JOURNAL OF ALGORITHMS **29,** 174–200 (1998)
ARTICLE NO. AL980964

Approximating Clique and Biclique Problems*

Dorit S. Hochbaum

Department of Industrial Engineering and Operations Research and Walter A. Haas School of Business, University of California, Berkeley, California 94720
E-mail: dorit@hochbaum.berkeley.edu

Received August 12, 1997; revised September 23, 1997

We present here 2-approximation algorithms for several node deletion and edge deletion biclique problems and for an edge deletion clique problem. The biclique problem is to find a node induced subgraph that is bipartite and complete. The objective is to minimize the total weight of nodes or edges deleted so that the remaining subgraph is bipartite complete. Several variants of the biclique problem are studied here, where the problem is defined on bipartite graph or on general graphs with or without the requirement that each side of the bipartition forms an independent set. The maximum clique problem is formulated as maximizing the number (or weight) of edges in the complete subgraph. A 2-approximation algorithm is given for the minimum edge deletion version of this problem. The approximation algorithms given here are derived as a special case of an approximation technique devised for a class of formulations introduced by Hochbaum. All approximation algorithms described (and the polynomial algorithms for two versions of the node biclique problem) involve calls to a minimum cut algorithm. One conclusion of our analysis of the NP-hard problems here is that all of these problems are MAX SNP-hard and at least as difficult to approximate as the vertex cover problem. Another conclusion is that the problem of finding the minimum node cut-set, the removal of which leaves two cliques in the graph, is NP-hard and 2-approximable.

Key Words: Approximation algorithm; half integrality; node deletion; edge deletion; biclique; maximum clique

1. INTRODUCTION

We present here new approximation algorithms based on a technique recently introduced by Hochbaum [Hoc96]. The technique relies on the integer programming formulation of the problem on constraints that involve up to three variables per constraint, where one of the three

* Research supported in part by National Science Foundation award DMI-9713482, and by SUN Microsystems.

0196-6774/98 $25.00

variables appears only in one constraint. Such problems have approximation algorithms easily derived by solving a certain minimum cut problem on a related network. The technique may also be used as a tool to identify the polynomiality of a problem via the easily recognized structure of its constraints, which we call *monotonicity*.

The collection of problems explored here is related to the maximum clique problem and to the biclique problem. A *biclique* is a complete bipartite graph. The maximum biclique problem was studied recently by Dawande et al. [DKT97]. They described interesting applications of finding the maximum edge weight subgraph that forms a biclique in bipartite graphs, and proved that the problem is NP-hard. We study this problem and several other problems of minimum node deletion or edge deletion so that the remaining subgraph is a biclique. The biclique problems discussed are listed in minimization form here:

- *Bipartite edge biclique*. Given a bipartite graph, the problem is to delete the minimum weight collection of edges so that the remaining subgraph forms a biclique.
- *General edge biclique*. Here the goal is to remove the minimum weight collection of edges from a general graph $G = (V, E)$ so that the remaining subgraph is a biclique. We consider two variants of the problem that are NP-hard. In one variant the edges in each side of the biclique may remain. In the second variant the nodes on each set of the bipartition must form an independent set and be pairwise nonadjacent in G.
- *Bipartite node biclique*. Given a bipartite graph, the goal is to delete a minimum weight collection of nodes, so that the remaining subgraph is a biclique. This problem is identified as solvable in polynomial time from the monotonicity of the formulation.
- *General node biclique*. Given a general graph, the goal is to delete the minimum weight collection of nodes so that the remaining subgraph is a biclique. As in the analogue edge problem, the biclique may or may not be required to have each set in the bipartition independent. Without this requirement the problem is shown to be solvable in polynomial time; with the requirement it is shown to be NP-hard and 2-approximable. This latter problem is also equivalent to a problem of a minimum node separator leaving two cliques in a graph.

In addition to the biclique problems, we consider an optimization-equivalent variant of the maximum clique problem. This problem is to delete minimum weight collection of edges so that the remaining subgraph is a clique. The node deletion clique problem is easily seen to be identical to a vertex cover problem, and is therefore not discussed here. The formulation structure of this edge deletion clique problem is technically similar to that of the edge biclique problems.

Although the biclique problems may seem at first to be more difficult than the *bipartization* problem that involves deleting nodes or edges so the remaining graph is bipartite, the approximation algorithms here are evidence that the opposite is the case: For the edge and node deletion bipartization problems, the best approximation algorithms known are of factor $O(\log n)$ ([GVY96], [GVY94]), where n is the number of nodes in the graph, whereas all problems discussed here are 2-approximable in polynomial time. The reader may verify that in our analysis the completeness restriction of a biclique plays a role in making the problem easier.

The paper is organized as follows. We first review the relevant technique for deriving approximation algorithms for the type of problems discussed here, IP2 problems. We then discuss the node biclique problems, then the edge biclique problems, and finally the clique problem. We present the full network for several selected problems.

One consequence of our analysis here is that since all NP-hard IP2 problems are also at least as hard to approximate as vertex cover [Hoc96a], then the problems addressed here are MAX SNP-hard and can be approximated by a factor better than 2 only if vertex cover has such approximation. To date, no such approximation is known, and it has been conjectured in [Hoc83] that 2-approximation is the best possible for the vertex cover problem.

Notation

We use either (i, j) or $\{i, j\}$ to denote an undirected edge. For a graph $G = (V, E)$ and a vertex $v \in V$ let $N(v) = \{u \,|\, (u, v) \in E\}$, the set of neighbors of v. Let $n_v = |N(v)|$, $N(u, v) = N(u) \cap N(v)$. We refer throughout to a *bipartition* as the two subsets of nodes that serve on each side of the biclique or any type of bipartite graph. We will use sans-serif acronyms to refer to formulations, and roman letters in the reference to problems.

2. THE IP2 ALGORITHM: AN APPROXIMATION TECHNIQUE

A class of integer programming formulations with up to three variables per inequality, called IP2, was analyzed for approximations in [Hoc96]. While any linear optimization problem can be written with at most three variables per inequality, the distinguishing feature of IP2 formulations is that two of the three variables (the so-called x-variables) may appear any number of times in other constraints, but the third one (the z-variable)

may appear only once. An IP2 problem is formulated as

$$
\text{(IP2)}\qquad
\begin{array}{lll}
\text{Min} & \sum_{j=1}^{n} w_j x_j + \sum e_i z_i & \\
\text{subject to} & a_i x_{j_i} + b_i x_{k_i} \geq c_i + d_i z_i & \text{for } i = 1, \ldots, m \\
& l_j \leq x_j \leq u_j & j = 1, \ldots, n \\
& z_i \text{ integer} & i = 1, \ldots, m_2 \\
& x_j \text{ integer} & j = 1, \ldots, n.
\end{array}
$$

It is assumed that among the constraints' coefficients, the values of d_i are integers. All other entries may attain arbitrary rational values. The range of values of the variables $U = \max_{j=1,\ldots,n}(u_j - l_j)$ is an important parameter in the complexity of algorithms for IP2. For the biclique and clique problems, the variables assume no more than three values, and the value of U is thus fixed. It is thus assumed throughout the discussion in this section that the value of U is fixed.

A crucial property of some IP2 problems is monotonicity.

DEFINITION 1. An inequality $ax - by \leq c + dz$ is monotone if $a, b \geq 0$ and $d = 1$.

Indeed, as proved in [Hoc96], monotone IP2 problems are solvable in polynomial time:

THEOREM 2.1 ([Hoc96]). *An IP2 problem on monotone constraints is solvable in integers in the time required to solve a minimum cut (or maximum flow) problem on a graph with $O(n)$ nodes and $O(m)$ edges.*

A monotone IP2 with all constraint coefficients in $\{-1, 0, 1\}$ is also *totally unimodular*. That means that all of the subdeterminants of the constraint matrix assume values in $\{-1, 0, 1\}$, and in particular, that all extreme points of the feasible solutions polytope are integral. Such IP2 problems can therefore be solved using any linear programming algorithm, and the optimal (basic, or extreme point) solution is guaranteed to be integer.

Some of the problems discussed here are "almost monotone," in the sense that the first part of the monotonicity requirement with respect to the x-variables applies. We shall call this form of restricted monotonicity *monotone with respect to the x-variables*. For such problems the violation of monotonicity is in the z-variables appearing in more than one constraint, or having coefficients, d, not equal to 1:

COROLLARY 2.1. *Consider an IP2 problem monotone with respect to the x-variables.*

(i) *If the z-variables appear up to p times, then there is a polynomial time algorithm attaining a superoptimal solution with the x-variables integral and the z-variables integer multiples of* $1/p$.

