Approximating the Minimum Chain Completion problem

Tomás Feder[*] Heikki Mannila[†] Evimaria Terzi[‡]

Abstract

A bipartite graph $G = (U, V, E)$ is a *chain graph* [9] if there is a bijection $\pi : \{1, \ldots, |U|\} \rightarrow U$ such that $\Gamma(\pi(1)) \supseteq \Gamma(\pi(2)) \supseteq \ldots \supseteq \Gamma(\pi(|U|))$, where Γ is a function that maps a node to its neighbors. We give approximation algorithms for two variants of the Minimum Chain Completion problem, where we are given a bipartite graph $G(U, V, E)$, and the goal is find the minimum set of edges F that need to be added to G such that the bipartite graph $G' = (U, V, E')$ ($E' = E \cup F$) is a chain graph.

Keywords: approximation algorithms, chain graphs, vertex cover, min cut, max flow

1 Introduction

A chain graph is a bipartite graph $G(U, V, E)$ where the sets of neighbors of nodes form a chain. That is, for any two nodes $u, v \in U$ (or $u, v \in V$) we have $\Gamma(u) \subseteq \Gamma(v)$ or vice versa. Here $\Gamma(u)$ denotes the set of neighbors of node u. The concept of chain graphs has been introduced by Golumbic [4] and Yannakakis [9]. Moreover, Yannakakis [9] showed that it is NP-hard to find the minimum number of edges that have to be added to an input bipartite graph to transform it into a chain graph.

The concept of chain graphs has – somewhat unexpectedly – applications in ecology. Namely, bipartite graphs can be used to describe the presence-absence relations of, say, species and islands, or foodweb interactions (predator and prey). The nestedness hypothesis of [7] states that species composition on sites with less species richness is a proper subset of those on sites with greater species richness. If this holds without exception, then the presence-absence matrix is said to be fully nested. This definition is equivalent to a chain graph. See [5] for more references on nestedness and its applications.

In this paper we consider the problem of transforming an input bipartite graph $G(U, V, E)$ to a chain graph by edge additions. There are two natural optimization measures: (a) the total number of edges in the resulting chain graph, and (b) the number of edges that have been added to the input graph in order to transform it into a chain graph. We call the corresponding optimization problems the Total Minimum Chain Completion problem (T-mcc) and the Additional Minimum Chain Completion problem (A-

[*]268 Waverley Street, Palo Alto, CA 94301. Email: tomas@theory.stanford.edu

[†]Helsinki Institute of Information Technology, University of Helsinki and Helsinki University of Technology, Finland. Email: heikki.mannila@tkk.fi

[‡]IBM ARC, San Jose, CA 95120. Email: eterzi@us.ibm.com

MCC). We first give a polynomial-time approximation algorithm for the T-MCC problem. Then, we show how we can use this algorithm as a subroutine in an approximation algorithm for the A-MCC problem.

We show that the T-MCC problem on graphs with maximum degree Δ has a polynomial-time algorithm with approximation factor $(2 - O(1/\Delta))$. We then consider the A-MCC problem. If the incremental degree (see Section 4) d of the input graph is a constant, we achieve an $O(d)$ approximation ratio in polynomial time.

Related Work: From the approximability point of view the A-MCC problem has been considered in [6]. In that paper a polynomial time approximation algorithm for the A-MCC problem is given. The algorithm achieves an approximation ratio of $8k$, where k is the cost of the optimal solution to the A-MCC problem. Notice that the approximation ratio of the algorithm proposed in [6] is not fixed but it depends on the cost of the optimal solution to the A-MCC problem. Chain graphs have also been investigated in [2]. However, the graph-modification problem arising in [2] is different from the ones studied in this paper.

The rest of this paper is organized as follows. In Section 2 we give the necessary definitions and describe the different ways of looking at the problem. Section 3 gives the basic algorithm for the T-MCC problem. Section 4 investigates that more difficult A-MCC problem. We conclude in Section 5.

