Gap refers to the percentage gap between the optimal value of the final LP and the integer optimum. The gaps tend to grow as the column generation progresses. Similar phenomena can be found in Johnson et al. (1993). However, even when the gap is very large, only a few branch-and-cut nodes are needed to be generated to find the integer optimum.

>> Insert Table 3 Here <<

6. Concluding Remarks

This paper presented a model and an algorithm to solve the node clustering problem for the design of two level survivable telecommunication networks. The proposed model can reflect various constraints. A column generation algorithm is proposed and tested on the set of real problems.

After the node clustering is completed, there arise the design problems of backbone and local networks. The backbone network design problem is to find cost-effective working and restoration routes between hub nodes, see Coan et al. (1991), Grover (1987), and Lee et al. (1998). When the network is sufficiently small (for example, when designing an intra-city network), the clustering and local network design problems can be
incorporated as one problem. In this case, our scheme of the algorithm can also be applied and the column generation problem finds a cost-effective configuration of the local network.

The node clustering algorithm presented in this paper will be incorporated with other design modules into an integrated network design tool.

Appendix: Proof of Proposition 2

We only sketch the proof. It is well-known that the node packing polytope is full dimensional, so is $NP_r$. $P(H)$ also can be shown to be full dimensional by noting that the zero vector together with the binary vectors corresponding to each single node and each single edge with its two end nodes constitute affinely independent feasible solutions. Moreover, we can assume $\pi_i \geq 0$ for all $i \in N_r$ and so, $\pi_0 > 0$. Since the inequality $\pi^T x \leq \pi_0$ defines a facet of $NP_r$ and $\pi_0 > 0$, there exist $|N_r|$ linearly independent points satisfying the inequality at equality. Let them be $\bar{x}^k$, $k = 1, \ldots, |N_r|$. For each $k$, let us define

$$D_k = \{r\} \cup \{j \in N_r \mid \bar{x}^k_j = 1\}.$$

Consider the following solutions $(x^k, y^k)$, $k = 1, \ldots, |N_r|$,

$$x^k_i = 1 \text{ if } i \in D_k,$$

$$0 \text{ otherwise,}$$

$$y^k_e = 1 \text{ if } e \in F(D_k),$$

25
0 otherwise,

where \( F(D_k) = \{(i, j) \in F \mid i, j \in D_k\} \). It can be easily shown that the above solutions satisfy the inequality (3.3) at equality. For each \( i \in V \setminus \{r\} \), consider the solution

\[
x_v = 1 \text{ if } v = i, 0 \text{ otherwise, and } y_e = 0 \text{ for all } e \in F.
\]

Finally for each \( f = (i, j) \in E, i \neq r \text{ and } j \neq r \), consider the solution

\[
x_v = 1 \text{ if } v \in \{i, j\}
\]

0 otherwise,

\[
y_e = 1 \text{ if } e = f
\]

0 otherwise.

With the above solutions and (0,0), we can complete the proof.

\[
\blacksquare
\]

Acknowledgment

This research was partially supported by Korea Telecom. We thank the associate editor and anonymous referees for their helpful comments to enhance the presentation of the paper.
References


Table 1. Characteristics of the Test Problems

| NAME   | |V| (|C|) | |E| | |F| | CON | HUB | q | L, U | #Hub |
|--------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| KO1    | 46 (32)         | 108             | 450             | 2               | D               | 5               | 3, 12           | 5               |
| KO2    | 46 (32)         | 108             | 450             | 2               | D               | 5               | 4, 11           | 5               |
| SE1    | 54 (47)         | 123             | 470             | 1               | S               | 8               | 5, 10           | 8               |
| SE2    | 54 (47)         | 123             | 470             | 1               | S               | 8               | 4, 11           | 8               |
| KN11S1 | 24 (22)         | 56              | 142             | 1               | S               | 4               | 4, 11           | 8               |
| KN11D1 | 24 (22)         | 56              | 240             | 1               | D               | 2               | 5, 15           | 2               |
| KN12S1 | 24 (22)         | 56              | 142             | 1               | S               | 4               | 5, 10           | 4               |
| KN12D1 | 24 (22)         | 56              | 240             | 1               | D               | 2               | 8, 12           | 2               |
| KN21S1 | 24 (22)         | 56              | 142             | 1               | S               | 5               | 3, 8            | 7               |
| KN21D1 | 24 (22)         | 56              | 240             | 1               | D               | 3               | 5, 15           | 8               |
| KN22S1 | 24 (22)         | 56              | 142             | 1               | S               | 5               | 4, 7            | 7               |
| KN22D1 | 24 (22)         | 56              | 240             | 1               | D               | 3               | 8, 12           | 8               |
| KN11S2 | 24 (22)         | 56              | 163             | 2               | S               | 4               | 4, 13           | 4               |
| KN11D2 | 24 (22)         | 56              | 240             | 2               | D               | 2               | 5, 15           | 8               |
| KN12S2 | 24 (22)         | 56              | 163             | 2               | S               | 4               | 5, 12           | 4               |
| KN12D2 | 24 (22)         | 56              | 240             | 2               | D               | 2               | 8, 12           | 2               |
| KN21S2 | 24 (22)         | 56              | 163             | 2               | S               | 5               | 4, 13           | 7               |
| KN21D2 | 24 (22)         | 56              | 240             | 2               | D               | 2               | 5, 15           | 2               |
| KN22S2 | 24 (22)         | 56              | 163             | 2               | S               | 5               | 5, 12           | 7               |
| KN22D2 | 24 (22)         | 56              | 240             | 2               | D               | 2               | 8, 12           | 8               |
| SL1    | 100 (41)        | 125             | 1644            | 2               | S               | 5               | 5, 12           | 6               |
| SL2    | 100 (41)        | 125             | 1644            | 2               | S               | 5               | 7, 10           | 6               |
| SL3    | 100 (41)        | 125             | 1644            | 2               | S               | 6               | 5, 12           | 6               |
| SL4    | 100 (41)        | 125             | 1644            | 2               | S               | 6               | 7, 10           | 6               |
Table 2. Computational Results

