More Reliable Protein NMR Peak Assignment via Improved 2-Interval Scheduling

Zhi-Zhong Chen [*] Tao Jiang [†] Guohui Lin [‡] Romeo Rizzi [§] Jianjun Wen [¶]
Dong Xu [‖] Ying Xu [**]

Abstract

Protein NMR peak assignment refers to the process of assigning a group of "spin systems" obtained experimentally to a protein sequence of amino acids. The automation of this process is still an unsolved and challenging problem in NMR protein structure determination. Recently, protein NMR peak assignment has been formulated as an *interval scheduling* problem, where a protein sequence $\mathcal{P}$ of amino acids is viewed as a discrete time interval $\mathcal{I}$ (the amino acids on $\mathcal{P}$ one-to-one correspond to the time units of $\mathcal{I}$), each subset S of spin systems that are known to originate from consecutive amino acids from $\mathcal{P}$ is viewed as a "job" j_S, the preference of assigning S to a subsequence P of consecutive amino acids on $\mathcal{P}$ is viewed as the profit of executing job j_S in the subinterval of $\mathcal{I}$ corresponding to P, and the goal is to maximize the total profit of executing the jobs (on a single machine) during $\mathcal{I}$. The interval scheduling problem is Max SNP-hard in general; but in the real practice of protein NMR peak assignment, each job j_S usually requires at most 10 consecutive time units, and typically the jobs that require one or two consecutive time units are the most difficult to assign/schedule. In order to solve these most difficult assignments, we present an efficient $\frac{13}{7}$-approximation algorithm for the special case of the interval scheduling problem where each job takes one or two consecutive time units. Combining this algorithm with a greedy filtering strategy for handling long jobs (*i.e.* jobs that need more than two consecutive time units), we obtain a new efficient heuristic for protein NMR peak assignment. Our experimental study shows that the new heuristic produces the best peak assignment in most of the cases, compared with the NMR peak assignment algorithms in the recent literature. The above algorithm is also the first approximation algorithm for a nontrivial case of the well-known interval scheduling problem that breaks the ratio 2 barrier.

Keywords: structural genomics, computational biology, protein NMR peak assignment, approximation algorithm, interval scheduling, constrained bipartite matching.

[*]Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan. Part of work done while visiting at University of Alberta. Email: chen@r.dendai.ac.jp.

[†]Supported in part by NSF Grants CCR-9988353 and ITR-0085910, and National Key Project for Basic Research (973). Department of Compute Science, University of California, Riverside, CA 92521, and Shanghai Center for Bioinformation Technology. Email: jiang@cs.ucr.edu.

[‡]Supported in part by NSERC grants RGPIN249633 and A008599, PENCE, and Startup Grant REE-P5-01-02-Sci from the University of Alberta. Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: ghlin@cs.ualberta.ca.

[§]Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Italy Email: romeo@science.unitn.it

[¶]Department of Computer Science, University of California, Riverside, CA 92521. Email: wjianju@cs.ucr.edu.

[‖]Supported by the Office of Biological and Environmental Research, U.S. Department of Energy, under Contract DE-AC05-00OR22725, managed by UT-Battelle, LLC. Protein Informatics Group, Life Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6480, USA. Email: xud@ornl.gov.

[**]Supported by the Office of Biological and Environmental Research, U.S. Department of Energy, under Contract DE-AC05-00OR22725, managed by UT-Battelle, LLC. Protein Informatics Group, Life Sciences Division and Computer Sciences and Mathematics Division, Oak Ridge National Laboratory. Oak Ridge, TN 37831-6480, USA. Email: xyn@ornl.gov.

1 Introduction

Due to the efforts of structural genomics [7], the NMR (nuclear magnetic resonance) technique has been used as a high-throughput technology to solve protein structures at a genome scale. Typically, protein structure determination via NMR involves the following steps:

- NMR spectral data generation, which produces
  - resonance peaks corresponding to amino acids in the target protein sequence. Peaks corresponding to a common amino acid are grouped into a *spin system*;
  - certain geometric relationships (*e.g.* distances and angles) between the spin systems;
- Peak picking, which identifies "real" resonance peaks (peaks generated from protein atoms rather than noise) from NMR spectral maps.
- Peak assignment, which assigns resonance peaks, typically peak groups, to individual residues of the target protein sequence.
- Structural restraint extraction, which extracts inter-residue distances, dihedral angles, etc., based on the peak assignment.
- Structure calculation, which calculates the protein structure, using molecular simulation and energy minimization, under the identified NMR restraints.

Among the five steps, the third one (namely, NMR peak assignment) is very time consuming. The process usually takes weeks or sometimes even months of manual work in order to produce a nearly complete assignment. The automation of the assignment process is still an unsolved and challenging problem in NMR protein structure determination.

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

- The likelihood (or weight) of the matching between a spin system and an amino acid on the protein sequence.
- The sequential adjacency (*i.e.*, consecutivity) information of some subsets of spin systems (*i.e.*, each such subset of spin systems should correspond to a subsequence of consecutive amino acids on the host protein sequence). Each maximal such subset is called a *segment of spin systems*. It is worth noting that each segment usually consists of at most 10 spin systems.

In a recently developed computational framework [8], the NMR peak assignment problem has been formulated as an *interval scheduling* problem [1] as follows. A protein sequence $\mathcal{P}$ of amino acids is viewed as a discrete time interval $\mathcal{I}$ (the amino acids on $\mathcal{P}$ one-to-one correspond to the time units of $\mathcal{I}$). Each segment S of spin systems is viewed as a job j_S. Each job j_S requires $|S|$ consecutive time units of $\mathcal{I}$ (this corresponds to the requirement that the spin systems in S should be assigned to $|S|$ consecutive amino acids on $\mathcal{P}$). For each time unit t of $\mathcal{I}$, the profit $w(j_S, t)$ of starting executing job j_S at time unit t and finishing at time unit $t + |S| - 1$ of $\mathcal{I}$ corresponds to the preference of assigning the spin systems in S to those $|S|$ consecutive amino acids on $\mathcal{P}$ that correspond to the time units $t, t+1, \ldots, t+|S|-1$. Given $\mathcal{I}$, the jobs j_S, and the profits $w(j_S, t)$, our goal is to maximize the total profit of the executed jobs (*i.e.* we want to find a maximum-likelihood assignment of the given spin systems to the amino acids on $\mathcal{P}$).

Unfortunately, the interval scheduling problem is Max SNP-hard [3, 4]. Indeed, for every integer $k \geq 2$, the special case of the interval scheduling problem (called the *k-interval scheduling problem* or *k-ISP* for short) where each job requires at most k consecutive time units is Max SNP-hard. On the other hand, several 2-approximation algorithms for the interval scheduling problem are known [2, 1, 3, 4]. Although these algorithms are theoretically sound, applying them to protein NMR peak assignment produces unsatisfactory assignments as demonstrated in [3]. A major reason why these algorithms do not have good performance in protein NMR peak assignment is that they ignore the following important observation:

[1]In [8], it was called the *constrained bipartite matching* problem.

- In the real practice of protein NMR peak assignment, long segments S of spin systems are typically easier to assign than shorter segments. In fact, many long segments have unique matches. On the other hand, segments consisting of one or two spin systems are often very difficult to assign.

The above observation suggests the following heuristic framework for protein NMR peak assignment: first try to assign segments consisting of at least $k+1$ spin systems for some small integer k (say, $k=2$), and then solve an instance of k-ISP. In [6], we have presented such a heuristic and have shown that it is very effective for protein NMR peak assignment. A major drawback of the heuristic in [6] is that it uses an inefficient branch-and-bound algorithm for k-ISP.

In order to improve the efficiency of the heuristic in [6], we present a new approximation algorithm for 2-ISP in this paper. This algorithm achieves an approximation ratio of $\frac{13}{7}$ and is the first approximation algorithm for a nontrivial case of the interval scheduling problem that breaks the ratio 2 barrier. Our algorithm is quite nontrivial – it consists of four separate algorithms and outputs the best solution returned by them. The main tool used in the algorithm design is maximum-weight bipartite matching and careful manipulation of the input instance. Substituting the new algorithm for the branch-and-bound algorithm in the heuristic in [6], we obtain a new heuristic for protein NMR peak assignment. We have performed extensive experiments on 70 instances of (pseudo) real NMR data derived from 14 proteins to evaluate the performance of our new heuristic in terms of (i) the weight of the assignment and (ii) the number of correctly assigned resonance peaks. The experimental results show that not only does the new heuristic run very fast, it also produces the best peak assignment on most of the instances, compared with the protein NMR peak assignment algorithms in the recent literature [3, 4, 6, 8]

The rest of the paper is organized as follows. The $\frac{13}{7}$-approximation algorithm for 2-ISP is presented in Section 2. In Section 3, we consider an interesting special profit function in interval scheduling, and present a $(1.5+\epsilon)$-approximation algorithm for 2-ISP under this special profit function for any $\epsilon > 0$. This improves on an approximation result in [4]. In Section 4, we describe our new heuristic for protein NMR peak assignment based on the $\frac{13}{7}$-approximation algorithm for 2-ISP, and give the experimental results. We end this paper with a short discussion in Section 5.

2 A new approximation algorithm for 2-ISP

Let $\mathcal{I}$ be the given discrete time interval. Without loss of generality, we may assume that $\mathcal{I} = [0, I]$. Let $\mathcal{J}_1 = \{v_1, v_2, \ldots, v_{n_1}\}$ be the given set of jobs requiring one time unit of $\mathcal{I}$. Let $\mathcal{J}_2 = \{v_{n_1+1}, v_{n_1+3}, \ldots, v_{n_1+2n_2-1}\}$ be the given set of jobs requiring two contiguous time units of $\mathcal{I}$. Note that $n_1 + n_2$ is the total number of given jobs. For each $1 \le i \le I$, let u_i denote the time unit $[i-1, i]$ of $\mathcal{I}$. Let $U = \{u_i \mid 1 \le i \le I\}$. Let $\mathcal{J}_2' = \{v_{n_1+2}, v_{n_1+4}, \ldots, v_{n_1+2n_2}\}$. Let $V = \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_2'$. We construct an edge-weighted bipartite graph G with color classes U and V as follows: For every $v_j \in \mathcal{J}_1$ and every $u_i \in U$ such that the profit of executing job v_j in time unit u_i is positive, (u_i, v_j) is an edge of G and its weight is the profit. Similarly, for every $v_j \in \mathcal{J}_2$ and every $u_i \in U$ such that the profit of executing job v_j in the two time units u_i, u_{i+1} is positive, both (u_i, v_j) and (u_{i+1}, v_{j+1}) are edges of G and the weight of each of them is half the profit. Figure 1 shows an example of G.

