Computing H/D-Exchange Speeds of Single Residues from Data of Peptic Fragments

Ernst Althaus Stefan Canzar Mark R. Emmett
Huimin Zhang Andreas Karrenbauer Alan G. Marshall
Anke Meyer-Baese

September 13, 2007

Abstract

Determining the hydrogen-deuterium exchange speeds of single residues from data for peptic fragments obtained by FT-ICS MS is currently mainly done by manual interpretation. We provide automated method based on combinatorial optimization. More precisely, we present an algorithm that enumerates all possible exchange speeds for single residues that explain the observed data of the peptic fragments.

1 Introduction

Hydrogen-deuterium exchange (also called H-D or H/D exchange) is a chemical reaction in which a covalently bonded hydrogen atom is replaced by a deuterium atom, or vice versa. Usually the examined protons are the amides in the backbone of a protein. The method gives information about the solvent accessibility of various parts of the molecule, and thus the tertiary structure of the protein.

In modern times H/D exchange has primarily been monitored by the methods: NMR spectroscopy and mass spectrometry. Each of these methods have their advantages and drawbacks. The main disadvantage of mass spectrometry is that one obtains exchange data for peptic fragments and assigning exchange speeds to single residues has to be done by manual interpretation.

We provide an automated method to resolve this problem. More precisely, we present an algorithm that enumerates all possible exchange speeds for single residues that explain the observed data of the peptic fragments.

As the number of possibilities is often very large, we combine sets of assignments to equivalence classes which are easily interpreted such that the number of equivalence classes is typically very small.

The assignment of exchange speeds to single residues from the data of the peptic fragments is a combinatorial problem. Hence, we applied methods from combinatorial optimization to it, i.e. we show how to formalize the problem into an integer linear program and propose a method to solve the problem.

The paper is organized as follows. In Section 2, we review the biochemical background of the underlying problem, motivating our research on this problem. Before giving an integer linear program for the combinatorial problem in Section 4, we make a formal description of the problem in Section 3. After that, we explain our solution method and the combination of possible solutions into equivalence classes in Section 5. In Section 6, we discuss the computational complexity of the problem and in Section 7, we show some experimental results of our algorithm. Finally, we give a short conclusion.

2 Biochemical Background

Determination of protein of protein-protein interaction is best accomplished by X-ray crystal diffraction and NMR [Zui02] because both methods provide the highest resolution of the sites of interaction. On the downside, both methods require large (milligram) quantities of protein. Other techniques rely on chemical or photo-induced reactions with MS analysis [LC02, KHJ$^+$06] to reveal functional groups that are exposed to the solvent. These methods also suffer from physical limitations.

Another method utilizes hydroxyl radical reactions with alkyl C-H bonds. The OH tends to react mainly with surface-exposed residues providing a good footprint of the solvent exposed surface of the protein(s) [GCA00, SBH04]. The modification is covalent and thus irreversible, but each modification can potentially change the conformation of the protein, thus skewing results.

Exchange of labile hydrogens for deuteriums (HDX) as a probe of protein surface accessibility does not change the conformation of the protein. Advantages over NMR and X-ray crystallography structural determination are the ability to work at low concentration and high molecular weight (removed from downside).

The experiment is initiated by dilution of the protein solution into a biological buffer made with D_2O. Solvent accessible hydrogens are exchanged

