

# Better Approximation Algorithms for NMR Spectral Peak Assignment

Zhi-Zhong Chen<sup>1</sup> \*, Tao Jiang<sup>2</sup> \*\*, Guohui Lin<sup>3</sup> \*\*\*, Jianjun Wen<sup>2</sup> †, Dong Xu<sup>4</sup> ‡, and Ying Xu<sup>4</sup> ‡

<sup>1</sup> Dept. of Math. Sci., Tokyo Denki Univ., Hatoyama, Saitama 350-0394, Japan.

<sup>2</sup> Dept. of Comput. Sci., Univ. of California, Riverside, CA 92521.

<sup>3</sup> Dept. of Comput. Sci., Univ. of Alberta, Edmonton, Alberta T6G 2E8, Canada.

<sup>4</sup> Protein Informatics Group, Oak Ridge National Lab., Oak Ridge, TN 37831-6480.

**Abstract.** We study a constrained bipartite matching problem where the input is a weighted bipartite graph  $G = (U, V, E)$ ,  $U$  is a set of vertices following a sequential order,  $V$  is another set of vertices partitioned into a collection of disjoint subsets, each following a sequential order, and  $E$  is a set of edges between  $U$  and  $V$  with non-negative weights. The objective is to find a matching in  $G$  with the maximum weight that satisfies the given sequential orders on both  $U$  and  $V$ , *i.e.* if  $u_{i+1}$  follows  $u_i$  in  $U$  and if  $v_{j+1}$  follows  $v_j$  in  $V$ , then  $u_i$  is matched with  $v_j$  if and only if  $u_{i+1}$  is matched with  $v_{j+1}$ . The problem has recently been formulated as a crucial step in an algorithmic approach for interpreting NMR spectral data [13]. The interpretation of NMR spectral data is known as a key problem in protein structure determination via NMR spectroscopy. Unfortunately, the constrained bipartite matching problem is NP-hard [13]. We first propose a 2-approximation algorithm for the problem, which follows directly from the recent result of Bar-Noy *et al.* [2] on interval scheduling. However, our extensive experimental results on real NMR spectral data illustrate that the algorithm performs poorly in terms of recovering the target-matching (*i.e.* correct) edges. We then propose another approximation algorithm that tries to take advantage of the “density” of the sequential order information in  $V$ . Although we are only able to prove an approximation ratio of  $3 \log_2 D$  for this algorithm, where  $D$  is the length of a longest string in  $V$ , the experimental results demonstrate that this new algorithm performs much better on real data, *i.e.* it is able to recover a large fraction of the target-matching

---

\* Supported in part by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan, under Grant No. 12780241. Email: chen@dendai.ac.jp. Part of the work done while visiting at UC Riverside.

\*\* Supported in part by a UCR startup grant and NSF Grants CCR-9988353 and ITR-0085910. Email: jiang@cs.ucr.edu.

\*\*\* Supported in part by NSERC grants RGPIN249633 and A008599, and Startup Grant REE-P5-01-02-Sci from the University of Alberta. Email: ghlin@cs.ualberta.ca.

† Supported by NSF Grant CCR-9988353. Email: wjianju@cs.ucr.edu.

‡ Supported by the Office of Biological and Environmental Research, U.S. Department of Energy, under Contract DE-AC05-00OR22725, managed by UT-Battelle, LLC. Email: xud,xyn@ornl.gov.

edges and the weight of its output matching is often in fact close to the maximum. We also prove that the problem is MAX SNP-hard, even if the input bipartite graph is unweighted. We further present an approximation algorithm for a nontrivial special case that breaks the ratio 2 barrier.

## 1 Introduction

The Human Genome Project [1] has led to the identification of a vast majority of protein-encoding genes in the human genome. To facilitate a systematic study of the biological functions of these proteins, the US National Institutes of Health (NIH) has recently launched another ambitious project, the Structural Genomics Project [9]. Its main goal is to solve about 100,000 protein structures within the next ten years, through the development and application of significantly improved experimental and computational technologies. Along with *X-ray crystallography*, *nuclear magnetic resonance* (NMR) spectroscopy has been one of the two main experimental methods for solving protein structures. Among the seven pilot Structural Genomics Centers set up by NIH, one center is devoted to protein structure determination via NMR.

Protein structure determination via NMR generally involves the following three key steps:

- NMR spectral data generation, which produces
  - resonance peaks corresponding to amino acids in the target protein sequence. Peaks corresponding to a common amino acid are grouped into a *spin system*;
  - certain geometric relationships (*e.g.* distances and angles) between the spin systems;
- NMR data interpretation, which involves relating the spin systems to the amino acids in the target protein sequence, providing both inter- and intra-amino acid distance and angle information;
- NMR structure calculation, which calculates the target protein structure through molecular dynamics (MD) and energy minimization (EM) under the constraints of the identified geometric relationships.

It typically takes several months to a year to solve a single protein structure by NMR, and a major part of that time is used for NMR data interpretation. Up until very recently, NMR data interpretation has been done mainly using manual procedures. Though a number of computer programs [4,6,7,12,14] have recently been developed to assist the data interpretation, most NMR labs are still doing the peak assignments manually or semi-manually for quality reasons. With the recent progress in NMR technologies for speeding up the data production rate, we expect that NMR data interpretation will soon become the sole bottleneck in a high-throughput NMR structure determination process.

Two key pieces of information form the foundation of NMR peak assignment:

- Each amino acid has a somewhat “unique” spin system <sup>1</sup>;
- The sequential adjacency information between spin systems in a protein sequence is often inferable from the spectral data. However, this type of information is generally incomplete, *i.e.* we may often be able to obtain the adjacency relationship between some of the spin systems but not all.