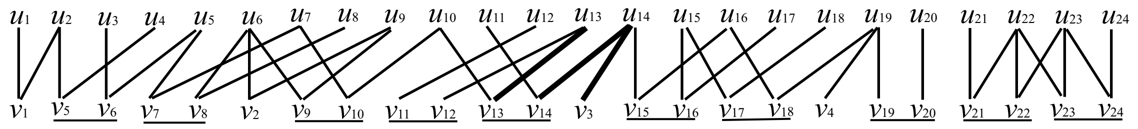

u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17} u_{18} u_{19} u_{20} u_{21} u_{22} u_{23} u_{24}
v_1 v_5 v_6 v_7 v_8 v_2 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_3 v_{15} v_{16} v_{17} v_{18} v_4 v_{19} v_{20} v_{21} v_{22} v_{23} v_{24}

Figure 1: An example of G. The three bold edges (u_{13}, v_{13}), (u_{14}, v_{14}), and (u_{14}, v_3) each have weight 3, and the other edges each have weight 1. Also, for each $v_j \in \mathcal{J}_2$, v_j and v_{j+1} are underlined together. In other words, $n_1 = 4$.

A *constrained matching* of G is a matching M of G such that for every $u_i \in U$ and every $v_j \in \mathcal{J}_2$, $(u_i, v_j) \in M$ if and only if $(u_{i+1}, v_{j+1}) \in M$. The objective of 2-ISP is equivalent to finding a maximum-weight constrained matching in G. For each edge (u_i, v_j) of G, let $w(u_i, v_j)$ denote the weight of the edge. For convenience, let $w(u_i, v_j) = 0$ for all $(u_i, v_j) \notin E$. For a (constrained or unconstrained) matching M of G, let $w_1(M)$ (respectively, $w_2(M)$) denote the total weight of edges $(u_i, v_j) \in M$ with $v_j \in \mathcal{J}_1$ (respectively, $v_j \in \mathcal{J}_2 \cup \mathcal{J}'_2$); let $w(M) = w_1(M) + w_2(M)$.

Let M^* be a maximum-weight constrained matching in G. In Sections 2.1, 2.3 through 2.5, we will design four algorithms each outputting a constrained matching in G. We will try to find a large constant ϵ such that the heaviest one among the four output matchings is of weight at least $(\frac{1}{2} + \epsilon)w(M^*)$. It will turn out that $\epsilon = \frac{1}{26}$. So, fix $\epsilon = \frac{1}{26}$ for the discussions in the rest of this section.

2.1 Algorithm 1

This algorithm will output a constrained matching of large weight when $w_2(M^*)$ is relatively large compared with $w_1(M^*)$. We first explain the idea behind the algorithm. Suppose that we partition the time interval $\mathcal{I}$ into shorter intervals, called *basic intervals*, in such a way that each basic interval, except possibly the first and the last (which may possibly consist of 1 or 2 time units), consists of 3 time units. There are exactly three such partitions of $\mathcal{I}$ (see Figure 2). Denote them by P_0, P_1, and P_2, respectively. With respect to each P_h with $0 \le h \le 2$, consider the problem $\mathcal{Q}_h$ of finding a constrained scheduling which maximizes the total profit of the executed jobs, but subject to the constraint that each basic interval in P_h can be assigned to at most one job and each executed job should be completed within a single basic interval in P_h. It is not so hard to see that each problem $\mathcal{Q}_h$ requires the computation of a maximum-weight (unconstrained) matching in a suitably constructed bipartite graph, and hence can be solved in polynomial time.

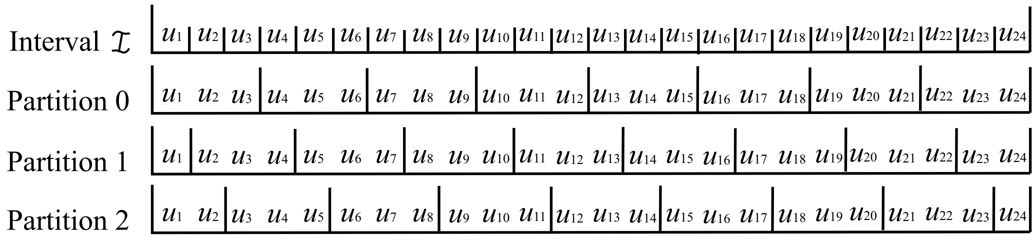

Interval $\mathcal{I}$
Partition 0
Partition 1
Partition 2
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17} u_{18} u_{19} u_{20} u_{21} u_{22} u_{23} u_{24}

Figure 2: An example of interval $\mathcal{I}$ and its partitions.

We claim that among the three problems $\mathcal{Q}_h$, the best one gives a scheduling by which the executed jobs achieve at least a total profit of $\frac{1}{2}w_1(M^*) + \frac{2}{3}w_2(M^*)$. This claim is actually easier to see, if we refer to a more constrained scheduling problem $\mathcal{Q}'_h$ than $\mathcal{Q}_h$ by adding the following constraint:

- For each job $v_j \in \mathcal{J}_1$ and for each basic interval b in P_h, only the *primary* time unit of b can be assigned to v_j, where the *primary* time unit of b, is u_i if b consists of three time units $u_{i-1}u_iu_{i+1}$, is u_1 if b consists of the first two time units u_1u_2 of $\mathcal{I}$, is u_I if b consists of the last two time units $u_{I-1}u_I$ of $\mathcal{I}$, is b itself if b consists of one time unit only.

  [Comment: The crux is that for each basic interval b consisting of at least two time units, each sub-interval of b consisting of two time units must contain the primary time unit of b. Thus, by this constraint, we are allowed to assign at most one job to each basic interval. In turn, $\mathcal{Q}'_h$ is more constrained than $\mathcal{Q}_h$.]

Consider an optimal (unconstrained) scheduling M^*. For each job $v_j \in \mathcal{J}_2$, if M^* assigns v_j to two time units u_iu_{i+1}, then this assignment of v_j is also valid in exactly two problems among $\mathcal{Q}'_0$, $\mathcal{Q}'_1$, and $\mathcal{Q}'_2$, because there are exactly two indices $h \in \{0, 1, 2\}$ such that some basic interval in P_h contains both time units u_iu_{i+1}. Similarly, for each job $v_j \in \mathcal{J}_1$, if M^* assigns v_j to one time unit u_i, then this assignment of v_j is also valid in at least one problem among $\mathcal{Q}'_0$, $\mathcal{Q}'_1$, and $\mathcal{Q}'_2$, because there is at least one index $h \in \{0,1,2\}$ such that u_i is the primary time unit of some basic interval in P_h. Thus, by inheriting from the optimal scheduling M^*, the three problems $\mathcal{Q}'_h$ have more-constrained schedulings M^*_h such that M^*_h is a sub-scheduling of M^* and the three schedulings M^*_h altogether achieve at least a total profit of $w_1(M^*) + 2w_2(M^*)$. Hence, the best more-constrained scheduling among M^*_1, M^*_2, and M^*_3 achieves at least a total profit of $\frac{1}{2}w_1(M^*) + \frac{2}{3}w_2(M^*)$. Indeed, we can prove the following better bound which is needed in later sections:

> The best more-constrained scheduling among M^*_1, M^*_2, and M^*_3 achieves a total profit of at least $\frac{1}{3}w_1(M^*) + \frac{2}{3}w_2(M^*) + \frac{1}{3}(p_1 + p_I)$, where $p_1 = 0$ (respectively, $p_I = 0$) if M^* assigns no job in $\mathcal{J}_1$ to u_1 (respectively, u_I), while p_1 (respectively, p_I) equals the weight of the edge of M^* incident to u_1 (respectively, u_I) otherwise.

To see why we have this better bound, first note that there are exactly two indices $h \in \{0,1,2\}$ such that u_1 is the primary time unit of a basic interval in P_h. Similarly, there are exactly two indices $h \in \{0,1,2\}$ such that u_I is the primary time unit of a basic interval in P_h. By these two facts, the better bound follows.

As it should be expected, the constrained scheduling problems $\mathcal{Q}_h$ may often lead to better experimental results than the more-constrained scheduling problems $\mathcal{Q}'_h$. However, as for general theoretical results, we don't know if there is a difference between the two types of problems. Moreover, $\mathcal{Q}'_h$ can be solved more efficiently than $\mathcal{Q}_h$. Hence, for simplicity, in the following exposition we will consider only the more-constrained scheduling problems $\mathcal{Q}'_h$.

It is not hard to see that each more-constrained scheduling problem $\mathcal{Q}'_h$ requires the computation of a maximum-weight (unconstrained) matching in a suitably constructed bipartite graph G_h, and hence can be solved in polynomial time. For clarity, we detail the construction of the graphs G_h below.

For each index $h \in \{0,1,2\}$, let G_h be the edge-weighted bipartite graph obtained from G as follows: (See Figure 3 for an example of G_2 constructed from graph G in Figure 1.)

1. For every $v_j \in \mathcal{J}_2$, merge the two vertices v_j and v_{j+1} into a single super-vertex $s_{j,j+1}$ (with all resulting multiple edges deleted).
2. For all i such that $h+1 \le i \le I-2$ and $i-1 \equiv h \pmod 3$, perform the following three sub-steps:
   (a) Merge u_i, u_{i+1}, and u_{i+2} into a single super-vertex $t_{i,i+1,i+2}$ (with all resulting multiple edges deleted).
   (b) For every $v_j \in \mathcal{J}_1$ that is a neighbor of $t_{i,i+1,i+2}$, if edge (u_{i+1}, v_j) is not in the original input graph, then delete the edge between $t_{i,i+1,i+2}$ and v_j; otherwise, assign a weight of $w(u_{i+1}, v_j)$ to the edge between $t_{i,i+1,i+2}$ and v_j.
   (c) For every $v_j \in \mathcal{J}_2$ such that $s_{j,j+1}$ is a neighbor of $t_{i,i+1,i+2}$, if neither $\{(u_i, v_j), (u_{i+1}, v_{j+1})\}$ nor $\{(u_{i+1}, v_j), (u_{i+2}, v_{j+1})\}$ is a matching in the original input graph, then delete the edge between $t_{i,i+1,i+2}$ and $s_{j,j+1}$; otherwise, assign a weight of $\max\{w(u_i, v_j) + w(u_{i+1}, v_{j+1}), w(u_{i+1}, v_j) + w(u_{i+2}, v_{j+1})\}$ to the edge between $t_{i,i+1,i+2}$ and $s_{j,j+1}$.
3. If neither u_1 nor u_2 was merged in Step 2a, then perform the following three sub-steps:
   (a) Merge u_1 and u_2 into a single super-vertex $t_{1,2}$ (with all resulting multiple edges deleted).
   (b) For every $v_j \in \mathcal{J}_1$ that is a neighbor of $t_{1,2}$, if edge (u_1, v_j) is not in the original input graph, then delete the edge between $t_{1,2}$ and v_j; otherwise, assign a weight of $w(u_1, v_j)$ to the edge between $t_{1,2}$ and v_j.
   (c) For every $v_j \in \mathcal{J}_2$ such that $s_{j,j+1}$ is a neighbor of $t_{1,2}$, if $\{(u_1, v_j), (u_2, v_{j+1})\}$ is not a matching in the original input graph, then delete the edge between $t_{1,2}$ and $s_{j,j+1}$; otherwise, assign a weight of $w(u_1, v_j) + w(u_2, v_{j+1})$ to the edge between $t_{1,2}$ and $s_{j,j+1}$.
4. If neither u_{I-1} nor u_I was merged in Step 2a, then perform the following three sub-steps:
   (a) Merge u_{I-1} and u_I into a single super-vertex $t_{I-1,I}$ (with all resulting multiple edges deleted).