2 Preliminaries

Definition 1 *A bipartite graph $G = (U, V, E)$ is a* chain graph *([9]) if there is a bijection $\pi : \{1, \ldots, |U|\} \rightarrow U$ (an* ordering *of U) such that $\Gamma(\pi(1)) \supseteq \Gamma(\pi(2)) \supseteq \ldots \supseteq \Gamma(\pi(|U|))$, where Γ is a function that maps a node to its neighbors.*

An equivalent definition of a chain graph is the following.

Definition 2 *A bipartite graph $G(U, V, E)$ is a chain graph if and only if it does not contain a pair of independent edges. In other words, a chain graph is $2K_2$ free.*

For a fixed permutation of the nodes in U and V, π_U and π_V respectively, we may represent $G(U, V, E)$ on the 2-dimensional plane so that node $u \in U$ with rank $\pi_U(u)$ corresponds to the interval $\big[(0, \pi_U(u) - 1), (0, \pi_U(u))\big]$ on the y-axis. Similarly, node $v \in V$ with rank $\pi_V(v)$ corresponds to interval $\big[(\pi_V(v) - 1, 0), (\pi_V(v), 0)\big]$ on the x-axis. Every possible edge (u, v) corresponds to a unit square defined by the points $(\pi_V(v) - 1, \pi_U(u) - 1)$, $(\pi_V(v) - 1, \pi_U(u))$, $(\pi_V(v), \pi_U(u))$, $(\pi_V(v), \pi_U(u) - 1)$. Visually, existing edges in $G(U, V, E)$ will correspond to shaded unit squares and absent edges to white unit squares. We call this representation of G a grid representation. We call this 2-dimensional representation of the bipartite graph G as M_G. We will often resort to this representation when explaining our algorithm. We can use the G and M_G representations interchangeably to represent graph G. In fact, M_G can also be represented as a matrix, where the square with its top right corner being at point (i, j) is represented by $M_G(i, j)$ and $M_G(i, j) = 1$ if the corresponding square is shaded and $M_G(i, j) = 0$ otherwise. We additionally use $M_G[u, :]$ to denote the u-th row of M_G and $M_G[:, v]$ to denote the v-th column of M_G.

Definition 3 *A 0–1 matrix M is* nested *if for any two rows i and j we have $M[i,:] \subseteq M[j,:]$ or $M[j,:] \subseteq M[i,:]$.*

Problem 1 (Minimum Chain Completion (Mcc)) *Given a bipartite graph $G(U,V,E)$, find the* minimum cardinality *set of edges F that need to be added to G such that the bipartite graph $G' = (U,V,E')$, where $E' = E \cup F$, is a chain graph.*

In terms of the 0–1 matrix M_G Problem 1 can be rephrased as the *optimal* number of 0-entries that need to be transformed into 1's so that the M_G matrix becomes nested. Different cost functions define optimization problems with different approximation properties. In this paper we discuss the following two.

Problem 2 (Total Minimum Chain Completion (T-mcc)) *Given a bipartite graph $G(U,V,E)$, find the minimum set of edges F that need to be added to G such that the bipartite graph $G' = (U,V,E')$, where $E' = E \cup F$, is a chain graph. The value of the solution is $|E'|$.*

Problem 3 (Additional Minimum Chain Completion (A-mcc)) *Given a bipartite graph $G(U,V,E)$, find the minimum set of edges F that need to be added to G such that the bipartite graph $G' = (U,V,E')$, where $E' = E \cup F$, is a chain graph. The value of the solution is $|F|$.*

Since the A-mcc problem is NP-hard ([9]) so is the T-mcc problem; it is easy to see that the optimal solution of the two problems are identical. However, the two problems have different approximation properties. Since our algorithms will connect the Mcc problem with vertex covers of bipartite graphs we give the following definition that will prove useful.