In a recently developed computational framework [13], the NMR peak assignment problem has been formulated as a constrained bipartite matching problem. In this framework, each amino acid (also called *residue*) is represented as a vertex of  $U$  and each spin system is represented as a vertex of  $V$  (and thus generally  $|U| = |V|$ ). A pair  $(u_i, v_j) \in U \times V$  of vertices that represents a potential assignment has a non-negative weight  $w_{i,j} = w(u_i, v_j)$ , which scores the preference of assigning spin system  $v_j$  to amino acid  $u_i$ . Let  $E$  denote the set of all potential assignments. Clearly  $G = (U, V, E \subseteq U \times V)$  is a bipartite graph. In general, the edges in  $E$  have different weights and  $G$  is said *weighted*. In the special case that the edges have equal weight,  $G$  is said *unweighted*. For more detailed information about the weighting scheme, we refer the reader to [13]. The MAXIMUM WEIGHT BIPARTITE MATCHING [5] provides a natural framework for the study of the NMR peak assignment problem. Nonetheless, some resonance peaks from a single NMR experiment are known to belong to atoms from consecutive amino acids and thus their host spin systems should be mapped to consecutive amino acids. Such spin systems that should be mapped consecutively are said to be *adjacent* and their corresponding vertices in  $V$  are required to follow a sequential order. For convenience, we number the amino acids consecutively in the order that they appear in the protein sequence, and number the spin systems in such a way that adjacent spin systems have consecutive indices. In this formulation, a *feasible* matching  $M$  in  $G$  is one such that if  $v_j$  and  $v_{j+1}$  are sequentially adjacent, then edge  $(u_i, v_j) \in M$  if and only if edge  $(u_{i+1}, v_{j+1}) \in M$ . The CONSTRAINED BIPARTITE MATCHING (CBM) problem is to find a feasible matching in  $G$  achieving the maximum weight.

We call a maximal set of vertices in  $V$  that are consecutively adjacent a *string*. Thus, the set  $V$  is partitioned into a collection of strings. The CBM problem in which the maximum length of strings in  $V$  is  $D$  is called the  $D$ -STRING CBM problem. Without loss of generality, assume  $D > 1$ . In the practice of NMR peak assignment,  $D$  is usually between 4 and 10. One may notice that the standard MAXIMUM WEIGHT BIPARTITE MATCHING problem is simply the 1-STRING CBM problem, and it is known to be solvable in polynomial time [5]. Unfortunately, the  $D$ -STRING CBM problem is intractable even when it is unweighted and  $D = 2$ .

**Theorem 1.** [13] *The unweighted 2-STRING CBM is NP-hard.*

---

<sup>1</sup> This information alone is not sufficient for a correct assignment since a protein sequence typically contains multiple copies of the same amino acid. Additional information is necessary in order to tell if a particular spin system corresponds to, for example, an Alanine at a particular sequence position.

A *two-layer* algorithm for  $D$ -STRING CBM has been proposed in [13] that attempts to fix likely assignments and filter out unlikely assignments for *long* strings (*i.e.* at least 3 spin systems) in the first layer of computation. In the second layer, it tries *all* possible combinations of assignments for long strings and extends them to perfect matchings (recall that  $|U| = |V|$ ) by exhaustive enumeration. A perfect matching with the maximum weight generated in this way is output as the result. The current implementation of the algorithm runs efficiently for cases where the number of long strings is relatively small and most of the long strings consist of at least 4 or 5 spin systems. Its running time goes up quickly (*i.e.* exponentially) when the instance has many strings consisting of 2 or 3 spin systems.

In this paper, we first propose a simple 2-approximation algorithm for  $D$ -STRING CBM that directly follows from the recent result of Bar-Noy *et al.* [2] on interval scheduling. However, our experimental results on 126 instances of NMR spectral data derived from 14 proteins illustrate that the algorithm performs poorly in terms of recovering the *target-matching edges* (*i.e.* matching edges that assign spin systems to their correct amino acids).<sup>2</sup> One explanation is that the algorithm looks for matching edges by scanning  $U$  from left to right, hence giving preference to edges close to the beginning of  $U$ . Consequently, it may miss many target-matching edges. We thus propose a second approximation algorithm that attempts to take advantage of the “density” of the spin system adjacency information in  $V$ . Although we are only able to prove an approximation ratio of  $3 \log_2 D$  for this algorithm, the experimental results demonstrate that this new algorithm performs much better than the 2-approximation algorithm on real data. In fact, it often recovers as many target-matching edges as the two-layer algorithm [13] (of exhaustive search nature) and the weight of its output matching is often close to the maximum. We then prove that unweighted 2-STRING CBM is MAX SNP-hard, implying that the problem has no polynomial-time approximation scheme (PTAS) unless  $P = NP$ . The proof extends to all constants  $D \geq 2$ . Although ratio 2 seems to be a barrier to polynomial-time approximation algorithms for  $D$ -STRING CBM, we show that this barrier can be broken for unweighted 2-STRING CBM, by presenting a  $\frac{5}{3}$ -approximation algorithm. We remark that unweighted  $D$ -STRING CBM could be interesting because it is simpler and is useful in NMR peak assignment when the edge weights fall into a small range. Moreover, since long strings in  $V$  are usually associated with good quality spectral data, algorithms that attempt to solve unweighted  $D$ -STRING CBM could yield reasonably good NMR peak assignment since they tend to favor long strings. We expect that the techniques developed in this work, in conjunction with the work of [13], will lead to a significantly improved capability for NMR data interpretation, providing a highly effective tool for high-throughput protein structure determination.

The paper is organized as follows. Section 2 describes the 2-approximation and the  $3 \log_2 D$ -approximation algorithms for  $D$ -STRING CBM, and compares

---

<sup>2</sup> Note that, target-matching edges are known in the simulations from BioMagRes-Bank [11] and they were used to generate the adjacency data.

their performances (as well as that of the two-layer algorithm) on 126 real NMR spectral data derived from 14 proteins. It also gives a proof of the MAX SNP-hardness of unweighted 2-STRING CBM. Section 3 presents an improved approximation algorithm for unweighted 2-STRING CBM. Section 4 concludes the paper with some future research directions.

