Heuristic Search in Constrained Bipartite Matching with Applications to Protein NMR Backbone Resonance Assignment

Theodore Tegos * Zhi-Zhong Chen † Guohui Lin * ‡

Abstract

The constrained bipartite matching (CBM) problem is a variant of the classical bipartite matching problem that has been well studied in the Combinatorial Optimization community. The input to CBM is an edge-weighted complete bipartite graph in which there are a same number of vertices on both sides and vertices on one side are sequentially ordered while vertices on the other side are partitioned and connected into disjoint directed paths. In a feasible matching a path must be mapped to consecutive vertices on the other side. The optimization goal is to find a maximum or a minimum weight perfect matching. Such an optimization problem has its applications to scheduling and protein Nuclear Magnetic Resonance peak assignment. It has been shown to be NP-hard and MAX SNP-hard if the perfectness requirement is dropped. In this paper, more results on the inapproximability are presented and IDA*, a memory efficient variant of the well known A* search algorithm, is utilized to solve the problem. Accordingly, search heuristics and a set of heuristic evaluation functions are developed to assist the search, whose effectiveness is demonstrated by a simulation study using real protein NMR backbone resonance assignment instances.

Keywords: Combinatorial Optimization, Constrained Bipartite Matching, Heuristic Search, IDA* Search, Evaluation Function, Protein NMR Backbone Resonance Assignment

1 Introduction

Determining the 3D structure of a protein is of extreme importance since it is this structure that largely defines the function of the protein. Although a huge amount of effort has been put into computational structure prediction, Nuclear Magnetic Resonance (NMR) spectroscopy and X-Ray crystallography, due to their accuracy, are still the dominant experimental techniques for determining protein structures at the atomic resolution. Typically, with the modern advents in NMR spectroscopy [1, 6, 9, 15], this technique is expected to play a more important role in Structural Genomics.

Protein NMR spectroscopy starts with the generation of spectral data which range from nuclei resonance frequency "fingerprints" to dihedral angles (spin-spin couplings) and inter-nuclear orientations (residual dipolar couplings). The local structural constraints have to be associated correctly to their host chemical units in order to calculate the three-dimensional protein structure. Such an

*Bioinformatics Research Group, Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada.

†Department of Mathematical Science, Tokyo Denki University. Hatoyama, Saitama 350-0394, Japan.

‡To whom correspondence should be addressed. Tel: (780) 492-3737, Fax: (780) 492-1071, Email: ghlin@cs.ualberta.ca

association can be done directly but due to the low accuracy it is usually guided by the NMR backbone resonance sequential assignment, which maps the identified spectral resonance peaks to the sets of nuclei that generate the peaks. Every resonance peak captures an atomic interaction, represented by a chemical shift tuple, involving two or more nuclei that may be within the same amino acid residue or a pair of neighboring amino acid residues. The peaks from a spectrum form a peak list and in general there are multiple peak lists used in the assignment process. The underlying principle making resonance peak assignment successful is that the chemical shift values for a specific nucleus in the target protein are unique across multiple spectra, despite possible degeneracy and reading errors. In fact, the chemical shift value of a nucleus depends only on the atomic type and its local chemical environment. The resonance peak assignment process tries to take advantage of such signature information to correctly associate resonance peaks, which can be regarded as vectors of chemical shifts, to their host nuclei.

A single piece of chemical shift value often is not enough to distinguish the amino acid residue type (out of 20 types) that the host nucleus resides in. One way to boost such signature information is to group together the chemical shifts from nuclei in a common amino acid residue and then use the sum of individual signature information to perform the assignment. Such a super vector of chemical shifts is referred to as a *spin system* and resonance peak assignment can be viewed as spin system assignment. The phase where spin systems are formed is referred to as peak grouping and it is expected to generate the same number of spin systems as the number of amino acid residues in the target protein. Mathematically, the spin system assignment can be cast into the classical *Bipartite Matching* problem that has been well studied in the Combinatorial Optimization community.

In our case, an instance of the Bipartite Matching problem is an edge-weighted complete bipartite graph $G = (A, S, E)$, where vertex set A consists of the linearly ordered amino acid residues in the target protein sequence, vertex set S consists of the identified spin systems and the weight of an edge (a_i, s_j) represents the quantified signature information of mapping spin system s_j to amino acid residue a_i. Note that we assume the ideal scenario where $|A| = |S|$, though in practice some true spin systems might be missing and some fake spin systems might be present in set S. Depending on the way of quantifying the spin system signature information, the desired perfect matching should have either a maximum weight or a minimum weight. For example, if the weight is the absolute logarithm of the probability of mapping a spin system to an amino acid residue type, then it is a minimization problem; if the weight is the shifted logarithm (to all positives), then it is a maximization problem.

The Bipartite Perfect Matching (BPM) problem (both the minimization and the maximization objectives) can be solved efficiently by a variety of existing algorithms. One of the fastest algorithms is CSA [8], developed by Goldberg and Kennedy, which is a cost-scaling push-relabel algorithm that finds minimum or maximum weight matchings of maximum cardinality (in our case, the input graph is complete and therefore the matching must be perfect).

However, spin system assignment through the BPM formulation is far from being solved and better mathematical models are still in need. What complicates things in the assignment is the existence of multiple copies of a type of amino acid residue in the target protein. Notice that there are only 20 types of amino acid residues while the length of target proteins, measured by the number of amino acid residues therein, is in the hundreds and approaching the thousands (in our experiments, protein lengths vary from 66 to 215). Since the signature information does not contain residual position, there are usually too many optimal matchings, in terms of matching weight, for a BPM instance and simply returning one by some algorithm doesn't satisfy the near completeness

requirement in structure calculation. Recall that some peaks capture atomic interaction among nuclei from two neighboring amino acid residues. At the peak grouping stage, these peaks provide evidence that some pair of spin systems should map to adjacent amino acid residues in the target protein. This is the so-called *adjacency information* among the spin systems and it has been noticed that adjacency information is crucial to the success of the resonance peak assignment [20] (for reviews of existing research taking advantage of modern triple resonance three- and four-dimensional NMR experiments for extracting adjacency information, see [1, 9, 15]). This adjacency information encapsulates information about proximity of residues in the protein sequence, not spatial closeness in 2D or 3D space. As a consequence, while the amino acid residues in vertex set A follow the sequential order as they show up in the target protein sequence, vertex set S contains groups of consecutive spin systems. The consecutive spin systems in a group form a directed *string* and they should be mapped to consecutive amino acid residues in vertex set A. Note that there are multiple strings and some strings might consist of only one spin system, due to missing adjacency information. Consequently, the spin system assignment problem has to be cast as a *Constrained Bipartite Perfect Matching* (CBPM) problem, with either the maximization or the minimization objective (denoted as max-CBPM or min-CBPM, respectively):