(b) For every $v_j \in \mathcal{J}_1$ that is a neighbor of $t_{I-1,I}$, if edge (u_I, v_j) is not in the original input graph, then delete the edge between $t_{I-1,I}$ and v_j; otherwise, assign a weight of $w(u_I, v_j)$ to the edge between $t_{I-1,I}$ and v_j.

(c) For every $v_j \in \mathcal{J}_2$ such that $s_{j,j+1}$ is a neighbor of $t_{I-1,I}$, if $\{(u_{I-1}, v_j), (u_I, v_{j+1})\}$ is not a matching in the original input graph, then delete the edge between $t_{I-1,I}$ and $s_{j,j+1}$; otherwise, assign a weight of $w(u_{I-1}, v_j) + w(u_I, v_{j+1})$ to the edge between $t_{I-1,I}$ and $s_{j,j+1}$.

5. If u_1 was merged in neither Step 2a nor Step 3a, then for every $v_j \in \mathcal{J}_2$ such that $s_{j,j+1}$ is a neighbor of u_1, delete the edge between u_1 and $s_{j,j+1}$.
6. If u_I was merged in neither Step 2a nor Step 4a, then for every $v_j \in \mathcal{J}_2$ such that $s_{j,j+1}$ is a neighbor of u_I, delete the edge between u_I and $s_{j,j+1}$.

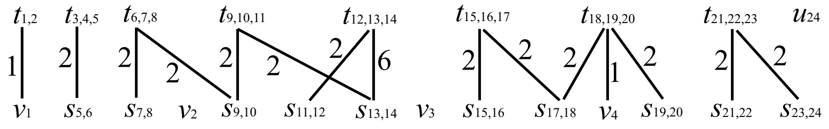


Figure 3: Graph G_2 constructed from graph G in Figure 1. The number beside each edge is the weight of that edge.

For each $h \in \{0, 1, 2\}$, let M_h be a maximum-weight matching in G_h. In our example (cf. Figure 3), M_2 may consist of the following edges: $(t_{1,2}, v_1)$, $(t_{3,4,5}, s_{5,6})$, $(t_{6,7,8}, s_{7,8})$, $(t_{9,10,11}, s_{9,10})$, $(t_{12,13,14}, s_{13,14})$, $(t_{15,16,17}, s_{15,16})$, $(t_{18,19,20}, s_{19,20})$, and $(t_{21,22,23}, s_{21,22})$. From each M_h, we can obtain a constrained matching $\bar{M}_h$ in the original input graph by performing the following steps in turn:

- Initialize $\bar{M}_h = \emptyset$.
- For each edge $(u_i, v_j) \in M_h$, add (u_i, v_j) to $\bar{M}_h$.
- For each edge $(t_{i,i+1,i+2}, v_j) \in M_h$, add (u_{i+1}, v_j) to $\bar{M}_h$.
- For each edge $(t_{1,2}, v_j) \in M_h$, add (u_1, v_j) to $\bar{M}_h$.
- For each edge $(t_{I-1,I}, v_j) \in M_h$, add (u_I, v_j) to $\bar{M}_h$.
- For each edge $(t_{i,i+1,i+2}, s_{j,j+1}) \in M_h$, if $w(u_i, v_j) + w(u_{i+1}, v_{j+1}) \geq w(u_{i+1}, v_j) + w(u_{i+2}, v_{j+1})$, then add edges (u_i, v_j) and (u_{i+1}, v_{j+1}) to $\bar{M}_h$; otherwise, add edges (u_{i+1}, v_j) and (u_{i+2}, v_{j+1}) to $\bar{M}_h$.
- For each edge $(t_{1,2}, s_{j,j+1}) \in M_h$, add edges (u_1, v_j) and (u_2, v_{j+1}) to $\bar{M}_h$.
- For each edge $(t_{I-1,I}, s_{j,j+1}) \in M_h$, add (u_{I-1}, v_j) and (u_I, v_{j+1}) to $\bar{M}_h$.

Note that $w(\bar{M}_h) = w(M_h)$. In our example (cf. Figures 1 and 3), if M_2 is as mentioned above, then $\bar{M}_2$ consists of the following edges: (u_1, v_1), (u_4, v_5), (u_5, v_6), (u_7, v_7), (u_8, v_8), (u_9, v_9), (u_{10}, v_{10}), (u_{13}, v_{13}), (u_{14}, v_{14}), (u_{15}, v_{16}), (u_{16}, v_{17}), (u_{19}, v_{19}), (u_{20}, v_{20}), (u_{21}, v_{21}), and (u_{22}, v_{22}).

In summary, we have established the following lemma:

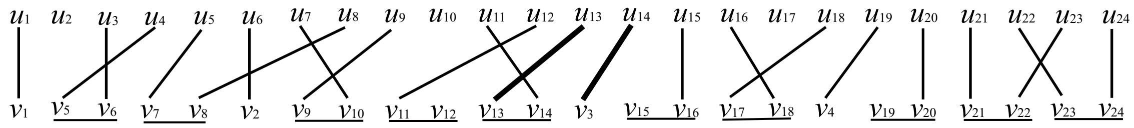
Lemma 2.1 *A constrained matching Z_1 in G can be found in $O(I(n_1 + n_2)\sqrt{I + n_1 + n_2})$ time, whose weight is at least $\frac{1}{3}w_1(M^*) + \frac{2}{3}w_2(M^*) + \frac{1}{3}(p_1 + p_I)$, where $p_1 = 0$ (respectively, $p_I = 0$) if u_1 (respectively, u_I) is not matched to a vertex of $\mathcal{J}_1$ by M^*, while p_1 (respectively, p_I) equals the weight of the edge of M^* incident to u_1 (respectively, u_I) otherwise.*

Corollary 2.2 *If $w_1(M^*) \leq (\frac{1}{2} - 3\epsilon)w(M^*)$, then $w(Z_1) \geq (\frac{1}{2} + \epsilon)w(M^*)$.*

Proof. Assume $w_1(M^*) \leq (\frac{1}{2} - 3\epsilon)w(M^*)$. Then, $w_2(M^*) = 1 - w_1(M^*) \geq (\frac{1}{2} + 3\epsilon)w(M^*)$. Moreover, by Lemma 2.1, $w(Z_1) \geq \frac{1}{3}w_1(M^*) + \frac{2}{3}w_2(M^*) = \frac{1}{3}w(M^*) + \frac{1}{3}w_2(M^*)$. Thus, $w(Z_1) \geq (\frac{1}{3} + \frac{1}{3}(\frac{1}{2} + 3\epsilon))w(M^*) \geq (\frac{1}{2} + \epsilon)w(M^*)$. □

2.2 Preparing for the other three algorithms

Before running the other three algorithms, we need to compute a maximum-weight unconstrained matching M^*_{un} of G. The unconstrained matching M^*_{un} will be an additional input to the other three algorithms. Therefore, before proceeding to the details of the algorithms, fix a maximum-weight unconstrained matching M^*_{un} of G. See Figure 4 for an example. The algorithms in Sec-



u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17} u_{18} u_{19} u_{20} u_{21} u_{22} u_{23} u_{24}
v_1 v_5 v_6 v_7 v_8 v_2 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_3 v_{15} v_{16} v_{17} v_{18} v_4 v_{19} v_{20} v_{21} v_{22} v_{23} v_{24}

Figure 4: A maximum-weight unconstrained matching M^*_{un} of the graph G in Figure 1.

tions 2.3 through 2.5 will use M^*_{un} in a sophisticated way. But first, we use M^*_{un} to define several subsets of U as follows.

- $U_0 = \{u_i \in U \mid u_i \text{ is not matched by } M^*_{\text{un}}\}$.
- $U_1 = \{u_i \in U \mid u_i \text{ is matched to a } v_j \in \mathcal{J}_1 \text{ by } M^*_{\text{un}}\}$.
- $U_{2,1} = \{u_i \in U \mid u_i \text{ is matched to a } v_j \in \mathcal{J}_2 \text{ by } M^*_{\text{un}}\}$.
- $U_{2,2} = \{u_i \in U \mid u_i \text{ is matched to a } v_j \in \mathcal{J}'_2 \text{ by } M^*_{\text{un}}\}$.
- $W = \{u_i \in U_1 \mid u_{i-1} \in U_{2,1} \text{ and } u_{i+1} \in U_{2,2}\}$.
- $W_L = \{u_i \in U \mid u_{i+1} \in W\}$ and $W_R = \{u_i \in U \mid u_{i-1} \in W\}$.

In our example (cf. Figures 1 and 4), $U_0 = \{u_2, u_{10}, u_{17}\}$, $U_1 = \{u_1, u_6, u_{14}, u_{19}\}$, $U_{2,1} = \{u_4, u_5, u_9, u_{12}, u_{13}, u_{18}, u_{21}, u_{23}\}$, $U_{2,2} = \{u_3, u_7, u_8, u_{11}, u_{15}, u_{16}, u_{20}, u_{23}, u_{24}\}$, $W = \{u_6, u_{14}, u_{19}\}$, $W_L = \{u_5, u_{13}, u_{18}\}$, and $W_R = \{u_7, u_{15}, u_{20}\}$. In general, whenever $u_i \in W$, we have $u_{i-1} \in W_L$ and $u_{i+1} \in W_R$. Moreover, since $W \subseteq U_1$, no two sets among W, W_L and W_R can intersect.

A common idea behind the forthcoming algorithms is to divide the weights $w_1(M^*)$ and $w_2(M^*)$ into smaller parts, based on the aforementioned subsets of U. The smaller parts are defined as follows.

- β_L is the total weight of all edges $(u_i, v_j) \in M^*$ such that $u_i \in W_L$ and $v_j \in \mathcal{J}_1$.
- β is the total weight of all edges $(u_i, v_j) \in M^*$ such that $u_i \in W$ and $v_j \in \mathcal{J}_1$.
- β_R is the total weight of all edges $(u_i, v_j) \in M^*$ such that $u_i \in W_R$ and $v_j \in \mathcal{J}_1$.
- $\bar{\beta} = w_1(M^*) - \beta_L - \beta - \beta_R$.
- α_0 is the total weight of all edges $(u_i, v_j) \in M^*$ such that either $v_j \in \mathcal{J}_2$ and $\{u_i, u_{i+1}\} \cap W = \emptyset$, or $v_j \in \mathcal{J}'_2$ and $\{u_{i-1}, u_i\} \cap W = \emptyset$.
- α_1 is the total weight of all edges $(u_i, v_j) \in M^*$ such that either $v_j \in \mathcal{J}_2$ and $\{u_i, u_{i+1}\} \subseteq W_L \cup W \cup W_R$, or $v_j \in \mathcal{J}'_2$ and $\{u_{i-1}, u_i\} \subseteq W_L \cup W \cup W_R$.