Definition 4 *Consider bipartite graph $G(U,V,E)$ and a sequence of vertex covers of G, $C_1 = (U_1,V_1),\ldots,C_k = (U_k,V_k)$, with $U_i \subseteq U$ and $V_i \subseteq V$. We say that $C_1,\ldots,C_k$ are* nested sequence *of vertex covers if $U_{i+1} \subseteq U_i$ and $V_{i+1} \supseteq V_i$ for all $1 \leq i < k$.*

3 Approximating the T-mcc problem

In this section we give a constant factor polynomial time approximation algorithm for the Total Minimum Chain Completion problem. We have the following result.

Theorem 1 *The* Total Minimum Chain Completion *problem on bipartite graphs of maximum degree Δ has a polynomial-time algorithm that achieves a $2 - O(1/\Delta)$ approximation ratio.*

The key idea of our algorithm is in defining a permutation π_u of the nodes in U and a permutation π_v of the nodes in V, such that when the rows in M_G are rearranged according to π_u and the columns according to π_v, then in this permuted version of M_G we only need to convert 0's to 1's if $M_G[i,j] = 0$ and there exists an $i' > i$ such that $M_G[i',j] = 1$. Therefore, the problem is in finding good permutations π_u and π_v. Could we find the best such permutations, our algorithm would optimal for T-MCC problem. However, we are only able to find *bucket orders* of the nodes rather than total orders[1].

We construct the bucket orders b_U and b_V, on the nodes of U and V, respectively, by constructing a sequence of consecutive and nested vertex covers of the input graph G. Let there be k such vertex covers $(U_1, V_1), \ldots, (U_k, V_k)$ such that for every $1 \leq i < k$ it holds that $U_i \supseteq U_{i+1}$ and $V_i \subseteq V_{i+1}$. Then, in bucket order b_U the i-th bucket consists of the nodes $U_i \setminus U_{i+1}$ and in bucket order b_V the i-th bucket contains the nodes $V_{i+1} \setminus V_i$. Our algorithm arranges the rows and the columns of M_G according to the bucket orders b_U and b_V and then converts 0's to 1's so that the rearranged M_G becomes nested.

Vertex covers of bipartite graphs: Consider a vertex cover (U', V') of the input bipartite graph $G\,(U, V, E)$, where $U' \subseteq U$ and $V' \subseteq V$. If we permute the nodes in U such that the nodes in U' are in the beginning of the ordering of U and also permute the nodes in V such that all nodes in V' are in the beginning of the ordering of V, then the grid representation of G will be as in Figure 1(a). Since (U', V') is a vertex cover, all the edges in G have at least one of their endpoints either in U' or in V'.

Therefore, knowing about vertex cover (U', V'), we know that there do not exist edges in the rectangle defined by lines $x = |V'|$, $y = |U'|$, $x = |V|$ and $y = |U|$. Let this rectangle be A'.

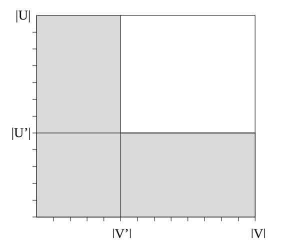
Now assume another vertex cover (U'', V'') of G and let vertex covers (U', V') and (U'', V'') be nested, so that $U'' \subseteq U'$ and $V' \subseteq V''$. If we rearrange the entries in U' so that the first $|U''|$ elements are the ones in U'' and elements in V'' so that the first $|V'|$ elements are the ones in V', then for this rearrangement the grid representation of G will look as in Figure 1(b). Due to (U', V') there are no shaded squares in the rectangle defined by the lines $x = |V''|$, $y = |U''|$, $x = |V|$ and $y = |U|$. Let this rectangle be A''.

Then, the combined knowledge of the nested vertex covers (U', V') and (U'', V'') allows us to conclude that all the unit squares in area defined by the union of A' and A'' will be white, and thus the corresponding edges do not exist in G. This is illustrated in Figure 1(c).