`Instance:`

An edge-weighted complete bipartite graph $G = (A, S, E)$, where vertex set A consists of linearly ordered amino acid residues in the target protein sequence and vertex set S consists of strings of spin systems;

`Goal:`

Find a perfect matching satisfying the adjacency constraint and achieving the maximum/minimum weight.

In the problem definitions of BPM and CBPM the input bipartite graph is required to be complete. Nonetheless, in practice, edge weights could be set to negative (for the maximization objective) or positive infinity (for the minimization objective) to indicate an infeasible mapping between a spin system and an amino acid residue. In such a case, the input bipartite graph becomes actually *not complete* (where the missing edges correspond to the infeasible mappings). For this reason, we will hereafter drop the completeness requirement from the problem definitions, unless otherwise specified.

Lemma 1 [21] *Given an instance $G = (A, S, E)$ of the* CBPM *problem, it is NP-hard to decide if G has a perfect matching satisfying the adjacency constraint.*

We note that in the application of NMR spin system assignment, instances of the CBPM problem constructed out of real (or simulated) NMR spectral data often do have a perfect matching (consisting of feasible mappings). Therefore, we are mostly interested in the special case of the CBPM problem where a perfect matching of the input graph G satisfying the adjacency constraint is guaranteed. However, theoretically, the CBPM problem is unbelievably hard. We prove that even approximating this special case within a polynomial factor is NP-hard. Therefore, heuristics (with no worst-case guarantees) are essential for tackling the problem. There has already been a considerable amount of effort to solve the spin system assignment problem using the CBPM representation or its variants. This includes an expert system [23], a statistical method [10], a two-layer search heuristic [21], a few approximation algorithms (with proven worst-case guarantees)

[3, 4, 5], and a branch-and-bound technique combined with greedy filtering [14]. An existing exact algorithm, branch-and-bound, has been described in [14] with its real performance not completely reported because of its high complexity.

We have taken a different approach and decided to use heuristic search to tackle the CBPM problem. After a theoretical analysis of the inapproximability of the CBPM problem with the maximization objective and the minimization objective (Section 2), we will present the use of the popular *iterative-deepening A** (IDA*) algorithm [12] to find the minimum weight perfect matching in a min-CBPM instance (Section 3). Towards our goal, we have developed a set of pruning heuristics that reduce the size of the search tree built by IDA* and consequently make the search more efficient. We have also developed four different heuristic evaluation functions to guide the search that provide different degrees of accuracy and speed (Section 3). The previous method most similar to ours would be the expert system described in [23], which also uses best-first search. However, the multi-step constraint satisfaction approach in that model may propagate assignment errors, so (contrary to our approach) it does not target globally optimal assignments. To verify the strength of our search method, we applied it to a test suite of 70 instances on 14 proteins obtained from [21] using the quantified spin system signature information from [3] (Section 4). In relation to the approximate methods [3, 14, 21] designed for the CBPM problem, our approach was able to retrieve more correct mappings (i.e. assignment edges). More significantly, our method was able to solve most of the instances faster than existing approaches that target optimal matchings and some existing approaches targeting only near optimal matchings [14]. We conclude the paper in Section 5 with some of our future work.

Throughout the remainder of the paper we will be referring to those groups containing more than one spin system as *strings*, whereas standalone spin systems will be called *singletons*. Note that a singleton can also be viewed as a string of size one. We will also use *run* to denote a maximal number of consecutive amino acid residues that have not been mapped to any spin systems yet.

2 Theoretical Inapproximability Results

Here we consider the special cases of the max-CBPM and the min-CBPM problems where a constrained perfect matching of the input graph G (i.e. a perfect matching of G satisfying the adjacency constraint) is given.

Let Π be a maximization (respectively, minimization) problem and let $f(n)$ be a function mapping each positive integer n to a positive real number. An algorithm $\mathcal{A}$ is said to be a *factor-$f(n)$ approximation algorithm* for Π if, on each instance I of Π, $\mathcal{A}$ outputs a solution of I whose weight is at least $1/f(n)$ times the optimal (respectively, at most $f(n)$ times the optimal), where $n = |I|$ measures the size of the instance.

Theorem 2 *If for some constant $\epsilon > 0$ the above special case of the* max-CBPM *problem has a factor-n^{ϵ} polynomial-time approximation algorithm, then $P = NP$.*

PROOF. The proof is done via a reduction from the *Constrained* max-3-*Dimensional Matching* (max-C3DM) problem defined as follows: An instance of the problem is a triple (M, M_0, M_1), where $M \subseteq W \times X \times Y$ (W, X, and Y are disjoint sets having the same number q of elements), M_0 is a *matching* of M (i.e. M_0 is a subset of M such that $|M_0| = q$ and no two elements of M_0 agree in any coordinate), and $M_1 \subseteq M$. Given (M, M_0, M_1), the objective is to compute a matching M^* of

M such that $|M^* \cap M_1|$ is maximized over all matchings of M. Zuckerman [24] showed that if for some $\epsilon > 0$ max-C3DM has a factor-n^ϵ polynomial-time approximation algorithm, then $P = NP$.

Our reduction needs another intermediate problem, namely, the 4-*Partitioning* problem. An instance of this problem is a pair (U, B), where U is a finite set of $4p$ positive integers, B is a positive integer, $\sum_{u \in U} u = pB$, and every $u \in U$ satisfies $B/5 < u < B/3$. Given (U, B), the objective is to decide if U has a 4-*partition*, i.e. a partition of U into p disjoint sets $U_1, \ldots, U_p$ such that $\sum_{u \in U_i} u = B$ for each $1 \leq i \leq p$. Note that each set U_i in a 4-partition of U must contain exactly four elements. The 4-Partitioning problem is strongly NP-complete [7], i.e. it remains NP-complete even if all the input integers are provided in unary.