(ii) *If the z-variables appear with coefficients* d_i *with* $D = \max_i |d_i| > 1$, *then there is a polynomial time algorithm attaining a superoptimal solution with the x-variables integral and the z-variables integer multiples of* $1/D$.

With Theorem 2.1 the proof is straightforward. For (i) each occurrence of a z-variable is interpreted as a different variable, $z^{(i)}$, and the cost of each such occurrence is $(1/p)c(z)$, where $c(z)$ is the cost coefficient of z in the objective function. The resulting system is monotone and solvable in integers where the value of z is then set equal to $z = (1/p)\sum_{i=1}^{p} z^{(i)}$. For (ii) the variable z is substituted by $z' = DZ$, and the problem is solved in integer z' as a monotone problem. The value of z is set to $z = (1/D)z'$ for z' integer.

Although the general IP2 is NP-hard, it is solvable in polynomial time in half integers in the x-variables. That solution is a lower bound to the integer optimum and thus is a *superoptimal* solution. Not only is the bound of better quality compared to a bound derived from a linear programming relaxation; it is also obtained by using a combinatorial algorithm of minimum cut on graph that runs in strongly polynomial time and more efficiently than a linear programming algorithm. Such a superoptimal solution is useful in approximation algorithms. The reader is referred to [Hoc96] for additional details.

In this paper we show that for all problems discussed we can derive a superoptimal half integral solution that can be rounded to a 2-approximate solution.

The case of IP2 problems with only two variables per inequality was analyzed in [HN94] and in [HMNT93]. Hochbaum and Naor [HN94] devised a polynomial time algorithm to solve the monotone problem in integers, when the coefficients a_i, b_i in constraint i are of opposite signs. Hochbaum et al. [HMNT93] described a polynomial time 2-approximation algorithm for the nonmonotone version which is NP-hard. Problems that are IP2 with no more than two variables per inequality always have a feasible rounding leading to a 2-approximation, provided that the problem has a feasible integer solution. This property is not shared with problems that have three variables per inequality. But if a feasible rounding exists, it frequently leads to a more efficient approximation algorithm. Several examples of this type are illustrated in this paper.

In the complexity expressions we let $T(n, m)$ be the time required to solve a minimum s, t cut problem on a graph with m arcs and n nodes. $T(n, m)$ may be assumed to be equal to $O(mn \log(n^2/m))$, which is the complexity of the push-relabel algorithm with dynamic tree data structure

[GT88]. When we refer to *half integral solutions*, these are feasible solutions with all components that are integer multiple of half.

THEOREM 2.2 ([Hoc96]). *Assume that in the given IP*2 *problem,* $d_i \leq 1$ *and* $U \leq 2$. *Then, a superoptimal half integral solution to the IP*2 *problem is attained in time* $T(n, m)$.

The half integral solution resulting from the solution is used to derive a 2-approximate solution by rounding its components to an integer feasible solution for each of the problems discussed. The 2-approximation algorithms presented in this paper are special cases of Theorem 2.2.

The technique for solving the IP2 problem in integer multiples of half involves transforming the formulation to another formulation where the constraints are monotone and their coefficients form a totally unimodular matrix. That, in turn, is solvable in polynomial time in integers. The transformation is such that only a factor of 2 is lost in the integrality of the x-variables if the original formulation was nonmonotone (for monotone formulations there is no loss of integrality). That is, when the inverse transformation is applied, every integer value of the variable is mapped to a half integer. This technique will be illustrated in detail for some of the problems discussed here.

The construction of the networks described here follows the method introduced in [Hoc96]. To facilitate the deciphering of the networks described, we mention only that each node is associated with some binary choice of values, and the rule of identifying a node value is to set it at its upper bound if and only if it is in the source set of a cut.

3. THE NODE BICLIQUE PROBLEM

3.1. *The Bipartite Node Biclique Problem*

The node biclique problem on a bipartite graph is solvable in polynomial time. This was first observed by Yannakakis [Yan81b]. The problem is equivalent to the maximum independent set on bipartite graphs that is known to be solvable by a minimum cut algorithm. The polynomiality is also evident from the formulation that is monotone and thus solvable in polynomial time in integers. To see this, consider a (maximization) formulation of the problem (previously given in [DKT97]) given on the bipartite graph $B = (V_1, V_2, E)$. Let $x_j = 1$ if node j is in the biclique.

(BNB)

$$\begin{array}{rll} \text{Max} & \sum_{j \in V} w_j x_j & \\ \text{subject to} & x_i + x_j \leq 1 & \text{for edge } \{i, j\} \notin E, \quad i \in V_1, \quad j \in V_2 \\ & x_j \in \{0, 1\} & \text{for all } j \in V. \end{array}$$

The constraints each involve two types of variables, those representing nodes in V_1 and those representing nodes in V_2. Thus multiplying one of these sets, say the variables in V_2, by -1 gives a formulation that is monotone. The network constructed for solving such a formulation is a bipartite network with only source and sink-adjacent arcs having finite capacity. The network is depicted in Fig. 1.

The formulation of BNB is identical to the formulation of the independent set problem on the bipartite complement $\bar{B} = (V_1, V_2, \bar{E})$. We sketch for the sake of completeness the basic idea of using a minimum cut algorithm for solving the independent set problem on bipartite graphs.

The minimum s,t-cut problem corresponding to the independent set problem is defined on a network where all nodes i in V_1 are linked to the source with arcs of capacities $u_{s,i} = w_i$, and all nodes j in V_2 are linked to a sink t with arcs of capacities $u_{j,t} = w_j$. All edges in the bipartite graphs are represented as directed arcs from V_1 nodes to V_2 nodes with infinite capacity. Given a finite capacity cut (S, T), the set $(S \cap V_1) \cup (T \cap V_2)$ is an independent set of weight $\sum_{j \in V_1 \cup V_2} w_j - (\sum_{i \in V_1 \cap T} u_{s,i} + \sum_{j \in V_2 \cap S} u_{j,t})$. Thus the weight of the independent set is equal to a constant minus the weight of the corresponding finite cut. Minimizing the capacity of the cut is therefore equivalent to maximizing the weight of the independent set. Although not immediately evident, this algorithm is a special case of the algorithm for solving problems on two variables per inequality of [HMNT93], which generates a network identical to that in Fig. 1.

The corresponding node deletion problem—the deletion of minimum weight collection of nodes so that the remaining bipartite graph is com-

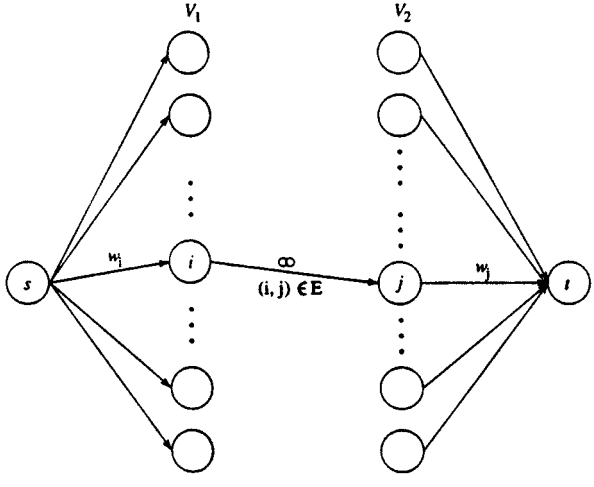


FIG. 1. The network used to solve the bipartite node biclique (BNB) problem.

plete—is obviously solvable in polynomial time as well. The optimal solution is the complement of the independent set corresponding to the minimum cut.

3.2. *General Node Biclique, without the Independence Requirement*

The general node biclique problem is to find in a general graph $G = (V,E)$, a node-induced subgraph that forms a biclique in that it defines two disjoint subsets of nodes, $V_1, V_2 \subset V$, that include *all* edges between V_1 and $V_2, V_1 \times V_2$. In this subsection we consider the version in which the biclique is not required to have the nodes on each side of the bipartition pairwise nonadjacent. We provide two different formulations for this problem, with the first having two variables per inequality, and the second having three variables per inequality. The second formulation leads to a more efficient algorithm and is more useful in extensions to other formulations discussed here. In the first formulation the objective is to maximize the weight of the nodes in the biclique. It is shown that the formulation is monotone, and thus the problem is solvable in polynomial time.

