Approximation algorithms for NMR spectral peak assignment

Zhi-Zhong Chen\textsuperscript{a,1}, Tao Jiang\textsuperscript{b,2}, Guohui Lin\textsuperscript{c,*,3}, Jianjun Wen\textsuperscript{d,4}, Dong Xu\textsuperscript{e,5}, Jinbo Xu\textsuperscript{f,6}, Ying Xu\textsuperscript{e,5}

\textsuperscript{a}Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan
\textsuperscript{b}Department of Computer Science, University of California, Riverside, CA 92521, USA
\textsuperscript{c}Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada
\textsuperscript{d}Department of Computer Science, University of California, Riverside, CA 92521, USA
\textsuperscript{e}Protein Informatics Group, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6480, USA
\textsuperscript{f}Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

Received 2 October 2001; received in revised form 11 February 2002; accepted 12 February 2002
Communicated by O.H. Ibarra

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 (IEEE Comput. Sci. Eng. 4 (2002) 50–62). 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 (IEEE Comput. Sci. Eng. 4 (2002) 50–62). We first propose a 2-approximation algorithm for the problem, which follows directly from the recent result of Bar-Noy et al. (Proc. 32nd ACM Symp. on Theory of Computing (STOC’00), 2000, pp. 735–744) on interval scheduling. However, our extensive experimental results on real NMR spectral data illustrate that the algorithm performs poorly in terms of recovering target-matching 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 target-matching 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.

**Keywords:** Computational biology; Approximation algorithm; Protein structure determination; NMR spectroscopy

1. Introduction

The Human Genome Project [10] 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 National Institutes of Health (NIH) has recently launched another ambitious project, the Structural Genomics Project [12]. 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 grouped into spin systems,
  - 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 [3,7,9,15,17] 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,\(^ \text{7} \)
- 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 [16], 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 [16]. The MAXIMUM WEIGHT BIPARTITE MATCHING [6] 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 \( (u_i, v_j) \in M \) iff 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. 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

\(^{7}\) 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.
standard Maximum Weight Bipartite Matching problem is simply the 1-String CBM problem, and it is known to be solvable in polynomial time [6]. Unfortunately, the D-String CBM problem is intractable even when it is unweighted and \( D = 2 \).

**Theorem 1.1** (Xu et al. [16]). The unweighted 2-String CBM is NP-hard.

A two-layer algorithm for D-String CBM has been proposed in [16] that attempts to fix likely assignments and filter out unlikely assignments for long strings in the first layer of computation. In the second layer, it tries all possible combinations of assignments for long strings (i.e. at least 3 spin systems) 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 short strings consisting of 1 or 2 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 target-matching edges. 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 (exhaustive) two-layer algorithm in [16] 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 1.7778-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 [16], 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 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 \( U = \{u_1, u_2, \ldots, u_n\} \), \( V = \{v_1 \cdots v_i, v_{i+1} \cdots v_j, \ldots, v_p \cdots v_q\} \), and \( E \subseteq U \times V \) is the set of edges. Here, \( v_{i-1} \cdots v_i \) in \( V \) denotes a string of consecutively adjacent spin systems. We may assume that for every substring \( v_jv_{j+1} \) of a string in \( V \), \((u_i, v_j) \in E \) iff \((u_{i+1}, v_{j+1}) \in E \), because otherwise \((u_i, v_j)\) cannot be in any feasible matching and thus can be deleted 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 \( v_jv_{j+1} \cdots v_k \in V \) 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 \( \{(u_{i+x}, v_{j+x}) | 0 \leq x \leq k-j\} \) in \( E \). For convenience, we call the subset \( \{(u_{i+x}, v_{j+x}) | 0 \leq x \leq k-j\} \) 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 \( v_1 \in V \) and \( v_2 \in V \) with \( j_1 \neq j_2 \) such that both \((u_i, v_1)\) and \((u_i, v_2)\) are innermost edges of \( G' \)). Define a leading innermost edge of \( G' \) to be an innermost edge \((u_i, v_j)\) such that \( i \) 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. \( i' < i \leq i' + s' - 1 < i + s - 1 \).
### Table 1

**Summary on the performances of the 2-approximation and \(3\log_2 D\)-approximation algorithms on 126 instances of NMR peak assignment**