Lemma 2.3 *$\alpha_0 + \alpha_1 = w_2(M^*)$ and $\beta_L + \beta + \beta_R + \bar{\beta} = w_1(M^*)$.*

PROOF. Note that when $\{u_i, u_{i+1}\} \cap W \neq \emptyset$, then $\{u_i, u_{i+1}\} \subseteq W_L \cup W \cup W_R$. The same holds for $\{u_{i-1}, u_i\}$. $\square$

Now, we are ready to explain how the four algorithms are related. The algorithm in Section 2.3, called Algorithm 2, will output a constrained matching of weight at least $\frac{1}{3}\bar{\beta} + \frac{2}{3}\alpha_0 + \beta + \frac{2}{3}(\beta_L + \beta_R)$. The algorithm in Section 2.4, called Algorithm 3, will output a constrained matching of weight at least $\beta + \bar{\beta} + \alpha_1$. Thus, if $\beta \geq (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$, then Algorithm 2 or 3 will output a constrained matching of weight at least $(\frac{1}{2} + \epsilon)w(M^*)$ (see Corollary 2.6 below). On the other hand, if $\beta < (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$, then Algorithm 1 or 4 will output a constrained matching of weight at least $(\frac{1}{2} + \epsilon)w(M^*)$ (see Section 2.6).

2.3 Algorithm 2

We first explain the idea behind the algorithm. The removal of the vertices in W leaves $|W| + 1$ blocks of U each of which consists of consecutive vertices of U. For each block b, we use the idea of Algorithm 1 to construct three graphs $G_{b,0}, G_{b,1}, G_{b,2}$. For each $h \in \{0, 1, 2\}$, we consider the graph $\cup_b G_{b,h}$ where b ranges over all blocks, and obtain a new graph G'_h from $\cup_b G_{b,h}$ by adding the vertices of W and the edges $\{u_i, v_j\}$ of G such that $u_i \in W$ and $v_j \in \mathcal{J}_1$. We then compute a maximum-weight (unconstrained) matching in each G'_h, and further convert it to a constrained matching $\bar{M}'_h$ of G as in Algorithm 1. The output of Algorithm 2 is the heaviest matching among $\bar{M}'_0, \bar{M}'_1, \bar{M}'_2$. In our example (cf. Figures 1 and 4), G'_2 is as shown in Figure 5, and $\bar{M}'_2$ may consist of the following edges: (u_1, v_1), (u_4, v_5), (u_5, v_6), (u_6, v_2), (u_7, v_7), (u_8, v_8), (u_9, v_9), (u_{10}, v_{10}), (u_{12}, v_{11}), (u_{13}, v_{12}), (u_{14}, v_3), (u_{15}, v_{17}), (u_{16}, v_{18}), (u_{19}, v_4), (u_{22}, v_{21}), and (u_{23}, v_{22}).

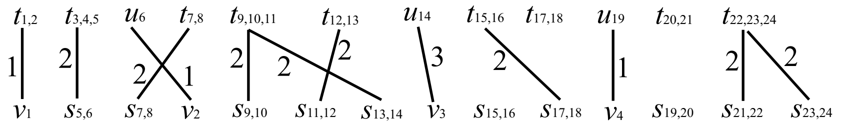


Figure 5: Graph G'_2 constructed from graph G in Figure 1 and matching M^*_{un} in Figure 4. The number beside each edge is the weight of that edge.

We next proceed to the details of Algorithm 2. Recall that the removal of the vertices in W leaves $|W| + 1$ blocks of U each of which consists of consecutive vertices of U. For each block b, let G_b be the subgraph of G induced by $V \cup \{u_i \in U \mid u_i \text{ is a vertex in block } b\}$.

1. For each block b, perform the following steps.
   (a) Delete all edges $\{u_i, v_j\}$ from G_b such that u_i is the first vertex in block b and $v_j \in \mathcal{J}'_2$; further delete all edges $\{u_i, v_j\}$ from G_b such that u_i is the last vertex in block b and $v_j \in \mathcal{J}_2$.
   (b) Construct three edge-weighted bipartite graphs $G_{b,0}, G_{b,1}, G_{b,2}$ from G_b in the same way as Algorithm 1 constructs the graphs G_0, G_1, G_2 from G.
2. For each $h \in \{0, 1, 2\}$, construct a new edge-weighted bipartite graph G'_h as follows. The vertex set of G'_h is the union of W and the vertex sets of the graphs $G_{b,h}$ where b ranges over all blocks. Note that even if a vertex appears in two or more of the graphs $G_{b,h}$, it appears in G'_h only once. The edges of the graphs $G_{b,h}$ where b ranges over all blocks are also edges in G'_h and inherit their weights to G'_h. Moreover, each edge (u_i, v_j) in G such that $u_i \in W$ and $v_j \in \mathcal{J}_1$ is also an edge in G'_h and inherits its weight from G to G'_h. G'_h has no other edges.
3. For each $h \in \{0, 1, 2\}$, compute a maximum-weight matching M'_h in G'_h, and then compute a constrained matching $\bar{M}'_h$ in G from M'_h in the same way as Algorithm 1 computes the constrained matching $\bar{M}_h$ in G from M_h.

4. Let Z_2 be the maximum-weight matching among the matchings M'_0, M'_1, M'_2. Output Z_2.

Lemma 2.4 $w(Z_2) \geq \frac{1}{3}\bar{\beta} + \frac{2}{3}\alpha_0 + \beta + \frac{2}{3}(\beta_L + \beta_R)$.

Proof. Immediate from Lemma 2.1 and Algorithm 2. □

2.4 Algorithm 3

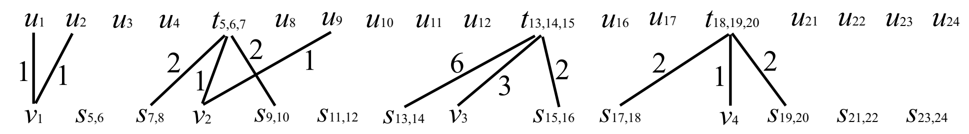
We first explain the idea behind Algorithm 3. Suppose that we partition the time interval $\mathcal{I}$ into shorter intervals in such a way that each shorter interval consists of either one time unit or three time units $u_{i-1}u_iu_{i+1}$ where $u_i \in W$. There is only one such partition of $\mathcal{I}$. Further suppose that we want to execute at most one job in each of the shorter intervals, while maximizing the total profit of the executed jobs. This problem can be solved in polynomial time by computing a maximum-weight (unconstrained) matching in a suitably constructed bipartite graph. We can prove that this matching results in a scheduling by which the executed jobs achieve at least a total profit of $\beta + \bar{\beta} + \alpha_1$.

We next proceed to the details of Algorithm 3. Algorithm 3 computes a constrained matching of G as follows. (See Figure 6 for an example.)

1. Construct a new edge-weighted bipartite graph G'' from G as follows:
   (a) For each $u_i \in W$, merge u_{i-1}, u_i and u_{i+1} into a super-vertex $t_{i-1,i,i+1}$ (with all resulting multiple edges deleted).
   (b) For each $v_j \in \mathcal{J}_2$, merge the two vertices v_j and v_{j+1} into a super-vertex $s_{j,j+1}$ (with all resulting multiple edges deleted).
   (c) For each edge $(t_{i-1,i,i+1}, v_j)$ such that $v_j \in \mathcal{J}_1$, if (u_i, v_j) is not an edge in the original input graph, then delete the edge $(t_{i-1,i,i+1}, v_j)$; otherwise, assign a weight of $w(u_i, v_j)$ to the edge $(t_{i-1,i,i+1}, v_j)$.
   (d) For each edge $(t_{i-1,i,i+1}, s_{j,j+1})$, if neither $\{(u_{i-1}, v_j), (u_i, v_{j+1})\}$ nor $\{(u_i, v_j), (u_{i+1}, v_{j+1})\}$ is a matching in the original input graph, then delete the edge $(t_{i-1,i,i+1}, s_{j,j+1})$; otherwise, assign a weight of $\max\{w(u_{i-1}, v_j) + w(u_i, v_{j+1}), w(u_i, v_j) + w(u_{i+1}, v_{j+1})\}$ to the edge $(t_{i-1,i,i+1}, s_{j,j+1})$.
   (e) Delete all edges $(u_i, s_{j,j+1})$. (Note that $u_i \notin W_L \cup W \cup W_R$.)
2. Compute a maximum-weight unconstrained matching M'' in G''.
3. Construct a constrained matching Z_3 in G from M'' as follows.
   (a) Initialize $Z_3 = \emptyset$.
   (b) For each edge $(u_i, v_j) \in M''$, add (u_i, v_j) to Z_3.
   (c) For each edge $(t_{i-1,i,i+1}, v_j) \in M''$, add (u_i, v_j) to Z_3.
   (d) For each edge $(t_{i-1,i,i+1}, s_{j,j+1}) \in M''$, if $w(u_{i-1}, v_j)+w(u_i, v_{j+1}) \geq w(u_i, v_j)+w(u_{i+1}, v_{j+1})$, then add edges (u_{i-1}, v_j) and (u_i, v_{j+1}) to Z_3; otherwise, add edges (u_i, v_j) and (u_{i+1}, v_{j+1}) to Z_3.
4. Output Z_3.

In our example (cf. Figures 1 and 6), M'' may consist of the following edges: (u_1, v_1), $(t_{5,6,7}, s_{7,8})$, (u_9, v_2), $(t_{13,14,15}, s_{13,14})$, and $(t_{18,19,20}, s_{17,18})$, and in turn Z_3 consists of the following edges: (u_1, v_1), (u_5, v_7), (u_6, v_8), (u_9, v_2), (u_{13}, v_{13}), (u_{14}, v_{14}), (u_{18}, v_{17}), and (u_{19}, v_{18}).

Lemma 2.5 $w(Z_3) \geq \beta + \bar{\beta} + \alpha_1$.



u_1 u_2 u_3 u_4 $t_{5,6,7}$ u_8 u_9 u_{10} u_{11} u_{12} $t_{13,14,15}$ u_{16} u_{17} $t_{18,19,20}$ u_{21} u_{22} u_{23} u_{24}
1 1 2 1 2 1 6 3 2 2 1 2
v_1 $s_{5,6}$ $s_{7,8}$ v_2 $s_{9,10}$ $s_{11,12}$ $s_{13,14}$ v_3 $s_{15,16}$ $s_{17,18}$ v_4 $s_{19,20}$ $s_{21,22}$ $s_{23,24}$

Figure 6: Graph G'' constructed from graph G in Figure 1 and matching M^*_{un} in Figure 4. The number beside each edge is the weight of that edge.