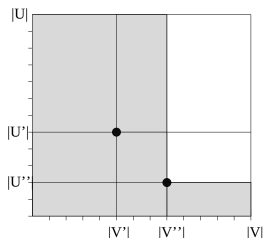
Vertex covers and min-cuts: Computing the vertex covers of bipartite graphs is central for our algorithm. The minimum vertex cover of the input graph can be computed using flow techniques. Given an input bipartite graph $G(U, V, E)$ construct the *extended graph* $\mathcal{H}_\alpha\,(x \cup U, y \cup V, E \cup E_H)$. Graph $\mathcal{H}_\alpha$ is directed. The set of directed edges E_H consists of the following edges: The edges $(u, v) \in E$ become directed edges $(u \rightarrow v)$. These edges are assigned weight ∞. The edges $(x \rightarrow u)$ for all $u \in U$. These edges are assigned weight α. Finally, the edges $(v \rightarrow y)$ for all $v \in V$. Those edges are assigned weight $1 - \alpha$. The following proposition is a straightforward result of this construction.

Proposition 1 *Let $U' \subseteq U$ and $V' \subseteq V$. Additionally, let $X = \{x\} \cup (U \setminus U') \cup V'$ and $Y = \{y\} \cup (V \setminus V') \cup U'$. We have that (U', V') is a vertex cover of $G\,(U, V, E)$ if and only if (X, Y) is a cut of finite capacity in the extended graph $\mathcal{H}_\alpha$, for fixed $\alpha \in [0, 1]$.*

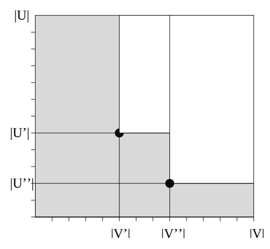
[1]Recall that in a bucket order there is a specific ordering of the buckets but not of the nodes in the same bucket. That is, every node in the i-th bucket is ordered before any node in the j-th bucket, for $j > i$, but there is no ordering for the nodes that belong in the same bucket.



(a) Arrangement of possible edges and definite non-edges G given vertex cover (U', V')



(b) Arrangement of possible edges and definite non-edges in G given two nested vertex covers (U', V') and (U'', V'')



(c) Arrangement of possible edges and definite non-edges in G given two nested vertex covers (U', V') and (U'', V'').

Figure 1: Arrangement of possible edges (shaded squares) and definite non-edges (white squares) in M_G, given (nested) vertex covers.

Notice that graph $\mathcal{H}_\alpha$ has a single parameter α. The following lemma is an immediate consequence of Lemma 2.4 in [3].

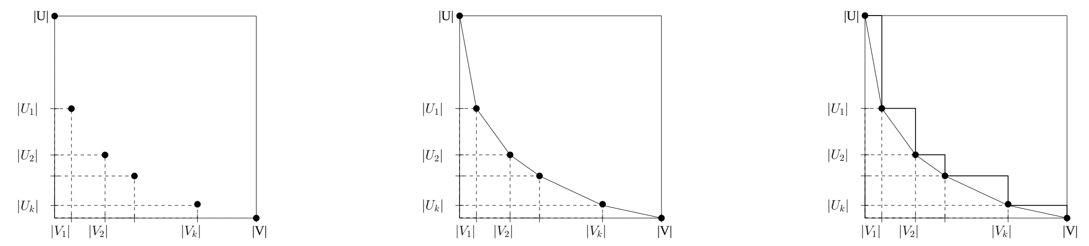
Lemma 1 *[Nested sequence of vertex covers] Let $G(U, V, E)$ be a bipartite graph and $\alpha_1, \ldots, \alpha_k$ a sequence of values for parameter α, such that $\alpha_1 < \alpha_2 < \ldots < \alpha_k$. Assume the min-cut algorithm from Proposition 1 that for each value α_i computes vertex cover (U_i, V_i). Then, this sequence of vertex covers is nested, that is, $U_i \supseteq U_{i+1}$ and $V_{i+1} \supseteq V_i$ for all $1 \leq i < k$.*

