

# GANA—A Genetic Algorithm for NMR Backbone Resonance Assignment

Hsin-Nan Lin, Kuen-Pin Wu, Jia-Ming Chang, Ting-Yi Sung and Wen-Lian Hsu\*

Institute of Information Science, Academia Sinica, Taipei, Taiwan

## Abstract

Automated backbone resonance assignment is a very challenging issue, because NMR data from different experiments often contains errors. In this paper, we present a method, called *GANA*, which uses a genetic algorithm to automatically perform backbone resonance assignment with high *precision* and *recall*. Precision is the number of correctly assigned residues divided by the number of assigned residues, and recall is the same numerator divided by the number of residues with known human curated answers. *GANA* takes spin systems as input data, and uses two data structures, *candidate lists* and *adjacency lists*, to assign the spin systems to each amino acid of a target protein. Using *GANA*, almost all spin systems can be mapped correctly onto a target protein, even if the data is noisy. We use the BMRB dataset (901 proteins) to test the performance of *GANA*. To test the robustness of *GANA*, we generate four additional datasets from the BMRB to simulate data errors of false positives, false negatives, linking errors, and a combination of these three error types to examine the fault tolerance of our method. The average precision rates of *GANA* on BMRB and the four simulated test cases are 99.61%, 99.55%, 99.34%, 99.35%, and 98.60%, respectively. The average recall rates of *GANA* on BMRB and the four simulated test cases are 99.26%, 99.19%, 98.85%, 98.87%, and 97.78%, respectively. We also test *GANA* on two real wet-lab datasets: hbSBD and hbLBD. The precision and recall rates of *GANA* on hbSBD are 95.12% and 92.86%, respectively; and those of hbLBD are 100% and 97.40%, respectively.

## 1. Introduction

Nuclear Magnetic Resonance spectroscopy (NMR) provides an alternative to X-ray diffraction for determining the three-dimensional structures of proteins in atomic resolution. NMR is also a powerful analytical tool for studying protein-ligand binding, protein-nucleic acid interactions, and protein dynamics, because it can probe protein molecules in a liquid environment. The first requirement for these studies is sequential resonance assignment on backbone structures. Researchers usually conduct several 3-D NMR experiments, such as CBCANH, CBCA(CO)NH, or HN(CO)CA, on  $^{13}\text{C}/^{15}\text{N}/^1\text{H}^{\text{N}}$ -labeled proteins, and 2-D NMR experiments, such as HSQC, on  $^{15}\text{N}/^1\text{H}^{\text{N}}$ -labeled proteins. These experiments are combined to construct sequential as-

---

\* Corresponding author. Email: [hsu@iis.sinica.edu.tw](mailto:hsu@iis.sinica.edu.tw)

signments. The multi-dimensional NMR spectra contain a mass of peaks, which in turn contain chemical shifts and corresponding intensities. Different kinds of NMR experiments provide different partial resonance information about particular atom groups on the backbone structure. For example, the 2-D HSQC experiment is used to detect whether there is a covalent bond between N and H<sup>N</sup>. If such a bond exists, a corresponding peak should appear in the spectrum, thereby showing the chemical shifts of the two atoms. The backbone resonance assignment problem is: how to identify the chemical shifts of particular atoms on the backbone structure from the connectivity information among the mass of isolated peaks. In the past, biologists had to make tedious backbone assignments manually or semi-manually during the spectra analysis process, but many automated tools using computational technologies are now available for the task. Even so, backbone resonance assignment is still very difficult in practice due to noise and errors in experimental NMR data.

NMR data often contains the four types of errors: noises (false positives), missing peaks (false negatives), clustered peaks, and inconsistent results among different experiments. Noisy peaks with high intensity could be accidentally regarded as real peaks, thereby creating false positive spin systems that coexist with real spin systems. A missing peak could be a weak spectrum peak mistakenly discarded due to its low intensity, or an empty spectrum peak that is not present due to the imperfect sensitivity of the NMR experiments. The third type of errors is the clustered peak, which occurs when the same kinds of atoms are located in similar environments. The last type of errors is inconsistency among different experiments. Theoretically, each atom on a backbone structure should have a fixed chemical shift. However, different NMR experiments on the same protein may generate slightly different chemical shifts for the same atom due to differences in experimental conditions, such as temperature, pH value, and isotope effects. As these four types of data errors appear simultaneously in the NMR spectra, the process of automated backbone resonance assignment is very challenging.