Proof. Similar to the proof of Lemma 2.1. $\square$

Corollary 2.6 *If* $\beta \geq (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$, *then* $\max\{w(Z_2), w(Z_3)\} \geq (\frac{1}{2} + \epsilon)w(M^*)$.

Proof. It suffices to show that if $\max\{w(Z_2), w(Z_3)\} < (\frac{1}{2} + \epsilon)w(M^*)$, then $\beta < (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$. So, assume that $\max\{w(Z_2), w(Z_3)\} < (\frac{1}{2} + \epsilon)w(M^*)$. Then, we have the following two inequalities:

$$w(Z_2) < (\frac{1}{2} + \epsilon)w(M^*). \tag{2.1}$$

$$w(Z_3) < (\frac{1}{2} + \epsilon)w(M^*). \tag{2.2}$$

Combining Inequality 2.1 and the inequality in Lemma 2.4, we obtain a new inequality, and further multiply it by a factor $\frac{3}{2}$ to obtain:

$$\alpha_0 + \frac{3}{2}\beta + \frac{1}{2}\bar{\beta} + \beta_L + \beta_R < \left(\frac{3}{4} + \frac{3}{2}\epsilon\right) w(M^*) \tag{2.3}$$

Moreover, combining Inequality 2.2 and the inequality in Lemma 2.5, we obtain:

$$\alpha_1 + \beta + \bar{\beta} < \left(\frac{1}{2} + \epsilon\right) w(M^*). \tag{2.4}$$

Now, adding Inequalities 2.3 and 2.4, we obtain

$$\alpha_0 + \alpha_1 + \frac{5}{2}\beta + \frac{3}{2}\bar{\beta} + \beta_L + \beta_R < \left(\frac{5}{4} + \frac{5}{2}\epsilon\right) w(M^*).$$

In turn, by Lemma 2.3, we have

$$w_1(M^*) + w_2(M^*) + \frac{3}{2}\beta + \frac{1}{2}\bar{\beta} < \left(\frac{5}{4} + \frac{5}{2}\epsilon\right) w(M^*).$$

Using the fact that $w_1(M^*) + w_2(M^*) = w(M^*)$ and $\bar{\beta} \geq 0$, we finally obtain

$$\frac{3}{2}\beta < \left(\frac{1}{4} + \frac{5}{2}\epsilon\right) w(M^*), \text{ or equivalently, } \beta < \left(\frac{1}{6} + \frac{5}{3}\epsilon\right) w(M^*),$$

which completes the proof. $\square$

2.5 Algorithm 4

The idea behind Algorithm 4 is to convert $M^*_{\rm un}$ to a constrained matching of G. To convert $M^*_{\rm un}$, we partition $U_1 \cup U_{2,1}$ (respectively, $U_1 \cup U_{2,2}$) into two subsets none of which contains two vertices u_i and u_{i+1} such that $u_i \in U_{2,1}$ (respectively, $u_{i+1} \in U_{2,2}$). The set of edges of $M^*_{\rm un}$ incident to the vertices of each such subset can be extended to a constrained matching of G. In this way, we obtain four constrained matchings of G. Algorithm 4 outputs the heaviest one among the four matchings. We can prove that the weight of the output matching is at least $w(M^*_{\rm un})/2$.

We next proceed to the details of Algorithm 4. Algorithm 4 computes a constrained matching in G as follows. (See Figure 7 for an example.)

1. Starting at u_1, divide U into segments each of which is in the following form:

$$u_{i-\ell}u_{i-\ell+1}\cdots u_{i-1}u_iu_{i+1}\cdots u_{i+r-1}u_{i+r},$$

where $u_j \in U_{2,1}$ for all $i-\ell \le j \le i-1$, $u_j \in U_{2,2}$ for all $i+1 \le j \le i+r$, $u_{i-\ell-1} \notin U_{2,1}$, $u_{i+r+1} \notin U_{2,2}$, and u_i has no restriction. Note that ℓ and/or r may be equal to zero. We call u_i the *center* of the segment. For each segment s, let $c(s)$ denote the integer i such that u_i is the center of s; let $\ell(s)$ denote the number of vertices in s that precede $u_{c(s)}$; let $r(s)$ denote the number of vertices in s that succeed $u_{c(s)}$.

[*Comment:* In our example (cf. Figure 7), U is divided into 8 segments. We name them from left to right as $s_1,\ldots,s_8$. For example, s_1 consists of only u_1 while s_3 consists of $u_4,\ldots,u_8$. Moreover, $c(s_1)=1$, $\ell(s_1)=r(s_1)=0$, $c(s_2)=2$, $\ell(s_2)=0$, $r(s_2)=1$, $c(s_3)=6$, $\ell(s_3)=r(s_3)=2$, $c(s_4)=10$, $\ell(s_4)=r(s_4)=1$, $c(s_5)=14$, $\ell(s_5)=r(s_5)=2$, $c(s_6)=17$, $\ell(s_6)=r(s_6)=0$, $c(s_7)=19$, $\ell(s_7)=r(s_7)=1$, $c(s_8)=22$, $\ell(s_8)=1$, and $r(s_8)=2$. Alternatively, it is also valid that $c(s_8)=23$, $\ell(s_8)=2$, and $r(s_8)=1$.

Given $M^*_{\rm un}$ (and hence the partition of U as $U_0\cup U_1\cup U_{2,1}\cup U_{2,2}$), the division of U into segments is unique. To see this, consider two relations $\mathcal{R}_{\rm left}$ and $\mathcal{R}_{\rm right}$ defined on U as follows: For every pair (u_i,u_j), $u_i\mathcal{R}_{\rm left}u_j$ if and only if $j=i+1$ and $u_i\in U_{2,1}$; $u_i\mathcal{R}_{\rm right}u_j$ if and only if $j=i-1$ and $u_i\in U_{2,2}$. Then, the segments one-to-one correspond to the equivalence classes of the symmetric and transitive closure of the relation $\mathcal{R}_{\rm left}\cup\mathcal{R}_{\rm right}$.]

2. For each segment s, compute two integers x_s and y_s as follows:

   - If $u_{c(s)} \in U_0$, then $x_s = c(s)-1$ and $y_s = c(s)+1$.
   - If $u_{c(s)} \in U_1$, then $x_s = y_s = c(s)$.
   - If $u_{c(s)} \in U_{2,1}$, then $x_s = c(s)$ and $y_s = c(s)+1$.
   - If $u_{c(s)} \in U_{2,2}$, then $x_s = c(s)-1$ and $y_s = c(s)$.

[*Comment:* In our example (cf. Figure 7), $x_{s_1}=y_{s_1}=1$, $x_{s_2}=1$, $y_{s_2}=3$, $x_{s_3}=y_{s_3}=6$, $x_{s_4}=9$, $y_{s_4}=11$, $x_{s_5}=y_{s_5}=14$, $x_{s_6}=16$, $y_{s_6}=18$, $x_{s_7}=y_{s_7}=19$, $x_{s_8}=22$, and $y_{s_8}=23$.

In other words, for each segment s, u_{x_s} is the rightmost vertex in s with $u_{x_s}\in U_{2,1}\cup U_1$, while u_{y_s} is the leftmost vertex in s with $u_{y_s}\in U_{2,2}\cup U_1$.]

3. Let

$$U^e_{2,1} = \bigcup_s \{u_i \mid (x_s - i) \bmod 2 = 0, c(s)-\ell(s) \le i \le x_s\},$$

$$U^o_{2,1} = \bigcup_s \{u_i \mid (x_s - i) \bmod 2 = 1, c(s)-\ell(s) \le i \le x_s\},$$

$$U^e_{2,2} = \bigcup_s \{u_i \mid (i - y_s) \bmod 2 = 0, y_s \le i \le c(s)+r(s)\},$$

$$U^o_{2,2} = \bigcup_s \{u_i \mid (i - y_s) \bmod 2 = 1, y_s \le i \le c(s) + r(s)\},$$

where s runs over all segments.

[*Comment:* In our example (cf. Figure 7), $U^e_{2,1} = \{u_1, u_4, u_6, u_9, u_{12}, u_{14}, u_{19}, u_{22}\}$, $U^o_{2,1} = \{u_5, u_{13}, u_{18}, u_{21}\}$, $U^e_{2,2} = \{u_1, u_3, u_6, u_8, u_{11}, u_{14}, u_{16}, u_{19}, u_{23}\}$, and $U^o_{2,2} = \{u_7, u_{15}, u_{20}, u_{24}\}$. Note that if a vertex $u_i \in U$ belongs to more than one of the four sets $U^e_{2,1}$, $U^o_{2,1}$, $U^e_{2,2}$, $U^o_{2,2}$, then $u_i \in U_1$, u_i is the center of the segment containing u_i, and u_i belongs to only $U^e_{2,1}$ and $U^e_{2,2}$.]

4. Let

$$M^e_{2,1} = \{(u_i, v_j) \in M^*_{\text{un}} \mid u_i \in U^e_{2,1}\} \cup \{(u_{i+1}, v_{j+1}) \mid u_i \in U^e_{2,1} \cap U_{2,1} \text{ and } \{u_i, v_j\} \in M^*_{\text{un}}\},$$

$$M^o_{2,1} = \{(u_i, v_j) \in M^*_{\text{un}} \mid u_i \in U^o_{2,1}\} \cup \{(u_{i+1}, v_{j+1}) \mid u_i \in U^o_{2,1} \cap U_{2,1} \text{ and } \{u_i, v_j\} \in M^*_{\text{un}}\},$$

$$M^e_{2,2} = \{(u_i, v_j) \in M^*_{\text{un}} \mid u_i \in U^e_{2,2}\} \cup \{(u_{i-1}, v_{j-1}) \mid u_i \in U^e_{2,2} \cap U_{2,2} \text{ and } \{u_i, v_j\} \in M^*_{\text{un}}\},$$

$$M^o_{2,2} = \{(u_i, v_j) \in M^*_{\text{un}} \mid u_i \in U^o_{2,2}\} \cup \{(u_{i-1}, v_{j-1}) \mid u_i \in U^o_{2,2} \cap U_{2,2} \text{ and } \{u_i, v_j\} \in M^*_{\text{un}}\}.$$

[*Comment:* $M^e_{2,1}, M^o_{2,1}, M^e_{2,2}, M^o_{2,2}$ are constrained matchings in G (cf. Lemma 2.7). In our example (cf. Figure 7), the edges in $M^e_{2,1}$ are (u_1, v_1), (u_4, v_5), (u_5, v_6), (u_6, v_2), (u_9, v_9), (u_{10}, v_{10}), (u_{12}, v_{11}), (u_{13}, v_{12}), (u_{14}, v_3), (u_{19}, v_4), (u_{22}, v_{23}), and (u_{23}, v_{24}); the edges in $M^o_{2,1}$ are (u_5, v_7), (u_6, v_8), (u_{13}, v_{13}), (u_{14}, v_{14}), (u_{18}, v_{17}), (u_{19}, v_{18}), (u_{21}, v_{21}), and (u_{22}, v_{22}); the edges in $M^e_{2,2}$ are (u_1, v_1), (u_2, v_5), (u_3, v_6), (u_6, v_2), (u_7, v_7), (u_8, v_8), (u_{10}, v_{13}), (u_{11}, v_{14}), (u_{14}, v_3), (u_{15}, v_{17}), (u_{16}, v_{18}), (u_{19}, v_4), (u_{22}, v_{21}), and (u_{23}, v_{22}); the edges in $M^o_{2,2}$ are (u_6, v_9), (u_7, v_{10}), (u_{14}, v_{15}), (u_{15}, v_{16}), (u_{19}, v_{19}), (u_{20}, v_{20}), (u_{23}, v_{23}), and (u_{24}, v_{24}).