3.2.1. *Formulation* 1

Let each node have three possible states indicated by the values $-1, 0, 1$. The values 1 and -1 imply that the node is in the bipartition, and specify which side of the bipartition it is in. The value 0 implies that node j is not in the biclique. A node contributes to the objective function if it is in the biclique, or if its value is -1 or 1. The variable $y_j^{(1)}$ is equal to 1 when $x_j = -1$ and $y_j^{(2)} = 1$ when $x_j = 1$. One trivial feasible solution is a single edge, the endpoints of which form a biclique. The formulation is given for each possible choice of such edge $\{s, t\} \in E$. The formulation is given first for the maximization problem. It models the maximum node biclique (MNB) problem on general graphs conditioned on a given pair s, t being in the biclique.

$(\mathrm{MNB}(s,t)_1)$

$$
\begin{array}{rll}
\text{Max} & \sum_{j \in V} w_j y_j^{(1)} + \sum_{j \in V} w_j y_j^{(2)} & \\
\text{subject to} & 1 - x_j \geq 2y_j^{(1)} & \text{for all } j \in V \\
& 1 + x_j \geq 2y_j^{(2)} & \text{for all } j \in V \\
& x_i - x_j \leq 1 & \text{for edge } \{i,j\} \notin E \\
& x_j - x_i \leq 1 & \text{for edge } \{i,j\} \notin E \\
& x_s = 1, \quad x_t = -1. & \\
& x_j \in \{-1, 0, 1\},\ y_j^{(1)}, y_j^{(2)} & \text{binary for all } j \in V.
\end{array}
$$

The first two sets of constraints ensure that nodes contribute weight to the biclique only when they are selected on either side of the bipartition.

LEMMA 3.1. *The formulation* $(\mathsf{MNB}(s,t)_1)$ *is monotone, and* (MNB) *is thus solvable in integers in* $O(m \cdot T(n, \binom{n}{2} - m))$.

Proof. The formulation has up to two variables per inequality, and their coefficients are of opposite signs. To see this, we replace the variables $y_j^{(1)}$ by their negatives $y_j'^{(1)} = -y_j^{(1)}$. The variables $y_j'^{(1)}$ thus assume values in $\{-1, 0\}$. Theorem 2.1 is applicable to this monotone formulation. The optimal integer solution can therefore be generated in polynomial time, by constructing a minimum cut solution on a certain network. The network for the minimization version, $\mathsf{DNB}(s,t)_1$, is given in Fig. 2.

The running time for solving the problem is m times the complexity of minimum cut on a graph with n nodes and $\overline{m} = \binom{n}{2} - |E|$ arcs. The overall complexity is thus $O(m \cdot T(n, \binom{n}{2} - m))$. ∎

To speed up the running time, one could employ ideas similar to those used by Hao and Orlin's algorithm for minimum cut in directed networks [HO94]. That algorithm involves switching the identity of the sink, yet it is necessary also to adapt it to switching the identity of the source. Such an adaptation was recently presented by Henzinger et al. in the context of node connectivity [HRG96]. Indeed, the node biclique problem is closely related to the node connectivity problem, as we show next.

The complementary problem to the maximum node biclique is the minimization of weight of nodes deleted so that the remaining subgraph is

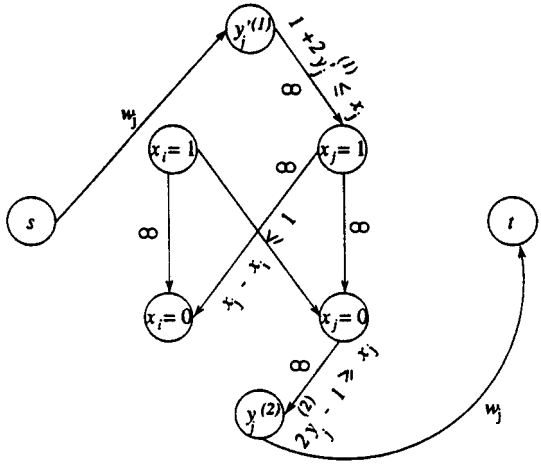


FIG. 2. The network used to solve $(\mathsf{DNB}(s,t)_1)$.

a biclique. This minimization problem is, of course, also solvable in polynomial time. The formulation of the deletion node biclique problem (DNB) corresponding to the formulation $(\mathrm{MNB}(s,t)_1)$ has one shortcoming—the objective value is not quite the weight of the deleted nodes. Here the value of the integer objective is $\sum_{j \in V} w_j + \sum_{x_j = 0} w_j$, which differs from the desired objective by a constant, $W = \sum_{j \in V} w_j$.

The corresponding set of variables for the deletion/minimization problem includes the variables $y_j^{(1)} = 1$ if $x_j = 0$ or -1, and $y_j^{(2)} = 1$ if $x_j = 1$ or 0.

$(\mathrm{DNB}(s,t)_1)$

$$
\begin{array}{lll}
\text{Min} & \sum_{j \in V} w_j y_j^{(1)} + \sum_{j \in V} w_j y_j^{(2)} & \\
\text{subject to} & 1 - x_j \le 2y_j^{(1)} & \text{for all } j \in V \\
& 1 + x_j \le 2y_j^{(2)} & \text{for all } j \in V \\
& x_i - x_j \le 1 & \text{for edge } \{i,j\} \notin E \\
& x_j - x_i \le 1 & \text{for edge } \{i,j\} \notin E \\
& x_s = 1, \quad x_t = -1. & \\
& x_j \in \{-1, 0, 1\},\ y_j^{(1)},\ y_j^{(2)} & \text{binary for all } j \in V.
\end{array}
$$

A description of the network is given in Fig. 2. For each node j we have four nodes in the network. One, indicated by $x_j = 0$, implies that $x_j \ge 0$ if the node is in the source set of a minimum cut. Similarly, the node indicated with $x_j = 1$ implies that $x_j = 1$ if this node is in the source set. The two other nodes correspond to $y_j'^{(1)}$ and $y_j^{(2)}$, and attain their upper bound, of 0 and 1, respectively, if the nodes are in the source set.

The network has $O(n)$ nodes and $O(\overline{m} + n)$ edges. The DNB problem is thus solvable in time $O(m \cdot T(n, \binom{n}{2} - m))$.

Although the problem is in P as a consequence of its monotonicity, it turns out that there is another explanation for the polynomiality:

LEMMA 3.2. *The deletion node biclique problem, DNB, without the independence restriction is equivalent to the weighted node connectivity problem on the complement graph.*[1]

Proof. A graph is complete bipartite (without independence restriction) if and only if its complement is disconnected—the two sides of the bipartition form unions of connected components in the complement graph. So, the minimum number of nodes whose deletion leaves a complete bipartite subgraph is equal to the node connectivity of the complement graph. ∎

[1] We gratefully acknowledge M. Yannakakis for pointing this out.

The equivalence of DNB to the node connectivity problem permits the use of the implementation described in [HRG96] to solve DNB in time $O(\kappa_1 \cdot T(n, \binom{n}{2} - m))$, where $\kappa_1 \leq \overline{m}/n$ is the unweighted node connectivity of G.

The directed version of the problem is also easy to represent: to formulate the *directed* node connectivity problem on $G = (V, A)$, we replace the pairs of constraints,

$$x_i - x_j \leq 1 \qquad \text{for edge } \{i, j\} \notin E$$
$$x_j - x_i \leq 1 \qquad \text{for edge } \{i, j\} \notin E,$$

by the single constraint, $x_i - x_j \leq 1$, for $(i, j) \notin A$. The directed node connectivity is equivalent to a *directed node biclique* problem with a complete set of arcs directed from one side of the bipartition to the other.

3.2.2. *Formulation* 2

This alternative formulation for the node biclique problem has the advantage of having an "exact" objective. The formulation relies again on the argument that the optimal biclique contains at least one pair of nodes in the graph and the edge that links them. This time, however, we take advantage of the restriction to include nodes s and t in the clique by removing a priori all nodes that cannot be present in the same biclique with these two nodes. For a pair of adjacent nodes s, t in the biclique, we consider the subgraph induced by the neighbors of s, $N(s)$, that contain t, and the neighbors of t, $N(t)$, that contain s. Any biclique containing s and t must have each side of its bipartition contained in $N(s)$ and $N(t)$, respectively. This construction appears to reduce the general graph problem to the bipartite version on $N(s)$, $N(t)$. This is not the case, however, as there are nodes in $N(s, t) = N(s) \cap N(t)$ that are candidates for either side of the bipartition.