## 2 Weighted Constrained Bipartite Matching

We first present two approximation algorithms for  $D$ -STRING CBM. Consider an instance of  $D$ -STRING CBM:  $G = (U, V, E)$ , where  $U = \{u_1, u_2, \dots, u_{n_1}\}$ ,  $V = \{v_1 \cdots v_{i_1}, v_{i_1+1} \cdots v_{i_2}, \dots, v_{i_p} \cdots v_{n_2}\}$ , and  $E \subseteq U \times V$  is the set of edges. Here,  $v_{i_{j-1}+1} \cdots v_{i_j}$  in  $V$  denotes a string of consecutively adjacent spin systems. We may assume that for every substring  $v_j v_{j+1}$  of a string in  $V$ ,  $(u_i, v_j) \in E$  if and only if  $(u_{i+1}, v_{j+1}) \in E$ , because otherwise  $(u_i, v_j)$  cannot be in any feasible matching and thus can be eliminated without further consideration. Based on  $G = (U, V, E)$ , we construct a new edge-weighted bipartite graph  $G' = (U, V, E')$  as follows: For each  $u_i \in U$  and each string  $v_j v_{j+1} \cdots v_k \in V$  such that  $(u_i, v_j) \in E$ , let  $(u_i, v_j)$  be an edge in  $E'$  and its weight be the total weight of edges  $\{(u_{i+x}, v_{j+x}) \mid 0 \leq x \leq k - j\}$  in  $E$ . For convenience, we call the subset  $\{(u_{i+x}, v_{j+x}) \mid 0 \leq x \leq k - j\}$  of  $E$  the *expanded matching* of edge  $(u_i, v_j)$  of  $E'$ .

We say that two edges of  $E'$  are *conflicting* if the union of their expanded matchings is not a feasible matching in  $G$ . Note that a set of non-conflicting edges in  $E'$  is always a matching in  $G'$  but the reverse is not necessarily true. A matching in  $G'$  is *feasible* if it consists of non-conflicting edges. There is an obvious one-to-one correspondence between feasible matchings in  $G$  and feasible matchings in  $G'$ . Namely, the feasible matching  $M$  in  $G$  corresponding to a feasible matching  $M'$  in  $G'$  is the union of the expanded matchings of edges in  $M'$ . Note that the weight of  $M$  in  $G$  is the same as that of  $M'$  in  $G'$ . Thus, it remains to show how to compute a feasible approximate matching in  $G'$ .

Define an *innermost edge* of  $G'$  to be an edge  $(u_i, v_j)$  in  $G'$  satisfying the following condition:

- $G'$  has no edge  $(u_{i'}, v_{j'})$  other than  $(u_i, v_j)$  such that  $i \leq i' \leq i' + s' - 1 \leq i + s - 1$ , where  $s$  (respectively,  $s'$ ) is the size of the expanded matching of  $(u_i, v_j)$  (respectively,  $(u_{i'}, v_{j'})$ ).

Note that for every  $u_i \in U$ ,  $G'$  has at most one innermost edge incident to  $u_i$  (*i.e.*, there cannot exist  $v_{j_1} \in V$  and  $v_{j_2} \in V$  with  $j_1 \neq j_2$  such that both  $(u_i, v_{j_1})$  and  $(u_i, v_{j_2})$  are innermost edges of  $G'$ ). Define a *leading innermost edge* of  $G'$  to be an innermost edge  $(u_i, v_j)$  such that index  $i$  is minimized. The crucial point is that for every leading innermost edge  $(u_i, v_j)$  of  $G'$  and every feasible matching  $M'$  in  $G'$ , at most two edges of  $M'$  conflict with  $(u_i, v_j)$ . To see this, let  $(u_{i'}, v_{j'})$  be an edge in  $M'$  that conflicts with  $(u_i, v_j)$ . Let  $s$  (respectively,  $s'$ ) be the size of the expanded matching of  $(u_i, v_j)$  (respectively,  $(u_{i'}, v_{j'})$ ). Since  $(u_i, v_j)$  is an innermost edge of  $G'$ , at least one of the following conditions holds:

1.  $j' = j$ .
2.  $i' \leq i \leq i + s - 1 \leq i' + s' - 1$ .
3.  $i < i' \leq i + s - 1 < i' + s' - 1$ .
4.  $i' < i \leq i' + s' - 1 < i + s - 1$ .

For each of these conditions,  $M'$  contains at most one edge  $(u_{i'}, v_{j'})$  satisfying the condition because  $M'$  is a feasible matching in  $G'$ . Moreover, if  $M'$  contains an edge  $(u_{i'}, v_{j'})$  satisfying Condition 2, then it contains no edge satisfying Condition 3 or 4. Furthermore,  $M'$  contains no edge  $(u_{i'}, v_{j'})$  satisfying Condition 4 or else there would be an innermost edge  $(u_{i''}, v_{j''})$  in  $G'$  with  $i' \leq i'' < i \leq i'' + s'' - 1 \leq i' + s' - 1$  (where  $s''$  is the size of the expanded matching of  $(u_{i''}, v_{j''})$ ), contradicting the assumption that  $(u_i, v_j)$  is a leading innermost edge in  $G'$ . Thus, at most two edges of  $M'$  conflict with  $(u_i, v_j)$ .

Using the above fact (that at most two edges of  $M'$  conflict with a leading innermost edge) and the *local ratio* technique in [3], we can construct a recursive algorithm to find a (heavy) feasible matching in  $G'$  as shown in Figure 1. The algorithm in fact, as we were informed very recently, follows directly from the recent result of Bar-Noy *et al.* [2] on interval scheduling.

2-APPROXIMATION on  $G'$ :

1. **if**  $(E(G') = \emptyset)$   
output the empty set and halt;
2. find a leading innermost edge  $e$  in  $G'$ ;
3.  $\Gamma = \{e\} \cup \{e' \mid e' \in E(G'), e' \text{ conflicts with } e\}$ ;
4. find the minimum weight  $c$  of an edge of  $\Gamma$  in  $G'$ ;
5. **for** (every edge  $f \in \Gamma$ )  
subtract  $c$  from the weight of  $f$ ;
6.  $F = \{e \mid e \in \Gamma, e \text{ has weight } 0\}$ ;
7.  $G'' = G' - F$ ;
8. recursively call 2-APPROXIMATION on  $G''$  and output  $M'_1$ ;
9. find a maximal  $M'_2 \subseteq F$  such that  $M'_1 \cup M'_2$  is a feasible matching in  $G'$ ;
10. output  $M'_1 \cup M'_2$  and halt.