Note that for each edge $(u_i, v_j) \in M^o_{2,1} \cup M^o_{2,2}$, we have $v_j \notin \mathcal{J}_1$. Indeed, $U^o_{2,1} \subseteq U_{2,1}$ and $U^o_{2,2} \subseteq U_{2,2}$.]

5. For the set $\bar{U}^o_{2,1}$ of vertices of U that are not matched by $M^o_{2,1}$, compute a maximum-weight matching $N^o_{2,1}$ between the vertices in $\bar{U}^o_{2,1}$ and the vertices in $\mathcal{J}_1$.

[*Comment:* $M^o_{2,1} \cup N^o_{2,1}$ is a constrained matching in G (cf. Lemma 2.7). In our example (cf. Figures 1 and 7), the vertices in $\bar{U}^o_{2,1}$ are $u_1, \ldots, u_4$, $u_7, \ldots, u_{12}$, $u_{15}, \ldots, u_{17}$, u_{20}, u_{23}, and u_{24}; $N^o_{2,1} = \{(u_1, v_1), (u_9, v_2)\}$.]

6. For the set $\bar{U}^o_{2,2}$ of vertices of U that are not matched by $M^o_{2,2}$, compute a maximum-weight matching $N^o_{2,2}$ between the vertices in $\bar{U}^o_{2,2}$ and the vertices in $\mathcal{J}_1$.

[*Comment:* $M^o_{2,2} \cup N^o_{2,2}$ is a constrained matching in G (cf. Lemma 2.7). In our example (cf. Figures 1 and 7), the vertices in $\bar{U}^o_{2,2}$ are $u_1, \ldots, u_5$, $u_8, \ldots, u_{13}$, $u_{16}, \ldots, u_{18}$, u_{21}, and u_{22}; $N^o_{2,2} = \{(u_2, v_1), (u_9, v_2)\}$.]

7. Let Z_4 be the maximum-weight matching among $M^e_{2,1}$, $M^o_{2,1} \cup N^o_{2,1}$, $M^e_{2,2}$, $M^o_{2,2} \cup N^o_{2,2}$. Output Z_4.

Lemma 2.7 *$M^e_{2,1}$, $M^o_{2,1} \cup N^o_{2,1}$, $M^e_{2,2}$ and $M^o_{2,2} \cup N^o_{2,2}$ are constrained matchings in G.*

Proof. Note that $U^e_{2,1} \subseteq U - U_{2,2}$. Thus, to prove that $M^e_{2,1}$ is a constrained matching in G, it suffices to prove that for every $u_i \in U^e_{2,1} \cap U_{2,1}$, $u_{i+1} \notin U^e_{2,1}$. Consider an arbitrary $u_i \in U^e_{2,1} \cap U_{2,1}$. By the definition of a segment, u_i and u_{i+1} belong to the same segment. So, by the definition of $U^e_{2,1}$, $u_{i+1} \notin U^e_{2,1}$. This completes the proof that $M^e_{2,1}$ is a constrained matching of G. Similarly, we can prove that $M^o_{2,1}$, $M^e_{2,2}$ and $M^o_{2,2}$ are constrained matchings in G.

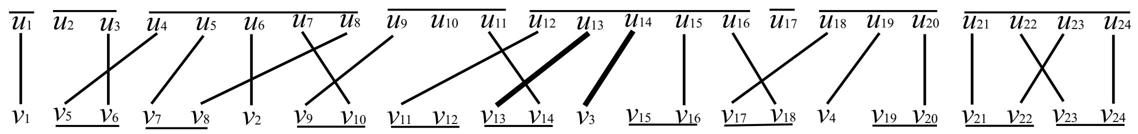

u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17} u_{18} u_{19} u_{20} u_{21} u_{22} u_{23} u_{24}
v_1 v_5 v_6 v_7 v_8 v_2 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_3 v_{15} v_{16} v_{17} v_{18} v_4 v_{19} v_{20} v_{21} v_{22} v_{23} v_{24}

Figure 7: The segments of U obtained from $M^*_{\rm un}$ in Figure 4. Each segment is shown by drawing a common line above the vertices of that segment.

To see that $M^o_{2,1} \cup N^o_{2,1}$ and $M^o_{2,2} \cup N^o_{2,2}$ are constrained matchings in G, first note that $U^o_{2,1} \subseteq U_{2,1}$ and $U^o_{2,2} \subseteq U_{2,2}$. Thus, for every edge $(u_i, v_j) \in M^o_{2,1} \cup M^o_{2,2}$, we have $v_j \notin \mathcal{J}_1$. In turn, $M^o_{2,1} \cup N^o_{2,1}$ is a constrained matching in G. Similarly, $M^o_{2,2} \cup N^o_{2,2}$ is a constrained matching in G. □

Lemma 2.8 $w(M^e_{2,1}) + w(M^o_{2,1}) + w(M^e_{2,2}) + w(M^o_{2,2}) \geq 2w(M^*_{\rm un})$.

PROOF. Consider an arbitrary edge $(u_i, v_j) \in M^*_{\rm un}$. We distinguish three cases as follows.

Case 1: $u_i \in U_1$ (i.e., $v_j \in \mathcal{J}_1$). Then, u_i must be the center of a segment and hence u_i is contained in both $U^e_{2,1}$ and $U^e_{2,2}$ by Step 3; consequently, edge (u_i, v_j) is contained in both $M^e_{2,1}$ and $M^e_{2,2}$ by Step 4.

Case 2: $u_i \in U_{2,1}$ (hence, $v_j \in \mathcal{J}_2$). Then, u_i and u_{i+1} must belong to the same segment, say s. If (u_{i+1}, v_{j+1}) is also in $M^*_{\rm un}$, then either u_i or u_{i+1} is the center of s. In either case, $x_s = i$ and $y_s = i + 1$, and hence (u_i, v_j) belongs to both $M^e_{2,1}$ and $M^e_{2,2}$ by Step 4 (and so does (u_{i+1}, v_{j+1})). On the other hand, if (u_{i+1}, v_{j+1}) is not in $M^*_{\rm un}$, then either $M^e_{2,1}$ or $M^o_{2,1}$ contains both (u_i, v_j) and (u_{i+1}, v_{j+1}). Since (u_i, v_j) and (u_{i+1}, v_{j+1}) have the same weight, we can think of (u_{i+1}, v_{j+1}) as a copy of (u_i, v_j).

Case 3: $u_i \in U_{2,2}$ (hence, $v_j \in \mathcal{J}'_2$). Similar to Case 2.

By the above case-analysis, we see that for each edge (u_i, v_j) of $M^*_{\rm un}$, either (u_i, v_j) belongs to two of $M^e_{2,1}, M^o_{2,1}, M^e_{2,2}, M^o_{2,2}$, or one of $M^e_{2,1}, M^o_{2,1}, M^e_{2,2}, M^o_{2,2}$ contains both (u_i, v_j) and its copy. This completes the proof of the lemma. □

Lemma 2.9 $(U - \bar{U}^o_{2,1}) \cap (U - \bar{U}^o_{2,2}) \subseteq W$.

PROOF. First note that $U - \bar{U}^o_{2,1}$ (respectively, $U - \bar{U}^o_{2,2}$) is the set of vertices in U that are matched by $M^o_{2,1}$ (respectively, $M^o_{2,2}$). Thus, $U - \bar{U}^o_{2,1} \subseteq \{u_{c(s)-\ell(s)}, \ldots, u_{x(s)-1}, u_{x(s)} \mid s \text{ is a segment}\}$ and $U - \bar{U}^o_{2,2} \subseteq \{u_{y(s)}, u_{y(s)+1}, \ldots, u_{c(s)+r(s)} \mid s \text{ is a segment}\}$. In turn, since segments are disjoint and $x(s) \leq y(s)$ for every segment s, it follows that for every $u_i \in (U - \bar{U}^o_{2,1}) \cap (U - \bar{U}^o_{2,2})$, we have $i = x(s) = y(s)$ for some segment s. Now, by the definitions of $x(s)$ and $y(s)$, the fact $i = x(s) = y(s)$ implies $u_i \in U_1$. Moreover, since $u_i \in (U - \bar{U}^o_{2,1}) \cap (U - \bar{U}^o_{2,2})$, u_i is matched by $M^o_{2,1}$ and so $u_{i-1} \in U_{2,1}$. For the same reason, u_i is matched by $M^o_{2,2}$ and so $u_{i+1} \in U_{2,2}$.

In summary, for every $u_i \in (U - \bar{U}^o_{2,1}) \cap (U - \bar{U}^o_{2,2})$, we have $u_i \in U_1$, $u_{i-1} \in U_{2,1}$, and $u_{i+1} \in U_{2,2}$; hence, $u_i \in W$. This completes the proof of the lemma. □

2.6 Performance of the algorithm when β is small

For a contradiction, assume the following:

Assumption 2.10 $\beta < (\frac{1}{6} + \frac{5}{3})w(M^*)$ *and* $\max\{w(Z_1), w(Z_4)\} < (\frac{1}{2} + \epsilon)w(M^*)$.

We want to derive a contradiction under this assumption. First, we derive three inequalities from this assumption and the lemmas in Section 2.5.