Most automated backbone resonance assignment programs comprise the following steps: 1) Filtering and referencing: filter peaks and relate resonances from different spectra. 2) Grouping: group resonances into spin systems. 3) Typing: identify the amino acid types of spin systems. 4) Linking: find and link sequential spin systems into segments. 5) Mapping: map spin-system segments to the primary sequence [1]. Note that the above procedures may be performed in a different order. Although many works have dealt with at least some of the above procedures [2--17], most models assume that the spin system grouping is already given and thus ignore a possible ambiguity problem in grouping. However, some works do perform all five procedures [18--24]. Readers are referred to [1] for a thorough survey.

In this paper, we present a method called GANA, which uses a genetic algorithm for automated backbone resonance assignment. GANA assumes that spin systems are given, and uses two data structures, *candidate lists* and *adjacency lists*, to assign spin systems to each amino acid of a target protein. In this way, almost all spin systems can be mapped onto a target protein.

To evaluate the performance of GANA, we use the precision and recall rates defined below:

$$\text{Precision} = \frac{\text{the number of correctly assigned residues}}{\text{the number of assigned residues}} \times 100\%$$

$$\text{Recall} = \frac{\text{the number of correctly assigned residues}}{\text{the number of residues with known answers}} \times 100\%$$

Note that the number of residues with known answers excludes Prolines, because they do not produce spectral peaks in the HSQC, CBCANH, and CBCA(CO)NH experiments. Both precision and recall are essential indicators for the performance of a backbone resonance assignment system. For example, consider a protein of length 100 with known assignments for all residues. If method A can only assign 40 positions of the entire sequence and all the assignments are correct, then the precision rate will be 100%, but the recall rate will be only 40%. If method B, on other hand, can assign 80 positions of the sequence and only 60 assignments are correct, then the precision will be 75% and the recall will be 60%. Recall rate in this assignment problem reflects the accuracy of the method as a whole, and precision rate indicates the reliability or confidence of the predicted assignments.

We tested GANA on the filtered BioMagResBank (BMRB) database of 901 proteins and two real NMR datasets: the substrate binding domain of BCKD (hbSBD) and the lipoic acid bearing domain of BCKD (hbLBD, [25]). To test the robustness of GANA, we also simulated real-world errors to generate the following four synthetic datasets from BMRB data set: false positives, false negatives, linking errors (explained in Section 3.2), and a combination of these three error types. When testing one round, the average precision rates of GANA on the BMRB dataset, false positives, false negatives, linking errors and the combination of test cases were 99.61%, 99.55%, 99.34%, 99.35%, and 98.60%, respectively; and the average recall rates were 99.26%, 99.19%, 98.85%, 98.87%, and 97.78%, respectively. Note that such a test of robustness has not been performed by previous methods. We believe it is crucial to be included for evaluation of different resonance assignment methods.

## 2. Method

### 2.1 NMR Experiments

In GANA, we use HSQC, CBCANH, and CBCA(CO)NH spectral data to assign chemical shifts to N, H<sup>N</sup>, C<sup>α</sup> and C<sup>β</sup> atoms on the backbone structure of a target protein. Figure 1a shows two consecutive residues, the (*i*−1)-th and the *i*-th residues, where only atoms along the backbone are depicted. For the *i*-th residue, the HSQC experiment detects H<sup>N</sup><sub>*i*</sub> and N<sub>*i*</sub> chemical shifts (Fig. 1b), the CBCANH experiment detects H<sup>N</sup><sub>*i*</sub>, N<sub>*i*</sub>, C<sup>α</sup><sub>*i*−1</sub>, C<sup>β</sup><sub>*i*−1</sub>, C<sup>α</sup><sub>*i*</sub> and C<sup>β</sup><sub>*i*</sub> chemical shifts (Fig. 1c), whereas the CBCA(CO)NH experiment detects H<sup>N</sup><sub>*i*</sub>, N<sub>*i*</sub>, C<sup>α</sup><sub>*i*−1</sub> and C<sup>β</sup><sub>*i*−1</sub> chemical shifts (Fig. 1d). By cross-referencing the HSQC, CBCANH, and CBCA(CO)NH peaks for the *i*-th residue, we can generate two consecutive spin systems, i.e., an inter-spin system, denoted by  $SS_{inter}(i)$ , and an intra-spin system, denoted by  $SS_{intra}(i)$ . The former contains the chemical shifts of C<sup>α</sup><sub>*i*−1</sub>, C<sup>β</sup><sub>*i*−1</sub>, and H<sup>N</sup><sub>*i*</sub>, N<sub>*i*</sub>, while the later contains the chemical shifts of C<sup>α</sup><sub>*i*</sub>, C<sup>β</sup><sub>*i*</sub> and H<sup>N</sup><sub>*i*</sub>, N<sub>*i*</sub>.