The construction of the induced bipartite graph has $N(s)$ for one side of the bipartition and $N(t)$ for the other. The nodes in $N(s, t)$ appear on both sides, with each node v duplicated as v in $N(s)$ and v' in $N(t)$, and each copy having all edges between v and all of the nodes adjacent to it on the opposite side of the bipartition. It is important to note that there is no edge between v and v' to prevent both copies from being present in the biclique.

We choose here decision variables x_j and y_j that are binary. The variable $x_j = 1$ if the node $j \in N(s) \cup N(t) \setminus N(s, t)$ is *deleted* from the graph and the variable $y_j = 1$ for $j \in N(s, t)$ if the node j is deleted from the graph. The determination of the side of the bipartition that j belongs

to follows immediately from its membership in either $N(s)$ or $N(t)$. The challenge is to make sure that a node that appears on both sides (because it is in $N(s,t)$) will not be charged for unless it appears on neither side of the selected biclique, and then charged for only once for its deletion. This is achieved first by setting a constraint $x_i + x_j \geq 1$ for $(i,j) \notin E$, which applies in particular to the pair v, v' as $x_v + x_{v'} \geq 1$, thus ensuring that at least one of the copies it deleted. Second, a node v in $N(s,t)$ is considered deleted only if both copies of the node are deleted and the corresponding variables' values are 1, in which case the value of the corresponding y variable is 1.

Let $V_{s,t} = N(s) \cup N(t)$, and $E_{s,t}$ be the set of edges with both endpoints in $V_{s,t}$.

$(\mathsf{DNB(s,t)}_2)$

$$
\begin{aligned}
z_{s,t} = \text{Min } & \textstyle\sum_{j \in V_{s,t} \setminus N(s,t)} w_j x_j + \sum_{j \in N(s,t)} w_j y_j \\
\text{subject to } & x_v + x_{v'} - 1 \leq y_v && \text{for } v \in N(s,t) \\
& x_i + x_j \geq 1 && \text{for edge } \{i,j\} \notin E_{s,t} \quad i \in N(t), \\
& && \quad j \in N(s) \\
& x_j \in \{0,1\}, && \text{for } j \in V_{s,t}.
\end{aligned}
$$

Remark 3.1. It is optional but not essential to include here the condition that $x_s = x_t = 0$. If one of these two nodes is deleted in the optimal solution, then the edge (s,t) is not a part of an optimal biclique.

LEMMA 3.3. *The formulation* $(\mathsf{DNB(s,t)}_2)$ *is monotone, and the linear programming relaxation's basic solutions are integral.*

Proof. To see that the formulation is equivalent to a monotone one, we multiply the variables x_j for j in $N(s)$ by -1 so that they attain values in $\{-1, 0\}$. The resulting formulation is

$(\mathsf{DNB(s,t)}_2)$

$$
\begin{aligned}
z_{s,t} = \text{Min } & \textstyle\sum_{j \in V_{s,t} \setminus N(s,t))} w_j x_j + \sum_{i \in N(s,t)} w_i y_i \\
\text{subject to } & -1 + (-x_v + x_{v'}) \leq y_v && \text{for } v \in N(s,t) \\
& x_i - x_j \geq 1 && \text{for edge } \{i,j\} \notin E, \\
& && \quad i \in N(t), \quad j \in N(s) \\
& x_j \in \{-1, 0\}, && \text{for } j \in N(s) \\
& x_i \in \{0,1\}, && \text{for } i \in N(t) \\
& y_j \in \{0,1\}, && \text{for } j \in N(s,t).
\end{aligned}
$$

This formulation is now monotone. Therefore a procedure involving minimum cut is delivering an integer solution. Furthermore, the constraint

matrix has all coefficients in $\{0, -1, 1\}$ and is monotone and therefore totally unimodular, as shown in [Hoc96], and as discussed in the Introduction, it is a cut polytope. Hence the linear programming optimal solution and all basic solutions are also integer. ▌

Solution method for general node biclique. We solve for each $s, t \in V$ such that $(s, t) \in E$ the formulation (DNB(s, t)$_2$). We then choose among the values $\sum_{j \in V \setminus V_{s,t}} w_j + z_{2,t}$ the smallest value of the relaxation. Since the formulation is monotone, the optimal solution delivered is in integers. The network is given in Fig. 3. Note that a node in $N(s)$ is in the source set if and only if its value is 0, and in the sink set if and only if its value is -1.

Complexity. For each s, t pair there are up to $2\binom{n}{2} - 2m + |N(s, t)| = O(n^2)$ edges in the constructed graph. The complexity is thus m times the complexity of solving a minimum cut problem on a graph with $O(n)$ nodes and $O(n^2)$ edges, $O(m \cdot T(n, n^2))$. This running time is a constant factor faster than for formulation 1. The difference in running time is attributed to having a formulation with three variables per inequality versus the two variables per inequality interpretation in the previous formulation. As we shall see, for edge biclique formulations this different interpretation may

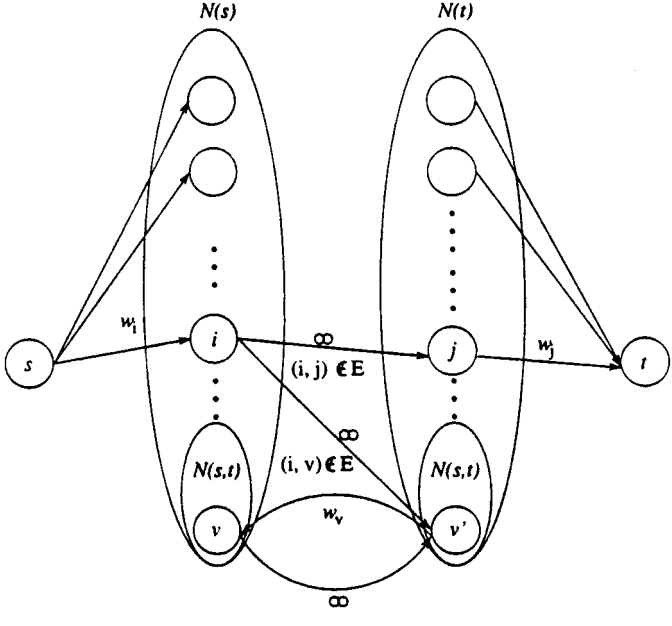


FIG. 3. The network used to solve (DNB(s, t)$_2$).

result in a more significant gap in the complexity of the approximation algorithm.

3.3. *General Node Biclique, with the Independence Requirement*

Adding the independence requirement lends the previous formulations nonmonotone: it is necessary to include constraints of the type $x_{i_1} + x_{i_2} \leq 1$ for $i_1, i_2 \in N(s)$, or for $i_1, i_2 \in N(t)$. Such constraints are no longer monotone, since the variables cannot be partitioned into two distinct sets so that one set's coefficients can be made negative. We verify that such a partition is impossible by demonstrating that the general node biclique problem is an NP-hard problem.

LEMMA 3.4. *The general node biclique problem with independence requirement is NP-hard.*

Proof. We reduce the independent set problem to this general node biclique problem. Given an independent set problem on a graph $G = (V, E)$, we construct a graph G^2 by duplicating the set of nodes V as V and V' and the edges as E and E'. Now join every node in V with every node in V'. A biclique in G^2 is any pair of independent sets in V and V'. In particular, the weight of the nodes in the biclique is maximized if the independent set in V and the one in V' are of maximum weight. ∎

For an alternative proof that the problem is NP-hard, observe that the biclique subgraph property is hereditary, and as such the complexity argument of Yannakakis [Yan81b] implies it is NP-hard.

To formulate the problem we employ the choice of variables x_j as binary variables equal to 1 and only if node j is deleted. As before, we construct the bipartite graph on $N(s), N(t)$ for any choice of adjacent nodes s and t. This time, since each side of the bipartition must be independent, all nodes of $N(s, t)$ are removed from the graph, as they are adjacent to both s and t and thus cannot be on either side of the bipartition.

We let $N'(s) = N(s) \setminus N(s, t)$ and $N'(t) = N(t) \setminus N(s, t)$, and $V_{s,t} = N'(s) \cup N'(t)$.