Lemma 2.11 *$w(M^o_{2,1}) + w(M^o_{2,2}) \geq (1-2\epsilon)w(M^*)$.*

PROOF. Assume, on the contrary, that $w(M^o_{2,1}) + w(M^o_{2,2}) < (1-2\epsilon)w(M^*)$. By Lemma 2.8 and the fact that $w(M^*_{\mathrm{un}}) \geq w(M^*)$, we have $w(M^e_{2,1}) + w(M^e_{2,2}) \geq (1+2\epsilon)w(M^*)$. But then $\max\{w(M^e_{2,1}), w(M^e_{2,2})\} \geq (\frac{1}{2}+\epsilon)w(M^*)$, contradicting Assumption 2.10. □

Lemma 2.12 *$w(N^o_{2,1}) + w(N^o_{2,2}) < 4\epsilon w(M^*)$.*

PROOF. Obviously, $w(M^o_{2,1} \cup N^o_{2,1}) + w(M^o_{2,2} \cup N^o_{2,2}) = w(M^o_{2,1}) + w(M^o_{2,2}) + w(N^o_{2,1}) + w(N^o_{2,2})$. By Assumption 2.10, $w(M^o_{2,1} \cup N^o_{2,1}) + w(M^o_{2,2} \cup N^o_{2,2}) < (1+2\epsilon)w(M^*)$. So, by Lemma 2.11, $w(N^o_{2,1}) + w(N^o_{2,2}) < 4\epsilon w(M^*)$. □

Lemma 2.13 *$\beta > w_1(M^*) - 4\epsilon w(M^*)$.*

PROOF. Let γ_1 be the total weight of all edges $(u_i, v_j) \in M^*$ such that $v_j \in \mathcal{J}_1$ and $u_i \in \bar{U}^o_{2,1}$. Let γ_2 be the total weight of all edges $(u_i, v_j) \in M^*$ such that $v_j \in \mathcal{J}_1$ and $u_i \in \bar{U}^o_{2,2}$. Let γ_3 be the total weight of all edges $(u_i, v_j) \in M^*$ such that $v_j \in \mathcal{J}_1$ and $u_i \in (U - \bar{U}^o_{2,1}) \cap (U - \bar{U}^o_{2,2})$. Clearly, $\gamma_1 + \gamma_2 + \gamma_3 \geq w_1(M^*)$. By Steps 5 and 6 in Algorithm 4, $\gamma_1 \leq w(N^o_{2,1})$ and $\gamma_2 \leq w(N^o_{2,2})$. So, by Lemma 2.12, $\gamma_1 + \gamma_2 < 4\epsilon w(M^*)$. Moreover, by Lemma 2.9, $\beta \geq \gamma_3$. Thus, $\beta \geq \gamma_3 \geq w_1(M^*) - \gamma_1 - \gamma_2 > w_1(M^*) - 4\epsilon w(M^*)$. □

Now, we are ready to get a contradiction. By Corollary 2.2 and Assumption 2.10, $w_1(M^*) > (\frac{1}{2} - 3\epsilon)w(M^*)$. Thus, by Lemma 2.13, $\beta > (\frac{1}{2} - 7\epsilon)w(M^*)$. On the other hand, by Assumption 2.10, $\beta < (\frac{1}{6} + \frac{5}{3})w(M^*)$. Hence,

$$\frac{1}{2} - 7\epsilon < \frac{1}{6} + \frac{5}{3}\epsilon,$$

contradicting our choice that $\epsilon = \frac{1}{26}$.

Therefore, we have

Theorem 2.14 *A constrained matching Z in G with $w(Z) \geq \frac{13}{7}w(M^*)$ can be found in $O(|E|\sqrt{I(n_1+n_2)})$ time.*

3 2-ISP with a special profit function

In this section, we consider *proportional 2-ISP*, where all the positive profits of executing a job are proportional to the lengths of the jobs. A $\frac{5}{3}$-approximation algorithm was recently presented in [4] for proportional 2-ISP. Here, we present a $(1.5+\epsilon)$-approximation algorithm for it for any $\epsilon > 0$.

Let U, $\mathcal{J}_1$, and $\mathcal{J}_2$ be as in Section 2. Let E be the set of those $(u_i, v_j) \in U \times \mathcal{J}_1$ such that the profit of executing job v_j in time unit u_i is positive. Let F be the set of those $(u_i, u_{i+1}, v_j) \in U \times U \times \mathcal{J}_2$ such that the profit of executing job v_j in time units u_i and u_{i+1} is positive.

Consider the hypergraph $H = (U \cup \mathcal{J}_1 \cup \mathcal{J}_2, E \cup F)$ on vertex set $U \cup \mathcal{J}_1 \cup \mathcal{J}_2$ and on edge set $E \cup F$. Obviously, proportional 2-ISP becomes the problem of finding a matching $E' \cup F'$ in H with $E' \subseteq E$ and $F' \subseteq F$ such that $|E'| + 2|F'|$ is maximized over all matchings in H. Our idea is to reduce this problem to the problem of finding a maximum cardinality matching in a 3-uniform hypergraph (*i.e.* each hyperedge consists of exactly three vertices). Since the latter problem admits a $(1.5+\epsilon)$-approximation algorithm [5] and our reduction is approximation preserving, it follows that proportional 2-ISP admits a $(1.5+\epsilon)$-approximation algorithm.

We now detail the approximation-preserving reduction. From H, we construct a 3-uniform hypergraph $\mathcal{H}$ as follows. Let

- $\hat{U} = \{\hat{u}_i \mid u_i \in U\}$ and $\overline{U} = \{\overline{u}_i \mid u_i \in U\}$;

- $\hat{\mathcal{J}}_2 = \{\hat{v}_j \mid v_j \in \mathcal{J}_2\}$ and $\overline{\mathcal{J}_2} = \{\overline{v}_j \mid v_j \in \mathcal{J}_2\}$;
- $\tilde{E} = \{\{\hat{u}_i, \overline{u}_i, v_j\} \mid (u_i, v_j) \in E\}$;
- $\hat{F} = \{\{\hat{u}_i, \hat{u}_{i+1}, \hat{v}_j\} \mid (u_i, u_{i+1}, v_j) \in F\}$;
- $\overline{F} = \{\{\overline{u}_i, \overline{u}_{i+1}, \overline{v}_j\} \mid (u_i, u_{i+1}, v_j) \in F\}$.

The vertex set of $\mathcal{H}$ is $\hat{U} \cup \overline{U} \cup \mathcal{J}_1 \cup \hat{\mathcal{J}}_2 \cup \overline{\mathcal{J}_2}$, and the edge set is $\tilde{E} \cup \hat{F} \cup \overline{F}$.

Lemma 3.1 *Let $E' \cup F'$ with $E' \subseteq E$ and $F' \subseteq F$ be a matching in H. Then, there is a matching in $\mathcal{H}$ of cardinality $|E'| + 2|F'|$.*

Proof. It suffices to check that $\tilde{E'} \cup \hat{F'} \cup \overline{F'}$ is a matching in $\mathcal{H}$, where $\tilde{E'} = \{\{\hat{u}_i, \overline{u}_i, v_j\} \mid (u_i, v_j) \in E'\}$, $\hat{F'} = \{\{\hat{u}_i, \hat{u}_{i+1}, \hat{v}_j\} \mid (u_i, u_{i+1}, v_j) \in F'\}$, and $\overline{F'} = \{\{\overline{u}_i, \overline{u}_{i+1}, \overline{v}_j\} \mid (u_i, u_{i+1}, v_j) \in F'\}$. □

Lemma 3.2 *Let $\tilde{E}' \cup \hat{F}' \cup \overline{F}'$ be a matching in $\mathcal{H}$ with $\tilde{E}' \subseteq \tilde{E}$, $\hat{F}' \subseteq \hat{F}$ and $\overline{F}' \subseteq \overline{F}$. Then, we can compute a matching $E' \cup F'$ in H with $E' \subseteq E$, $F' \subseteq F$, and $|E'| + 2|F'| \geq |\tilde{E}'| + |\hat{F}'| + |\overline{F}'|$.*

Proof. Let $\tilde{E}' \cup \hat{F}' \cup \overline{F}'$ be a matching in $\mathcal{H}$ as in the lemma. Consider $E' = \{(u_i, v_j) \mid \{\hat{u}_i, \overline{u}_i, v_j\} \in \tilde{E}'\}$. Clearly, E' is a matching in H, that is, no vertex in $U \cup \mathcal{J}_1$ belongs to more than one pair in E'. Moreover, if $u_i \in U$ is a vertex belonging to some pair in E', then neither $\hat{u}_i$ belongs to some triple in $\hat{F}'$ nor $\overline{u}_i$ belongs to some triple in $\overline{F}'$.

Now, either $|\hat{F}'| \geq |\overline{F}'|$ or $|\hat{F}'| \leq |\overline{F}'|$. We assume $|\hat{F}'| \geq |\overline{F}'|$; the other case is similar. Consider $F' = \{(u_i, u_{i+1}, v_j) \mid \{\hat{u}_i, \hat{u}_{i+1}, \hat{v}_j\} \in \hat{F}'\}$. Clearly, F' is a matching in H, that is, no vertex in $U \cup \mathcal{J}_2$ belongs to more than one triple in F'. Note that $E' \cup F'$ is also a matching in H and

$$|E'| + 2|F'| = |\tilde{E}'| + 2\max\{|\hat{F}'|, |\overline{F}'|\} \geq |\tilde{E}'| + |\hat{F}'| + |\overline{F}'|.$$

□

By the above two lemmas, we have

Theorem 3.3 *For every $\epsilon > 0$, there is a polynomial-time $(1.5 + \epsilon)$-approximation algorithm for proportional 2-ISP.*

4 A new heuristic for protein NMR peak assignment

As mentioned in Section 1, the $\frac{13}{7}$-approximation algorithm for 2-ISP can be easily incorporated into a heuristic framework for protein NMR peak assignment introduced in [6]. The heuristic first tries to assign "long" segments of three or more spin systems that are under the consecutivity constraint to segments of the host protein sequence, using a simple greedy strategy, and then solves an instance of 2-ISP formed by the remaining unassigned spin systems and amino acids. The first step of the framework is also called *greedy filtering* and may potentially help improve the accuracy of the heuristic significantly in practice because we are often able to assign long segments of spin systems with high confidence. We have tested the new heuristic based on the $\frac{13}{7}$-approximation algorithm for 2-ISP and compared the results with two of the best approximation and heuristic algorithms in [3, 4, 6], namely the 2-approximation algorithm for the interval scheduling problem [3, 4] and the branch-and-bound algorithm (augmented with greedy filtering) [6]. The test data consists of 70 (pseudo) real instances of NMR peak assignment derived from 14 proteins, each with 5 (density) levels of consecutivity constraints, as shown in Table 1. Each protein is represented as an entry in the BioMagResBank database [9], *e.g.* bmr4027, and the consecutivity level is represented by the underscore symbol following the BioMagResBank entry. For example, _5 means that the number of pairs of consecutive spin systems in the input is 50% of the total number of spin systems. Hence, the higher the consecutivity level index, the more the constraint.