### 2.2 Chemical Shift Ranges of Amino Acids

Different amino acid residues may have different chemical shift ranges. TATAPRO II [19] uses BMRB statistical data for <sup>13</sup>C<sup>α</sup> and <sup>13</sup>C<sup>β</sup> chemical shifts to group the 20 amino acids into 8 categories (Table 1). In order to associate more spin systems with amino acids, we extend the upper and lower bounds by 10%. Although this introduces more false positives, their effect is more than offset by the reduction in missing peaks. The minimum and maximum chemical shifts are set to 13.5 and 79.75 respectively. For each amino acid, *x*, we use  $Range(x)$  to represent the chemical shift ranges of *x*. For example, the modified chemical shift range of Alanine (Ala) is  $13.5 \leq C^{\beta} \leq 26.4$ , denoted by  $Range(Ala)$ .

### 2.3 Spin System Groups

A spin system contains the chemical shifts of atoms within a residue. In GANA, we denote a set of spin systems as an  $SSGroup$  in which all systems have identical chemical shifts of H<sup>N</sup> and N. An ideal  $SSGroup$  should contain only one pair of  $SS_{inter}$  and  $SS_{intra}$  systems. In practice, however, an  $SSGroup$  may contain several ambiguous spin systems due to errors. The record of  $SSGroup$  is headed by a pair of chemical shifts of <sup>15</sup>N and <sup>1</sup>H<sup>N</sup>, followed by two or more spin systems, each of which is flagged. The flag indicates the type of spin system, either an inter- or intra-spin system, as denoted by 0 and 1, respectively. An example of an  $SSGroup$ , which is the basic unit of GANA, is shown in Figure 2. Hereafter, we assume that the whole system has *n* spin system groups, i.e.,  $SSGroup_1, \dots, SSGroup_n$ .

## 2.4 Data Structures

### 2.4.1 Candidate Lists

Automated backbone resonance assignment requires a typing step to associate spin systems with amino acid types. Most methods try to associate each spin system with its potential residue type(s). However, doing so may generate a large search space while performing linking and mapping due to the ambiguity of the spin systems. Instead of applying such a typing mechanism, GANA uses a data structure, *candidate lists*, to record potential spin systems for each residue in a target sequence.

For each residue  $i$  in a target protein, the candidate list  $CL(i)$  records all the  $k$ 's such that  $SSGroup_k$  matches residue  $i$  according to  $Range(i)$  and  $Range(i-1)$ . In other words, if some  $SS_{inter}$  (spin systems type 0) of  $SSGroup_k$  is within the  $Range(i-1)$  and some  $SS_{intra}$  (spin systems type 1) is within the  $Range(i)$ , then  $k$  is added to  $CL(i)$ . Note that if no such  $SSGroup_k$  is found for residue  $i$ , then  $CL(i) = \emptyset$  (an empty set). Intrinsically, Proline has no NMR spectral peaks, so its candidate list is  $\emptyset$ . Figure 3 shows the candidate lists of four residues.

### 2.4.2 Adjacency Lists

We construct an adjacency list for each  $SSGroup_i$ , denoted by  $AL(i)$ , to express the connectivity relations between  $SSGroup_i$  and all  $SSGroup_j$ . Two groups,  $SSGroup_i$  and  $SSGroup_j$ , are said to be *connected* if the absolute differences of their corresponding chemical shifts are within predefined thresholds. That is,

$$\left|C_{i,inter}^{\alpha} - C_{j,intra}^{\alpha}\right| < \delta_{\alpha} \text{ and } \left|C_{i,inter}^{\beta} - C_{j,intra}^{\beta}\right| < \delta_{\beta} \quad (1)$$

or

$$\left|C_{i,intra}^{\alpha} - C_{j,inter}^{\alpha}\right| < \delta_{\alpha} \text{ and } \left|C_{i,intra}^{\beta} - C_{j,inter}^{\beta}\right| < \delta_{\beta}, \quad (2)$$

where  $C_{i,inter}^{\alpha}$  denotes the  $C^{\alpha}$  chemical shift of the inter-spin system of  $SSGroup_i$ , and the other terms are similarly denoted, and  $\delta_{\alpha}$  and  $\delta_{\beta}$  are the predefined thresholds.