| \(|M^∗|\) | \(w(M^∗)\) | \(|M_2|\) | \(w(M_2)\) | \(R_2\) | \(R_1\) | \(|M_3|\) | \(w(M_3)\) | \(R_3\) | \(|M^∗|\) | \(w(M^∗)\) | \(|M_2|\) | \(w(M_2)\) | \(R_2\) | \(R_1\) | \(|M_3|\) | \(w(M_3)\) | \(R_3\) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bmr4027.1 | 158 | 1896284 | 158 | 1871519 | 28 | 11 | 158 | | | | | | | | | | | |
| bmr4027.2 | 157 | 1849500 | 7 | 18 | 158 | 1927193 | | | | | | | | | | | | |
| bmr4027.3 | 158 | 1841683 | 8 | 18 | 158 | 1930119 | 23 | | | | | | | | | | | |
| bmr4027.4 | 158 | 1829367 | 11 | 43 | 158 | 1925237 | 36 | | | | | | | | | | | |
| bmr4027.5 | 156 | 1827498 | 3 | 33 | 158 | 1923556 | 37 | | | | | | | | | | | |
| bmr4027.6 | 157 | 1818131 | 8 | 36 | 158 | 1916814 | 48 | | | | | | | | | | | |
| bmr4027.7 | 155 | 1784027 | 44 | 79 | 158 | 1885779 | 90 | | | | | | | | | | | |
| bmr4027.8 | 154 | 1671475 | 19 | 113 | 158 | 1875058 | 117 | | | | | | | | | | | |
| bmr4027.9 | 155 | 1652859 | 60 | 155 | 158 | 1896606 | 156 | | | | | | | | | | | |
| bmr4288.1 | 105 | 1249465 | 105 | 1238612 | 8 | 9 | 105 | 1208142 | 6 | | | | | | | | | |
| bmr4288.2 | 105 | 1220481 | 8 | 9 | 105 | 1194198 | 9 | | | | | | | | | | | |
| bmr4288.3 | 103 | 1206995 | 17 | 16 | 105 | 1199374 | 17 | | | | | | | | | | | |
| bmr4288.4 | 105 | 1185685 | 5 | 33 | 105 | 1214237 | 21 | | | | | | | | | | | |
| bmr4288.5 | 103 | 1169907 | 6 | 38 | 105 | 1211226 | 34 | | | | | | | | | | | |
| bmr4288.6 | 102 | 1179110 | 15 | 52 | 105 | 1217006 | 52 | | | | | | | | | | | |
| bmr4288.7 | 103 | 1122888 | 22 | 55 | 105 | 1230117 | 62 | | | | | | | | | | | |
| bmr4288.8 | 101 | 1133554 | 35 | N/A | 105 | 1232331 | 66 | | | | | | | | | | | |
| bmr4288.9 | 100 | 1051817 | 48 | 105 | 105 | 1249465 | 105 | | | | | | | | | | | |
| bmr4309.1 | 178 | 2048987 | 178 | 2066506 | 4 | 6 | 178 | 2118482 | 4 | | | | | | | | | |
| bmr4309.2 | 178 | 2023648 | 9 | 10 | 178 | 2108291 | 4 | | | | | | | | | | | |
| bmr4309.3 | 177 | 2013099 | 9 | 33 | 178 | 2115356 | 22 | | | | | | | | | | | |
| bmr4309.4 | 176 | 2024268 | 14 | 34 | 178 | 2107417 | 18 | | | | | | | | | | | |
| bmr4309.5 | 174 | 1954955 | 13 | 46 | 178 | 2090346 | 31 | | | | | | | | | | | |
| bmr4309.6 | 177 | 1924727 | 12 | 59 | 178 | 2074540 | 55 | | | | | | | | | | | |
| bmr4309.7 | 174 | 1885986 | 24 | 122 | 178 | 2078322 | 114 | | | | | | | | | | | |
| bmr4309.8 | 173 | 1868338 | 55 | 106 | 178 | 2026479 | 112 | | | | | | | | | | | |
| bmr4309.9 | 170 | 1796864 | 95 | 176 | 175 | 1999734 | 153 | | | | | | | | | | | |
| bmr4318.1 | 215 | 2390881 | 215 | 2418440 | 17 | 8 | 215 | 2495022 | 2 | | | | | | | | | |
| bmr4318.2 | 215 | 2398412 | 0 | 5 | 215 | 2481997 | 6 | | | | | | | | | | | |
| bmr4318.3 | 214 | 2409316 | 17 | N/A | 215 | 2481867 | 10 | | | | | | | | | | | |
| bmr4318.4 | 213 | 2394682 | 3 | 23 | 215 | 2481099 | 12 | | | | | | | | | | | |
| bmr4318.5 | 215 | 2355926 | 2 | 38 | 215 | 2473707 | 27 | | | | | | | | | | | |

<table>
<thead>
<tr>
<th>Instance</th>
<th>Start</th>
<th>End</th>
<th>Length</th>
<th>Number of Adjacent Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmr4318_6</td>
<td>214</td>
<td>231</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>bmr4318_7</td>
<td>210</td>
<td>225</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>bmr4318_8</td>
<td>212</td>
<td>221</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>bmr4318_9</td>
<td>207</td>
<td>215</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_1</td>
<td>66</td>
<td>71</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_2</td>
<td>66</td>
<td>72</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_3</td>
<td>66</td>
<td>72</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_4</td>
<td>65</td>
<td>68</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_5</td>
<td>64</td>
<td>68</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_6</td>
<td>66</td>
<td>69</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_7</td>
<td>66</td>
<td>68</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_8</td>
<td>64</td>
<td>66</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bmr4391_9</td>
<td>66</td>
<td>68</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_1</td>
<td>86</td>
<td>95</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_2</td>
<td>86</td>
<td>93</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_3</td>
<td>85</td>
<td>92</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_4</td>
<td>86</td>
<td>93</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_5</td>
<td>85</td>
<td>89</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_6</td>
<td>86</td>
<td>91</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_7</td>
<td>86</td>
<td>87</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_8</td>
<td>83</td>
<td>87</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bmr4579_9</td>
<td>83</td>
<td>76</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_1</td>
<td>68</td>
<td>88</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_2</td>
<td>68</td>
<td>84</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_3</td>
<td>68</td>
<td>83</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_4</td>
<td>67</td>
<td>82</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_5</td>
<td>67</td>
<td>79</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_6</td>
<td>67</td>
<td>82</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_7</td>
<td>66</td>
<td>75</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_8</td>
<td>65</td>
<td>73</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>bmr4752_9</td>
<td>67</td>
<td>81</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

The number after the underscore symbol in the name of each instance indicates the number of adjacent pairs of spin system in the instance (more precisely, \_*represents the target assignment, and \_M means that the number of adjacent pairs of spin systems is 50% of the total number of residues), \_M represents the target assignment, and \_M is the assignment computed by the two-layer (2-approximation, or 3 \log_2 D-approximation, respectively) algorithm. The parameters \( R_1 = |M^* \cap M_1| \), \( R_2 = |M^* \cap M_2| \), and \( R_3 = |M^* \cap M_3| \) show how many target-matching edges were recovered by the two-layer, 2-approximation, and 3 \log_2 D-approximation algorithms, respectively, on each instance. Each N/A indicates that the computation of the two-layer algorithm was taking too long (on a supercomputer) and had to be killed before any result could be obtained.
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 inner most 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 [1], we can construct a recursive algorithm to find a (heavy) feasible matching in $G'$ as shown in Fig. 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.