Remark: Even when the values of α_i's are not known in advance, but we know that for every i, $\alpha_i \in [0, 1]$, then the results of [3] allow us to compute a sequence of at most $n - 1$ nested cuts in polynomial time. Let $f(\alpha)$ be the function that gives the capacity of the minimum cut as a function of α. Due to Proposition 1 this is also the cost of the minimum vertex cover in $G(U, V, E)$. Additionally, the capacities of the edges in $\mathcal{H}_\alpha$ satisfy conditions (i)–(iii) of Section 2.3 in [3]. Therefore, $f(\alpha)$ is a linear concave function. That is, if $n = |U| + |V|$, then $f(\alpha)$ has at most $n - 2$ breakpoints, see [1, 3, 8][2]. The $n - 1$ or fewer line segments forming the graph of $f(\alpha)$ correspond to $n - 1$ distinct cuts (and thus distinct vertex covers). Using the results of [3] we can compute these breakpoints in polynomial time.

We can now prove Theorem 1. Using the results of [3] and Proposition 1, for all values of $\alpha \in [0, 1]$ we can produce a linear number of vertex covers $(U_1, V_1), \ldots, (U_k, V_k)$ with $U_{i+1} \subseteq U_i$ and $V_i \subseteq V_{i+1}$. By Lemma 1 we know that this sequence of vertex covers is a nested sequence. We may rearrange the nodes in U and in V so that all nodes in U_i are grouped together and the first $|U_{i+1}|$ nodes of the group correspond to the nodes in U_{i+1}. Similarly, the nodes in V are rearranged so that all nodes in V_{i+1} are grouped together and the first $|V_i|$ nodes of the group are the nodes in V_i. Due to this rearrangement, the marked points in Figure 2(a) correspond to the vertex covers of G.

Proposition 2 *Define a curve $C_\triangle$ by joining $(|V_i|, |U_i|)$ to $(|V_{i+1}|, |U_{i+1}|)$ with a straight line for all i. This curve is convex up, and no vertex cover (U', V') has $(|U'|, |V'|)$ strictly underneath the curve.*

[2]By a breakpoint we mean a value of α at which the slope of $f(\alpha)$ changes.



(a) Vertex covers of G (b) Curve $C_{\triangle}$ (c) Curve $C_{\square}$

Figure 2: Figure 2(a): A sequence of optimal nested vertex covers for bipartite graph $G(U, V, E)$. Figure 2(b): Curve $C_{\triangle}$ obtained by connecting successive vertex-cover points by lines. Figure 2(c): Curve $C_{\square}$ obtained by constructing the rectangles in which the lines forming $C_{\triangle}$ are used as diagonals.

An example of such a curve is shown in Figure 2(b). The following lemma relates the area underneath this curve with the cost of the optimal solution with respect to $|E'|$.

Lemma 2 *If $A^{\triangle}$ is the area underneath curve $C_{\triangle}$, and A^* is the cost of the optimal solution with respect to the total 1's appearing in the nested matrix, then $A^{\triangle} \leq A^*$.*

Proof. The problem is equivalent to finding orderings of U and V so that the total number of 1's in the area is A^*. Such an ordering defines a sequence of vertex covers $\left(U_j^*, V_j^*\right)$ for $j = 1, \ldots, n^* \leq n$. If $A^* \leq A^{\triangle}$, then at least one of the points $\left(\left|U_j^*\right|, \left|V_j^*\right|\right)$ would have to be below $C_{\triangle}$, contradicting Proposition 2. $\square$

Now assume the curve $C_{\square}$ defined by joining $(|V_i|, |U_i|)$ to $(|V_{i+1}|, |U_i|)$ and then $(|V_{i+1}|, |U_{i+1}|)$, as in Figure 2(c). We have the following.