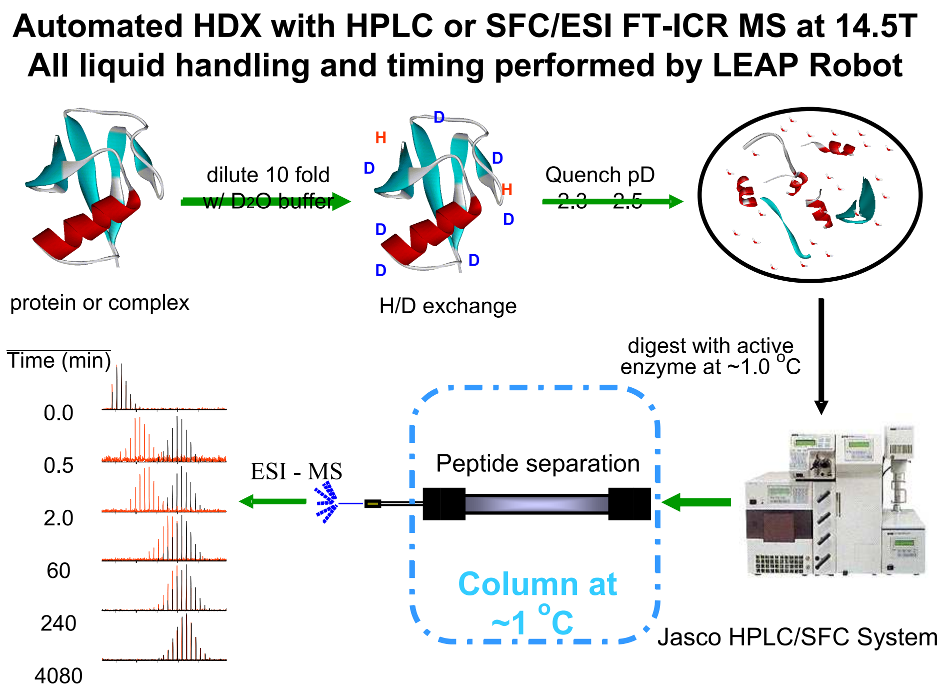
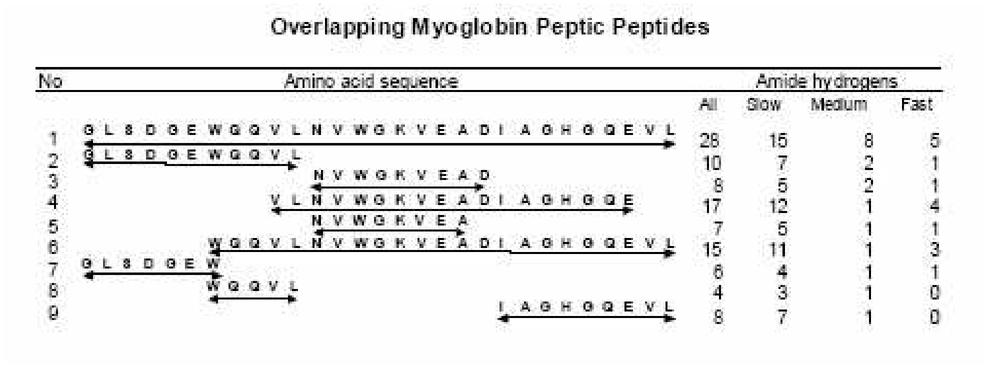

Automated HDX with HPLC or SFC/ESI FT-ICR MS at 14.5T
All liquid handling and timing performed by LEAP Robot
dilute 10 fold w/ D2O buffer
protein or complex
H/D exchange
Quench pD 2.3 – 2.5
digest with active enzyme at ~1.0 °C
Time (min)
0.0
0.5
2.0
60
240
4080
ESI - MS
Peptide separation
Column at ~1 °C
Jasco HPLC/SFC System

Figure 1: Schematic diagram of procedures for the current HDX experiment. The experiment starts by diluting the protein solution 10 fold into deuterated buffer, after which exchange occurs for predetermined time points, and is then quenched by lowering the temperature and dropping the pH. The protein is then digested and injected onto the HPLC for separation and mass analysis.

with deuterium. The exchange is quenched (greatly slowed) by dropping the pH to pH 2.3 and lowering the temp to 0-4° C. The protein complex is digested with pepsin or protease type XIII (Cravello et.al 2003 and Zhang H. et.al 2007 submitted) and on-line liquid chromatography is performed directly to the FT-ICR MS (Figure 1). Deuterium incorporation is monitored by the increase in mass of each peptic fragment as the deuteron is added.

The data sets produced are large and each spectrum has hundreds of overlapping peptic fragments. From this data, the exchange rate is easily determined for the same peptic fragments from protein and protein complex. [LLE$^+$02]. When peptic fragments are not directly comparable, but are overlapping (Figure 2) manual interpretation must be performed to assign



Overlapping Myoglobin Peptic Peptides

No	Amino acid sequence	Amide hydrogens			
		All	Slow	Medium	Fast
1	G L S D G E W Q Q V L N V W G K V E A D I A G H G Q E V L	28	15	8	5
2	G L S D G E W Q Q V L	10	7	2	1
3	N V W G K V E A D	8	5	2	1
4	V L N V W G K V E A D I A G H G Q E	17	12	1	4
5	N V W G K V E A	7	5	1	1
6	W Q Q V L N V W G K V E A D I A G H G Q E V L	15	11	1	3
7	G L S D G E W	6	4	1	1
8	W Q Q V L	4	3	1	0
9	I A G H G Q E V L	8	7	1	0

Figure 2: Overlap of peptic fragments as seen in an HDX of the model protein myoglobin. The table on the right shows the total number of amide hydrogens exchanged in each peptide and the number of amide hydrogens exchanged in each peptide and the number of amide hydrogens predicted to be either slow, medium or fast as predicted by maximum entropy method (MEM) evaluation of the H/D exchange rate distribution (Zhang 1997). Since the peptides are not directly comparable, we plan to apply novel data analysis techniques to localize where the fast, medium or slow amide hydrogens are located based on the overlap of peptic fragments.

exchange speed to single residues. HDX data analysis is the greatest bottleneck in these experiments, thus automated data analysis is necessary.