**Theorem 2.1** (Bar-Noy et al. [2]). The algorithm described in Fig. 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 [14]. For each protein, we generated 9 instances of spin-system adjacency by randomly adding links between spin systems that correspond to adjacent residues. These linked spin systems form strings in the instances. The number of links added in an instance ranged from 10% of the number of residues to 90% of the number of residues. 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. In the tests, the target assignments are matchings consisting of edges of form $(u_i, v_i)$. 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 $U$ 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 for the poor performance is due to the particular scoring/weighting function used in the experiment. Although the original objective of the scoring function is that target assignments always receive the largest weights, this goal is hard to achieve in practice, given the statistical nature of the scoring function. This is why even the two-layer algorithm was only able to recover a small number of target matching-edges in many cases, although it is supposed to be able to produce matchings with the largest weights. We are presently working on the improvement of the scoring function (which will not be discussed in this paper). As the scoring function improves, we would expect that target assignments will typically have weights close to the maximum, especially when the number of pairs of adjacent spin systems in an instance is sufficiently large. Observe that the two-layer algorithm did pretty well in the above experiment with the present scoring function on instances where the number of pairs of adjacent spin systems was at least 70% of the number of residues.
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. $\Gamma = \{e\} \cup \{e' | e' \in E(G'), e' \text{ conflicts with } e\}$;
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. $F = \{e | e \in \Gamma, e \text{ has weight 0}\}$;
7. $G'' = G' - F$;
8. recursively call 2-APPROXIMATION on $G''$ and output $M_1'$;
9. find a maximal $M_2' \subseteq F$ s.t. $M_1' \cup M_2'$ 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 $G'$.

3log$_2$D-APPROXIMATION on $G'$:
1. compute ratio $r = \frac{\text{the maximum length of strings in } V}{\text{the minimum length of strings in } F}$;
2. partition $V$ into $\ell = \max\{1, \log_4 r\}$ subsets $V_1, V_2, \ldots, V_\ell$ such that a string $s$ is included in subset $V_i$ if and only if $4^{i-1} \leq |s| < 4^i$;
3. for (every $i \in \{1, 2, \ldots, \ell\}$)
   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 $M_1', M_2', \ldots, M_\ell'$ and halt.

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

In trying to improve the performance on recovering 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 Fig. 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 2.2.** The algorithm described in Fig. 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.
Thus an approximation algorithm for D-STRING CBM with ratio $3 \log_2 D$.

Proof. For each $i \in \{1, 2, \ldots, \ell\}$, consider the bipartite graph $G_i' = (U, V, E_i)$. 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 $e = (u, v) \in E_i$ added to $M_i'$ in Step 3.3.2. Let $e' = (u', v')$ be an edge conflicting with $e$. Let $s$ (respectively, $s'$) 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. $x' < x + s - 1 \leq x' + s' - 1$ or $x' < x \leq x + s - 1 < x' + s' - 1$.
6. $x < x' \leq x' + s' - 1 \leq x + s - 1$ or $x \leq x' < x + s - 1 < x' + s' - 1$.

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 $\tilde{M}'$ denote an optimal feasible matching for graph $G'$, and $\tilde{M}_i^*$ be the sub-matching of $\tilde{M}^*$ in edge set $E_i$. Suppose without loss of generality that $\tilde{M}_1^*$ is the heaviest one among $\tilde{M}_1^*, \tilde{M}_2^*, \ldots, \tilde{M}_\ell^*$. Clearly, we have $w(\tilde{M}_i^*) \geq \frac{1}{6}w(\tilde{M}^*)$. Thus, $w(M') \geq \frac{1}{6}w(M_i^*) \geq 1/6w(\tilde{M}^*)$. The time complexity analysis is straightforward. \qed

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. 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. 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 [16] 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 2.3.** For all $D \geq 2$, unweighted D-STRING CBM is MAX SNP-hard.

**Proof (Sketch).** 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 [11]. The proof can be easily extended to any constant $D \geq 2$.

**Maximum Bounded 3-Dimensional Matching** (MB3DM): Given a universal set $U = \{1, 2, \ldots, m\}$ and a collection of subsets $S_1, S_2, \ldots, S_n$, where $S_i \subseteq U$, $|S_i| = 3$, and every element $u \in U$ 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 $U$ appears in at most 3 subsets. For each subset $S_i$, construct 7 vertices $a_{i,1}, a_{i,2}, \ldots, a_{i,7}$ in set $U$ and for each element $i \in U$ construct a 2-vertex string $b_{i,1}b_{i,2}$ in set $V$. We will also have in $V$ $q$ 1-vertex strings $f_1, f_2, \ldots, f_q$ and $3n$ 2-vertex strings $c_{1,1}c_{1,2}$, $c_{1,3}c_{1,4}$, $c_{1,6}, \ldots, c_{n,1}c_{n,2}$, $c_{n,3}c_{n,4}$, $c_{n,9}c_{n,6}$. Finally, for every $i = 1, 2, \ldots, m$, 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 $i = 1, 2, \ldots, q$ and every $j = 1, 2, \ldots, n$, connect string $f_j$ to $a_{j,1}$; and for every $i = 1, 2, \ldots, n$ and every $j = 1, 2, \ldots, n$, 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 [13] 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 $a_{i,1}, a_{i,2}, \ldots, a_{i,7}$ 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.

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 $< 2$. In this section, we answer the question affirmatively for a special case, namely, unweighted 2-STRING CBM. More specifically,
we will give a 1.7778-approximation algorithm for unweighted 2-STRING CBM, using a quite nontrivial construction. Part of the ideas in the construction are based on Berman’s recent work in [4], where he presented a $((k + 1)/2 + \varepsilon)$-approximation algorithm for weighted $k$-SET PACKING.

**Weighted $k$-SET PACKING:** Given a base set $X$ and a collection $\mathcal{S}$ of subsets of $X$, where every subset $S \in \mathcal{S}$ is of size at most $k$ and is associated with a non-negative weight $w(S)$, find out a subcollection $\mathcal{A} \subset \mathcal{S}$ of disjoint subsets such that its weight is the maximum. The weight of a collection is the sum of the weights of the subsets therein.