**Fig. 1.** A recursive algorithm for finding a feasible matching in  $G'$ .

**Theorem 2.** [2] *The algorithm described in Figure 1 outputs a feasible matching of the graph  $G' = (U, V, E')$  with weight at least half of the optimum.*

We have implemented the algorithm and tested it on a set of 14 proteins from BioMagResBank [11]. For each protein, we randomly generated 9 instances of spin-system adjacency by adding links (getting from the correct adjacency from BioMagResBank) between neighboring spin systems. If the spin systems are connected by the links, they will map to the sequence as a string together. We increased the number of links from 10% of the sequence length to 90% of the sequence length. In other words, the algorithm was tested on 126 bipartite

graphs with positive edge weights and adjacency constraints. The test results are summarized in Table 1 (Columns 5–7). In the tests, the *target* assignments are matchings consisting of edges of form  $(u_i, v_i)$  that assign spin systems to correct amino acids from BioMagResBank. Although these target assignments do not always have the maximum weights, their weights are not far from the maxima. As can be seen from the table, although the algorithm did very well in terms of maximizing the weight of its output matching, it recovered very few target-matching edges and is thus almost useless in practice. A possible explanation of the poor performance of the algorithm in this experiment is that the algorithm looks for edges by scanning amino acids in  $U$  from left to right, hence giving preference to edges close to the beginning of  $U$ . As a consequence, it may miss many target-matching edges. Another reason of the poor performance is probably due to the scoring function that was used. The goal of the scoring function is to force the correct assignment to have the maximum score. However, given the statistical nature of the scoring function, this goal can not be achieved completely currently. That is why even the “two-layer” algorithm [13] (briefly described in the Introduction section. For the detail description of the algorithm, please refer to [13].) recovers small numbers of correct assignments (Table 1, Column 4) in many cases, although as the number of links between adjacent spin systems increases, the performance improves. The development of the scoring function, which we are working on, will not be addressed in this paper. As the scoring function improves, the correct assignment should get closer to the maximum score, especially when the number of links between adjacent spin systems is large.

In trying to improve the performance on recovering the target-matching edges, we next present a second approximation algorithm that tries to take advantage of the presence of many long strings in the instance, as described in Figure 2. Basically, the algorithm partitions the strings in  $V$  into groups of strings of approximately the same length, greedily finds a maximal feasible matching in each group, and then greedily extends the matching to a maximal feasible matching in  $G'$ . It outputs the heaviest one among the matchings found for all groups.

**Theorem 3.** *The algorithm described in Figure 2 outputs a feasible matching in  $G'$  with weight at least  $\frac{1}{3 \max\{1, \log_2 r\}}$  of the maximum weight in  $\tilde{O}(|U||V|)$  (i.e. quadratic up to a poly-logarithmic factor) time, where  $r$  is as defined in Figure 2. It is thus an approximation algorithm for  $D$ -STRING CBM with performance ratio  $3 \log_2 D$ .*

*Proof.* For each  $i \in \{1, 2, \dots, g\}$ , consider the bipartite graph  $G'_i = (U, V_i, E_i)$ . Let  $M_i^*$  denote an optimal feasible matching for graph  $G'_i$ . Right before the execution of Step 3.4 of the algorithm,  $M'_i$  is clearly a feasible matching for graph  $G'_i$ , and its weight is at least  $\frac{1}{6}$  of that of  $M_i^*$  because we can claim that each execution of Step 3.3.2 only rules out at most 6 edges of  $M_i^*$  from further consideration. To see the claim, consider an edge  $e = (u_x, v_y)$  added to  $M'_i$  in Step 3.3.2. Let  $e' = (u_{x'}, v_{y'})$  be an edge conflicting with  $e$ . Let  $s$  (respectively,





$3 \log_2 D$ -APPROXIMATION on  $G'$ :

1. compute ratio  $r = \frac{\ell_{max}}{\ell_{min}}$ , where  $\ell_{max}$  (respectively,  $\ell_{min}$ ) is the maximum (respectively, minimum) length of strings in  $V$ ;
2. partition  $V$  into  $g = \max\{1, \log_4 r\}$  subsets  $V_1, V_2, \dots, V_g$  such that a string  $s$  is included in subset  $V_i$  if and only if  $4^{i-1} \leq \frac{|s|}{\ell_{min}} \leq 4^i$ ;  
(Note:  $V_{i-1}$  and  $V_i$  may not be disjoint.)
3. **for** (every  $i \in \{1, 2, \dots, g\}$ )
  - 3.1 compute the set  $E_i$  of edges of  $G'$  incident to strings in  $V_i$ ;
  - 3.2 initialize  $M'_i = \emptyset$ ;
  - 3.3 **while** ( $E_i \neq \emptyset$ )
    - 3.3.1 find an edge  $e \in E_i$  of maximum weight;
    - 3.3.2 add  $e$  to  $M'_i$ , and delete  $e$  and all edges conflicting with  $e$  from  $E_i$ ;
  - 3.4 greedily extend  $M'_i$  to a maximal feasible matching of  $G'$ ;
4. output the heaviest one among  $M'_1, M'_2, \dots, M'_g$  and halt.

**Fig. 2.** A new algorithm for finding a feasible matching in  $G'$ .

$s'$ ) be the size of the expanded matching of  $e$  (respectively,  $e'$ ). Then, at least one of the following conditions 1 through 6 holds:

1.  $y' = y$ .
2.  $x' = x$  and  $s' = s$ .
3.  $x' < x \leq x' + s' - 1 < x + s - 1$ .
4.  $x < x' \leq x + s - 1 < x' + s' - 1$ .
5.  $x' < x \leq x + s - 1 \leq x' + s' - 1$  or  $x' \leq x \leq x + s - 1 < x' + s' - 1$ .
6.  $x < x' \leq x' + s' - 1 \leq x + s - 1$  or  $x \leq x' \leq x' + s' - 1 < x + s - 1$ .