3 Mathematical Abstraction

In an idealized setting, we are considering the following problem. We are given a sequence $(1, \ldots, n)$ of residues, a set $\mathcal{F} \subseteq \{(i, \ldots, j) \mid 1 \leq i \leq j \leq n\}$ of peptic fragments, and a set of possible exchange speeds $\mathcal{S} = \{1, \ldots K\}$. We denote a fragment $(i, \ldots, j)$ by (i, j) and refer to an exchange speed as a color. For each fragment $(i, j) \in \mathcal{F}$ and each color $k \in \mathcal{S}$, we are given the number $b^k_{(i,j)}$ of residues with color k within the fragment (i, j). The corresponding vector b^k is referred to as the right hand side for color k. In our experimental data, we consider exactly three different colors (slow, medium, and fast), i.e. $K = 3$.

We have to compute an assignment $\pi : \{1, \ldots, n\} \mapsto \mathcal{S}$ which assigns a color to each residue. This assignment has to respect our knowledge on the peptic fragments, i.e. $\pi : \{1, \ldots, n\} \mapsto \mathcal{S}$ such that $|\{i \leq l \leq j \mid \pi(l) =$

$k\}| = b^k_{(i,j)}$ for all given fragments $(i,j) \in \mathcal{F}$ and all possible colors $k \in \mathcal{S}$.

In real setting we will have errors in the data from our experiments. Hence we want to compute all assignments that minimize the total sum of errors, i.e. the assignments minimizing

$$\sum_{k \in \mathcal{S}} \sum_{(i,j) \in \mathcal{F}} e^k_{(i,j)},$$

where $e^k_{(i,j)} = |(|\{i \leq l \leq j \mid \pi(l) = k\}| - b^k_{(i,j)})|$.

Furthermore, we are interested in all such assignments, as we want to determine protein conformation study, protein/protein interaction, and protein/ligand interactions. This data will be useful in the design and synthesis of small molecules to be used as therapeutic agents.

4 Mathematical Model

First we formulate the idealized problem above as an integer linear program. More precisely, we show an integer linear program whose feasible solutions correspond to the feasible assignments of colors to residues.

Let $\pi : \{1, \ldots, n\} \mapsto \mathcal{S}$ be an assignment of colors to residues and for each $k \in \mathcal{S}$ let $x^k \in \{0,1\}^n$ be a set of binary variables and let $x = (x^k)_{k \in \mathcal{S}} \in \{0,1\}^{Kn}$. Let furthermore $x^k_i = 1$ if $\pi(i) = k$ and $x^k_i = 0$ otherwise. Notice that all assignments satisfy $\sum_{k \in \mathcal{S}} x^k_i = 1$ for all $i \in \{1, \ldots, n\}$ as every residue has exactly one color assigned. Furthermore every $\{0,1\}$-assignment to $x \in \{0,1\}^{Kn}$ satisfying $\sum_{k \in \mathcal{S}} x^k_i = 1$ for all $i \in \{1, \ldots, n\}$ corresponds to an assignment.

An $\{0,1\}$-assignment x corresponds to a feasible π, iff furthermore $\sum_{l=i}^{j} x^k_l = b^k_{(i,j)}$ for all $(i,j) \in \mathcal{F}$ and $k \in \mathcal{S}$.

Assume now, we want to compute an assignment with minimum number of errors. Notice that we want to minimize a sum of absolute values. We use a standard trick to formulate such a problem as an integer linear program. Assume we have a variable $e^k_{i,j}$ for every color $k \in \mathcal{S}$ and every fragment $(i,j) \in \mathcal{F}$ whose value should be the error of the assignment x for the color k and the fragment (i,j). Hence we want to minimize $\sum_{k \in \mathcal{S}} \sum_{(i,j) \in \mathcal{F}} e^k_{(i,j)}$. It suffices to formulate conditions on $e^k_{(i,j)}$ that enforce it to be at least the error. As we minimize, the value will not be larger than the error in an optimal solution. To enforce $e^k_{(i,j)}$ to be at least the error, the two linear constraints $e^k_{(i,j)} \geq \sum_{l=i}^{j} x^k_l - b^k_{(i,j)}$ and $e^k_{(i,j)} \geq -\sum_{l=i}^{j} x^k_l + b^k_{(i,j)}$ are sufficient.

Hence the integer linear program, we are looking at is

$$\begin{array}{lll} \min & \sum_{k\in\mathcal{S}}\sum_{(i,j)\in\mathcal{F}} e^k_{(i,j)} & \\ \text{s.t.} & e^k_{(i,j)} \geq \sum_{l=i}^{j} x^k_l - b^k_{(i,j)} & \text{for all } k\in\mathcal{S}, (i,j)\in\mathcal{F} \\ & e^k_{(i,j)} \geq -\sum_{l=i}^{j} x^k_l + b^k_{(i,j)} & \text{for all } k\in\mathcal{S}, (i,j)\in\mathcal{F} \\ & \sum_{k\in\mathcal{S}} x^k_l = 1 & \text{for all } 1\leq l\leq n \\ & x\in\{0,1\}^{Kn} & \end{array}$$

We refer to this integer linear program as *basic-ILP*.