Note that each  $SSGroup$  may contain more than one inter-spin (or intra-spin) system, but as long as at least one pair of inter-spin and intra-spin systems satisfy either of the above condition,  $SSGroup_i$  and  $SSGroup_j$  are said to be connected.

Each  $AL(i)$  contains two kinds of lists: an  $L$ -List, denoted by  $AL_L(i)$ , and an  $R$ -List, denoted by  $AL_R(i)$ . The  $L$ -List records all  $SSGroup_j$  whose  $SS_{intra}$  are connected with  $SS_{inter}$  of  $SSGroup_i$ . In this case, we have a connected fragment,  $ji$ , of length two. The  $R$ -List records all  $SSGroup_k$  whose  $SS_{inter}$  are connected with  $SS_{intra}$  of  $SSGroup_i$ . In this case, we have a connected fragment,  $ik$ , of length two.  $AL_L(i)$  and  $AL_R(i)$  are then sorted in ascending order by the sum of absolute differences on the left-hand side of Inequalities (1) and (2). The purpose of sorting these lists is to ensure that closer spin

systems are used for assignments later in our genetic algorithm. Figure 4 shows an example of the adjacency lists of  $n$  *SSGroups*.

## 2.5. Our Genetic Algorithm Model

Genetic algorithms (GAs) proposed by Holland [26] simulate the process of biological evolution in computers. A great deal of research has shown that GAs are efficient for solving problems that have a very large search space [27]. GAs usually comprise chromosome initialization, reproduction, crossover, and mutation operations. They also require a fitness function.

To apply GANA, we are given a target protein of length  $l$ ,  $n$  *SSGroups* numbered from 1 through  $n$ , and first generate the candidate and adjacency lists. Note that the lists are regarded as two assignment constraints of the genetic algorithm in GANA. The former are used to select *SSGroup* candidates for residues, and the latter are used to construct or extend the connected fragments from a specific *SSGroup*. In this section, we use  $i$  to denote the residue located at the  $i$ th position of the target protein,  $x_i$  to denote an *SSGroup* in  $CL(i)$  and  $1 \leq x_i \leq n$ .

In the iterative procedure of chromosome generation of GANA, we assign one of the *SSGroups* in  $CL(i)$  to each residue,  $i$ . Note that each *SSGroup* may be found in more than one candidate list, but it can only be assigned to at most one residue. If all *SSGroups* in  $CL(i)$  are already assigned to other residues, then residue  $i$  will be assigned  $-1$ , and the chemical shifts of  $C_i^\alpha$ ,  $C_i^\beta$  will be determined by  $SS_{inter}(i+1)$ .

### 2.5.1 Chromosome initialization

Each chromosome in GANA represents a candidate solution for backbone resonance assignment, and *SSGroups* are the basic units (*genes*) of such solutions. A chromosome,  $ch$ , has  $l$  components corresponding to all residues of the target protein. Each component of  $ch$  is denoted by  $ch[i]$ , which is assigned an *SSGroup* or  $-1$ , and written as  $ch[i] = x_i$  or  $-1$ .

To create a new chromosome,  $ch$ , we initially set all  $ch[i]$  as undefined and then iteratively perform the following steps: (1) Randomly select a residue,  $i$ , of a target protein that has not been assigned an *SSGroup* or  $-1$ , where  $-1$  indicates that no available *SSGroup* can be assigned to a specific residue in the target protein by our method. (2) Given a residue  $i$ , randomly select an *SSGroup*  $x_i$  from  $CL(i)$  that has not been assigned to any other residue and assign  $x_i$  to  $ch[i]$  (define  $ch[i] = x_i$ ). (3) Extend connected fragments by examining  $AL(x_i)$ . When performing Step (2), if no *SSGroup*  $x_i$  from  $CL(i)$  can be found for residue  $i$ , i.e., all *SSGroups* have been assigned to other residues, assign  $-1$  to  $ch[i]$ . When all residues have been assigned an *SSGroup*  $x_i$  or  $-1$ , i.e.,  $ch[i] = x_i$  or  $-1$ , return the  $ch$ , which is a new chromosome.