Since  $M_i^*$  is a feasible matching of  $G'_i$ ,  $M_i^*$  may contain at most one edge satisfying Condition 1, at most one edge satisfying Condition 2, at most one edge satisfying Condition 3, at most one edge satisfying Condition 4, at most one edge satisfying Condition 5, and at most four edges satisfying Condition 6 (because of the construction of  $V_i$ ). Due to the same reason, if  $M_i^*$  contains an edge satisfying Condition 2 (respectively, 5), then  $M_i^*$  contains no edge satisfying Condition 6. Similarly, if  $M_i^*$  contains an edge satisfying Condition 3 or 4, then  $M_i^*$  contains at most three edges satisfying Condition 6 (because of the construction of  $V_i$ ). So, in the worse case (where  $M_i^*$  contains the largest number of edges conflicting with  $e$ ),  $M_i^*$  may contain one edge satisfying Condition 1, one edge satisfying Condition 3, one edge satisfying Condition 4, and three edges satisfying Condition 6. This proves the claim.

Let  $M'$  denote the output matching of the algorithm. Let  $\bar{M}^*$  denote an optimal feasible matching for graph  $G'$ , and  $\bar{M}_i^*$  be the sub-matching of  $\bar{M}^*$  in edge set  $E_i$ . Suppose without loss of generality that  $\bar{M}_j^*$  is the heaviest one among  $\bar{M}_1^*, \bar{M}_2^*, \dots, \bar{M}_g^*$ . Clearly, we have  $w(\bar{M}_j^*) \geq \frac{1}{g}w(\bar{M}^*)$ . Thus,  $w(M') \geq \frac{1}{6}w(M_j^*) \geq \frac{1}{6g}w(\bar{M}^*)$ . The time complexity analysis is straightforward.  $\square$

The above  $3 \log_2 D$ -approximation has been implemented and tested on the same set of 126 instances of NMR peak assignment. The test results are also summarized in Table 1 (Columns 8–10). It is quite clear that this algorithm is much more superior to the 2-approximation algorithm both in terms of maximizing the weight of the output matching and in terms of maximizing the number of target-matching edges recovered. In fact, on over half of the instances (more precisely, 65 out of the 126 instances), the  $3 \log_2 D$ -approximation algorithm recovered at least as many target-matching edges as the (exhaustive) two-layer algorithm. Because the  $3 \log_2 D$ -approximation algorithm is much more efficient than the two-layer algorithm, it will be very useful in NMR peak assignment.

Observe that the (feasible) matchings found by the approximation algorithms have weights greater than that of the target assignments on quite a few instances, especially when the adjacency information is sparse. This implies that the weighting scheme as formulated in [13] may not work very well when the adjacency information is sparse, and more work on weighting scheme is needed in the future.

A natural question is if  $D$ -STRING CBM admits a  $\rho$ -approximation algorithm for some constant  $\rho < 2$ . Our next theorem shows that there is a constant  $\rho > 1$  such that  $D$ -STRING CBM does not admit a  $\rho$ -approximation algorithm for every  $D \geq 2$ , unless  $P = NP$ , even if the input bipartite graph is unweighted.

**Theorem 4.** *For all  $D \geq 2$ , unweighted  $D$ -STRING CBM is MAX SNP-hard.*

*Proof.* We prove the theorem for  $D = 2$  by a simple  $L$ -reduction from MAXIMUM 3-DIMENSIONAL MATCHING (3DM), which is known to be MAX SNP-complete [8]. The proof can be easily extended to any constant  $D \geq 2$ .

MAXIMUM BOUNDED 3-DIMENSIONAL MATCHING (MB3DM): Given a universal set  $\mathcal{U} = \{1, 2, \dots, m\}$  and a collection of subsets  $S_1, S_2, \dots, S_n$ , where  $S_i \subseteq \mathcal{U}$ ,  $|S_i| = 3$ , and every element  $u \in \mathcal{U}$  is contained in at most 3 subsets, find a largest subcollection of pairwise disjoint subsets.

Given an instance of MB3DM, without loss of generality, suppose that  $m = 3q$  and  $n \geq q$ . Observe that  $n \leq m$ , because every element of  $\mathcal{U}$  appears in at most 3 subsets. For each subset  $S_i$ , construct 7 vertices  $a_{i,1}, a_{i,2}, \dots, a_{i,7}$  in set  $U$  and for each element  $i \in \mathcal{U}$  construct a 2-vertex string  $b_{i,1}b_{i,2}$  in set  $V$ . We will also have in  $V$   $q$  1-vertex strings  $f_1, f_2, \dots, f_q$  and  $3n$  2-vertex strings  $c_{1,1}c_{1,2}, c_{1,3}c_{1,4}, c_{1,5}c_{1,6}, \dots, c_{n,1}c_{n,2}, c_{n,3}c_{n,4}, c_{n,5}c_{n,6}$ . Finally, for every  $i = 1, 2, \dots, m$ , we connect string  $b_{i,1}b_{i,2}$  to  $a_{j,2k}a_{j,2k+1}$  (*i.e.* connect vertex  $b_{i,1}$  to vertex  $a_{j,2k}$  and vertex  $b_{i,2}$  to vertex  $a_{j,2k+1}$ ), for each  $1 \leq k \leq 3$ , if  $i \in S_j$ ; for every  $i = 1, 2, \dots, q$  and every  $j = 1, 2, \dots, n$ , connect string  $f_i$  to  $a_{j,1}$ ; and for every  $i = 1, 2, \dots, n$  and every  $j = 1, 2, \dots, n$ , connect string  $c_{i,2k-1}c_{i,2k}$  to  $a_{j,2k-1}a_{j,2k}$ , for each  $1 \leq k \leq 3$ . All the edges have the unit weight. This forms an instance of unweighted 2-STRING CBM:  $G = (U, V, E)$ , where  $|U| = 7n$ ,  $|V| = 7q + 6n$ .