We describe our reduction next. Let (M, M_0, M_1) be an instance of the max-C3DM problem. Let $W = \{w_1, \ldots, w_q\}$, $X = \{x_1, \ldots, x_q\}$, and $Y = \{y_1, \ldots, y_q\}$. Let $M = \{e_1, \ldots, e_m\}$. From M, we construct an instance (U, B) of the 4-Partitioning problem in the same way as in the proof of the strong NP-completeness of the 4-Partitioning problem described in [7, Pages 96–99]. Let us review some details in this typical construction. U consists of $4m$ integers, one for each triple in M and one for each occurrence of an element of $W \cup X \cup Y$ in a triple in M. The integers of U corresponding to a particular element $z \in W \cup X \cup Y$ are denoted by $z[1], \ldots, z[n_z]$, where n_z is the number of triples from M in which z occurs. For convenience, $z[1]$ is regarded as the "actual" element corresponding to z, while $z[2]$ through $z[n_z]$ are regarded as the "dummy" elements corresponding to z. Moreover, we abuse the notation to let each $e_\ell \in M$ denote the integer (of U) corresponding to e_ℓ. By setting up the values as in [7, Page 97], (U, B) has the following two properties:

P1: Suppose M has a matching M'. Then, a 4-partition $\mathcal{U}$ of U can be computed easily from M' as follows:

- P1a: for each $e_\ell = (w_i, x_j, y_k) \in M'$, the corresponding set in $\mathcal{U}$ is $\{e_\ell, w_i[1], x_j[1], y_k[1]\}$;
- P1b: for each $e_\ell = (w_i, x_j, y_k) \notin M'$, the corresponding set in $\mathcal{U}$ contains e_ℓ and three "dummy" elements corresponding to w_i, x_j, y_k.

P2: Suppose $\mathcal{U} = \{U_1, \ldots, U_m\}$ is a 4-partition of U. For every set $U_h \in \mathcal{U}$, we conclude that

- P2a: either it consists of an e_ℓ and the "actual" elements corresponding to the elements of e_ℓ;
- P2b: or it consists of an e_ℓ and three "dummy" elements corresponding to the elements of e_ℓ.

Consequently, those $U_h \in \mathcal{U}$ satisfying property (P2a) form a matching of M.

Next, from (M, M_0, M_1) and its associated (U, B), we construct an instance $G = (A, S, E)$ of the max-CBPM problem and a constrained perfect matching N_0 of G as follows. A is a sequence of mB amino acid residues $a_1, a_2, \ldots, a_{mB}$ ordered in this order. S consists of $4m$ (ordered) strings $C_1, \ldots, C_{4m}$ of spin systems. The strings in S one-to-one correspond to the integers of U and the string C_i $(1 \leq i \leq 4m)$ corresponding to $u \in U$ has length $|C_i| = u$. Without loss of generality, we may assume that each C_i with $1 \leq i \leq m$ corresponds to $e_i \in M$ and each C_j with $m+1 \leq j \leq m+3q$ corresponds to an "actual" element of U.

For convenience, we divide sequence A into m subsequences $A_1, \ldots, A_m$ each of which consists of B consecutive amino acid residues. We call $A_1, \ldots, A_q$ the *primary* subsequences and call $A_{q+1}, \ldots, A_m$ the *secondary* subsequences. We say that an $a_i \in A$ and a $C_j \in S$ are *matchable* if

Q1: the subsequence A_k with $a_i \in A_k$ has at least $|C_j| - 1$ amino acid residues following a_i,

and exactly one of the following three conditions holds (where u is the element of U corresponding to C_j):

Q2a: $u \in M$ (u is an integer for a triple);

Q2b: A_k is primary and u is an "actual" element of U;

Q2c: A_k is secondary and u is a "dummy" element of U.

For each $a_i \in A$ and for each $C_j \in S$, if they are matchable, then for each $1 \leq h \leq |C_j|$, G has an edge between a_{i+h-1} and the h-th spin system of C_j. G has no other edges. For each edge (a_h, s_ℓ) of G, if the subsequence A_k with $a_h \in A_k$ is primary and the string $C_j \in S$ with $s_\ell \in C_j$ corresponds to a triple $e_r \in M_1$, then edge (a_h, s_ℓ) has weight $1/|C_j|$; otherwise, edge (a_h, s_ℓ) has weight 0. This completes the construction of $G = (A, S, E)$.

Suppose M' is a matching of M. We can use M' to construct a constrained perfect matching N of G as follows. First, we use M' to construct a 4-partition $\mathcal{U}$ of U satisfying properties (P1a) and (P1b) above. Then, for each $e_\ell = (w_i, x_j, y_k) \in M'$ and its corresponding set $\{e_\ell, w_i[1], x_j[1], y_k[1]\}$ in $\mathcal{U}$, we pick a primary subsequence A_h and one-to-one match its amino acid residues to the spin systems in the strings corresponding to e_ℓ, $w_i[1]$, $x_j[1]$, and $y_k[1]$. We then do the same for each $e_\ell = (w_i, x_j, y_k) \in M - M'$ and its corresponding set in $\mathcal{U}$, except that we pick a secondary subsequence A_h this time. This results in a constrained perfect matching N of G whose weight is exactly $|M' \cap M_1|$. In particular, this results in our desired N_0 when $M' = M_0$.