(NBI(s,t))

$$
\begin{array}{lll}
z_{s,t} = \text{Min} & \sum_{j \in V_{s,t}} w_j x_j & \\
\text{subject to} & x_i + x_j \geq 1 & \text{for edge } \{i, j\} \notin E, \quad i \in N'(t), \\
 & & \quad j \in N'(s) \\
 & x_i + x_j \geq 1 & \text{for edge } \{i, j\} \in E \quad i, j \in N'(t) \\
 & x_i + x_j \geq 1 & \text{for edge } \{i, j\} \in E \quad i, j \in N'(s) \\
 & x_j \in \{0, 1\}, & \text{for } j \in V_{s,t}.
\end{array}
$$

The first set of constraints says that for any edge missing in the bipartition, at least one endpoint is deleted so as not to violate the complete bipartite requirement. The two other sets of constraints say that for any edge within $N(s)$ or $N(t)$, at least one endpoint is not in the biclique, as otherwise the independence requirement will be violated.

Complexity and solution method. There are a couple of alternative ways of solving this problem. In one we monotonize and solve m minimum cut problems on a graph with $2(n_s + n_t)$ nodes and $O(n^2)$ edges for a total complexity of $O(m \cdot T(n, n^2))$. Alternatively, observe that the formulation (NBI(s, t)) is that of a vertex cover on a graph containing the set of edges induced by $N'(s)$ and $N'(t)$ and the complement of the edge set in the bipartition. Each of these m vertex cover problems is 2-approximable in polynomial time. The 2-approximation for the general node biclique problem is the minimum of $\sum_{j \in V \setminus V_{s,t}} w_j + z_{s,t}$ for all pairs s and t. It is possible to use Bar-Yehuda and Even's 2-approximation algorithm for vertex cover [BYE81], which runs linear in time in the number of elements/edges that need to be covered. Here this number is $O(n^2)$. The procedure has to be run for each selected edge (s, t), and thus the overall complexity is $O(mn^2)$. The appropriate network is depicted in Fig. 4.

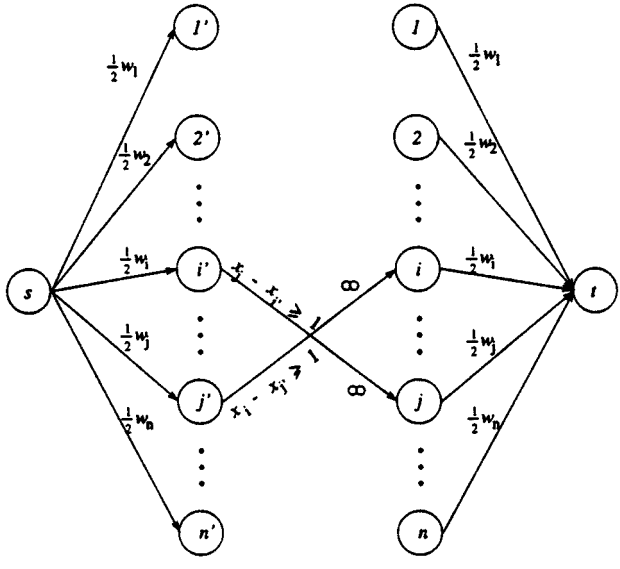


FIG. 4. The network used to solve the node biclique problem with independence requirement (NBI(s, t)). Here $\{i, j\} \notin E$, or $i, j \in N'(t)$, or $i, j \in N'(s)$. The set $V_{s,t}$ is assumed to contain n nodes. $V_{s,t} = \{1, \ldots, n\}$.

Therefore we have a polynomial 2-approximation algorithm for minimizing node biclique on general graphs.

Remark 3.2. Consider the clique vertex connectivity problem, which is to find a minimum weight node separator, the removal of which leaves two disconnected **cliques**. That problem is identical to the general node biclique with the independence requirement (NBI) on the complement graph. (To see this, apply the same arguments as in Lemma 3.2). The clique vertex connectivity problem is hence NP-hard and 2-approximable, as a consequence of the discussion above.

This is remarkable in that the node deletion problem that leaves a single clique in a graph is equivalent to the vertex cover problem and thus is 2-approximable. Here we require that the deleted node set leaves two cliques, and yet the problem is still 2-approximable without an increase in complexity. In contrast, the node deletion problem to two cliques that are not required to be fully disconnected is the bipartization node deletion problem. For this problem the best approximation factor known to date is $O(\log n)$ [GVY94].

4. THE EDGE BICLIQUE PROBLEM ON BIPARTITE GRAPHS

The edge-weighted biclique problem is to delete from a bipartite graph $B = (V_1, V_2, E)$ a minimum weight collection of edges so that the remaining edges induce a complete bipartite graph—a biclique. We refer to this problem with the acronym BEB (bipartite edge biclique). Dawande et al. proved that the weighted version of this problem is NP-complete by reduction from maximum clique [DKT97]. For the sake of completeness, we sketch this reduction.

LEMMA 4.1 ([DKT97]). *Edge biclique on bipartite graph is an NP-hard problem.*

Proof. The reduction is from the maximum clique problem defined on a graph $G = (V, E)$. Construct a bipartite graph (V, V, E') with the set of edges $E' = \{(u, v) | (u, v) \in E, \text{ or } u = v\}$. The edges of the form (u, u) get the weight of 1, and the others the weight of 0. A maximum weight biclique corresponds to a maximum clique with a number of nodes equal to the weight of the biclique. ∎

We present two alternative formulations of the problem. There is a trade-off between the two formulations, with one leading to a superoptimal half integral solution faster than the other. Yet the slower formulation provides a tighter lower bound but the same approximation factor.

4.1. *Formulation* 1

Let a node variable x_j be 1 if node j is in the biclique and 0 otherwise. Let variable z_{ij} be defined for each edge $(i, j) \in E$ as equal to 1 if the edge is deleted or 0 otherwise. Equivalently stated, $z_{ij} = 1$ if and only if $x_i + x_j < 2$. The following formulation of bipartite biclique is an IP2:

(BEB1)

$$\begin{array}{lll} \text{Min} & \sum_{(i,j)\in E} c_{ij} z_{ij} & \\ \text{subject to} & 2 - (x_i + x_j) \le 2 z_{ij} & \text{for } (i,j) \in E \\ & x_i + x_j \le 1 & \text{for } (i,j) \notin E \quad i \in V_1, j \in V_2. \\ & x_i, z_{ij} & \text{binary for all } i, j. \end{array}$$

The first set of constraints guarantees that unless an edge has both endpoints in the biclique, it must be deleted. The second set ensures that every pair of nodes included in the biclique, on opposite sides of the bipartition, must have an edge between them. Together these constraints say that the set of nodes selected is a biclique, and that edges not in the biclique are deleted. In each constraint there is one node variable that belongs to V_1 and one to V_2. These can thus be made to appear with opposite signs. Only the coefficient 2 of z_{ij} destroys the total unimodularity of the constraint matrix: all entries in a totally unimodular matrix must be 0, 1 or -1. We can thus solve in polynomial time the problem in integer x-variables and half integral z-variables as in Corollary 2.1 (ii). The network and its construction are discussed later in Subsection 4.3.1.

4.2. *Formulation* 2

Using the same variables as in formulation 1, we state the problem in an equivalent *disaggregate* formulation:

(BEB2)

$$\begin{array}{lll} \text{Min} & \sum_{(i,j)\in E} c_{ij} z_{ij} & \\ \text{subject to} & 1 - x_i \le z_{ij} & \text{for } (i,j) \in E \\ & x_i + x_j \le 1 & \text{for } (i,j) \notin E \quad i \in V_1, \quad j \in V_2. \\ & x_i, z_{ij} & \text{binary for all } i, j. \end{array}$$

This formulation is identical to (BEB1), except that the first set of constraints is split into twice as many equivalent constraints, each enforcing the requirement that if an endpoint of an edge is not in the biclique, then the edge must be deleted. The formulation is tighter than (BEB1), since a fractional feasible solution to (BEB2) is also feasible for (BEB1),

but not the other way around. (BEB2) is in general slower to solve. If we cast it as a problem in two variables per inequality, then the number of nodes in the network created is $O(m + n)$, as opposed to $O(n)$ nodes in (BEB1). The number of arcs in the networks is the same for both formulations: $O(\overline{m} + n)$. However, we can treat (BEB2) as a formulation with up to three variables per inequality, while the double appearance of the variable z_{ij} can be considered as two different variables as in Corollary 2.1(i). The resulting network corresponding to this formulation would then be equivalent to that of (BEB1), which will be discussed in detail later.