**Unweighted $k$-SET PACKING:** Weighted $k$-SET PACKING where every subset has the same weight $1$.

Given an instance graph $G = (U, V, E)$ of unweighted 2-STRING CBM, we can construct an instance $\mathcal{S}$ of weighted 3-SET PACKING as follows: For each string $v_i \cdots v_{i+t-1} \in V$ with $1 \leq t \leq 2$ such that $\{(u_i, v_i), (u_{i+t-1}, v_{i+t-1})\} \subseteq E$, we construct a $(t + 1)$-set $\{u_i, u_{i+t-1}, v_j\}$. The weight associated with this $(t + 1)$-set is $t$. Let $\mathcal{S}$ denote the collection of all 2- and 3-sets constructed.

Note that every 3-set $S \in \mathcal{S}$ contributes 2 units of weight to the solution and every 2-set $S \in \mathcal{S}$ contributes 1 unit of weight to the solution. Define a weight function $w : \mathcal{S} \rightarrow \mathbb{Z}^+$ as: $w(S) = |S| - 1$. For convenience, let $w(\mathcal{A}) = \sum_{S \in \mathcal{A}} w(S)$, where $\mathcal{A} \subset \mathcal{S}$ is a subcollection of disjoint sets. Our goal is to find a subcollection of disjoint sets that achieves the maximum weight. Define another new (square) weight function $w^2 : \mathcal{S} \rightarrow \mathbb{Z}^+$ in the way that $w^2(S) = (w(S))^2$ for every $S \in \mathcal{S}$; and similarly, let $w^2(\mathcal{A}) = \sum_{S \in \mathcal{A}} w^2(S)$ for any subcollection $\mathcal{A}$ of disjoint sets in $\mathcal{S}$.

Define a claw $\mathcal{C}$ of $\mathcal{S}$ to be a subcollection of disjoint sets that overlap (i.e. intersect) with a common other set not in $\mathcal{S}$; and define the common overlapped set to be the center of claw $\mathcal{C}$. If there are $d$ sets in $\mathcal{C}$, then $\mathcal{C}$ is called a $d$-claw. It is easy to see that in our constructed $\mathcal{S}$, $1 \leq d \leq 3$.

Let $\mathcal{A}$ be a subcollection of disjoint sets. We say that another subcollection of disjoint sets $\mathcal{C}$ (in the most special case, $\mathcal{C}$ is a claw), where $\mathcal{A} \cap \mathcal{C} = \emptyset$, improves $w(\mathcal{A})$ if $w((\mathcal{A} - N(\mathcal{C}, \mathcal{A})) \cup \mathcal{C}) > w(\mathcal{A})$, where $N(\mathcal{C}, \mathcal{A})$ is the collection of sets in $\mathcal{A}$ each of which overlaps with some set in $\mathcal{C}$. We may replace the weight function $w(\cdot)$ by other functions. For example, we say that $\mathcal{C}$ improves $\mathcal{A}$ if $|\mathcal{A} - N(\mathcal{C}, \mathcal{A})| \geq |\mathcal{A}|$; it improves $w^2(\mathcal{A})$ if $w^2((\mathcal{A} - N(\mathcal{C}, \mathcal{A})) \cup \mathcal{C}) > w^2(\mathcal{A})$; and it improves $f(\mathcal{A}) = w^2(\mathcal{A}) + 3|\mathcal{A}|$ if $f((\mathcal{A} - N(\mathcal{C}, \mathcal{A})) \cup \mathcal{C}) > f(\mathcal{A})$.

Consider the unweighted 3-SET PACKING problem. A straightforward greedy way to find a collection of disjoint sets is to start with the empty collection and at each round to add in a set while maintaining the disjointness. To formalize, for any $\mathcal{A}$, if there is a set $S$ such that $\{S\}$ improves $|\mathcal{A}|$, then we say $\{S\}$ 1-improves $\mathcal{A}$ or $\{S\}$ is a 1-improvement for $\mathcal{A}$. If $\mathcal{A}$ does not have any 1-improvement, then it is 1-maximal. A 1-maximal collection of disjoint sets output in this greedy manner could contain a number of sets only $\frac{1}{3}$ of the optimum. Fortunately, we do have a way to improve it. This needs the following more general definition of $t$-improvement [5,8]. For a collection of disjoint sets $\mathcal{A}$, if there exists a size-$t$ subcollection of disjoint sets $\mathcal{C}$ improving $|\mathcal{A}|$, then we say $\mathcal{C}$ $t$-improves $\mathcal{A}$, where $t \geq 2$. Similarly, when there is no $t$-improvement for $\mathcal{A}$, we say that $\mathcal{A}$ is $t$-maximal. When $\mathcal{C}$ $t$-improves $\mathcal{A}$, and every
set $S \in \mathcal{C}$ intersects at most two sets in $\mathcal{A}$, we say that $\mathcal{C}$ strictly $t$-improves $\mathcal{A}$. It is trivial to note that a $t$-improvement is also a strict $t$-improvement, for $t \leq 3$.

Let $\mathcal{A}$ be a strictly $t$-maximal collection of disjoint sets, where $t \geq 2$, and $\mathcal{A}^*$ be an optimal collection of disjoint sets. Then every set in $\mathcal{A}^*$ should intersect some set in $\mathcal{A}$. For simplicity, let the number of sets in $\mathcal{A}$ intersecting a set $S \in \mathcal{A}^*$ be the degree of $S$ (with respect to $\mathcal{A}$), denoted by $d(S)$. We can also let $d(T)$ denote the degree (number of sets in $\mathcal{A}^*$ intersecting $T$) of set $T \in \mathcal{A}$ (with respect to $\mathcal{A}^*$). Clearly, at most $|\mathcal{A}|$ sets in $\mathcal{A}^*$ can have degree 1, and all the others must have degree 2 or 3. Letting $\mathcal{A}_i^*$ denote the subcollection of sets in $\mathcal{A}^*$ having degree $i$, for $i = 1, 2, 3$, we have the following lemma.