Conversely, suppose N is a constrained perfect matching of G. For each (primary or secondary) subsequence A_h, the integers of U corresponding to the strings matched to A_h form a subset of U. By condition (Q1) above, these subsets altogether form a 4-partition $\mathcal{U}$ of U. Now, by property (P2) and conditions (Q2a, Q2b, Q2c) above, we can use the subsets in $\mathcal{U}$ corresponding to the primary subsequences of A to construct a matching M' of M such that $|M' \cap M_1|$ is exactly the weight of N. □

Theorem 3 *If for some polynomial-time computable function $f(n)$ the above special case of the* min-CBPM *problem has a factor-$f(n)$ polynomial-time approximation algorithm, then $P = NP$.*

PROOF. The proof is done via a similar reduction from the 4-Partitioning problem. Given an instance (U, B) of the 4-Partitioning problem, we construct an instance $G = (A, S, E)$ of the min-CBPM as follows. A and S are constructed and divided into subsequences and strings as in the proof of Theorem 2 except that the integer m is replaced by t/B where $t = \sum_{u \in U} u$. E is constructed so that G is a complete bipartite graph. For each $(a, s) \in E$, if the subsequence A_k of A with $a \in A_k$ has at least $|C| - 1$ amino acid residues following a where C is the string in S with $s \in C$, then the weight of edge (a, s) is 1; otherwise, the weight of (a, s) is $f(|G|)t$. This completes the reduction.

Obviously, if U has a 4-partition, then G has a constrained perfect matching of weight t; otherwise, the weight of each constrained perfect matching of G must be at least $f(|G|)t + (t-1) > f(|G|)t$. This completes the proof. □

3 IDA* Search for Optimal Solutions

Previous search-based approaches for solving the CBPM problem optimally (typically, max-CBPM), rely on branch-and-bound techniques [14]. While branch-and-bound methods employ heuristic

estimates [3] to cut off unpromising nodes in the search tree, they still have to explore the whole tree in order to find the optimal solution. This makes them inefficient for large search trees [14]. In many cases, best-first searches are a better alternative because they lead to a solution faster (possibly at the expense of increased memory requirements). We have chosen to follow a best-first search approach for the assignment problem, in order to make search more efficient (i.e. faster). In the following we will present in detail how search is applied to min-CBPM.

One of the most popular best-first heuristic search algorithms is A* [11]. It has been used extensively in many areas of Artificial Intelligence and has been successfully applied to various bioinformatics problems, the most notable of which is probably multiple sequence alignment [17]. For each state (which is a partial solution to min-CBPM, i.e. a partial matching), A* uses both the exact distance (which is the sum of weights of edges in the partial solution) from the start state (which is the empty solution, i.e. null matching), denoted as g, and a heuristic estimate of the remaining distance (estimated by the *heuristic evaluation function*) to the goal (which is a perfect matching), denoted as h. The state with the smallest $(g+h)$ value is always expanded next and the algorithm is guaranteed to find the optimal (i.e. minimum weight) solution provided that h never overestimates the true distance to the goal state.

The main disadvantage of the A* algorithm is that it requires a lot of memory. Taking this into account, we turned to a more memory-efficient variant of A*, namely Iterative-Deepening A* (IDA*) [12], partly because its recursive implementation is fairly straightforward without requiring a lot of bookkeeping. Instead of a best-first search, IDA* performs a series of depth-first searches that are bounded by a distance threshold, which is increased successively. A node in the search tree is cut off if its $(g + h)$ value exceeds the current threshold.

We present the details of our implementation of IDA* in pseudocode form in Figure 1, but the reader may also refer to [16] for a more detailed account of the main components of IDA*. In the following two subsections we describe our main contributions to heuristic search for the min-CBPM problem, which are the development of a search-enhancing heuristic and four different heuristic evaluation functions.

3.1 Search Heuristic

The effectiveness of a search method depends on the size of the search tree that gets built. Assuming a solution of length ℓ and an average branching factor b, the size of the search tree built by IDA* is $O(b^\ell)$. We have developed a pruning technique that produces a smaller search tree and consequently a faster search.

The CBPM problem is computationally hard due to the existence of adjacency information among the spin systems, though it is informative in terms of identifying correct mappings. The pruning heuristic is based on the fact that once a state that contains only singleton spin systems has been encountered, there is no need to continue with the IDA* search. Instead, we can calculate the corresponding minimum weight *unconstrained* bipartite perfect matching, which can be done very efficiently by existing algorithms. This way, we are not only able to skip whole subtrees of the search tree but we also get an accurate evaluation distance for the root node of those subtrees.

In order to utilize the full power of this heuristic, we generate all the states corresponding to string mappings first, ignoring singleton mappings. Note that individual string/singleton assignments are independent of each other, so we can enforce this special way of building the tree without losing optimality. In fact, this way of building the search tree also reduces the running time by at least one order of magnitude. In the resulting search tree, all the internal nodes correspond

```
function Iterative_Deepening_A_Star( )
    thresh = h(root);
    repeat
        thresh = DFS(root, thresh);
    until Solution_found( )

function DFS(state, thresh)
    if Solution_found( )
        return 0;
    new_depth = ∞;
    for each successor u_i of state do
        if cost(state, u_i) + h(u_i) ≤ thresh
            if u_i contains no strings
                depth = cost(state, u_i) + CSA( );
            else
                depth = cost(state, u_i) + DFS(u_i, thresh − cost(state, u_i));
        else
            depth = cost(state, u_i) + h(u_i);
        if Solution_found( )
            return depth;
        new_depth = min(new_depth, depth);
    return new_depth;
```

Figure 1: IDA* implementation for the min-CBPM problem. *thresh* is the distance threshold, which is updated iteratively; Function Solution_found( ) checks if a solution has been found; Function $cost(state, u_i)$ returns the exact distance of going from node *state* to node u_i; Function $h(\ )$ is the heuristic evaluation function; Function CSA( ) calls the minimum weight bipartite matching algorithm CSA [8]. Our implementation also uses a transposition table for efficiency but the code has been left out for simplicity.

to mappings of strings and all the leaf nodes contain only singleton spin systems. Thus, heuristic search stops at those leaf nodes and we get an accurate evaluation for them very fast by calling the CSA algorithm to calculate the minimum weight unconstrained bipartite perfect matching. Our experimental results show that the application of this heuristic has reduced the running time of the algorithm by more than 2 orders of magnitude.

3.2 Evaluation Functions

The heuristic evaluation function is the most important component of an IDA* implementation since it can possibly provide orders of magnitude of savings. The ideal combination is an evaluation function that is accurate and at the same time inexpensive to compute. We have developed four different heuristic evaluation functions for the min-CBPM problem, which are described in the following paragraphs. The adjacency information implies that in a feasible matching the edges incident at a string must be *parallel*. For clarity, we will be referring to these parallel edges as forming a *compound* edge.