As all coefficients are in $\{-1, 0, 1\}$, it is not obvious that the constraints of (BEB2) cannot be written as a monotone system. This would be a consequence of the NP-hardness of the problem. We settle this directly in the following lemma:

LEMMA 4.2. *The constraint matrix of* (BEB2) *is not totally unimodular.*

Proof. The following subset of constraints creates a 6-cycle with corresponding determinant equal to 2. The constraints involve the nodes $i_1, i_2 \in V_1, j_1, j_2 \in V_2$, and the edges $(i_1, j_2), (i_2, j_1) \in E$, the edges $(i_1, j_1), (i_2, j_2) \notin E$. The six inequalities creating a 6-cycle are

$$\begin{aligned} 1 - x_{i_1} &\le z_{i_1 j_2} \\ 1 - x_{j_2} &\le z_{i_1 j_2} \\ x_{i_2} + x_{j_2} &\le 1 \\ 1 - x_{i_2} &\le z_{i_2 j_1} \\ 1 - x_{j_1} &\le z_{i_2 j_1} \\ x_{i_1} + x_{j_1} &\le 1. \end{aligned}$$

The determinant of the 6×6 submatrix defined by the coefficients of these constraints is 2, and thus the matrix is not totally unimodular. ∎

4.3. *Solving BEB*1

Either formulation leads to a 2-approximation algorithm. We show how this is done for formulation 1.

To transform the constraints into a monotone system, we apply a transformation on the variables:

$$\begin{aligned} x_i^+ &= x_i \qquad \text{for } i \in V_1 \\ x_j^- &= -x_j \quad \text{for } j \in V_2 \\ q_{ij} &= 2 z_{ij}. \end{aligned}$$

With this transformation $q_{ij} \in \{0, 2\}$. Substituting this requirement by $0 \le q_{ij} \le 2$ and integer, the constraints are of the form

(relaxed BEB1)

$$
\begin{aligned}
& x_j^- - x_i^+ \le -2 + q_{ij} && \text{for } (i,j) \in E \\
& x_i^+ - x_j^- \le 1 && \text{for } (i,j) \notin E \quad i \in V_1, \quad j \in V_2. \\
& x_i^+ \in \{0,1\}, x_j^- \in \{-1,0\} && \\
& q_{ij} \in \{0,1,2\}, && \text{for all } i,j.
\end{aligned}
$$

The transformed constraints form a relaxation of BEB1, relaxed BEB1. The constraints' coefficients constitute a totally unimodular matrix: a matrix with one 1 and one -1 in each row, appended with the identity matrix. All extreme points of such polytopes are integral. Therefore this problem is solvable in integers using linear programming. Having integer extreme points means that q_{ij} assumes values in $\{0, 1, 2\}$ rather than in $\{0, 2\}$.

The integer optimal solution to relaxed-BEB1 provides a superoptimal half integral solution to the problem BEB, in which the x variables assume integer values and z_{ij} assume values in $\{0, \frac{1}{2}, 1\}$.

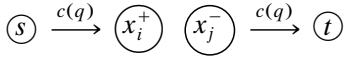
We now show how to solve relaxed BEB1 as a minimum cut problem on a certain network.

4.3.1. *The Network for Relaxed BEB*1

As previously mentioned, a node in the network belongs in the source set if and only if the corresponding variable value is at the upper bound, which is 1 for x_i^+ and 0 for x_j^-.

With this interpretation, a constraint from the second set $x_i^+ - x_j^- \le 1$ is represented by an arc going from x_i^+ to x_j^-. The arc has infinite capacity, enforcing the requirement that if $x_i^+ = 1$, then $x_j^- = 0$. A constraint from the first set, $x_j^- - x_i^+ \le -2 + q_{ij}$, is represented by an arc from the source to $x_i^+ = 1$ of capacity $\frac{1}{2}c_{ij}$ and an arc from $x_j^- = 0$ to the sink of the same capacity, $\frac{1}{2}c_{ij}$. The gadget used to represent such a constraint, with $c(q)$ representing the cost per unit of q in the objective, is

$$x_j^- - x_i^+ \le -2 + q,\ q \in \{0, 1, 2\}$$



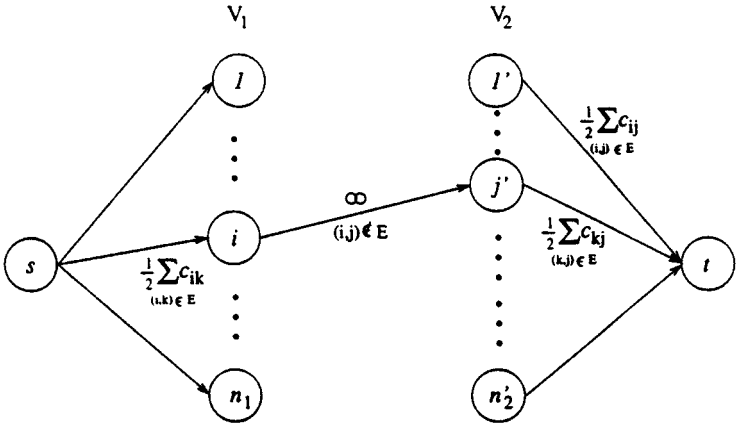


FIG. 5. The network uses to solve the relaxed bipartite edge biclique, relaxed BEB1 problem.

All arcs (i, j) adjacent to node i are consolidated into one arc from s to x_i^+ and from x_i^- to t. Figure 5 illustrates the entire bipartite network in which a minimum cut corresponds to an optimal solution to the relaxed BEB1. This is proved in the next lemma, which is a special case of Theorem 2.2.

LEMMA 4.3. *Any finite cut* $(S, \bar{S})$ *corresponds to a feasible solution to relaxed BEB*1, *and the minimum cut corresponds to an optimal solution to relaxed BEB*1.

Proof. Recall that variables are in the source set if and only if they are at their upper bound. Namely,

$$x_i^- = \begin{cases} 0 & x_i^- \in S \\ -1 & x_i^- \in \bar{S} \end{cases}$$

$$x_i^+ = \begin{cases} 1 & x_i^+ \in S \\ 0 & x_i^+ \in \bar{S}. \end{cases}$$

An arc is charged to the cut if it is an arc (s, i) and $x_i^+ \in \bar{S}$, or if it is an arc (j, t) and $x_j^- \in S$.

Consider the four possible cases for the values of the x variables associated with nodes i and j and their corresponding membership in source or sink sets.

x_i^+	x_j^-	q_{ij}	Arcs in cut $(S, \bar{S})$	Cost of cut arcs
$1, S$	$-1, \bar{S}$	0	$\varnothing$	0
$1, S$	$0, S$	1	(x_j^-, t)	$\frac{1}{2}c_{ij}$
$0, \bar{S}$	$-1, \bar{S}$	1	(s, x_i^+)	$\frac{1}{2}c_{ij}$
$0, \bar{S}$	$0, S$	2	$(s, x_i^+), (x_j^-, t)$	c_{ij}

Thus the value of the cut is the same as the value of the feasible solution, and vice versa. The minimum cut thus provides the optimal integer solution to the problem with the variables q_{ij}. ∎

The minimum cut solution provides a superoptimal half integral solution in which only z_{ij} may be fractional, whenever $q_{ij} = 1$. A feasible rounding is achieved by rounding z_{ij} up.

The number of nodes in the network used to derive the half integral solution is $n_1 + n_2 = |V_1| + |V_2|$. The number of arcs is $\overline{m} + n_1 + n_2$, where $\overline{m}$ is the number of arcs in the complement graph $\overline{m} = n_1 n_2 - m$. We thus have a 2-approximation algorithm of complexity $T(n_1 + n_2, n_1 + n_2 + \overline{m})$ for this NP-hard problem.

The readers familiar with the vertex cover problem may notice that the bipartite network used to solve the bipartite edge biclique approximately is the same network as would be used to solve the vertex cover problem on a bipartite graph where the weight of node i is $\frac{1}{2}\sum_{(i,k)\in E} c_{ik}$, half the sum of weights of the adjacent edges. The nonedges (those in $\bar{E}$) are the ones to be covered. Indeed, this vertex cover problem is a factor of 2 relaxation of the edge biclique problem on bipartite graphs: for every nonedge $(i, j) \notin E$, delete the set of edges adjacent to either i or j, so that the total cost of the deleted edges is minimum. If the edges are deleted because of only one endpoint, then the cost charged is half of the cost of the edge. This renders the solution a lower bound that is also within a factor of 2 of a value of a feasible solution that is an upper bound.