We now explain Step (3) in detail. Given  $ch[i] = x_i$ , we extend a connected fragment from  $x_i$  leftward (called *left extension*) and rightward (called *right extension*). To perform left extension, we sequentially select an *SSGroup*  $x_{i-1}$  from  $AL_L(x_i)$  that is also in  $CL(i-1)$  and has not been assigned yet. Then, we assign *SSGroup*  $x_{i-1}$  to residue  $i-1$  (define  $ch[i-1] = x_{i-1}$ ) and obtain a connected fragment of spin systems  $x_{i-1}x_i$  for residues  $i-1$  and  $i$ . We repeat the above process for residues  $i-1, i-2, \dots$  for left extension until no further extension is possible. Similarly, for  $ch[i] = x_i$  we use  $AL_R(x_i)$  to extend connected fragments to the right for residues  $i+1, i+2, \dots$  until no further extension is possible. In other words, we sequentially select an *SSGroup*  $x_{i+1}$  that is also in  $CL(i+1)$  and has not been assigned yet. Then, we assign *SSGroup*  $x_{i+1}$  to residue  $i+1$  (define  $ch[i+1] = x_{i+1}$ ).

We repeat chromosome initialization many times to generate a population of chromosomes. There are two advantages to creating chromosomes  $ch$  this way: 1) all *SSGroups* assigned to residues satisfy the chemical shift ranges of each corresponding residue; and 2) a  $ch$  contains several connected fragments.

### 2.5.2 Fitness Function

The fitness function guides the evolutionary direction of a population. Thus, the more connected fragments a  $ch$  has, and the longer they are, the higher its fitness score will be.

To define the fitness score of  $ch$ , we first examine each residue  $i$  with  $ch[i] = x_i$  for  $i = 1, 2, \dots, l$  to determine whether or not  $x_i$  can connect with  $x_{i-1}$  ( $= x_a$ ) and with  $x_{i+1}$  ( $= x_b$ ). We then define two variables,  $D_L(i)$  and  $D_R(i)$ , to reflect the degree of connectivity between  $(x_i, x_a)$  and  $(x_i, x_b)$ , respectively:

$$D_L(i) = \begin{cases} \delta, & \text{if } x_a \text{ does not connect with } x_i \\ |C_{i,inter}^\alpha - C_{a,intra}^\alpha| + |C_{i,inter}^\beta - C_{a,intra}^\beta|, & \text{otherwise} \end{cases}$$

$$D_R(i) = \begin{cases} \delta, & \text{if } x_i \text{ does not connect with } x_b \\ |C_{i,intra}^\alpha - C_{b,inter}^\alpha| + |C_{i,intra}^\beta - C_{b,inter}^\beta|, & \text{otherwise,} \end{cases}$$

where  $\delta$  is a special flag symbol. Each connectivity is then given a score:

$$S_L(i) = \begin{cases} -3, & \text{if } D_L(i) = \delta \\ 5, & \text{if } D_L(i) < 0.1 \\ 4, & \text{if } 0.1 \leq D_L(i) < 0.3 \\ 3, & \text{if } 0.3 \leq D_L(i) < 0.5 \\ 2, & \text{if } 0.5 \leq D_L(i) < 0.7 \\ 1, & \text{otherwise.} \end{cases}$$

$S_R(i)$  is defined similarly. The closer the two connected *SSGroups* are, the higher the score they receive. Now, we define the linking score for each  $x_i$  as follows:

$$LS(x_i) = \begin{cases} 0, & \text{if } x_i = -1 \\ 1, & \text{if } x_i \neq -1, x_a = x_b = -1 \\ S_L(i), & \text{if } x_i, x_a \neq -1, x_b = -1 \\ S_R(i), & \text{if } x_i, x_b \neq -1, x_a = -1 \\ S_L(i) + S_R(i), & \text{otherwise.} \end{cases}$$

The fitness score of  $ch$  is then defined as:

$$FitnessScore(ch) = \sum_{i=1}^l LS(x_i).$$

### 2.5.3 Reproduction Operation

For the reproduction operation, GANA uses a selection procedure to produce the next generation according to the fitness score of each chromosome in the current population. After ranking chromosomes according to their fitness scores, we keep the top 50% of them for the next generation. They are also treated as *parent candidates* in the crossover operation (to be explained in Section 2.5.4). The remaining 50% of chromosomes in the next generation, called *offspring*, are produced by a crossover operation through parent candidates or random chromosome initialization. We use a parameter called the *CrossoverRate* (0–100) to denote the proportion of offspring produced by the crossover operation. If the *CrossoverRate* is 40, it means that 40% of the offspring are produced by the crossover operation, and the remaining 60% are produced by chromosome initialization. In summary, the chromosomes of the next generation consist of 50% from parent candidates, 20% ( $= 0.4 \times 0.5$ ) from crossover operations, and 30% from chromosome initialization.