<table>
<thead>
<tr>
<th>NAME</th>
<th>#LP</th>
<th>#CLUSTER</th>
<th>Gap (%)</th>
<th>CPU (min:sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KO1</td>
<td>11</td>
<td>32</td>
<td>0.0</td>
<td>32.6</td>
</tr>
<tr>
<td>KO2</td>
<td>14</td>
<td>35</td>
<td>0.0</td>
<td>38.1</td>
</tr>
<tr>
<td>SE1</td>
<td>35</td>
<td>164</td>
<td>0.0</td>
<td>22:24.8</td>
</tr>
<tr>
<td>SE2</td>
<td>33</td>
<td>160</td>
<td>0.0</td>
<td>10:58.6</td>
</tr>
<tr>
<td>KN11S1</td>
<td>24</td>
<td>61</td>
<td>0.0</td>
<td>32.5</td>
</tr>
<tr>
<td>KN11D1</td>
<td>17</td>
<td>25</td>
<td>0.0</td>
<td>44.1</td>
</tr>
<tr>
<td>KN12S1</td>
<td>21</td>
<td>57</td>
<td>0.0</td>
<td>30.9</td>
</tr>
<tr>
<td>KN12D1</td>
<td>31</td>
<td>48</td>
<td>0.0</td>
<td>2:35.2</td>
</tr>
<tr>
<td>KN21S1</td>
<td>18</td>
<td>72</td>
<td>0.0</td>
<td>1:16.9</td>
</tr>
<tr>
<td>KN21D1</td>
<td>20</td>
<td>84</td>
<td>0.0</td>
<td>3:16.0</td>
</tr>
<tr>
<td>KN22S1</td>
<td>17</td>
<td>70</td>
<td>0.0</td>
<td>1:44.8</td>
</tr>
<tr>
<td>KN22D1</td>
<td>21</td>
<td>83</td>
<td>0.0</td>
<td>4:00.6</td>
</tr>
<tr>
<td>KN11S2</td>
<td>17</td>
<td>46</td>
<td>0.0</td>
<td>38.0</td>
</tr>
<tr>
<td>KN11D2</td>
<td>16</td>
<td>53</td>
<td>0.0</td>
<td>4:25.4</td>
</tr>
<tr>
<td>KN12S2</td>
<td>20</td>
<td>51</td>
<td>0.0</td>
<td>56.6</td>
</tr>
<tr>
<td>KN12D2</td>
<td>17</td>
<td>24</td>
<td>0.0</td>
<td>3:25.5</td>
</tr>
<tr>
<td>KN21S2</td>
<td>17</td>
<td>55</td>
<td>0.0</td>
<td>2:48.1</td>
</tr>
<tr>
<td>KN21D2</td>
<td>14</td>
<td>20</td>
<td>0.0</td>
<td>77.9</td>
</tr>
<tr>
<td>KN22S2</td>
<td>19</td>
<td>58</td>
<td>0.0</td>
<td>3:00.0</td>
</tr>
<tr>
<td>KN22D2</td>
<td>19</td>
<td>71</td>
<td>0.0</td>
<td>10:07.7</td>
</tr>
<tr>
<td>SL1</td>
<td>41</td>
<td>148</td>
<td>0.19</td>
<td>5:20.5</td>
</tr>
<tr>
<td>SL2</td>
<td>23</td>
<td>94</td>
<td>0.0</td>
<td>4:10.6</td>
</tr>
<tr>
<td>SL3</td>
<td>38</td>
<td>153</td>
<td>0.19</td>
<td>16:50.3</td>
</tr>
<tr>
<td>SL4</td>
<td>30</td>
<td>128</td>
<td>0.0</td>
<td>6:10.6</td>
</tr>
</tbody>
</table>
### Table 3. Cluster Generation Problem Results

<table>
<thead>
<tr>
<th>(1) KN11S1 – Subproblem 1</th>
<th>(2) KN11D1 – Subproblem 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(#Var = 69, #Row = 189)</td>
<td>(#Var = 203, #Row = 589)</td>
</tr>
<tr>
<td><strong>NO</strong></td>
<td><strong>#CUT</strong></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(3) KN11S2 – Subproblem 3</th>
<th>(4) KN11D2 – Subproblem 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(#Var = 118, #Row = 346)</td>
<td>(#Var = 180, #Row = 513)</td>
</tr>
<tr>
<td><strong>NO</strong></td>
<td><strong>#CUT</strong></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>8</td>
</tr>
<tr>
<td>46</td>
<td>5</td>
</tr>
</tbody>
</table>
Figure 1. An Example of the Target Configuration
Hub Pairs : \{(2,6), (2,8), (6,8), (8,12), (12,15), (15,18), (15,19), (18,19)\}
Transit Nodes : \{23, 24\}

Figure 2. The physical graph for the problem KN11D2
Figure 3. The complement of the compatibility graph for the problem KN11D2
Figure 4. Clustering Result for the Problem KN11D2