**Lemma 3.1.** For $t = 2k - 1$ or $2k$, where $k \geq 2$, the subcollection $\mathcal{A}_2^*$ can be partitioned into $k$ disjoint subcollections: $\mathcal{A}_2^* = \bigcup_{i=1}^{k} \mathcal{Y}_i$, such that for every set $S \in \mathcal{Y}_i$, $1 \leq i \leq k - 1$, $S$ intersects exactly one set in $N(\mathcal{Y}_{i-1}, \mathcal{A})$, denoted as $N^1(S)$, and exactly one set not in $\bigcup_{j=0}^{i-1} N(\mathcal{Y}_j, \mathcal{A})$, denoted as $N^2(S)$. Here $\mathcal{Y}_0 = \mathcal{A}_1^*$. Furthermore, for every two sets $S_1, S_2 \in \mathcal{Y}_i$, $1 \leq i \leq k - 1$, if $t = 2k - 1$, then either $N^1(S_1) \neq N^1(S_2)$ or $N^2(S_1) \neq N^2(S_2)$; if $t = 2k$, then $N^2(S_1) \neq N^2(S_2)$.

**Proof.** We will partition $\mathcal{A}_2^*$ inductively. First of all, recall that every set in $\mathcal{Y}_0 = \mathcal{A}_1^*$ intersects a distinct set in $\mathcal{A}$. Let $\mathcal{Y}_1$ denote the subcollection (which could be empty) of sets in $\mathcal{A}_2^*$ each of which intersects a set in $N(\mathcal{Y}_0, \mathcal{A})$. Trivially, no set in $\mathcal{Y}_1$ can intersect two sets in $N(\mathcal{Y}_0, \mathcal{A})$, otherwise it would implies a 3-improvement to $\mathcal{A}$. For two sets $S_1, S_2 \in \mathcal{Y}_1$ with $N^1(S_1) = N^1(S_2)$, for the same reason that there should be $N^2(S_1) \neq N^2(S_2)$. Furthermore, when $t = 2k$, then $N^2(S_1) \neq N^2(S_2)$ whether $N^1(S_1) = N^1(S_2)$ or not, since otherwise it would imply a strict 4-improvement to $\mathcal{A}$. Therefore, the lemma holds for $k = 2$.

For larger $k$, we may further partition $\mathcal{A}_2^* - (\mathcal{Y}_1 \cup \mathcal{Y}_0)$ just the same as in the above to get $\mathcal{Y}_2, \mathcal{Y}_3, \ldots, \mathcal{Y}_{k-1}$, and let $\mathcal{Y}_k = \mathcal{A}_2^* - \bigcup_{i=1}^{k-1} \mathcal{Y}_i$. $\square$

**Corollary 3.2.** (i) For any $1 \leq i \leq k - 1$, $|\mathcal{Y}_i| \leq 2|\mathcal{Y}_{i-1}|$; (ii) $3(|\mathcal{A}| - \sum_{i=0}^{k-2} |\mathcal{Y}_i|) - |\mathcal{Y}_{k-1}| \geq 2|\mathcal{Y}_k|$; (iii) when $t = 2k$, $|\mathcal{A}| \geq \sum_{i=0}^{k-1} |\mathcal{Y}_i|$.

**Proof.** The proof is straightforward according to the above lemma and the fact that every set in $\mathcal{A}$ has degree between 1 and 3. $\square$

Trivially, if there is some $\mathcal{C} = \{S\}$ (strictly) 1-improving $\mathcal{A}$, then it also improves $w^2(\mathcal{A})$, and thus it improves $f(\mathcal{A})$ as well. The following lemma concerns strict $t$-improvements, for $t \geq 2$.

**Lemma 3.3.** Let $\mathcal{A}$ be a strictly $(t-1)$-maximal collection of disjoint sets. If $\mathcal{C}$ strictly $t$-improves $\mathcal{A}$, then $\mathcal{C}$ improves $f(\mathcal{A})$.

**Proof.** Let $H(\mathcal{A}, \mathcal{C}, E)$ denote the bipartite graph which takes sets in $\mathcal{A}$ and $\mathcal{C}$ as vertices, and two vertices $S \in \mathcal{C}$ and $T \in \mathcal{A}$ are adjacent if and only if they intersect.
Let $H'$ be the subgraph of $H$ induced by the vertex-subset $\mathcal{C} \cup N(\mathcal{C}, \mathcal{A})$. From the assumption that $\mathcal{A}$ is strictly $(t-1)$-maximal, we know that $|N(\mathcal{C}, \mathcal{A})| = t - 1$ and every vertex/set $T \in N(\mathcal{C}, \mathcal{A})$ must have degree at least 2. From the fact that $\mathcal{C}$ strictly $t$-improves $\mathcal{A}$, we know that every vertex/set $\mathcal{C}$ must have degree at most 2. It follows that in $H'$, there are $x$ degree-3 vertices in $\mathcal{A}$ if and only if there are $(2-x)$ degree-1 vertices in $\mathcal{C}$, where $x$ can be either 0, 1, or 2. Also from the strict $(t-1)$-maximality of $\mathcal{A}$, we conclude that $H'$ must be connected.

Case 1: $x = 0$. Then, $H'$ is actually a path with two ending vertices both in $\mathcal{C}$. Assume without loss of generality that this path is $S_1 - T_1 - S_2 - T_2 - \cdots - T_{t-1} - S_t$. We claim that if $T_i$ is a 3-set, then the next 3-set along the path, if exists, must be a set in $\mathcal{C}$. To prove the claim, we assume without loss of generality that $i = 1$: $T_1$ is a 3-set. If $S_2$ is not a 3-set, it means the element in $T_1 \cap S_2$ must be an element in set $U$. Therefore, the element in $S_2 \cap T_2$ is in set $V$, indicating that $T_2$ is neither a 3-set. The argument can be repeated to show that if $S_3$ is not a 3-set, then $T_3$ neither, etc. Thus, the next 3-set must be a set in $\mathcal{C}$, proving the claim.