After producing the new generation, all the chromosomes go through a mutation operation (described in Section 2.5.5). We use a parameter called the *MutationRate* (0-1000) to denote the probability of mutation for each locus in a  $ch$ . If the *MutationRate* is 5, then each locus has 0.005 probability of being mutated to another value. Finally, we check each pair of chromosomes,  $ch_i$  and  $ch_j$ , with  $i \neq j$  in the population to determine whether or not they are identical; if they are identical, then one of them will be re-produced by chromosome initialization.

### 2.5.4 Crossover Operation

Our crossover operation produces an offspring  $ch$  that has a higher fitness score and has inherited as many connected fragments as possible from its parents. To achieve this, we randomly select two different chromosomes from the *parent candidates*, say  $p_1$  and  $p_2$ . Let  $ch$  denote the new offspring chromosome to be produced by the crossover operation from  $p_1$  and  $p_2$ . Initially, all  $ch[i]$  are undefined. The iterative procedure of crossover operation is as follows:

1. Randomly select a residue  $i$  of the target protein that has not been assigned an *SSGroup* or  $-1$ . If all  $l$  residues have been assigned, proceed to Step 4.
2. Randomly select a parent  $p$  ( $p = p_1$  or  $p_2$ ). If  $p[i] = -1$ , then assign  $-1$  to  $ch[i]$  (define  $ch[i] = -1$ ) and return to Step 1. Otherwise, proceed as follows: if  $p[i]$  has not been assigned to any other residue of  $ch$ , then assign  $p[i]$  to  $ch[i]$ , i.e., define  $ch[i] = p[i]$ , and go to Step 3. Otherwise, assign  $ch[i]$  a special symbol  $\Delta$  (define  $ch[i] = \Delta$ ) to indicate that it will be re-assigned later with another *SSGroup* that has not been used yet, and return to Step 1.
3. Extend the connected fragment from  $ch[i]$  by referencing  $p$  and return to Step 1.
4. To process those residues with  $ch[i] = \Delta$ , randomly select a residue,  $i$ , such that  $ch[i] = \Delta$ . Then randomly select an *SSGroup*  $x_i$  from  $CL(i)$  that has not been assigned to any other residue. If all *SSGroups* in  $CL(i)$  have been assigned, assign  $-1$  to  $ch[i]$ ; otherwise, assign  $x_i$  to  $ch[i]$ . Repeat this step until all residues with  $ch[i] = \Delta$  have been processed and return the  $ch$ , which is the new offspring chromosome.

Given that  $ch[i] = p[i]$ , Step 3 is similar to that of chromosome initialization. Left extension proceeds as follows. We examine  $p[i-1]$  to determine whether it has not been used in  $ch$  yet. If so, we define  $ch[i-1] = p[i-1]$  and repeat the left extension procedure for  $i-2, \dots$  until no further extension is possible; otherwise, we stop left extension. Then, we proceed to do right extension, which is similar to left extension.

### 2.5.5 Mutation Operation

Mutation operations may alter the values of some positions in a chromosome, so we use a modified chromosome to replace the original chromosome. This operation provides a population with reasonable diversity and prevents the offspring from resembling their parents so that GAs do not fall into a local maximum. Though most mutation operations are single or multiple point mutations, we discard such mutations since they may split a connected fragment into pieces. Instead, we use a different mutation strategy: once a locus is mutated, our operation also mutates its subsequent loci.

Let  $mch$  denote a new mutated chromosome to be generated. Our mutation operation starts with the first locus, i.e.,  $i = 1$ . We distinguish two cases. First, residue  $i$  is mutated by the following steps: (1) Randomly select an *SSGroup*  $x_i$  from  $CL(i)$  that has not been assigned to any other residue of  $mch$ , and define  $mch[i] = x_i$ . If no such  $x_i$  from  $CL(i)$  can be found, define  $mch[i] = -1$  and proceed to the next residue  $i + 1$ . (2) Perform only the right extension from  $x_i$  by referencing  $AL(x_i)$  until  $i = k$  ( $k$  is an integer  $\leq l$ ) for no further extension is possible. This extension procedure is the same as that for chromosome initialization, described in Section 2.5.1. In the second case, residue  $i$  is not mutated. Then we examine  $ch[i]$  to determine whether or not it has

been assigned to *mch* yet. If it has not been assigned to *mch*, we assign  $ch[i]$  to  $mch[i]$  and proceed to the next residue  $i + 1$ ; otherwise, we perform Steps (1) and (2) above until  $i = k$ , or no further extension is possible.

We then repeat the above mutation operation with  $i = i + 1$  or  $k + 1$  (depending on whether extension is performed) until all residues have been processed, and output the resulting *mch* to replace the original *ch*.