3.2.1 MIN_WEIGHTS_EVAL

Let s be a string (or a singleton); Let w_s be the minimum weight of the compound edge (or normal edge, respectively) incident at s. Then obviously, the summation of w_s over all strings and singletons, $\sum_s w_s$, is a lower bound on the optimal solution. Note that these edges might not constitute a feasible solution since some amino acid residues might be incident to more than one of them. This evaluation function may not be very accurate but it is inexpensive (about linear time) to compute. So accuracy is compensated for by speed.

3.2.2 UBM_EVAL

We have mentioned earlier that it is the adjacency constraint that complicates the problem. One natural consideration would be to drop the adjacency constraint in order to design an efficient evaluation function. It is easy to see that the minimum weight perfect matching of the corresponding unconstrained problem is always a lower bound on the minimum weight perfect matching of the min-CBPM problem. This is because the adjacency constraint in the spin systems can only prevent the selection of some of the edges in the unconstrained matching and thus increase the total weight of the constrained matching (or leave it the same). So we let this evaluation function return the weight of the unconstrained perfect matching, which can be calculated in polynomial time by a variety of existing algorithms. Typically, in our implementation we choose the CSA algorithm by Goldberg and Kennedy [8], which runs in about cubic time.

3.2.3 COLLAPSED_UBM_EVAL

Both the MIN_WEIGHTS_EVAL and the UBM_EVAL heuristic have flaws that make them inaccurate. In particular, MIN_WEIGHTS_EVAL does not resolve conflicts between the edges whereas UBM_EVAL does not use any information about strings. The COLLAPSED_UBM_EVAL heuristic was developed in order to remedy those drawbacks by incorporating string information into the UBM_EVAL heuristic. This is done by replacing parallel edges with compound edges. Let $s_{i,i+k}$ denote a string (of length $k+1$) that can be matched to a run of amino acid residues a_j through a_{j+k}. The compound edge is created to connect s_{i+k} and a_{j+k} with its weight being the sum of the weights of those $k+1$ parallel edges. Subsequently, those k parallel edges connecting s_{i+p} and a_{j+p}, for $0 \leq p < k$, are set to have weight 0. The edges out of singleton spin systems remain unchanged. The weight of every other edge in the graph (i.e. one that hasn't been processed in the steps just described) is set to infinity. This transformed edge-weighted bipartite graph is then passed to UBM_EVAL.

We want to remark that the weights of the parallel edges are set to zero in order not to affect the weight of the matching returned by UBM_EVAL. Also, if any edges with a weight of infinity are selected in the matching, it is an indication that the corresponding tree branch is a dead-end and we have a cut-off in the search. Notice that we could remove the edges with infinite weight from the graph altogether. However, the current implementation of CSA requires the existence of a perfect matching otherwise it goes into an infinite loop. Removing the edges with infinite weight may exclude the presence of a perfect matching when there is a dead-end.

3.2.4 PARTIAL_UBM_EVAL

This fourth heuristic implements another way to combine MIN_WEIGHTS_EVAL and UBM_EVAL. It first asks MIN_WEIGHTS_EVAL to get the weight for a partial matching involving some of the strings

(typically, long strings) and then asks UBM_EVAL to get the weight for the matching in the remaining bipartite graph. In more detail, some of the strings are selected and MIN_WEIGHTS_EVAL is used to find the minimum weight edges coming out of them in the usual fashion. The weights of these (compound) edges are then added together, producing a distance W_m. The next step is to set the weight of all edges that are incident at the spin systems in strings that have been processed by MIN_WEIGHTS_EVAL to 0. This is necessary since we do not want these edges to affect the matching calculated for the remaining bipartite graph. UBM_EVAL is then applied to the remaining graph to produce another distance W_u. The final evaluation distance is equal to $W_m + W_u$.

4 Experimental Results and Discussions

In order to evaluate our implementation we have used a test suite of 70 instances. These consist of 14 proteins extracted from BioMagResBank (`http://www.bmrb.wisc.edu`) without solved atomic structures and 5 adjacency information densities for each protein. The adjacency densities range from 50% to 90%, which together with the protein length represent the size of the search space. Note that the protein data obtained from BioMagResBank contains complete (i.e. 100%) adjacency information. For our simulation purposes, we created instances with different levels of adjacency densities, where a density of p% indicates that p% of the adjacent spin system pairs are randomly retained while $(100 - p)$% are left out to mimic missing adjacency.

In detail, the min-CBPM instances were generated in the following way: We obtained the backbone chemical shifts of H, N, C_α, and C_β for each of those 14 proteins from BioMagResBank to generate the bipartite graphs using the scoring scheme trained in [19] and used the five levels of pseudo-adjacency densities presented in [21]. We used this typical combination of chemical shifts only because they can be obtained from three NMR spectra HSQC, HNCACB, and CBCACONH, which are typically done in most NMR labs. We remark that these 14 proteins were not used to derive the scoring scheme so as to avoid any possible bias. This specific scoring scheme uses extra information, additional to the spin systems, to better differentiate between residue types and has been typically designed to handle missing chemical shifts. All experiments were performed on an AIX IBM cluster with 32GB of memory and 16 PowerPC processors running at 1.7GHz. The running times reported here are all user times in seconds.