The network for the disaggregate formulation (BEB2) has a node for each variable, for a total of $2m + n$ nodes, and two arcs for each constraint, for a total of $2\overline{m} + 2m$ arcs. The detailed description of the network is omitted. Interpreting the variables z_{ij} in the formulation as a "third" z-variable results in exactly the same network as the one for (BEB1) in Fig. 5.

5. EDGE BICLIQUE ON GENERAL GRAPHS

The aim in the edge biclique problem is to delete a minimum weight collection of edges from a graph $G = (V, E)$ so that the remaining edge-induced subgraph $(V_1, V_2, E_{1,2})$ forms a biclique. When defined on general graphs the problem has two possible interpretations, mentioned earlier in the Introduction:

- Version 1: Nodes in V_1 and V_2 must form an independent set in the graph $G = (V, E)$.
- Version 2: V_1 and V_2 may form any subgraph in G, and the edges within them need not be eliminated to create a biclique.

The two versions are NP-hard, since the general graph problem generalizes the bipartite case (the bipartite case is reducible to the general case). Version 1 is also NP-hard because of the independence requirement for each side of the bipartition. To see this we construct a new graph formed by duplicating G twice, with nodes V and V' and all edge weights set to zero, and placing all possible edges between all nodes of V and V' with edge weights equal to 1. Clearly, any pair of subsets of V and V' forms a biclique. If each side of the biclique is required to be independent, the problem is equivalent to maximizing the size of an independent set in G.

5.1. *Formulation and approximation of Version* 1

An optimal edge biclique contains at least one edge. The formulations are therefore given for each possible guess of such an edge, $(s, t) \in E$. The presence of (s, t) in the biclique implies that the nodes in one side of the biclique are in the set of neighbors of s, $N(s)$, and the nodes on the other side are in the set of the neighbors of t, $N(t)$. Let $N(s, t) = N(s) \cap N(t)$ as before.

Here each side of the bipartition must form an independent set in $G = (V, E)$. Nodes of $N(s, t)$ are adjacent to both s and t and thus cannot be on the same biclique with this pair of nodes on opposite sides. Therefore the candidate nodes for one side of the bipartition are in the set $N'(s) = N(s) \setminus N(s, t)$, and for the other side the nodes are in $N'(t) = N(t) \setminus N(s, t)$.

The edge biclique problem with the requirement that (s, t) is in the biclique is denoted by $\mathrm{EB}_{s,t}$ and the optimal value by $z_{s,t}$. This problem is defined on the set of nodes $V_{s,t} = N'(s) \cup N'(t)$ and the set of edges $E_{s,t}$ that have both endpoints in $V_{s,t}$. All other edges are deleted. The optimal solution to the edge biclique will be $\min_{(s,t) \in E} \sum_{(i,j) \in E \setminus E_{s,t}} c_{ij} + z_{s,t}$.

Let $x_j = 1$ if node j is deleted, s and t be on opposite sides of the biclique, and z_{ij} be a binary edge variable equal to 1 if and only if edge

(i, j) is deleted:

$(\mathrm{EB}^{(1)}_{s,t})$

$$
\begin{aligned}
z_{s,t} = \text{Min } & \textstyle\sum_{(i,j)\in E_{s,t}} c_{ij} z_{ij} \\
\text{subject to } & x_i \le z_{ij} && \text{for } (i,j) \in E_{s,t}, \quad i \in V_{s,t} \\
& x_i + x_j \ge 1 && (i,j) \in E_{s,t} \quad \text{and} \quad i,j \in N'(s) \\
& x_i + x_j \ge 1 && (i,j) \in E_{s,t} \quad \text{and} \quad i,j \in N'(t) \\
& x_i + x_j \ge 1 && \text{for } (i,j) \notin E_{s,t} \quad i \in N'(t),\ j \in N'(s). \\
& x_i \in \{0,1\} && \text{for } i \in V_{s,t} \\
& z_{ij} \in \{0,1\} && \text{for } (i,j) \in E_{s,t}.
\end{aligned}
$$

The first set of constraints says that all edges adjacent to a node not in the biclique are deleted. The second and third sets ensure the independence of the nodes on each side of the biclique. The fourth set of constraints ensures that the biclique contains all possible edges and is a complete bipartite graph.

This formulation has no more than two variables per inequality. A 2-approximation algorithm thus follows immediately by solving a minimum cut on a graph on $O(n + m)$ nodes, and $O(\binom{n}{2})$ edges as in [HMNT93]. The half integral optimal solution is rounded up for $z_{ij} = \frac{1}{2}$ and for $x_j = \frac{1}{2}$.

Once each instance for a pair $(s, t) \in E$ has been 2-approximated with an objective value $\bar{z}_{s,t}$, and the weight of all deleted edges in $E \setminus E_{s,t}$ is added to this value, the minimum is selected across all s, t pairs. Formally, given a 2-approximation to $\mathrm{EB}^{(1)}_{s,t}$ of value $\bar{z}_{s,t} \le 2 z_{s,t}$, the $\min_{(s,t)\in E} \sum_{(i,j)\in E\setminus E_{s,t}} c_{ij} + \bar{z}_{s,t}$ is a 2-approximation to version 1 of the edge biclique problem.

Remark 5.1. If the first set of constraints is interpreted as three variables per inequality, then the complexity of solving the problem is improved, but since each variable z_{ij} appears in two constraints, this leads to a 4-approximation algorithm.

5.2. *Formulation and Approximation of Version* 2

Here the sets $N(s)$, $N(t)$ are again potential two sides of the biclique for a given adjacent pair of nodes s, t. In this problem we permit adjacent nodes on each side of the bipartition. Therefore, unlike the case for version 1, the nodes in the set $N(s, t)$ are also candidates for inclusion in the biclique. Consequently, it is essential to make sure that only one of the two copies of such nodes is selected. Moreover, some of the edges (those that connect nodes on $N(s, t)$) are duplicated, but should not be charged for twice. There is a charge for such deleted edge only if both copies are deleted.

Let $x_j = 1$ when node j is deleted from the biclique and $z_{ij} = 1$ when edge (i, j) is deleted:

($\mathrm{EB}^{(2)}_{s,t}$)

$$
\begin{array}{lll}
z_{s,t} = \text{Min } \sum_{(i,j) \in E_{s,t} \cap (N(s) \times N(t))} c_{ij} z_{ij} & & \\
\text{subject to} \quad x_i \le z_{ij} & \text{for } (i,j) \in E_{s,t} & i \in V_{s,t} \setminus N(s,t) \\
x_v + x_{v'} - 1 \le z_{vj} & \text{for } v \in N(s,t), & j \in V_{s,t} \\
x_i + x_j \ge 1 & \text{for } (i,j) \notin E_{s,t}, & \\
 & i \in N(t),\ j \in N(s) & \\
x_i \in \{0,1\} & \text{for } i \in V_{s,t} & \\
z_{ij} \in \{0,1\} & \text{for all } i, j. &
\end{array}
$$

The first set of constraints enforces the deletion of edges adjacent to deleted nodes not in $N(s, t)$. The edges adjacent to nodes in $N(s, t)$ are deleted only if both copies of the node are deleted, as in the second set of constraints. The third set of constraints ensures the completeness of the biclique.

This formulation has up to three variables per inequality. In the next lemma we show that the formulation is "almost monotone," except that the variables z_{ij} may appear twice.