5 Solution of the mathematical model

We implemented our approach using the C++-Library SCIL [SCI04] to solve integer linear programs. SCIL uses the libraries LEDA [MN99] and SCIP [Ach04]. SCIP uses CPLEX [CPL07] or SoPlex [SoP04] as solver for linear programs. The underlying solution method is branch-&-bound, which works roughly as follows. We ignore the integrality-condition on the variables and solve the so called linear relaxation, i.e. we replace the condition $x^k_{(i,j)} \in \{0,1\}$ by $0 \leq x^k_{(i,j)} \leq 1$ for all $k \in \mathcal{S}, (i,j) \in \mathcal{F}$. Linear programs can be solved efficiently. If all variables have integral values in the solution of the linear program, we are done. If the objective function value of the linear relaxation is larger than the objective of the best solution found so far, we stop our search. Otherwise we choose a variable $x^k_{(i,j)}$ with a fractional value in the solution of the linear relaxation and solve the problems with the additional condition $x^k_{(i,j)} = 0$ and $x^k_{(i,j)} = 1$ recursively. For details, we refer to [Wol98].

In order to find all solutions within a given error bound e, we add the constraint $\sum_{k\in\mathcal{S}}\sum_{(i,j)\in\mathcal{F}} e^k_{(i,j)} \leq e$ to the integer linear program and hence are faced with the problem of computing all feasible solutions of an integer linear program. We do this with a branching-approach similar to the branch-&-bound approach described above. We solve the linear relaxation. If the linear relaxation if infeasible, we stop the search on this branch. If the solution is integral, we store it (if we haven't found this solution jet). If there is a binary variable which is not fixed so far (i.e. not set to 0 or 1), we pick one such variable x^k_l and solve the two subproblems where we fix the variable to 0 or 1 recursively. Notice that it is possible that we branch on a variable which already has an integral value. In this case, the solution of the linear relaxation of the subproblem will be the same as in problem itself. Nevertheless, we will terminate, as there are only a finite number of variables to branch on.

In our experiments, it turns out that finding a single solution is very fast, whereas finding all solutions takes quite some time (see Table 1 in Section 7). The reason is mainly that the number of solutions is quite large. This is as there are quite large intervals such that no fragment starts or ends within an interval. Let P be the partition of $\{1, \ldots, n\}$ into a minimal number of intervals, such that for each element of $p \in P$ and each fragment $f \in F$ either $p \subseteq f$ or $p \cap f = \emptyset$. Notice that for an assignment π, we can get further assignments with the same total error, if we permute the colors within these intervals, i.e. if $i, j \in p$ for $p \in P$ and π is a feasible assignment than π' with $\pi'(i) = \pi(j), \pi'(j) = \pi(i)$ and $\pi'(l) = \pi(l)$ for $l \neq i, j$ is a feasible assignment. We call two assignments equivalent, if one can be obtained by the other by iteratively applying this rule.

Hence we modify our integer linear program in order to enumerate equivalent solutions only once. For $k \in \mathcal{S}$ and $p \in P$, we replace the binary variables $(x_l^k)_{l \in p}$ by a single integer variable y_p^k with $y_p^k := \sum_{l \in p} x_l^k$. Moreover, let A be the $\mathcal{F} \times P$ matrix, i.e. for all $f \in \mathcal{F}$ and $p \in P$

$$a_{f,p} = \begin{cases} 1 & \text{if } p \subseteq f \\ 0 & \text{otherwise} \end{cases}$$

In matrix notation the corresponding constraints are then of the form

$$\begin{aligned} -Ay^k + e^k &\geq -b^k \\ Ay^k + e^k &\geq b^k \end{aligned}$$

for all $k \in \mathcal{S}$. Hence our integer linear program gets

$$\begin{aligned} \min \quad & \sum_{k \in \mathcal{S}} \sum_{f \in \mathcal{F}} e_f^k \\ \text{s.t.} \quad & -Ay^k + e^k \geq -b^k && \text{for all } k \in \mathcal{S} \\ & Ay^k + e^k \geq b^k && \text{for all } k \in \mathcal{S} \\ & \sum_{k \in \mathcal{S}} y^k = P \\ & y \geq 0, \text{ integer} \end{aligned} \tag{1}$$

where P is the vector that contains $|p|$ for each component $p \in \mathcal{P}$.