We claim that the above construction is an  $L$ -reduction [10] from MB3DM to unweighted 2-STRING CBM. It is straightforward to see that each subcollection of  $p$  (where  $p \leq q$ ) disjoint subsets implies a constrained matching in  $G$  of weight

$7p + 6(n - p) = 6n + p$ . To complete the proof of the claim, we only need to observe that, for any given constrained matching in the above bipartite graph, we can always rearrange it without decreasing the weight so that each group of vertices  $a_{i,1}, a_{i,2}, \dots, a_{i,7}$  are matched either with three  $c$ -type strings or with a combination of one  $f$ -type string and three  $b$ -type strings, due to the special construction of the edges. This completes the  $L$ -reduction.  $\square$

### 3 Unweighted Constrained Bipartite Matching

As noted in the last section, a natural question is to ask if  $D$ -STRING CBM admits an approximation algorithm with ratio less than 2. In this section, we answer the question affirmatively for a special case, namely, unweighted 2-STRING CBM. More specifically, we will give a  $\frac{5}{3}$ -approximation algorithm for unweighted 2-STRING CBM.

Consider an instance of unweighted  $D$ -STRING CBM:  $G = (U, V, E)$ , where  $U = \{u_1, u_2, \dots, u_{n_1}\}$ ,  $V = \{v_1, v_2, \dots, v_k, v_{k+1}v_{k+2}, v_{k+3}v_{k+4}, \dots, v_{k+2\ell-1}v_{k+2\ell}\}$ , and  $E \subseteq U \times V$  is the set of edges in  $G$ . Here,  $v_j$  alone forms a 1-string in  $V$  for each  $1 \leq j \leq k$ , while  $v_{k+2j-1}v_{k+2j}$  is a 2-string in  $V$  for each  $1 \leq j \leq \ell$ . Note that  $k \geq 0$  and  $\ell \geq 0$ . Let  $n_2 = k + 2\ell$  and  $m = |E|$ . We may assume that for every  $u_i \in U$  and every 2-string  $v_{k+2j-1}v_{k+2j}$  in  $V$ ,  $(u_i, v_{k+2j-1}) \in E$  if and only if  $(u_{i+1}, v_{k+2j}) \in E$ , because otherwise  $(u_i, v_{k+2j-1})$  or  $(u_i, v_{k+2j})$  cannot be in any feasible matching and thus can be eliminated without further consideration.

Fix a maximum size feasible matching  $M^*$  in  $G$ . Let  $m_1^*$  be the number of edges  $(u_i, v_j) \in M^*$  with  $1 \leq j \leq k$ . Similarly, let  $m_2^*$  be the number of edges  $(u_i, v_{k+2j-1}) \in M^*$  with  $1 \leq j \leq \ell$ . Then,  $|M^*| = m_1^* + 2m_2^*$ .

**Lemma 1.** *A feasible matching in  $G$  can be found in  $O(m\sqrt{n_1 n_2})$  time, whose size is at least  $m_1^* + m_2^*$ .*

*Proof.* Construct a new bipartite graph  $G' = (U, V, E_1 \cup E_2)$ , where  $E_1 = \{(u_i, v_j) \in E \mid 1 \leq j \leq k\}$  and  $E_2 = \{(u_i, v_{k+2j-1}) \in E \mid 1 \leq j \leq \ell\}$ . Let  $M'$  be a maximum matching in  $G'$ . Obviously, we can obtain a matching in  $G'$  from  $M^*$  by deleting all edges  $(u_i, v_{k+2j}) \in M^*$  with  $1 \leq i \leq n_1$  and  $1 \leq j \leq \ell$ . So,  $|M'| \geq m_1^* + m_2^*$ . To obtain a feasible matching  $M$  of  $G$  from  $M'$ , we perform the following steps:

1. Initialize  $M = \emptyset$ .
2. Construct an auxiliary graph  $H$  as follows. The vertex set of  $H$  is  $M'$ . The edge set of  $H$  consists of all  $(e_1, e_2)$  such that  $e_1 \in M'$ ,  $e_2 \in M'$ , and  $e_1$  conflicts with  $e_2$ . [*Comment:* Each connected component of  $H$  is a path  $P$  (possibly consisting of a single vertex); if  $P$  contains two or more vertices, then there exist integers  $i, j_1, \dots, j_h$  ( $h \geq 2$ ) such that the vertices of  $P$  are  $(u_i, v_{j_1}), (u_{i+1}, v_{j_2}), \dots, (u_{i+h-1}, v_{j_h})$ , and each of  $v_{j_1}$  through  $v_{j_{h-1}}$  is the leading vertex of a 2-string in  $V$ .]
3. For each connected component of  $H$  formed by only one vertex  $(u_i, v_j) \in M'$ , if  $v_j$  is a 1-string in  $V$ , then add edge  $(u_i, v_j)$  to  $M$ ; otherwise, add edges  $(u_i, v_j)$  and  $(u_{i+1}, v_{j+1})$  to  $M$ .

4. For each connected component  $P$  of  $H$  formed by two or more vertices, perform the following three substeps:
  - (a) Let the vertices of  $P$  be  $(u_i, v_{j_1}), (u_{i+1}, v_{j_2}), \dots, (u_{i+h-1}, v_{j_h}) \in M'$ .
  - (b) If  $h$  is even or  $v_{j_h}$  is the leading vertex of a 2-string in  $V$ , then for each  $1 \leq x \leq \lceil \frac{h}{2} \rceil$ , add edges  $(u_{i+2x-2}, v_{j_{2x-1}})$  and  $(u_{i+2x-1}, v_{j_{2x-1}+1})$  to  $M$ .
  - (c) If  $h$  is odd and  $v_{j_h}$  alone forms a 1-string in  $V$ , then for each  $1 \leq x \leq \frac{h-1}{2}$ , add edges  $(u_{i+2x-2}, v_{j_{2x-1}})$  and  $(u_{i+2x-1}, v_{j_{2x-1}+1})$  to  $M$ ; further add edge  $(u_{i+h-1}, v_{j_h})$  to  $M$ .