With the above claim, it is easy to notice that $N(\mathcal{C}, \mathcal{A})$ can have at most one more 3-set than $\mathcal{C}$ has. Furthermore, if the first 3-set along the path is in $\mathcal{C}$, then the number of 3-sets in $N(\mathcal{C}, \mathcal{A})$ is no greater than that in $\mathcal{C}$. This means that $\mathcal{C}$ can reduce $w^c(\mathcal{A})$ by at most 2 and thus it certainly improves $f(\mathcal{A})$.

Case 2: $x = 1$. Then, we can decompose $H'$ into a simple cycle and a simple path, where only the degree-3 vertex is shared by them. It is easy to verify that the claim in Case 1 applies to every simple path, and every simple cycle, in $H'$ as well. In particular, for each simple cycle, the number of 3-sets in $\mathcal{C}$ which are vertices on the cycle is not less than the number of 3-sets in $N(\mathcal{C}, \mathcal{A})$ which are vertices on the cycle. So, the number of 3-sets in $N(\mathcal{C}, \mathcal{A})$ is at most one greater than the number of 3-sets in $\mathcal{C}$. Thus, $\mathcal{C}$ improves $f(\mathcal{A})$.

Case 3: $x = 2$. Then, we can decompose $H'$ into either two vertex-disjoint simple cycles and a path connecting the two cycles, or two simple cycles sharing a simple path. In either case, each degree-3 vertex is included in some cycle. We conclude that in either case, the number of 3-sets in $N(\mathcal{C}, \mathcal{A})$ is at most one greater than the number of 3-sets in $\mathcal{C}$. Thus, $\mathcal{C}$ improves $f(\mathcal{A})$.

Definition 3.1. Let $\mathcal{A}$ be a subcollection of disjoint sets, and $S \in \mathcal{A} - \mathcal{A}$ such that $w(S) > \frac{1}{2}w(N(S, \mathcal{A}))$.

- $N(S, \mathcal{A}) = N(\{S\}, \mathcal{A})$.
- $m(S)$ is the subcollection of sets $T \in N(S, \mathcal{A})$ having the maximum weight $w(T)$ in $N(S, \mathcal{A})$.
- $\text{charge}(S, T) = \begin{cases} \frac{1}{|m(S)|}(w(S) - \frac{1}{2}w(N(S, \mathcal{A}))), & \text{if } T \in m(S), \\ 0, & \text{otherwise.} \end{cases}$
- $\mathcal{C}$ is a good claw if either (1) $N(\mathcal{C}, \mathcal{A}) = \emptyset$, or (2) the center of $\mathcal{C}$ is $T \in \mathcal{A}$, $w(S) > \frac{1}{2}w(N(S, \mathcal{A}))$ for all $S \in \mathcal{C}$, and $\sum_{S \in \mathcal{C}} \text{charge}(S, T) > \frac{1}{2}w(T)$.
- $\mathcal{C}$ is a nice claw if it is a minimal good claw.

It is not hard to derive all possible configurations for a set $S \in \mathcal{A}$ having a positive charge, which are listed in Table 2. The next lemma follows naturally:
Table 2
Possible configurations for a set $S$ outside $\mathcal{A}$ having a positive charge

<table>
<thead>
<tr>
<th>$w(T)$</th>
<th>$w(S)$</th>
<th>$w(N(S, \mathcal{A}'))$</th>
<th>$w^2(S) - w^2(N(S, \mathcal{A} - {T}))$</th>
<th>charge($S, T$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3 = 2 + 1</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2 = 1 + 1</td>
<td>3</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>

---

$f$-IMP($t$):
1. $\mathcal{A} \leftarrow \emptyset$; $f_0 = -1$; $f_1 = 0$;
2. while ($f_1 > f_0$)
   2.1 $f_0 \leftarrow f_1$;
   2.2 while (there exists a nice claw $\mathcal{C}$ for $\mathcal{A}$)
      2.2.1 $\mathcal{A} \leftarrow (\mathcal{A} - N(\mathcal{C}, \mathcal{A})) \cup \mathcal{C}$;
   2.3 $i = 2$;
   2.4 maximal$_i = \text{false}$;
   2.5 while ($i \leq t$ & & maximal$_i = \text{false}$)
      2.5.1 if (there exists a strict $i$-improvement $\mathcal{C}$ for $\mathcal{A}$)
         2.5.1.1 $\mathcal{A} \leftarrow (\mathcal{A} - N(\mathcal{C}, \mathcal{A})) \cup \mathcal{C}$;
         2.5.1.2 maximal$_i = \text{true}$;
      2.5.2 else
         2.5.2.1 $i \leftarrow i + 1$;
   2.6 $f_1 \leftarrow f(\mathcal{A})$;

Fig. 3. A high-level description of algorithm $f$-IMP($t$).

**Lemma 3.4.** Let $\mathcal{A}$ be a subcollection of disjoint sets. If $\mathcal{C}$ is a nice claw with respect to $\mathcal{A}$, then it improves $f(\mathcal{A})$.

A high-level description of our approximation algorithm, called $f$-IMP($t$), for $t \geq 5$, is given in Fig. 3. From Lemmas 3.3 and 3.4, we conclude that $f$-IMP($t$), where $t \geq 5$, terminates when $\mathcal{A}$ is a strictly $t$-maximal collection of disjoint sets with respect to which there exists no nice claw. Examining the existence of a nice claw takes $O(m^3)$ time and examining the existence of a strictly $i$-improving collection takes $O(m^3)$ time, where $m = |\mathcal{A}|$. Since for every collection $\mathcal{A}$ of disjoint sets, $f(\mathcal{A}) \leq 4m + 3m = 7m$, $f$-IMP($t$) runs in $O(m^3)$ time.