We refer to this integer linear program as *improved-ILP*. We compute all solutions within a certain error bound by following basically the same approach as described above. The number of solutions is just a fraction of the number of solutions of the original integer linear program (see Section 7).

6 Computational Complexity

We first consider the case with two colors. That is, we have the constraints $y_p^1 + y_p^2 = |p|$ for all $p \in P$. This allows us to simplify the linear program considerably. We replace $y_p^2 = |p| - y_p^1$ and omit the superscript of the y-variables in the following. This yields

$$\begin{aligned} -Ay + e^1 &\geq -b^1 & \qquad Ay + e^2 &\geq F - b^2 \\ Ay + e^1 &\geq b^1 & -Ay + e^2 &\geq -F + b^2 \end{aligned}$$

where F is the vector of fragment sizes. We may get rid of half of the constraints by the following observation. Let $b := \max\{b^1, F - b^2\}$ and $\bar{b} := \max\{-b^1, -F + b^2\}$ where the maximum is taken component-wise. Let y be an arbitrary feasible solution with minimum total error $\sum_{f \in \mathcal{F}} e_f^1 + e_f^2$. We may consider the contribution of each fragment independently for that particular y. We may rename the error variables e^1 and e^2 component-wise according to b and $\bar{b}$, i.e.

$$e_f := \begin{cases} e_f^1 & \text{if } b = b^1 \\ e_f^2 & \text{otherwise} \end{cases} \qquad \bar{e}_f := \begin{cases} e_f^1 & \text{if } \bar{b} = b^1 \\ e_f^2 & \text{otherwise} \end{cases} \tag{2}$$

For each $f \in \mathcal{F}$ with $\bar{b}_f \leq a_f^T y \leq b_f$, we have $e_f^1 + e_f^2 = b - \bar{b}$. If $a_f^T y > b_f$, we get $e_f^1 + e_f^2 = 2e_f + b_f - \bar{b}_f$. Analogously, we get $e_f^1 + e_f^2 = 2\bar{e}_f + b_f - \bar{b}_f$ if $a_f^T y < \bar{b}_f$. Hence, it is sufficient to optimize the following linear program

$$\begin{aligned} \min \quad & \sum_{f \in \mathcal{F}} e_f + \bar{e}_f \\ \text{s.t.} \quad & Ay + e \geq b \\ & -Ay + \bar{e} \geq \bar{b} \\ & -y \geq -P \\ & y, e, \bar{e} \geq 0 \end{aligned} \tag{3}$$

which is integral if b and $\bar{b}$ are integral since the constraint matrix is totally unimodular. The corresponding dual LP is given by

$$\begin{aligned} \max \quad & b^T f^1 + \bar{b}^T f^2 - P^T f^3 \\ \text{s.t.} \quad & A^T f^1 - A^T f^2 - f^3 \leq 0 \\ & 0 \leq f^{1,2} \leq 1 \\ & 0 \leq f^3 \end{aligned} \tag{4}$$

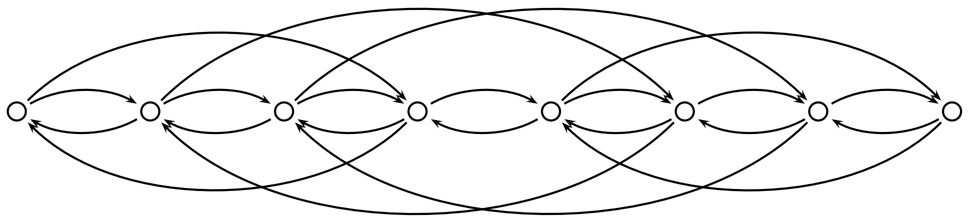


Figure 3: Example of a fragment graph with $|P| = 7$. The corresponding fragments are $(1, 3)$, $(2, 5)$, $(3, 6)$, and $(5, 7)$.

which is equivalent to (multiplying the objective function by -1 and introducing slack variables)

$$\begin{aligned} -\min \quad & -b^T f^1 - \bar{b}^T f^2 + P^T f^3 \\ \text{s.t.} \quad & A^T f^1 - A^T f^2 - f^3 + f^4 = 0 \\ & 0 \le f^{1,2} \le 1 \\ & 0 \le f^{3,4} \end{aligned} \tag{5}$$

We will show next that this LP is a Minimum Cost Circulation Problem. To this end, let M be the matrix of the equality constraints, i.e.

$$M := \begin{pmatrix} A^T & -A^T & -I & I \end{pmatrix}$$

Note that this matrix has the column-wise consecutive-ones property. By row operations like in Gaussian elimination, we can easily transform M such that each column contains exactly one 1 and one -1, as follows. We add the dummy constraint $0 = 0$ at the end and subtract from each row its predecessor. The resulting matrix, say $\bar{M}$, can be considered as the node-arc-incidence matrix of a directed graph. Since the right hand side remains unchanged, we get a Minimum Cost Circulation problem on a graph with $|\mathcal{P}| + 1$ nodes and $O(|\mathcal{P}| + |\mathcal{F}|)$ arcs [AMO93]. As a matter of fact, we have for each variable y_p two arcs corresponding to the constraint $0 \le y_p \le |p|$ and for each fragment (i, j) the arcs $(i, j + 1)$ and $(j + 1, i)$ as depicted in Figure 3.