Lemma 3 *If $A^{\square}$ is the area underneath curve $C_{\square}$, then $A^{\square} \leq 2A^{\triangle}$.*

Proof. Compared to $C_{\triangle}$, the area underneath $C_{\square}$ has extra triangles $(|V_i|, |U_i|)$, $(|V_{i+1}|, |U_i|)$, $(|V_{i+1}|, |U_{i+1}|)$ (see Figure 2(c)). These triangles have the same area as the corresponding triangles $(|V_i|, |U_i|)$, $(|V_i|, |U_{i+1}|)$, $(|V_{i+1}|, |U_{i+1}|)$. But these latter triangles are already included in $C_{\triangle}$. Thus $A^{\square} \leq 2A^{\triangle}$. $\square$

The preceding analysis suggests the following 2-approximation algorithm for the T-MCC problem:

1. For input bipartite graph $G(U, V, E)$ that is not a chain graph construct the extended graph $\mathcal{H}_\alpha$.

2. Apply the parametric min-cut max-flow algorithm of [3] on the extended graph $\mathcal{H}_\alpha$ and obtain a sequence $(V_1, U_1), \ldots, (V_k, U_k)$ of nested vertex covers of G, for $k \leq n - 1$.

3. Permute the nodes of V so that for every $1 \leq i \leq k$ the nodes in V_i are grouped together in the beginning of the ordering of V. Similarly, permute the nodes of U so that for every $1 \leq i \leq k$ the nodes in U_i are grouped together in the beginning of the ordering of U.

4. For this permutations of the nodes of U and V, construct matrix M_G. Convert to 1 every entry $M_G[i, j] = 0$ for which there exists an $i' > i$ such that $M_G[i', j] = 1$.

Saving a factor of $1/\Delta$: An additional $O(1/\Delta)$ term can be saved as follows. Let $y_i = |U_i| - |U_{i+1}|$ and $x_i = |V_{i+1}| - |V_i|$, and assume $x_i \geq y_i$ (the case $x_i \leq y_i$ is similar). The line that moves x_i forward and y_i down delineating A' can be replaced with a line that repeatedly advances some distance x' and then goes down distance 1. The two sets $U_i \setminus U_{i+1}$ and $V_{i+1} \setminus V_i$ have sizes y_i and x_i respectively. Lets assume that the first set has a vertex of degree Δ. We may start from point $(|V_i|, |U_i|)$ and proceed Δ units to the right and one unit down, to obtain another vertex cover, by substituting the vertex of degree at most Δ for its neighbors. This reduces x_i by Δ and y_i by 1. We repeat this operation until we get to the opposite side of the rectangle at point $(|V_{i+1}|, |U_{i+1}|)$. In that way we save area of size at least $\ell x_i - (\ell(\ell-1)/2)\,\Delta$ where ℓ is the number of times the operation is repeated. For $\ell = x_i/\Delta$ we have that the total saving is $x_i^2/(2\Delta) + x_i/2$. Therefore, the total area A''' covered by this improved algorithm is $A''' \leq x_i y_i - x_i^2/(2\Delta) \leq x_i y_i - x_i y_i/(2\Delta)$. Therefore, the approximation factor of this improved algorithm is $\frac{A'''}{A} \leq \frac{1}{2}\left(1 - \frac{1}{2\Delta}\right)$.

Corollary 1 *In the special case where the input bipartite graph $G(U, V, E)$ is a forest, the* T-MCC *problem can be approximated in polynomial time within a factor of* $1 + (\sqrt{2}-1)^2 < 1.1716$.