From a computational complexity point of view, 0% adjacency density is an easy case (i.e. the BPM problem). The complexity increases as the adjacency density rises but decreases after the adjacency density passes some limit and eventually becomes another easy case when the adjacency density reaches 100% (i.e. the CBPM problem with a single string of all spin systems). In this sense, we might claim that for each protein, instances with higher adjacency density (for example 90%) are easier than instances with lower adjacency density (for example 50%). This has been verified by the computational experiments. Table 1 presents the detailed results for the spin system assignment instances. In summary, our IDA* algorithm was able to solve 65 out of 70 instances and 43 of them were solved in less than 15 seconds. The remaining 5 instances could not be solved within the time allocated for each instance, which was 2 days. These instances are the most difficult instances for the proteins involved and their adjacency densities are 50% and 60%. In practice, the percentage of adjacency information available to NMR labs is usually 70% or more, for example, from the three NMR spectra HSQC, HNCACB, and CBCACONH. Therefore, these unsolved instances do not diminish the power of our approach. On the contrary, we have re-run the branch-and-bound algorithm [14] on the same computer and within the same 2-day time limit, the number of instances

	$\|A\|$	W_M	$T_{\text{IDA}*}$	T_{BnB}	$T_{\text{GF-BnB}}$	$M_{\text{IDA}*}$	$M_{\text{GF-BnB}}$	HEF
bmr4027_9	158	10366	1	1	1	**158**	156	MW
bmr4144_9	78	6556	1	1	1	**78**	78	MW
bmr4288_9	105	7923	1	1	1	**105**	105	MW
bmr4302_9	115	7736	1	1	1	**115**	115	MW
bmr4309_9	178	8056	1	170	1	**178**	178	MW
bmr4316_9	89	4704	1	1	1	**89**	89	MW
bmr4318_9	215	14674	1	5	1	**215**	215	MW
bmr4353_9	126	8068	1	1	1	**126**	126	MW
bmr4391_9	66	4858	1	1	1	**66**	66	MW
bmr4393_9	156	5781	1	120	1	**156**	156	MW
bmr4579_9	86	5418	1	1	1	**86**	86	MW
bmr4670_9	120	7892	1	1	1	**120**	120	MW
bmr4752_9	68	4094	1	1	1	**68**	68	MW
bmr4929_9	114	7113	1	2	1	**114**	114	MW
bmr4027_8	158	10366	1	41	4	**158**	156	MW
bmr4144_8	78	6556	1	1	1	**78**	76	MW
bmr4288_8	105	7923	1	1	1	**105**	105	MW
bmr4302_8	115	7694	1	70	1	**112**	112	MW
bmr4309_8	178	8038	12	17233	42	**172**	173	MW
bmr4316_8	89	4704	1	7	1	**89**	89	MW
bmr4318_8	215	14667	5	*	12	**213**	211	MW
bmr4353_8	126	8068	1	149	1	**126**	126	MW
bmr4391_8	66	4858	1	1	1	**66**	66	MW
bmr4393_8	156	5773	13	*	3	**139**	144 (5784)	MW
bmr4579_8	86	5418	1	1	1	**86**	86	MW
bmr4670_8	120	7892	1	366	1	**120**	120	MW
bmr4752_8	68	4094	1	1	1	**68**	68	MW
bmr4929_8	114	7113	1	57	2	**114**	114	MW
bmr4027_7	158	10341	11	*	18528	**149**	150	MW
bmr4144_7	78	6554	1	245	36	**70**	68	MW
bmr4288_7	105	7921	2	34	35	**99**	97	MW
bmr4302_7	115	7683	7	*	495	**106**	106	MW
bmr4309_7	178	7853	496	*	*	**150**	142 (8304)	MW
bmr4316_7	89	4704	1	73	1	**89**	87	MW
bmr4318_7	215	14661	1028	*	*	**212**	168 (15233)	MW

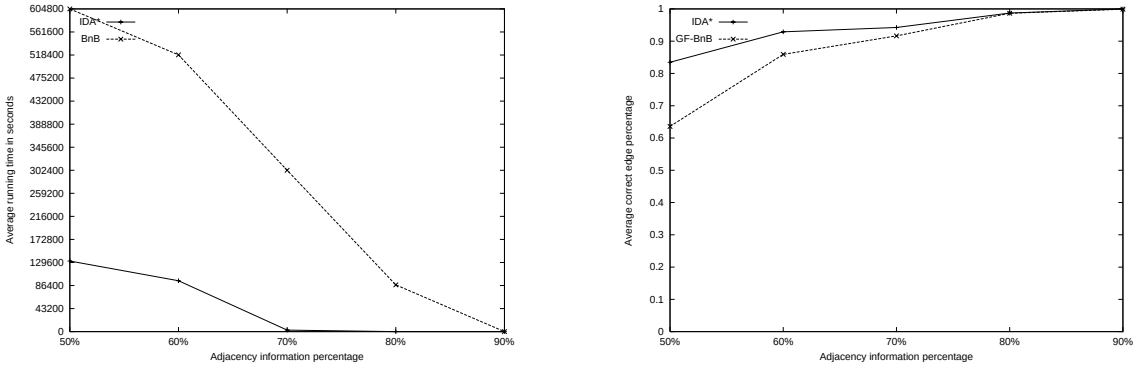
	$\|A\|$	W_M	$T_{\text{IDA}*}$	T_{BnB}	$T_{\text{GF-BnB}}$	$M_{\text{IDA}*}$	$M_{\text{GF-BnB}}$	HEF
bmr4353_7	126	8066	2	*	6	**124**	122	MW
bmr4391_7	66	4858	1	114	1	**66**	64	MW
bmr4393_7	156	5718	38355	*	122963	**111**	112 (5771)	MW
bmr4579_7	86	5418	1	48	7	**86**	86	MW
bmr4670_7	120	7880	2	*	695	**118**	116	MW
bmr4752_7	68	4094	1	1	1	**68**	68	MW
bmr4929_7	114	7111	1	425	47	**112**	112	MW
bmr4027_6	158	10321	779	*	518400	**144**	131 (10971)	MW
bmr4144_6	78	6504	27	2558	177989	**73**	69	PART
bmr4288_6	105	7921	65	*	696182	**99**	76 (8282)	PART
bmr4302_6	115	7635	111	*	*	**101**	100	PART
bmr4309_6	178	-	*	*	355450	**-**	100 (8530)	-
bmr4316_6	89	4704	1	1062	66	**89**	84	MW
bmr4318_6	215	14438	123896	*	355913	**178**	143 (15957)	COLL
bmr4353_6	126	8063	258	*	*	**121**	109 (8155)	MW
bmr4391_6	66	4847	1	*	9	**60**	51 (4876)	MW
bmr4393_6	156	-	*	*	*	**-**	77 (6245)	-
bmr4579_6	86	5407	211	*	72	**79**	80	MW
bmr4670_6	120	7773	9	*	381454	**113**	114	MW
bmr4752_6	68	4094	2	*	482	**68**	66	MW
bmr4929_6	114	7068	8	*	109082	**105**	103	MW
bmr4027_5	158	10211	24645	*	355912	**121**	103 (11518)	COLL
bmr4144_5	78	6440	40	*	*	**64**	21 (9214)	PART
bmr4288_5	105	7914	169	*	432000	**97**	50 (12107)	PART
bmr4302_5	115	7621	1782	*	355897	**95**	83 (8089)	PART
bmr4309_5	178	-	*	*	255874	**-**	61 (9044)	-
bmr4316_5	89	4703	1	*	*	**87**	76	PART
bmr4318_5	215	-	*	*	355874	**-**	89 (18454)	-
bmr4353_5	126	7919	2090	*	*	**91**	76 (8958)	COLL
bmr4391_5	66	4811	14	*	708	**52**	51 (4815)	MW
bmr4393_5	156	-	*	*	355837	**-**	65 (6325)	-
bmr4579_5	86	5310	2223	*	*	**57**	52 (5635)	PART
bmr4670_5	120	7736	6442	*	172800	**108**	65 (9488)	MW
bmr4752_5	68	4090	4	*	77233	**60**	58	UBM
bmr4929_5	114	7003	810	*	432000	**104**	73 (8330)	MW