It is clear that for each connected component  $P$  of  $H$ , we add at least as many edges to  $M$  as the number of vertices in  $P$ . Thus,  $|M| \geq |M'| \geq m_1^* + m_2^*$ .  
□

**Lemma 2.** *A feasible matching in  $G$  can be found in  $O(m\sqrt{n_1 n_2})$  time, whose size is at least  $\frac{m_1^*}{3} + \frac{4m_2^*}{3}$ .*

*Proof.* For each index  $i \in \{0, 1, 2\}$ , let  $G_i$  be the edge-weighted bipartite graph obtained from  $G$  as follows:

1. For every 2-string  $v_j v_{j+1}$  in  $V$ , merge the two vertices in the string into a single super-vertex  $s_{j,j+1}$  (with all resulting multiple edges deleted).
2. For all  $j$  such that  $i + 1 \leq j \leq n_1 - 2$  and  $j - 1 \equiv i \pmod{3}$ , perform the following three substeps:
  - (a) Merge  $u_j, u_{j+1}$ , and  $u_{j+2}$  into a single super-vertex  $t_{j,j+1,j+2}$  (with all resulting multiple edges deleted).
  - (b) For every 1-string  $v_h$  that is a neighbor of  $t_{j,j+1,j+2}$ , if edge  $\{u_{j+1}, v_h\}$  is not in the original input graph, then delete the edge between  $t_{j,j+1,j+2}$  and  $v_h$ ; otherwise, assign a weight of 1 to the edge between  $t_{j,j+1,j+2}$  and  $v_h$ .
  - (c) For every 2-string  $v_h v_{h+1}$  such that  $s_{h,h+1}$  is a neighbor of  $t_{j,j+1,j+2}$ , if neither  $\{(u_j, v_h), (u_{j+1}, v_{h+1})\}$  nor  $\{(u_{j+1}, v_h), (u_{j+2}, v_{h+1})\}$  is a matching in the original input graph, then delete the edge between  $t_{j,j+1,j+2}$  and  $s_{h,h+1}$ ; otherwise, assign a weight of 2 to the edge between  $t_{j,j+1,j+2}$  and  $s_{h,h+1}$ .
3. If neither  $u_1$  nor  $u_2$  was merged in Step 2a, then perform the following three substeps:
  - (a) Merge  $u_1$  and  $u_2$  into a single super-vertex  $t_{1,2}$  (with all resulting multiple edges deleted).
  - (b) For every 1-string  $v_h$  that is a neighbor of  $t_{1,2}$ , if edge  $\{u_1, v_h\}$  is not in the original input graph, then delete the edge between  $t_{1,2}$  and  $v_h$ ; otherwise, assign a weight of 1 to the edge between  $t_{1,2}$  and  $v_h$ .
  - (c) For every 2-string  $v_h v_{h+1}$  such that  $s_{h,h+1}$  is a neighbor of  $t_{1,2}$ , if  $\{(u_1, v_h), (u_2, v_{h+1})\}$  is not a matching in the original input graph, then delete the edge between  $t_{1,2}$  and  $s_{h,h+1}$ ; otherwise, assign a weight of 2 to the edge between  $t_{1,2}$  and  $s_{h,h+1}$ .
4. If neither  $u_{n_1-1}$  nor  $u_{n_1}$  was merged in Step 2a, then perform the following three substeps:

- (a) Merge  $u_{n_1-1}$  and  $u_{n_1}$  into a single super-vertex  $t_{n_1-1, n_1}$  (with all resulting multiple edges deleted).
- (b) For every 1-string  $v_h$  that is a neighbor of  $t_{n_1-1, n_1}$ , if edge  $\{u_{n_1}, v_h\}$  is not in the original input graph, then delete the edge between  $t_{n_1-1, n_1}$  and  $v_h$ ; otherwise, assign a weight of 1 to the edge between  $t_{n_1-1, n_1}$  and  $v_h$ .
- (c) For every 2-string  $v_h v_{h+1}$  such that  $s_{h, h+1}$  is a neighbor of  $t_{n_1-1, n_1}$ , if  $\{(u_{n_1-1}, v_h), (u_{n_1}, v_{h+1})\}$  is not a matching in the original input graph, then delete the edge between  $t_{n_1-1, n_1}$  and  $s_{h, h+1}$ ; otherwise, assign a weight of 2 to the edge between  $t_{n_1-1, n_1}$  and  $s_{h, h+1}$ .

For each  $i \in \{0, 1, 2\}$ , let  $M_i$  be a maximum-weighted matching in  $G_i$ . From each  $M_i$ , we can obtain a feasible matching  $\bar{M}_i$  in the original input graph by performing the following steps in turn:

- Initialize  $\bar{M}_i = \emptyset$ .
- For each edge  $(u_j, v_h) \in M_i$ , add  $(u_j, v_h)$  to  $\bar{M}_i$ .
- For each edge  $(t_{j, j+1, j+2}, v_h) \in M_i$ , add  $(u_{j+1}, v_h)$  to  $\bar{M}_i$ .
- For each edge  $(t_{1, 2}, v_h) \in M_i$ , add  $(u_1, v_h)$  to  $\bar{M}_i$ .
- For each edge  $(t_{n_1-1, n_1}, v_h) \in M_i$ , add  $(u_{n_1}, v_h)$  to  $\bar{M}_i$ .
- For each edge  $(t_{j, j+1, j+2}, s_{h, h+1}) \in M_i$ , if  $\{(u_j, v_h), (u_{j+1}, v_{h+1})\}$  is a matching in the original input graph, then add edges  $(u_j, v_h)$  and  $(u_{j+1}, v_{h+1})$  to  $\bar{M}_i$ ; otherwise, add edges  $(u_{j+1}, v_h)$  and  $(u_{j+2}, v_{h+1})$  to  $\bar{M}_i$ .
- For each edge  $(t_{1, 2}, s_{h, h+1}) \in M_i$ , add edges  $(u_1, v_h)$  and  $(u_2, v_{h+1})$  to  $\bar{M}_i$ .
- For each edge  $(t_{n_1-1, n_1}, s_{h, h+1}) \in M_i$ , add  $(u_{n_1-1}, v_h)$  and  $(u_{n_1}, v_{h+1})$  to  $\bar{M}_i$ .