LEMMA 5.1. *The formulation* ($\mathrm{EB}^{(2)}_{s,t}$) *has monotone inequalities with respect to the x-variables, with the z-variables appearing in two constraints each.*

Proof. To show the monotonicity of the constraints, we let the variables for nodes in $N(t)$ assume values in $\{0, -1\}$ with $x_j = -1$ if node $j \in N(t)$ is deleted. The formulation now has the two variables appearing with opposite signs in every constraint:

(monotone $\mathrm{EB}^{(2)}_{s,t}$)

$$
\begin{array}{lll}
z_{s,t} = \text{Min } \sum_{(i,j) \in E_{s,t} \cap (N(s) \times N(t))} c_{ij} z_{ij} & & \\
\text{subject to} \quad x_i \le z_{ij} & \text{for } (i,j) \in E_{s,t}, & \\
 & i \in N(s) \setminus N(s,t),\ j \in N(t) & \\
-x_j \le z_{ji} & \text{for } (i,j) \in E, & \\
 & j \in N(t) \setminus N(s,t),\ i \in N(s) & \\
x_i - x_j \ge 1 & \text{for } (i,j) \notin E_{s,t}, & \\
 & i \in N(s),\ j \in N(t) & \\
-x_v + x_{v'} - 1 \le z_{vj} & \text{for } v \in N(s,t), & j \in V_{s,t} \\
x_i \in \{0,1\} & \text{for } i \in N(s) & \\
x_j \in \{-1,0\} & \text{for } j \in N(t) & \\
z_{ij} \in \{0,1\} & \text{for all } i, j. &
\end{array}
$$

This is a monotone formulation, except that each z variable appears twice: once for each endpoint constraint. ∎

To recover the monotonicity of constraints, we substitute for each edge the two occurrences of z_{ij} by $z_{ij}^{(1)}$ and $z_{ij}^{(2)}$, where $z_{ij} = z_{ij}^{(1)}$ and $z_{ji} = z_{ij}^{(2)}$. As in Corollary 2.1(i),

$$z_{ij} = \tfrac{1}{2}\left(z_{ij}^{(1)} + z_{ij}^{(2)}\right).$$

This monotone formulation is solvable optimally in integers. When the values of z_{ij} are recovered, they are in $\{0, \frac{1}{2}, 1\}$ (the values of the x-variables are integers). Rounding the values up provides a feasible solution that is at most twice the optimum.

Notice that the constructed network here is the same as in Fig. 3, except that the weights w_i are replaced by $\frac{1}{2}\sum_{(i,k)\in E} c_{ik}$. The complexity of the resulting 2-approximation algorithm is $O(m \cdot T(n, n^2))$.

6. A 2-APPROXIMATION FOR A PROBLEM EQUIVALENT TO MAXIMUM CLIQUE

The maximum clique problem is a well-known optimization problem that is notoriously hard to approximate, as shown by Håstad [Ha96]. The problem is to find in a graph $G = (V, E)$ the largest set of nodes that form a clique—a complete graph.

An equivalent statement of the clique problem is to find the complete subgraph that maximizes the number (or sum of weights) of the edges in the subgraph. There is a clique of size k if and only if there is a clique on $\binom{k}{2}$ edges.

The complement of this edge-variant of the maximum clique problem is to find a minimum weight set of edges to delete so the remaining edge-induced subgraph is a clique.

Let x_j be a variable that is 1 if node j is *not* in the clique, and 0 otherwise. Let z_{ij} be 1 if edge $(i, j) \in E$ is deleted. The first set of constraints ensures that if an edge has an endpoint not in the clique, then it must be deleted. The second set of constraints says that the set of nodes selected forms a clique by requiring that if an edge is not in the graph, then both of its endpoints cannot be in the clique:

(Clique)

$$\begin{array}{rll} \text{Min} & \sum_{(i,j)\in E} c_{ij} z_{ij} & \\ \text{subject to} & x_j \le z_{ij} & \text{for } (i,j) \in E, \quad j \in V \\ x_i + x_j \ge 1 & & \text{for } (i,j) \notin E \\ x_i,\, z_{ij} & & \text{binary for all } i, j. \end{array}$$

With this formulation each inequality has no more than two variables. Thus the problem is 2-approximable, since the results of [HMNT93] apply directly. In the network we have $2(m + n)$ nodes (one for each variable in the monotonized version) and $4m + 2\overline{m}$ edges. The resulting complexity of the 2-approximation algorithm is therefore $T(2(m + n), 4m + 2\overline{m})$, which is $O(mn^2 \log n)$.

The formulation (Clique) has, like all problems in two variables per inequality, an interpretation as 2SAT with the clauses $(\bar{x}_i, \bar{x}_j)$ for each $(i, j) \notin E$ and $(\bar{x}_i, z_{ij})$ for every node i and $(i, j) \in E$. Furthermore, it is reducible to a vertex cover problem by using the transformation described in [Hoc96a, p. 132]. The resulting bipartite (monotonized) vertex cover problem has the same number of nodes $O(m + n)$ as above. The number of edges of this vertex cover problem is quadratic in the number above, i.e., $O(n^4)$.

EPILOGUE

The original version of the paper contained a 4-approximation algorithm to the clique problem. Following a presentation of this result, I received a number of suggestions regarding the improvements of the approximation factor of the Clique problem from 4 to 2. Among these, Reuven Bar-Yehuda was the first to point this fact out by restating the problem: for each nonedge and each pair of edges adjacent to the nonedge, at least one edge of the pair must be deleted. That problem is a vertex cover problem in which the edges of E play the role of the vertices that must cover each nonedge. The set of constraints is thus

$$z_{ip} + z_{jk} \geq 1 \qquad \text{for } (i, j) \notin E, \quad (i, p), (j, k) \in E.$$

The number of variables is $m = |E|$, and the number of constraints is $\overline{m}n^2$ (for $\overline{m} = |\overline{E}|$). The running time required for the 2-approximation of this vertex cover problem is thus $O(\overline{m}n^2)$.

The SODA98 program committee provided the 2SAT interpretation, which motivated the formulation presented here. A formulation identical to ours, was proposed independently by the referee.

ACKNOWLEDGMENT

This is to express my gratitude to an anonymous referee on this paper. His insightful comments led to significant improvements in the scope and content of the results presented.

In particular, the referee pointed out an error in an earlier formulation of the general node biclique.

REFERENCES

[BYE81] R. Bar-Yehuda and S. Even, A linear time approximation algorithm for the weighted vertex cover problem, *J. Algorithms* **2** (1981), 198–203.

[DKT97] M. Dawande, P. Keskinocak, and S. Tayur, "On the Biclique Problem in Bipartite Graphs," GSIA working paper 1996-04, Carnegie-Mellon University, 1997.

[GVY94] N. Garg, V. V. Vazirani, and M. Yannakakis, Multiway cuts in directed and node weighted graphs, *in* "Proceedings of the 21st International Colloquium on Automata, Languages and Programming, 1994," pp. 487–498.

[GVY96] N. Garg, V. V. Vazirani, and M. Yannakakis, Approximate max-flow min-(multi)cut theorems and their applications, *SIAM J. Comput.* **25**(2) (1996), 235–251.

[GT88] A. V. Goldberg and R. E. Tarjan, A new approach to the maximum flow problem, *J. Assoc. Comput. Mach.* **35** (1988), 921–940.

[HO94] J. Hao and J. B. Orlin, A faster algorithm for finding the minimum cut in a graph, *J. Algorithms* **17** (1994), 424–446.

[Ha96] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, *in* "Proceedings of the 37th IEEE Symposium on Foundations of Computer Science, 1996," pp. 627–636.

[HRG96] M. R. Henzinger, S. Rao, and H. N. Gabow, Computing vertex connectivity: New bounds from old techniques, *in* "Proceedings of the 37th IEEE Symposium on Foundations of Computer Science, 1996," pp. 462–471.

[Hoc83] D. S. Hochbaum, Efficient bounds for the stable set, vertex cover and set packing problems, *Discrete Appl. Math.* **6** (1983), 243–254.

[Hoc96] D. S. Hochbaum, A framework for half integrality and 2-approximations, manuscript, UC Berkeley, 1996.

[HN94] D. S. Hochbaum and J. Naor, Simple and fast algorithms for linear and integer programs with two variables per inequality, *SIAM J. Comput.* **23**(6) (1994), 1179–1192.

[HMNT93] D. S. Hochbaum, N. Megiddo, J. Naor, and A. Tamir, Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality, *Math. Programming* **62** (1993), 69–83.

[Hoc96a] D. S. Hochbaum, Approximating covering and packing problems: Set cover, vertex cover, independent set and related problems, *in* "Approximation Algorithms for NP-Hard Problems" (D. S. Hochbaum, Ed.), PWS, Boston, 1996.

[Yan81a] M. Yannakakis, Edge deletion problems, *SIAM J. Comput.* **10** (1981), 297–309.

[Yan81b] M. Yannakakis, Node deletion problems on bipartite graphs, *SIAM J. Comput.* **10** (1981), 310–327.