We may use any algorithm that solves the Minimum Cost Circulation problem, e.g. Cycle Canceling or Successive Shortest Path (see [AMO93] for further reference). Both approaches have their advantages. The former always maintains a feasible circulation, i.e. we start with the zero flow and augment flow along negative cycles in the residual network until no negative cycle remains. Since the residual network with respect to an optimal circulation does not contain a directed negative circuit, we can find node potentials, i.e. a corresponding dual solution, using the Bellman-Ford algorithm in $O(|\mathcal{P}| \cdot |\mathcal{F}|)$ time. The difference between the potential of two neighboring nodes then yields the value of the corresponding y-variable. The errors are determined straight forward. If there is a solution without error this approach yields a solution within the running time of Bellman-Ford. On the other hand, the Successive Shortest Path algorithm maintains similar node potentials such that the arc-weights remain non-negative. Since the total excess is bounded by $|\mathcal{P}|$ in our case, the running time of that algorithm is $O(|\mathcal{P}| \cdot |\mathcal{F}| + |\mathcal{P}|^2 \log|\mathcal{P}|)$.

For three or more colors the complexity is open. The totally unimodularity of the constraint matrix is destroyed, i.e. there are instances with fractional vertices, e.g. the one from Figure 3 with the appropriate right hand sides. Moreover there is an instance which have an error, but the value of the LP is 0. Hence the integrality gap is infinite.

7 Experiments

We applied our solution method to several real instances and to randomly generated instances, as we only have a limited number of real instances at hand.

The real instances had between 28 and 57 residues and between 16 and 50 fragments. The solutions with a minimal number of errors could be computed in less that 0.1 seconds for all instances. All (non-equivalent) solutions with a minimal number of errors, which are between 6 and 62, could be computed in less than 5 seconds, where the number of solutions and the running time greatly depend on the number of solutions (see Table 1). Computing the all solutions using the basic-ILP takes much longer as with the improved-ILP.

The results for the real instances are very promising as the small number of easily interpretable classes of equivalent solutions can be used in protein structure prediction tools and for manual inspection.

To evaluate the running time of our approach more closely, we created the random instances as follows. We generated a sequence of a given number of residues. For each residue, we randomly chose a color with a biased coin. We have chosen a probability of 0.6 for slow, and 0.2 for medium and fast, which reflect approximately the numbers we observed in the real instances. Then we generated random fragments, i.e. we chose $i, j \in \{1, \dots, n\}$ with $i < j$ at random and repeated this $n/2$ times. In our experiments, we use $n = 50, 100, 150, 200, 250, 300, 500, 1000$. Note that the actual numbers of variables for the Improved-ILP is lower than n. However, it is only a slight difference due to our random choice of the fragments. We computed

Instance				Basic-ILP			Improved-ILP		
n	$\|\mathcal{F}\|$	$\|\mathcal{P}\|$	ϵ	T(One)	T(All)	#-Sol	T(One)	T(All)	#-Sol
28	16	12	19	0.02	418.75	102600	0.02	0.13	11
57	34	28	10	0.05	>3600	> 300000	0.02	0.32	6
57	50	37	35	0.06	215.13	8568	0.04	4.32	62
37	17	16	9	0.03	2084.12	4529151	0.01	0.22	18

Table 1: We give the characteristics of the instance, i.e. the number of residues (n), the number of fragments ($\mathcal{F}$), number of intervals ($\mathcal{P}$), and the minimal error of an solution (ϵ). Furthermore, we give for the basic and the improved integer linear programming formulation the solution times for finding one solution (T(One)) and for finding all solutions (T(All)) in seconds, and the number of solutions found (#-Sol). Notice that the number of solutions found by the improved integer program are far less than for the basic integer program as we enumerate equivalent solutions only once.

the numbers of residues for the fragments and added a random Gaussian noise on these numbers to reflect the errors in the measurement. Since those artificial instances come are generated by a rather simple model of real measurements, we have to be careful about a quantitative analysis. We generated one series of instances without noise and three further series with gaussian noise of mean 0 and different standard deviations.