Table 1: Computational results for all 70 instances. The number beside the underscore in the instance name indicates the adjacency information (e.g. _5 indicates an adjacency density of 50%). '$|A|$' is the number of amino acid residues (the length of the protein), while 'W_M' is the weight of an optimal matching (from IDA*). '$T_{\text{algorithm}}$' is the running time, in seconds, for the first solution found by the corresponding algorithm and '$M_{\text{algorithm}}$' is the number of correct matching edges retrieved accordingly. Note that a * in the '$T_{\text{algorithm}}$' column indicates that the algorithm didn't terminate in 2 days and the numbers inside the parentheses in the $M_{\text{GF-BnB}}$ column are the distances returned by GF-BnB (not optimal). Finally, 'HEF' is the name of the evaluation function that performed the best for that instance. 'MW' stands for MIN_WEIGHTS_EVAL, 'UBM' stands for UBM_EVAL, 'COLL' stands for COLLAPSED_UBM_EVAL and 'PART' stands for PARTIAL_UBM_EVAL.

solved was only 35 out of 70; even worse, there were 2 _8 instances and 7 _7 instances that couldn't be solved. The average running times for different levels of adjacency densities of our method and the branch-and-bound algorithm are plotted in Figure 2(a), where the running times for those unsolved instances were all set to 7 days (i.e. 604800 seconds). It should be noted that our method is several orders faster than the branch-and-bound algorithm, something that becomes even more significant when the adjacency density drops.

The above comparison is made to demonstrate the efficiency of our IDA* algorithm in searching for optimal solutions to the min-CBPM problem. The following comparison presents the advantage of searching for optimal solutions over near-optimal solutions, quantified by the number of spin systems correctly mapped to their host amino acid residues in the target protein. This comparison also shows the strength of the CBPM formulation of the NMR resonance assignment problem.

The solutions found by IDA* were compared to those produced by greedy filtering followed by branch-and-bound (GF-BnB) [14]. The results show that our method succeeded in retrieving more correct mappings for every instance. This should come as no surprise though since the use of the greedy heuristics — greedy filtering — degrades the quality of the solution in terms of assignment confidence. More importantly though, in most instances our approach ran faster even compared to GF-BnB, as can be seen in Table 1. The statistics in this table were obtained by re-running the GF-BnB algorithm on the same computer. The average percentages of correct edges recovered for different levels of adjacency densities by our method and by the GF-BnB algorithm are plotted in Figure 2(b), where those instances not solved by our method were excluded from the statistics. From that, we might be able to claim that, prior to our method, there existed no approach for the



(a) Average running time comparison between IDA* and BnB: IDA* significantly outperforms BnB when the adjacency density decreases.

(b) Average percentage of correct edges recovered by IDA* and GF-BnB: GF-BnB performs much worse than IDA* when the adjacency density decreases.

Figure 2: Performance comparison between IDA* and (GF-)BnB.

NMR resonance assignment problem that combined optimality of solutions with such small running times.

As mentioned above, for the same protein, the instances get harder as the number of strings increases while their length decreases, i.e. the adjacency density decreases. The instances not solved by our method in 2 days are the ones with the largest number of amino acid residues and contain more strings and singletons than other instances in their group. We have modified the search algorithm to continue searching after finding the first solution so that all optimal solutions are retrieved. For the unsolved instances, the solved instances in the same category (that is, constructed for the same protein but with higher adjacency densities) have a very large number of optimal solutions. This evidence may provide an additional explanation for the difficulty of a particular instance. It is an indication that sometimes the scoring scheme does not differentiate very well between alternative matchings (something that might be inherent in the spectra data), so the search effort needed to produce the optimal solution is much greater. Nonetheless, as the reader can imagine, using the same greedy filtering as in [14], our IDA* search algorithm can easily find the *conditional* optimal solutions for those five unsolved instances too. So far we don't have the complete set of results on this aspect, so more experimental results will be reported later.

Coming back to the IDA* algorithm itself, it is characteristic that no single evaluation function

outperforms all the others in the experiments. It is clear though that MIN_WEIGHTS_EVAL, although rather simplistic, cannot be beaten for the easy instances, (that is, instances with adjacency density $\geq 70\%$). However, the performance of the evaluation functions changes as the instances become more difficult. This is due to the fact that the strings increase in number but become shorter, so UBM_EVAL-based evaluations become more accurate. By keeping track of the computations within the 2-day period, we have observed that COLLAPSED_UBM_EVAL and PARTIAL_UBM_EVAL clearly outperform the other evaluation functions in the unsolved instances. It is characteristic that in some of the hard instances MIN_WEIGHTS_EVAL appeared to be faster initially but other evaluation functions proved more efficient in the long-run.