**Lemma 3.5.** Let $\mathcal{A}$ be a subcollection of disjoint sets, with respect to which there is no nice claw; and $\mathcal{A}^*$ an optimal subcollection. Assume that there are $m_1$ 2-sets and $m_2$ 3-sets in $\mathcal{A}$. Then $w(\mathcal{A}^*) \leq \frac{3}{2}m_1 + 4m_2$. 
Proof. We will distribute \( w(\mathcal{A}^*) \) among sets in \( \mathcal{A} \) in such a way that no 2-set \( T \in \mathcal{A} \) receives more than \( \frac{1}{2}w(T) \) and no 3-set \( T \in \mathcal{A} \) receives more than \( 2w(T) \). The distribution consists of two phases. In the first phase, every set \( S \in \mathcal{A}^* \) sends to each \( T \in N(S, \mathcal{A}) \) a portion of weight equal to \( \frac{1}{2}w(T) \). Note that \( N(S, \mathcal{A}) \) should not be empty, otherwise \( \{S\} \) would be a nice claw. It is clear that in this phase, set \( S \) sends off weight equal to \( \frac{1}{2}w(N(S, \mathcal{A})) \). Therefore, \( S \) still has an amount \( w(S) - \frac{1}{2}w(N(S, \mathcal{A})) \) of weight, which is positive only when \( w(S) > \frac{1}{2}w(N(S, \mathcal{A})) \). In the second phase, if \( w(S) > \frac{1}{2}w(N(S, \mathcal{A})) \), \( S \) sends to every set \( T \in m(S) \) a portion of weight equal to \( \text{charge}(S, T) \). Note that after the second phase, set \( S \) sent off all its weight.

Consider the receiving side. In the first phase, every set \( T \in \mathcal{A} \) gets weight \( \frac{1}{2}w(T) \) from each neighbor \( S \in \mathcal{A}^* \). Therefore, from the fact that \( \mathcal{A}^* \) is a collection of disjoint sets, every 2-set \( T \) would get weight in total at most \( w(T) \) and every 3-set \( T \) would get weight in total at most \( \frac{1}{2}w(T) \). During the second phase, \( T \) gets at most \( \frac{1}{2}w(T) \), since otherwise the sets that send (positive) charges to \( T \) would form a good claw, which would imply the existence of a nice claw. In other words, \( T \) in total receives at most an amount \( (|T| + 1)/2w(T) \) of weight, implying that \( w(\mathcal{A}^*) \leq \frac{1}{2}m_1 + 4m_2 \). \( \square \)

Lemma 3.6. Let \( \mathcal{A} \) be a strictly \( t \)-maximal subcollection of disjoint sets, where \( t \geq 5 \), with respect to which there is no nice claw; and \( \mathcal{A}^* \) an optimal subcollection. Assume that there are \( m_1 \) (\( m_1^* \), respectively) 2-sets and \( m_2 \) (\( m_2^* \), respectively) 3-sets in \( \mathcal{A} \) (in \( \mathcal{A}^* \), respectively). Then

\[
(7 \cdot 2^{k-1} - 9)m_1 + (10 \cdot 2^{k-1} - 15)m_2 \geq (5 \cdot 2^{k-1} - 6)m_1^* + (6 \cdot 2^{k-1} - 9)m_2^*,
\]

where \( k = [\frac{1}{2}] \).

Proof. Let \( \mathcal{A}^* \) be decomposed in the way as in Lemma 3.1, with respect to \( \mathcal{A} \). Clearly, every 2-set (3-set, respectively) in \( \mathcal{A} \) has degree at most 2 (3, respectively). Therefore,

\[
m_1 + 3m_2 \geq \sum_{T \in \mathcal{A}} d(T) = \sum_{S \in \mathcal{A}^*} d(S) = |\mathcal{A}^*| - 2|\mathcal{A}_1^*| - |\mathcal{A}_2^*| = 3(m_1^* + m_2^*) - 2|\mathcal{Y}_0| - \sum_{i=1}^k |\mathcal{Y}_i|.
\] (3.1)

From Corollary 3.2(ii) we have

\[
3m_1 + 3m_2 \geq 3 \sum_{i=0}^{k-2} |\mathcal{Y}_i| + |\mathcal{Y}_{k-1}| + 2|\mathcal{Y}_k|.
\] (3.2)

In addition, from the fact that there is no nice claw with respect to \( \mathcal{A} \), we conclude that a degree-1 3-set (in \( \mathcal{Y}_0 \)) cannot intersect a 2-set in \( \mathcal{A} \). That is,

\[
m_2 \geq |\mathcal{Y}_0| - m_1^*.
\] (3.3)
It follows that \((3.1) \times (3+\alpha)/2 + (3.2) + (3.3) \times \alpha\), where \(0 \leq \alpha \leq 1\), gives the following:

\[
(6 + \alpha)m_1 + \frac{15 + 5\alpha}{2} m_2 \\
\geq 9 + \frac{9 + 3\alpha}{2} m_1^* + \frac{9 + 3\alpha}{2} m_2^* + \frac{3 - \alpha}{2} \sum_{i=1}^{k-2} |\gamma_i| - \frac{1 + \alpha}{2} |\gamma_{k-1}|
\]

\[+ \frac{1 - \alpha}{2} |\gamma_k|. \tag{3.4}
\]

By setting \(3 - \alpha = (2^{k-2}/2^{k-2} - 1)(1 + \alpha)\), or equivalently \(\alpha = (2^{k-1} - 3)/2^{k-1} - 1\), in (3.4), together with Corollary 3.2(i), we have the following:

\[
(7 \cdot 2^{k-1} - 9)m_1 + (10 \cdot 2^{k-1} - 15)m_2 \geq (5 \cdot 2^{k-1} - 6)m_1^* + (6 \cdot 2^{k-1} - 9)m_2^*.
\]

This proves the lemma. \(\Box\)

In order to analyze \(f\)-\(\text{IMP}(t)\), we need one more lemma.

**Lemma 3.7.** Let graph \(G = (U, V, E)\) be an instance of the unweighted 2-STRING CBM problem such that the set \(V\) consists of strings of equal length (namely, 2), where \(n_1 = |U|\), \(n_2 = |V|\), and \(m = |E|\). Then, a feasible matching in \(G\) can be found in \(O(m\sqrt{n_1n_2})\) time, whose size is at least \(\frac{2}{3}\) of the optimum.