Proof. (SKETCH) As before let $r = |U|$ and $s = |V|$ and let $r \geq s$. The total number of edges in the forest is at most $r+s-1$. Let x out of the r vertices of U have degree at least 2, and the remaining $r-x$ have degree 1. Then the total number of edges is $2x + (r-x) = r + x \leq r + s - 1$, or $r - x \geq r - s + 1$. Thus, at least one of the s vertices is adjacent to at least $k = \lceil (r-s+1)/s \rceil = \lfloor r/s \rfloor$ vertices of degree 1 (leaves). Thus, if we move k units to the right and one unit down from the upper left corner, we find another vertex cover. More generally, if $r = ks + \ell$ with $0 \leq \ell < s$ we can proceed $s - \ell$ times with k and ℓ times with $(k+1)$ units to the right. The resulting area underneath this curve is $r/2 + (s-\ell)(k+1)\ell + \ell(k+1)\ell/2 + (s-\ell)k(s-\ell)/2 = r/2 + rs/2 + \ell(s-\ell)/2$. The ratio $\ell(s-\ell)/(rs)$ is maximized at $k = 1, \ell = (\sqrt{2}-1)s$, where it equals $(\sqrt{2}-1)^2 < 0.1716$. □

4 Approximating the A-MCC problem

In this section we give a polynomial time factor $O(d)$ approximation algorithm for the A-MCC problem. Here d is the *incremental* degree of the input bipartite graph G. The incremental degree of a graph G is the least d such that every subgraph of G has a vertex of degree at most d; forests are the graphs G of incremental degree 1.

Let $|E|$ be the *total* number of edges in the input bipartite graph, $|E'|$ the total number of edges in the optimal solution of Problem 3 and $|E''|$ the total number of edges in the chain graph created by an approximation algorithm to the A-MCC problem. Then, if there exist $\alpha > 1, \beta > 1$ such that $|E| = \alpha(|E'| - |E|)$ and $|E''| = \beta|E'|$, then we have that the A-MCC problem can be approximated with ratio

$$\frac{(|E''| - |E|)}{(|E'| - |E|)} = \beta + \alpha(\beta - 1). \tag{1}$$

We can now prove the following approximation result for the A-MCC problem.

Theorem 2 *Consider bipartite graphs $G(U,V,E)$ with constant incremental degree d. The optimal solution to the* A-MCC *problem can be approximated in polynomial time within $8d+2$.*

Proof. Let $r = |U|$ and $s = |V|$, and $r \geq s$. Now consider the following algorithm: Consider all possible permutations of nodes in U that have degree at most $r/4$ and nodes in V that have degree at least $r/2$. For each such permutation of vertices, start removing those vertices from the original graph in the order given by this permutation. Stop the vertex-removal process and call the 2-approximation algorithm of Theorem 1 on the remaining graph when one of the following two cases occur: (a) A vertex with degree less than (or equal to) $r/4$ is removed while a vertex with degree greater than (or equal to) $r/2$ still remains. (b) The remaining graph does not have $r/4$ isolated vertices and at the same time there is no remaining vertex of degree at least $r/2$. Note that the case where there are $r/4$ isolated vertices simply goes into the recursion and these vertices are simply removed.

Consider case (a) above: if we remove a vertex of degree at most $r/4$, while a vertex of degree at least $r/2$ remains, then we must add at least $r/4 \geq n/8$ edges[3]. Since $|E| \leq nd$ we have that $|E'| - |E| \geq n/8 \geq |E|/(8d)$ and therefore $|E| \geq 8d(|E'| - |E|)$. Since in the last step we call the 2-approximation algorithm of Theorem 1 we also have that $|E''| = 2|E'|$. Using Equation (1) for $\alpha = 8d$ and $\beta = 2$ we get approximation factor $8d+2$.