Our model focuses mainly on string mappings since they have a higher degree of confidence. Singleton mappings are not as reliable but can affect the subsequent structure determination, so our implementation offers a few options to handle them better. Users can specify either that no singletons be mapped (and just get a list of all singletons and remaining residues) or that all possible singleton mappings be returned (currently we use IDA* together with CSA to enumerate them, another alternative enumerating algorithm would be from [18]). The latter is aimed at providing extra aid in the manual mapping of singletons. Moreover, in a parallel work of research, we have developed an evaluation model [13] that can assign confidence levels to the mapping edges contained in the optimal assignment, thus providing a quantified measure for the quality of the proposed mappings.

5 Future Work

Although our current work focuses mainly on the computational side of the NMR backbone resonance assignment problem, one of our objectives is to develop a fully automated tool for NMR backbone resonance assignment that will be both robust and efficient. This will considerably speed up the protein structure determination process via NMR spectroscopy and it will transform it from a time-consuming method to a high-throughput technique.

With this in mind, we are currently trying to extend our approach into a more general framework that will be oriented more towards full protein structure determination. One step towards this is the integration of side-chain nuclei into the backbone assignment, as our experiments do not currently take full advantage of them. Moreover, we have been examining the possibility of utilizing structural constraints (i.e. NOE spectral peaks, J-coupling constants, Residual Dipolar Coupling constants) in order to improve the assignment. We are aware though that this might introduce "false" adjacency information, possibly requiring a modification of the CBM model.

On the computational side, we believe the most promising area of improvement lies in the search algorithm. We are currently in the process of implementing other variants of the A* algorithm, such as *Partial-Expansion A** [22], to see how they perform on this particular problem, especially on the unsolved instances. On the other hand, we are always looking into developing better evaluation functions. There is existing work [3, 4] on approximating the max-CBM problem. We are currently pursuing how we may take advantage of it.

Another direction we would like to follow is to better evaluate our approach against existing approximate methods, as well as methods proposed for other formulations of the NMR resonance assignment problem [2, 10, 23]. Such an evaluation becomes necessary when they are applied to real peak assignment instances where the correct solutions are unknown to us. Along this line, we are in the process of acquiring some real peak assignment data to facilitate our experiments. We

are also planning to conduct an extensive study on greedy filtering [3, 4] followed by IDA*.

Acknowledgments

The authors are grateful to two referees for their many helpful suggestions and comments. The research of TT and GL was supported by NSERC, PENCE, AICML, and CFI. The research of ZZC was supported in part by the Grant-in-Aid for Scientific Research of the Ministry of Education of Japan, under Grant No. 14580390.

References

[1] M. C. Baran, Y. J. Huang, H. N. Moseley, and G. T. Montelione. Automated analysis of protein NMR assignments and structures. *Chemical Reviews*, 104:3541–3556, 2004.

[2] 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, 1997.

[3] Z.-Z. Chen, T. Jiang, G.-H. Lin, J. J. Wen, D. Xu, J. Xu, and Y. Xu. Approximation algorithms for NMR spectral peak assignment. *Theoretical Computer Science*, 299:211–229, 2003.

[4] Z.-Z. Chen, T. Jiang, G.-H. Lin, J. J. Wen, D. Xu, and Y. Xu. Improved approximation algorithms for NMR spectral peak assignment. In *Proceedings of the* 2*nd Workshop on Algorithms in Bioinformatics (WABI 2002)*, LNCS 2452, pages 82–96, 2002.

[5] Z.-Z. Chen, G.-H. Lin, R. Rizzi, J. J. Wen, D. Xu, Y. Xu, and T. Jiang. More reliable protein NMR peak assignment via improved 2-interval scheduling. *Journal of Computational Biology*, 12(2):129–146, 2005.

[6] A. E. Ferentz and G. Wagner. NMR spectroscopy: a multifaceted approach to macromolecular structure. *Quarterly Review Biophysics*, 33:29–65, 2000.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[8] A. V. Goldberg and R. Kennedy. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71:153–178, 1995.

[9] W. Gronwald and H. R. Kalbitzer. Automated structure determination of proteins by NMR spectroscopy. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 44:33–96, 2004.

[10] 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.

[11] N. J. Hart, P. E. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.

[12] R. E. Korf. Depth-first iterative deepening: an optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[13] G.-H. Lin, X. Wan, T. Tegos, and Y. Li. Statistical evaluation of NMR backbone resonance assignment. *International Journal of Bioinformatics Research and Applications*, 2005. In press.

[14] G.-H. Lin, D. Xu, Z. Z. Chen, T. Jiang, J. J. Wen, and Y. Xu. An efficient branch-and-bound algorithm for the assignment of protein backbone NMR peaks. In *Proceedings of the First IEEE Computer Society Bioinformatics Conference (CSB 2002)*, pages 165–174. IEEE Computer Society Press, 2002.

[15] D. Malmodin, C. H. Papavoine, and M. Billeter. Fully automated sequence-specific resonance assignments of hetero-nuclear protein spectra. *Journal of Biomolecular NMR*, 27:69–79, 2003.

[16] A. Reinefeld and T. A. Marsland. Enhanced iterative-deepening search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):701–710, 1994.

[17] I. Takahiro and I. Hiroshi. Fast A* algorithms for multiple sequence alignment. In *Genome Informatics Workshop 94*, pages 90–99, 1994.

[18] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Processings of the 8th International Symposium on Algorithms and Computation (ISAAC 1997)*, LNCS 1350, pages 92–101, 1997.

[19] X. Wan, D. Xu, C. M. Slupsky, and G.-H. Lin. Automated protein NMR resonance assignments. In *Proceedings of the Second IEEE Computer Society Bioinformatics Conference (CSB 2003)*, pages 197–208. IEEE Computer Society Press, 2003.

[20] K. Wüthrich. *NMR of Proteins and Nucleic Acids*. Wiley, John & Sons, New York, 1986.

[21] 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.

[22] T. Yoshizumi, T. Miura, and T. Ishida. A* with partial expansion for large branching factor problems. In *National Conference on Artificial Intelligence (AAAI-00)*, pages 923–929, 2000.

[23] 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.

[24] D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *Proceedings of the 8th IEEE Conference on Structure in Complexity Theory*, pages 305–312, 2003.