### 2.5.6 Creating Backbone Resonance Assignment from the Best Chromosome

After evaluating the fitness function of all chromosomes in the current generation, we select the chromosome, *ch*, that has the highest fitness score as the candidate solution for backbone resonance assignment. (Recall that  $n$  denotes the number of spin systems.) For each residue  $i$ , if  $ch[i] = x_i$  where  $x_i \in \{1, 2, \dots, n\}$ , we report that *SSGroup*  $x_i$  is assigned to residue  $i$ . In other words, the chemical shifts of  $N_i$  and  $H_i^N$  are the chemical shifts of  $^{15}N$  and  $^1H^N$  in *SSGroup*  $x_i$ . Also,  $SS_{intra}(i)$ , which represents the chemical shift of  $C_i^\alpha$  and  $C_i^\beta$ , is the intra-spin system of *SSGroup*  $x_i$ . If there is more than one intra-spin system in *SSGroup*  $x_i$ , to assign  $SS_{intra}(i)$ , we arbitrarily select from any intra-spin system that connects  $SS_{inter}(i + 1)$  when residue  $i + 1$  is assigned with another *SSGroup*.

## 3. Experimental Results

GANa, which was developed under Linux Redhat 9.0, is implemented as a standard C++ program. We use BMRB datasets and real wet-lab datasets to estimate the precision and recall rates of GANA. The C++ source code of GANA and all synthetic datasets are available at <http://bioinformatics.iis.sinica.edu.tw/GANA/>.

The parameters used in each single *round* of GANA are as follows: the number of chromosomes in each generation = 600, the number of generations for evolution in a single round = 500, the *CrossoverRate* = 70, and the *MutationRate* = 2. Because GAs may fall into a local maximum, we perform multiple rounds to select the chromosome with the highest fitness score as the final assignment for each protein. Each round of the GA is a complete and independent assignment procedure. In the tables in this section, we use P and R to denote the precision rate and the recall rate, respectively, expressed in percentage.

### 3.1 Results of raw datasets

#### 3.1.1 The BMRB dataset

We downloaded the full BMRB dataset containing 3,129 proteins on September 10, 2004. Since protein lengths in NMR experiments are in general less than 400, we choose proteins of lengths 50 to 400 that have at least 50% residues with known hu-

man curated answers as our dataset. The resulting dataset contains 901 proteins, the average length of which is 128.17, and the average proportion of residues with known answers in a protein sequence is 86.3%.

For each test protein, we generated simulated *SSGroups* according to the chemical shifts assigned to each residue. Note that if the chemical shifts of N or H<sup>N</sup> on residue  $i$  were unavailable, we did not generate a corresponding *SSGroup* for residue  $i$ .

The single round precision and recall rates of GANA for the dataset are 99.61% and 99.26%, respectively; and after ten rounds they are 99.67% and 99.34%, respectively. The improvement after ten rounds compared to a single round was small, which implies that GANA is less likely to be trapped in a local optimum.

Although our BMRB dataset contains proteins with lengths less than 400, we also tested GANA on a very challenging 723-residue Malate Synthase G, BMRB #5471, to demonstrate its ability to handle long proteins. In addition to the original data, we use synthetic data with false positives, false negatives and linking errors (explained in 3.2). The precision and recall rates are both 100% for the original data; 100% and 99.8% respectively for the data with false positives; 98.9% and 97.7% respectively for the data with false negatives; 98.92% and 97.57% respectively for the data with linking errors.

As both MARS [28] and GANA take spin systems as input, we compare these two methods. In [28], MARS is tested its performance on 11 proteins of BMRB; thus, we test GANA under the same conditions, except that BMRB #547 and #4106 are removed because they lack C $\beta$  and N chemical shifts, respectively. The results are given in Table 2. (Since MARS [28] reports only the number of correctly assigned residues, the experimental results are reported in terms of recall rate, i.e., the accuracy rate.)

Table 2: Recall rates of GANA and MARS on 9 proteins of the original data and with linking errors

Protein ID	$n$	MARS		GANA	
		Original data	Linking errors	Original data	Linking errors
5471	723	97.71	95.11	100.00	97.57
4354	370	99.40	98.51	100.00	100.00
4384	262	98.64	98.64	100.00	100.00
4022	260	99.59	99.59	100.00	100.00
4457	227	80.24	81.44	100.00	100.00
4402	210	81.58	81.05	92.71	98.44
4341	192	95.73	95.73	100.00	100.00
4082	139	100.00	100.00	100.00	100.00

4136	110	94.59	86.49	95.35	93.02
Average	277	94.16	92.95	98.67	98.78

$n$  = the number of residues in the protein.