Note that the total weight of edges in  $M_i$  is exactly  $|\bar{M}_i|$ . Let  $\bar{M}$  be the maximum size one among  $\bar{M}_0, \bar{M}_1, \bar{M}_2$ . We claim that  $|\bar{M}| \geq \frac{m_1^*}{3} + \frac{4m_2^*}{3}$ . To see this, for each  $i \in \{0, 1, 2\}$ , let  $M_i^*$  be the union of the set  $\{(u_j, v_h) \in M^* \mid j+1 \equiv i \pmod{3} \text{ and } v_h \text{ is a 1-string in } V\}$  and the set  $\{(u_j, v_h), (u_{j+1}, v_{h+1}) \in M^* \mid u_j \text{ and } u_{j+1} \text{ belong to the same super-vertex in } G_i \text{ and } v_h v_{h+1} \text{ is a 2-string in } V\}$ . It holds that  $\sum_{i=0}^2 |M_i^*| = m_1^* + 4m_2^*$ , because each edge in  $M^*$  incident to a 1-string belongs to exactly one of  $M_0^*, M_1^*, M_2^*$  while each edge in  $M^*$  incident to a 2-string belongs to exactly two of  $M_0^*, M_1^*, M_2^*$ . This implies that the maximum size one among  $M_0^*, M_1^*, M_2^*$  has size at least  $\frac{m_1^*}{3} + \frac{4m_2^*}{3}$ . On the other hand, for each  $i \in \{0, 1, 2\}$ , we can obtain a matching  $\tilde{M}_i$  in  $G_i$  by modifying  $M_i^*$  as follows:

- For each edge  $(u_j, v_h) \in M_i^*$  such that  $v_h$  is a 1-string in  $V$ , replace  $u_j$  by the super-vertex of  $G_i$  to which  $u_j$  belongs.
- For each pair of edges  $(u_j, v_h), (u_{j+1}, v_{h+1}) \in M_i^*$  such that  $v_h v_{h+1}$  is a 2-string in  $V$ , replace the two edges by a single edge between  $s_{h, h+1}$  and the super-vertex in  $G_i$  to which both  $u_j$  and  $u_{j+1}$  belong.

The total weight of edges in  $\tilde{M}_i$  is exactly  $|M_i^*|$ . On the other hand, the total weight of edges in  $M_i$  is larger than or equal to that of edges in  $\tilde{M}_i$ , because  $M_i$

is a maximum-weighted matching in  $G_i$ . Thus, the total weight of edges in  $M_i$  is at least  $|M_i^*|$ . In turn,  $|\bar{M}_i| \geq |M_i^*|$ . Therefore,

$$|\bar{M}| \geq \frac{1}{3} \sum_{i=0}^2 |\bar{M}_i| \geq \frac{1}{3} \sum_{i=0}^2 |M_i^*| = \frac{1}{3}(m_1^* + 4m_2^*).$$

This completes the proof of the claim and hence that of the lemma.  $\square$

Combining Lemmas 1 and 2, we now have:

**Theorem 5.** *We can compute a feasible matching in  $G$  whose size is at least  $\frac{2}{5}|M^*|$ , in  $O(m\sqrt{n_1n_2})$  time. Consequently, there is a  $\frac{5}{3}$ -approximation algorithm for the unweighted 2-STRING CBM problem; it runs in  $O(m\sqrt{n_1n_2})$  time.*

## 4 Concluding Remarks

It would be interesting to test if the  $\frac{5}{3}$ -approximation algorithm works well in practice. An obvious open question is if  $D$ -STRING CBM admits a  $\rho$ -approximation algorithm for some constant  $\rho < 2$ .

In the real NMR spectral peak assignment, we want to assign every amino acid a spin system. Therefore, the desired output matchings are perfect matchings. So far our theoretical analysis on the approximation algorithms does not involve this requirement, although during the implementation we did put priority on perfect matchings. Designing algorithms that guarantee to output perfect matchings (given that the real data is a complete weighted bipartite graph) is a practical consideration. On the other hand, not putting the perfect requirement on approximation algorithms could be one of the reasons that they produce heavier matchings than the correct assignments.

## References

1. International Human Genome Sequencing Consortium. Initial Sequencing and Analysis of the Human Genome. *Nature*, 409:860-921, 2001.
2. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48:1069–1090, 2001.
3. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.
4. C. Bartels, P. Güntert, M. Billeter, and K. Wüthrich. GARANT-A general algorithm for resonance assignment of multidimensional nuclear magnetic resonance spectra. *Journal of Computational Chemistry*, 18:139–149, 1996.
5. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.
6. P. Güntert, M. Salzmann, D. Braun, and K. Wüthrich. Sequence-specific NMR assignment of proteins by global fragment mapping with the program mapper. *Journal of Biomolecular NMR*, 18:129–137, 2000.

7. K. Huang, M. Andrec, S. Heald, P. Blake, and J.H. Prestegard. Performance of a neural-network-based determination of amino acid class and secondary structure from  $^1\text{H}$ - $^{15}\text{N}$  NMR data. *Journal of Biomolecular NMR*, 10:45–52, 1997.
8. V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27–35, 1991.
9. National Institute of General Medical Sciences. Pilot projects for the protein structure initiative (structural genomics). June 1999. Web page at ‘‘<http://www.nih.gov/grants/guide/rfa-files/RFA-GM-99-009.html>’’.
10. C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.
11. University of Wisconsin. BioMagResBank. ‘‘<http://www.bmrwisc.edu>’’ . 2001.
12. J. Xu, S.K. Straus, B.C. Sanctuary, and L. Trimble. Use of fuzzy mathematics for complete automated assignment of peptide  $^1\text{H}$  2D NMR spectra. *Journal of Magnetic Resonance*, 103:53–58, 1994.
13. Y. Xu, D. Xu, D. Kim, V. Olman, J. Razumovskaya, and T. Jiang. Automated assignment of backbone NMR peaks using constrained bipartite matching. *IEEE Computing in Science & Engineering*, 4:50–62, 2002.
14. D.E. Zimmerman, C.A. Kulikowski, Y. Huang, W.F.M. Tashiro, S. Shimotakahara, C. Chien, R. Powers, and G.T. Montelione. Automated analysis of protein NMR assignments using methods from artificial intelligence. *Journal of Molecular Biology*, 269:592–610, 1997.