First, we evaluated the running times to find one optimal solution (see Figure 4). As one can see, the noise has a growing effect the more variables we consider. At a first glance, this log-log-plot suggests a power law behavior. Hence, we fitted a power function for each series. For the sake of illustration, we only show the straight line representing the best fit of a power function to running times of the instances without noise. However, it demonstrates that the measurement points tend to follow a slight curvature to the left. Moreover, the exponent of this fit is roughly 2.1 and for the other series it tends towards 3 with growing noise. Though integer linear programming is exponential in general, this is appearantly not the case in the considered range $n \leq 1000$. It seems that the running times are dominated by solving linear programs. Since this involves solving systems of equations, we assume a cubic polynomial for a further fit of the running times. In fact, the curve that one can see in Figure 4, which corresponds to the measurements with the most noise, nicely fits the data.

The effect of the noise on the running times becomes more apparant when we enumerate all optimal solutions (see Figure 5). One reason for this is that the higher the minimum error the more optimal solutions exist at this

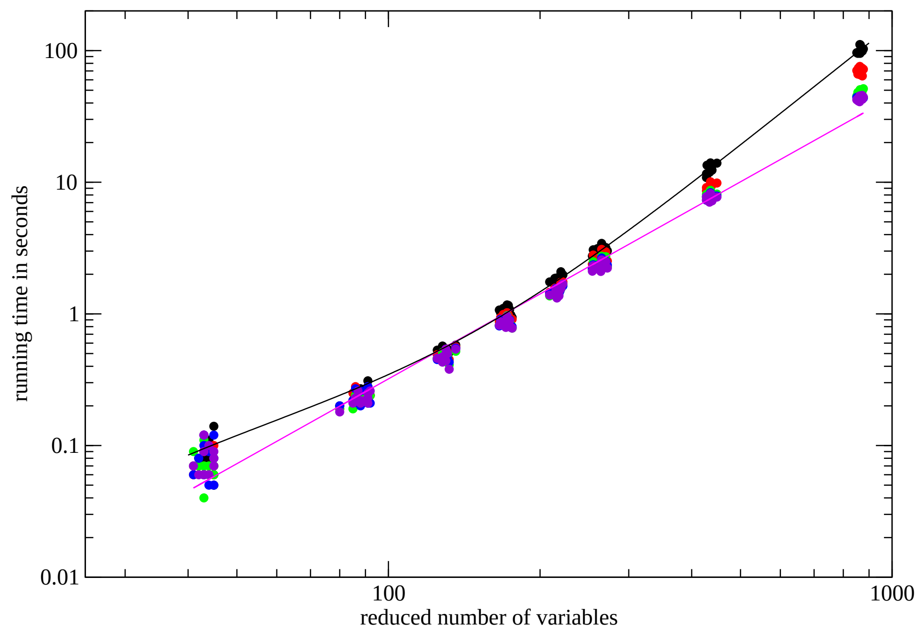


Figure 4: Running time of the Improved-ILP approach.

value. Moreover, the number of optimal solutions also grows with increasing number of variables. The straight lines in the log-log-plot show fits of power functions with exponents ranging from 3.2 to 4.3 roughly. However, the distribution of the running times is too broad to get a significant result with the limited number of measurements.

Anyways, it would be more interesting to have more real world instances than the randomly generated ones to evaluate our method and future work such as generalizing the combinatorial approach to more than two colors. However, gathering the experimental data involves a considerable effort. For the sake of completeness, we briefly explain the applied techniques in the following.

The entire HDX experiment was automated with a LEAP robot (HTS PAL, Leap Technologies, Carrboro, NC). To investigate digestion by proteases and provide control for deuterium incorporation analysis, a blank control was used. In the blank control, samples are diluted in H_2O solvent, instead of D_2O solvent, followed by digestion with proteases under quench conditions.

A triplicate blank control was performed to monitor digested fragments of 4 μM myoglobin in 50 mM sodium phosphate buffer, pH = 8.0. Initial en-