**Proof.** Let \(U = \{u_1, u_2, \ldots, u_{n_1}\}\). For each index \(i \in \{0, 1, 2\}\), let \(G_i\) be the bipartite graph obtained from \(G\) as follows:

1. For every string in \(V\), merge the two vertices in the string into a single super-vertex (with all resulting multiple edges deleted).
2. For all \(j\) such that \(i + 1 \leq j \leq n_1 - 2\) and \(j \mod 3 = i\), merge \(u_i\), \(u_{i+1}\), and \(u_{i+2}\) into a single super-vertex (with all resulting multiple edges deleted); and for every neighbor \(s_h\) of the new super-vertex, if the original string (in \(V\)) corresponding to \(s_h\) can be matched to neither \(u_ju_{j+1}\) nor \(u_{j+1}u_{j+2}\), then delete the edge between the new super-vertex and \(s_h\).
3. If neither \(u_1\) nor \(u_2\) was merged in Step 2, then merge \(u_1\) and \(u_2\) into a single super-vertex (with all resulting multiple edges deleted); and for every neighbor \(s_h\) of the new super-vertex, if the original string (in \(V\)) corresponding to \(s_h\) cannot be matched to \(u_1u_2\), then delete the edge between the new super-vertex and \(s_h\).
4. If neither \(u_{n_1-1}\) nor \(u_{n_1}\) was merged in Step 2 or 3, then merge \(u_{n_1-1}\) and \(u_{n_1}\) into a single super-vertex (with all resulting multiple edges deleted); and for every neighbor \(s_h\) of the new super-vertex, if the original string (in \(V\)) corresponding to \(s_h\) cannot be matched to \(u_{n_1-1}u_{n_1}\), then delete the edge between the new super-vertex and \(s_h\).

It is clear that every matching in \(G_i\) can be easily transformed into a feasible matching of the same size in \(G\). So, for each \(i \in \{0, 1, 2\}\), we compute a maximum matching \(M_i\) in \(G_i\), and transform it into a feasible matching \(\tilde{M_i}\) of the same size in \(G\). We claim that \(|\tilde{M}| \geq \frac{2}{3}|M^*|\), where \(\tilde{M}\) is the maximum-sized one among \(\tilde{M_0}, \tilde{M_1}, \tilde{M_2}, \) and
$M^*$ is a maximum-sized feasible matching in $G$. To see this, for each $i \in \{0, 1, 2\}$, let $M^*_i$ denote the subset of matches $(u_h u_{h+1}, v_{i+1}) \in M^*$ such that $u_h$ and $u_{h+1}$ belong to the same super-vertex in $G_i$. It holds that $\sum_{i=0}^{2} |M^*_i| = 2|M^*|$ because each match $(u_h u_{h+1}, v_{i+1}) \in M^*$ belongs to exactly two of $M^*_0, M^*_1, M^*_2$. This implies that the maximum-sized one among $M^*_0, M^*_1, M^*_2$ has size at least $\frac{2}{3}|M^*|$. On the other hand, for each $i \in \{0, 1, 2\}$, if we modify $M^*_i$ by merging $u_h, u_{h+1}$ into a super-vertex and merging $v_{i+1}$ into a super-vertex for every match $(u_h u_{h+1}, v_{i+1}) \in M^*$, the resulting matching is a matching in $G_i$ and has size $|M^*_i|$; hence $|M^*_i| \leq |M| = |\tilde{M}|$ because $M$ is a maximum matching in $G_i$. Therefore,

$$\frac{2}{3} |M^*| \leq \max \{ |M^*_0|, |M^*_1|, |M^*_2| \} \leq \max \{ |\tilde{M}_0|, |\tilde{M}_1|, |\tilde{M}_2| \} = |\tilde{M}|.$$  

This completes the proof of the claim and hence that of the lemma. □

**Theorem 3.8.** For any positive $\epsilon$, the unweighted 2-STRING CBM problem can be approximated within ratio $\frac{16}{9} + \epsilon$ in polynomial time.

**Proof.** Let $\mathcal{A}$ denote the subcollection of disjoint sets output by algorithm $f$-IMP($t$), where $t \geq 5$. We have known that $\mathcal{A}$ is strictly $t$-maximal and there is no nice claw with respect to $\mathcal{A}$. Let $\mathcal{A}^*$ denote an optimal subcollection. Assume there are $m_1$ (or $m_1^*$) 2-sets and $m_2$ (or $m_2^*$, respectively) 3-sets in $\mathcal{A}$ ($\mathcal{A}^*$, respectively). Let $k = \lceil \frac{t}{2} \rceil$.

We have by Lemma 3.5

$$\frac{3}{2} m_1 + 4m_2 \geq w(\mathcal{A}^*) \quad (3.5)$$

and by Lemma 3.6

$$(7 \cdot 2^{k-1} - 9)m_1 + (10 \cdot 2^{k-1} - 15)m_2 \geq (5 \cdot 2^{k-1} - 6)m_1^*$$

$$+ (6 \cdot 2^{k-1} - 9)m_2^* \quad (3.6)$$

Moreover, by Lemma 3.7, we may assume that

$$w(\mathcal{A}) \geq \frac{4}{3} m_2^* \quad (3.7)$$

These three inequalities $((3.5) \times (4 \cdot 2^{k-1} - 3) + (3.6) + (3.7) \times \frac{3}{4} (4 \cdot 2^{k-1} - 3))$ imply that

$$w(\mathcal{A}) \geq \frac{36 \cdot 2^{k-1} - 36}{64 \cdot 2^{k-1} - 63} w(\mathcal{A}^*).$$

Since $f$-IMP($t$) terminates in $O(m^t)$ time, where $m = |\mathcal{S}|$, it is an approximation algorithm for the unweighted 2-STRING CBM problem and its worst-case performance ratio is $(64 \cdot 2^{k-1} - 63)/(36 \cdot 2^{k-1} - 36)$, which approaches $\frac{16}{9}$ when $t$ (and thus $k$) approaches $+\infty$. □
4. Concluding remarks

It would be interesting to test if the 1.7778-approximation algorithm works well in practice. An obvious open question is if $D$-STRING CBM admits a $\rho$-approximation algorithm for some constant $\rho<2$.

References