Consider now case (b): we may assume that while removing vertices of degree at least $r/2$ we do not remove any vertex of degree at most $r/4 \geq n/8$. In a graph there are at most $8d$ vertices with degree at least $r/2$. We thus remove at most $8d$ vertices. When we are done, the next vertex to be removed will be of degree less than $r/2$, so unless we already have $r/4$ isolated vertices we will add $r/4 \geq n/8$ edges as before. By calling the 2-approximation algorithm of Theorem 1 we get the $8d+2$ factor as before.
Running Time: The algorithm thus reduces to producing all possible sequences of at most $8d$ vertices of degree at least $r/4$, say $f(d)$ such sequences, and then reducing r to $3r/4$. The recurrence gives $g(rs) = f(d)g(3rs/4)$, which is $n^{O(\log f(d))}$, a polynomial running time. $\square$
As for the T-MCC problem, we have the following result for the approximability of the A-MCC problem for input graphs that are forests.

Corollary 2 *If the input bipartite graph $G(U,V,E)$ is a forest (having incremental degree $d=1$), the optimal solution to the* A-MCC *problem can be approximated in polynomial time within factor $1+5(\sqrt{2}-1)^2 + O(1/|E|) < 1.8579$.*

Proof. (SKETCH) Let $r = |U|$ and $s = |V|$, and suppose $r \geq s$. The set V contains at most one vertex of degree at least $1 + r/2$. If this vertex is not removed first, then the vertex removed will add at least $r/2 - 1 \geq |E|/4 - 1$ edges, giving the result by Theorem 2 with $\alpha = 4 + O(1/m)$ and $\beta = 1 + (\sqrt{2}-1)^2$. The remaining case removes first the vertex of degree $1 + r/2$ and proceeds inductively after removing the isolated vertices. $\square$

[3]this is because $n \leq 2r$

5 Concluding remarks

We have studied the approximation properties of the Minimum Chain Completion problem. More specifically, we showed that when we want to minimize the total number of edges in the output chain graph there is a polynomial time factor $2 - O(1/\Delta)$-approximation algorithm, where Δ is the maximum degree of the graph. For the case where the goal is to minimize just the additional number of edges added to the input graph so that it becomes a chain graph, we gave an polynomial time $O(d)$-approximation algorithm, where d is the incremental degree of the input graph. The approximation algorithm for the A-mcc problem uses as a subroutine the approximation algorithm for the T-mcc problem.

Several problems remain open. For example, the bipartite complement of a chain graph is also a chain bipartite graph. Thus the problem that asks for the minimum number of edges to remove from a bipartite graph so that it becomes a chain graph is as hard as the A-mcc problem. However, this problem seems harder to approximate, just as independent set is harder to approximate than vertex cover. Only the case of very dense graphs seems easier, just as sparse graphs were easier for the A-mcc problem. The case where simultaneous additions and deletions are allowed in the input graph and the problem is to make the minimum number of edge modifications in an input bipartite graph so that a chain graph is a subject of further study.

References

[1] M. J. Eisner and D. G. Severance. Mathematical techniques for efficient record segmentation in large shared databases. *J. ACM*, 23(4):619–635, 1976.

[2] D. Fasulo, T. Jiang, R. M. Karp, and N. Sharma. Constructing maps using the span and inclusion relations. In *RECOMB*, pages 64–73, 1998.

[3] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, 1989.

[4] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[5] H. Mannila and E. Terzi. Nestedness and segmented nestedness. In *KDD*, pages 480–489, 2007.

[6] A. Natanzon, R. Shamir, and R. Sharan. A polynomial approximation algorithm for the minimum fill-in problem. *SIAM J. Comput.*, 30(4):1067–1079, 2000.

[7] B. Patterson and W. Atmar. Nested subsets and the structure of insular mammalian faunas and archipelagos. *Biological Journal of the Linnean Society*, 28(1-2), 1986.

[8] H. S. Stone. Critical load factors in two-processor distributed systems. *IEEE Trans. Softw. Eng.*, 4(3):254–258, 1978.

[9] M. Yannakakis. Computing the Minimum Fill-In is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79, 1981.