Note that, both the new heuristic algorithm and the 2-approximation algorithm are very fast in general while the branch-and-bound algorithm can be much slower because it may have to explore much of the entire search space. On a standard Linux workstation, it took seconds to hours for each assignment by the branch-and-bound algorithm in the above experiment, while it took a few seconds consistently using either the new heuristic algorithm or the 2-approximation algorithm. Table 1 shows the comparison of the performance of the three algorithms in terms of (i) the weight of the assignment and (ii) the number of correctly assigned spin systems. Although measure (i) is the objective in the interval scheduling problem, measure (ii) is what it counts in NMR peak assignment. Clearly, the new heuristic outperformed the 2-approximation algorithm in both measures by large margins. Furthermore, the new heuristic outperformed the branch-and-bound algorithm in measure (ii), although the branch-and-bound algorithm did slightly better in measure (i). More precisely, the new heuristic was able to assign the same number of or more spin systems correctly than the branch-and-bound algorithm on 53 out of the 70 instances, among which the new heuristic algorithm improved over the branch-and-bound algorithm on 39 instances. [2] Previously, the branch-and-bound algorithm was known to have the best assignment accuracy (among all heuristics proposed for the interval scheduling problem) [6]. The result demonstrates that this new heuristic based on the $\frac{13}{7}$-approximation algorithm for 2-ISP will be very useful in the automation of NMR peak assignment. In particular, the good assignment accuracy and fast speed allow us to tackle some large-scale problems in experimental NMR peak assignment within realistic computation resources. As an example of application, the consecutivity information derived from experiments may sometimes be ambiguous. The new heuristic algorithm makes it possible for the user to experiment with different interpretations of consecutivity and compare the resulting assignments.

5 Discussion

The computational method, presented in this paper, provides a more accurate and more efficient technique for NMR peak assignment, compared to our previous algorithms [3, 4, 6, 8]. While this algorithm will prove to be useful to protein NMR experimentalists for their data assignment procedures, it could be used to help protein structure study directly. We are in the process of incorporating this algorithm into a computational pipeline for fast protein fold recognition and structure determination, using an iterative procedure of NMR peak assignments and protein structure prediction. The basic idea of this pipeline can be briefly outlined as follows.

Recent developments in applications of *residual dipolar coupling* (RDC) data to protein structure determination have indicated that RDC data alone may be adequate for accurate resolution of protein structures [11], bypassing the expensive and time-consuming step of NOE (nuclear Overhauser effect) data collection and assignments. We have recently demonstrated (unpublished results) that if the RDC data/peaks are accurrately assigned, we can accurately identify the correct fold of a target protein in the PDB database [10] even when the target protein has lower than 25% of sequence identity with the corresponding PDB protein of the same structural fold. In addition, we have found that RDC data can be used to accurately rank sequence-fold alignments (alignment accuracy), suggesting the possibility of protein backbone structure prediction by combining RDC data and fold-recognition techniques like protein threading [13].

By including RDC data in our peak assignment algorithm (like [12]), we expect to achieve two things: (a) an improved accuracy of peak assignments with the added information, and (b) an assignment (possibly partial) of the RDC peaks. Using assigned RDC peaks and the aforementioned strategy, we can identify the correct structural folds of a target protein in the PDB database.

[2] It is not completely clear to us why the new heuristic did better on these 39 instances, although the weighting function between spin systems and amino acids could be a factor.

	W_1	R_1	W_2	R_2	W_3	R_3		W_1	R_1	W_2	R_2	W_3	R_3
bmr4027_5	1873820	40	1827498	3	1934329	33	bmr4144_5	919419	11	921816	17	997603	16
bmr4027_6	1854762	64	1818131	8	1921093	37	bmr4144_6	923546	21	897500	11	993361	11
bmr4027_7	1845477	89	1784027	44	1910897	74	bmr4144_7	954141	68	842073	2	954633	64
bmr4027_8	1900416	151	1671475	19	1894532	128	bmr4144_8	953741	69	804531	5	954585	67
bmr4027_9	1896606	156	1652859	60	1896606	156	bmr4144_9	952241	75	837519	35	952241	75
bmr4288_5	1243144	36	1169907	6	1255475	12	bmr4302_5	1275787	31	1219920	11	1331391	16
bmr4288_6	1197106	49	1179110	15	1261696	26	bmr4302_6	1282789	51	1174564	0	1324395	43
bmr4288_7	1232771	65	1112288	22	1251020	57	bmr4302_7	1310324	78	1181267	8	1323495	62
bmr4288_8	1201192	68	1133554	35	1238344	66	bmr4302_8	1308217	112	1152323	27	1308217	103
bmr4288_9	1249465	105	1051817	48	1249465	105	bmr4302_9	1250300	111	1293954	107	1298321	110
bmr4309_5	1974762	35	1954955	13	2117910	25	bmr4316_5	999920	43	890944	2	1009329	30
bmr4309_6	1960424	48	1924727	12	2110992	57	bmr4316_6	967526	59	863207	13	1022505	35
bmr4309_7	2046029	119	1885986	24	2093595	77	bmr4316_7	925817	75	882818	9	1029287	79
bmr4309_8	1962114	121	1868338	55	2067295	101	bmr4316_8	1005898	75	957378	62	1029287	89
bmr4309_9	2048987	178	1796864	95	2048987	178	bmr4316_9	1029827	89	984774	85	1029287	89
bmr4318_5	2338383	19	2355926	2	2497294	20	bmr4353_5	1468772	20	1417351	8	1532518	17
bmr4318_6	2265090	34	2312260	13	2481789	35	bmr4353_6	1428944	23	1421633	18	1524784	24
bmr4318_7	2268700	73	2259377	52	2444439	52	bmr4353_7	1461648	56	1370235	14	1516244	44
bmr4318_8	2217936	92	2214174	63	2420829	62	bmr4353_8	1443261	78	1337329	9	1472871	80
bmr4318_9	2339582	201	2158223	122	2383453	201	bmr4353_9	1474022	124	1273988	15	1483781	126
bmr4391_5	691804	10	688400	5	753046	18	bmr4393_5	1816837	49	1742954	3	1874095	41
bmr4391_6	680959	7	699066	8	745501	10	bmr4393_6	1843685	71	1772955	42	1871616	59
bmr4391_7	699199	17	684953	37	735683	26	bmr4393_7	1847874	102	1722026	22	1862221	76
bmr4391_8	688368	38	663147	30	723111	42	bmr4393_8	1832576	129	1709538	65	1853749	130
bmr4391_9	710914	66	687290	45	710914	66	bmr4393_9	1837340	142	1527885	3	1851298	152
bmr4579_5	913713	18	894084	2	967647	15	bmr4670_5	1365873	32	1309727	11	1435721	22
bmr4579_6	889118	35	911564	8	976720	32	bmr4670_6	1326082	35	1290812	13	1429449	30
bmr4579_7	903586	48	873884	17	958335	44	bmr4670_7	1353618	78	1239001	6	1402335	38
bmr4579_8	933371	72	877556	26	956115	63	bmr4670_8	1391055	116	1236726	19	1391055	116
bmr4579_9	950173	86	760356	0	950173	86	bmr4670_9	1391055	120	1237614	60	1391055	116
bmr4752_5	881020	21	796019	8	884307	21	bmr4929_5	1410017	17	1408112	4	1496460	23
bmr4752_6	877313	32	824289	6	892520	32	bmr4929_6	1391418	36	1385673	12	1496954	32
bmr4752_7	866896	43	752633	3	887292	41	bmr4929_7	1427122	69	1378166	30	1490155	56
bmr4752_8	882755	68	730276	17	882755	68	bmr4929_8	1459368	82	1281548	18	1481593	88
bmr4752_9	882755	68	812950	44	882755	68	bmr4929_9	1477704	114	1178499	20	1477704	114

Table 1: The performance of the new heuristic comprising greedy filtering and the $\frac{13}{7}$-approximation algorithm for 2-ISP in comparison with two of the best approximation and heuristic algorithms in [3, 4, 6] on 70 instances of NMR peak assignment. The number after the underscore symbol in the name of each instance indicates the density level of consecutivity constraints in the instance. W_1 and R_1 represent the total assignment weight and number of spin systems correctly assigned by the new heuristic, respectively. W_2 and R_2 (W_3 and R_3) are corresponding values for the 2-approximation algorithm for the interval scheduling problem (the branch-and-bound algorithm augmented with greedy filtering, respectively).

Then based on the identified structural fold and a computed sequence-fold alignment, we can back-calculate the theoretical RDC peaks of the predicted backbone structure (possibly partial). Through matching the theoretical and experimental RDC peaks, we can establish an iterative procedure for NMR data assignment and structure prediction. Such a process will iterate until most of the RDC peaks are assigned and a structure is predicted. We expect that such a procedure will prove to be highly effective for fast and accurate protein fold and backbone structure predictions, using NMR data from only a small number of NMR experiments.

References

[1] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48:1069–1090, 2001.

[2] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *Proceedings of the 31th Annual ACM Symposium on Theory of Computing (STOC'99)*, pp. 622–631, ACM, 1999.

[3] Z.-Z. Chen, T. Jiang, G. Lin, J. Wen, D. Xu, J. Xu, and Y. Xu. Approximation algorithms for NMR spectral peak assignment. To appear in *Theoretical Computer Science*, 2002.

[4] Z.-Z. Chen, T. Jiang, G. Lin, J. Wen, D. Xu, and Y. Xu. Improved approximation algorithms for NMR spectral peak assignment. *Proceedings of the 2nd Workshop on Algorithms in Bioinformatics (WABI'2002)*, Lecture Notes in Computer Science, Vol. 2452, pp. 82–96, 2002.

[5] C.A.J. Hurkens and A. Schrijver. On the size of systems of sets of every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68-72, 1989.

[6] G. Lin, D. Xu, Z.-Z. Chen, T. Jiang, J. Wen, and Y. Xu. An efficient branch-and-bound algorithm for the assignment of protein backbone NMR peaks. *Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB'2002)*, Palo Alto, CA. Pages 165-174. Aug. 2002.

[7] National Institute of General Medical Sciences. Pilot projects for the protein structure initiative (structural genomics). *http://www.nih.gov/grants/guide/rfa-files/RFA-GM-99-009.html*, June:RFA GM–99–009, 1999.

[8] 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.

[9] University of Wisconsin. *BioMagResBank. http://www.bmrb.wisc.edu.* University of Wisconsin, Madison, Wisconsin, 2001.

[10] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, The Protein Data Bank: A Computer Based Archival File for Macromolecular Structures, *J. Mol. Biol.*, 112:535 - 542, 1977.

[11] J.C. Hus, D. Marion and M. Blackledge, Determination of protein backbone structure using only residual dipolar couplings, *J. Am. Chem. Soc*, 123:1541-1542, 2001.

[12] J.C. Hus, J.J. Prompers, and R. Bruschweiler, Assignment strategy for proteins with known structure, *Journal of Magnetic Resonnance*, 157:119-123, 2002.

[13] Y. Xu and D. Xu, Protein Threading using PROSPECT: design and evaluation, *Protein: Structure, Function, Genetics*, 40:343 - 354, 2000.