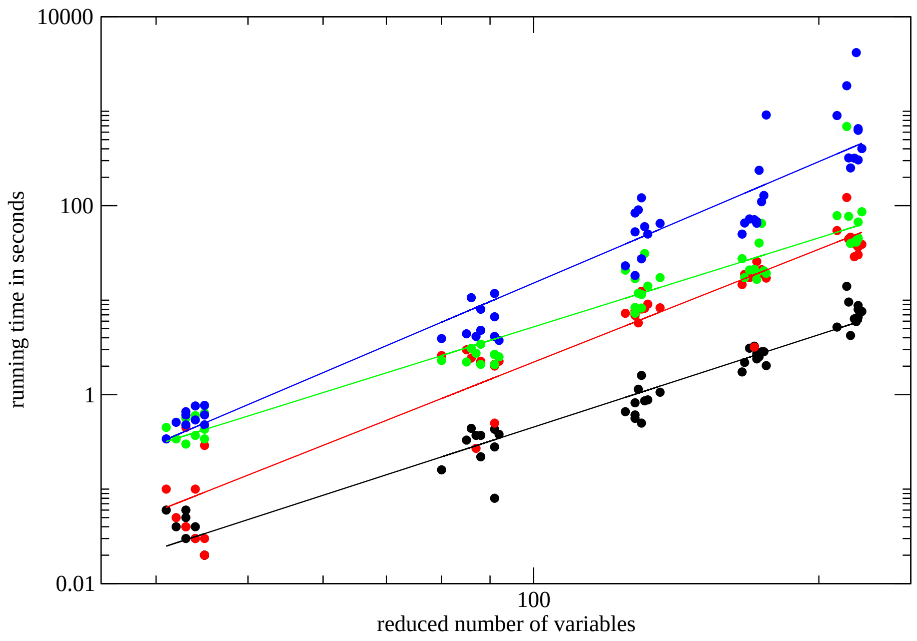


Figure 5: Time to enumerate all solution with minimum error for instances

zyme/protein ratios (w/w) were 1, 10.5 and 17 for pepsin, protease type XIII and protease type XVIII, respectively. Further diluted protease concentrations were also investigated. After digestion, the protein digest was injected from a 10 μL loop to either a 1 mm x 50 mm C5 column (Phenomenex) or a Pro-Zap Prosphere HP C18 HR 1.5u 10 mm x 2.1 mm (Alltech). A rapid gradient 2% B to 95% B in 1.5 min (A: acetonitrile/H2O/formic acid 5/94.5/0.5, B: acetonitrile/H2O/formic acid 95/4.5/0.5) was used to elute peptides.

The eluent was post-column split and infused by microelectrospray ionization into a custom built 14.5 T LTQ FT-ICR mass spectrometer. Data was analyzed by an in-house analysis package (Sasa 2007 paper, submitted).

8 Conclusion

We proposed an approach to assign exchange speeds to single residues from data of peptic fragments based on combinatorial optimization. The resulting algorithm is very efficient. Furthermore we gave a combinatorial algorithm for the case of two exchange speeds.

In the case of three or more different exchange speeds, the complexity of the problem remains open. Furthermore, we want to extend the combinatorial approach for three exchange speeds and so that it is able to enumerate all solutions even faster, which could be important as the sizes of the problems will probably increase in the future.

References

[Ach04] T. Achterberg. SCIP - a framework to integrate constraint and mixed integer programming. Technical Report 04-19, Zuse Institute Berlin, 2004. `http://www.zib.de/bib/pub/pw`.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice Hall Inc., Englewood Cliffs, NJ, 1993.

[CPL07] Ilog-cplex 10.0, 2007. `www.ilog.com`.

[GCA00] M.B. Goshe, Y.H. Chen, and V.E. Anderson. Identification of the sites of hydroxyl radical reaction with peptides by hydrogen/deuterium exchange: Pr evalence of reactions with the side chains. *Biochemistry*, 39(7):1761–1770, 2000.

[KHJ+06] S. Kang, A.M. Hawkridge, K.L. Johnson, D.C. Muddiman, and P.E. Prevelige. Identification of subunit-subunit interactions in bacteriophage p22 procapsids by chemical cross-linking and mass spectrometry. *Journal of Proteome Research*, 5(2):370–377, 2006.

[LC02] J.F. Leite and M. Cascio. Probing the topology of the glycine receptor by chemical modification coupled to mass spectrometry. *Biochemistry*, 41(19):6140–6148, 2002.

[LLE+02] T.T. Lam, J.K. Lanman, M.R. Emmett, C.L. Hendrickson, Marshall A.G., and P.E. Prevelige. Mapping of protein:protein contact surfaces by hydrogen/deuterium exchange, followed by online high-performance liquid chromatography-electrospray ionization fourier-transform ion-cyclotron-resonance mass analysis. *Journal of Chromatography A*, 982(1):85–95, 2002.

[MN99] K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, Cambridge, 1999. See also `http://www.mpi-sb.mpg.de/LEDA/`.

[SBH04] J.S. Sharp, J.M. Becker, and R.L. Hettich. Analysis of protein solvent accessible surfaces by photochemical oxidation and mass spectrometry. *Analytical Chemistry*, 76(3):672–683, 2004.

[SCI04] SCIL (Symbolic Constraints for Integer Linear programming), 2004. `http://www.mpi-sb.mpg.de/SCIL`.

[SoP04] SoPlex (Sequential object-oriented simplex class library), 2004. `www.zib.de/Optimization/Software/Soplex`.

[Wol98] Laurence A. Wolsey. *Integer programming.* Wiley-interscience series in discrete mathematics and optimization. Wiley & Sons, New York, 1998.

[Zui02] E.R.P. Zuiderweg. Mapping protein-protein interactions in solution by nmr spectroscopy. *Biochemistry*, 41(1):1–7, 2002.