### 3.1.2 Two real wet-lab datasets

In addition to the BMRB dataset, we also used two real wet-lab datasets: the substrate binding domain of BCKD (hbSBD) and the lipoic acid bearing domain of BCKD (hbLBD, [25]). Each dataset contains more than 50% false positives and false negatives together. Details of the two datasets are given in Table 3.

The single round precision and recall rates of GANA for hbSBD was 95.12% and 92.86%, respectively; and for hbLBD they were 100% and 97.40%, respectively.

## 3.2 Simulated datasets

To compare different backbone resonance assignment methods, researchers prefer to use real wet-lab datasets for testing; however, such datasets are quite scarce. Thus, tests have also been conducted using the BMRB dataset, which contains a large number of proteins with known human curated answers. The raw BMRB data contains no errors and can be regarded as perfect data. To simulate real-world noisy data, we generate a large dataset from the BMRB dataset to simulate false positives and false negatives. Though real data contains the error type of clustered peaks, we do not simulate this error type since the problem of clustered peaks is an intrinsic property of NMR data. Instead, we simulate linking errors, as in PACES [7], and a combination of the above error types. In summary, we modify the original BMRB data to generate four kinds of synthetic datasets simulating error types: (1) false positives, (2) false negatives, (3) linking errors, and (4) a combination of the previous three cases. These datasets, which – to the best of our knowledge – are unique to our approach, are constructed as follows.

To create a false positive dataset, we add synthetic intra- and inter-spin systems to the *SSGroups*. We then define two types of synthetic spin systems:  $\alpha$  and  $\beta$ . An  $\alpha$  (respectively,  $\beta$ ) spin system contains an artificial  $C^\alpha$  (respectively,  $C^\beta$ ) and an inherent  $C^\beta$  (respectively,  $C^\alpha$ ) chemical shift. We randomly select 25% of the *SSGroups* and add an  $\alpha$  intra-spin system to each of them. Similarly, we add  $\beta$  intra-,  $\alpha$  inter-, and  $\beta$  inter-spin systems to 25%, 12.5%, and 12.5% of the *SSGroups*, respectively. Note that all selections are random, but not independent, so it is possible for an *SSGroup* to be added to more than one type of synthetic spin system. Figure 5a shows an original *SSGroup*, while Figure 5b shows the modified *SSGroup* of a false positive case, where the first underlined spin system is an extra  $\beta$  inter-spin system and the second underlined spin system is an extra  $\alpha$  intra-spin system. In this dataset, 75% of the data

contains false positive errors.

To create a false negative dataset, we assume that some CBCA(CO)NH peaks are missing. In this situation, we cannot recognize which  $C^\alpha$  or  $C^\beta$  peak in CBCANH experiments belongs to the inter-residue. Thus, we generate all possible combinations of spin systems to solve the problem. We then randomly select 50% of the *SSGroups* to simulate the case where either the  $C^\alpha$  or the  $C^\beta$  peak in CBCA(CO)NH experiments is missing (25% for each). For example, in Fig. 5c we assume the  $C^\alpha$  peak in the CBCA(CO)NH experiment is missing, so we generate all possible spin systems.

To create a linking error dataset, we modify the  $C^\alpha$  and  $C^\beta$  chemical shifts of the inter-spin systems for all *SSGroups*. Each modified  $C^\alpha$  (respectively,  $C^\beta$ ) chemical shift differs by  $\pm 0.2$  (respectively,  $\pm 0.4$ ) ppm from the original data. Furthermore, these chemical shift differences follow normal distributions with mean 0 and standard deviations of 0.08 and 0.16 ppm for  $C^\alpha$  and  $C^\beta$ , respectively. Figure 5d shows the modified *SSGroup* of the linking error case.

To create a combined-error dataset, we use the above three modification methods.

The experiment results of GANA for the synthetic datasets are listed in Table 4.

PACES [7] only reported experiment results for synthetic data of linking errors using 21 proteins from the 901-protein BMRB dataset. To compare GANA with PACES, we tested it on the same data with a similar parameter setting. PACES can only handle 20 of the proteins (BMRB #4402 is excluded), whereas GANA can handle all 21 proteins. Note that the final results of PACES are post-edited by human experts, whereas GANA is fully automated. The test results the 21 proteins tested by GANA and the 20 proteins tested by PACES are listed in Table 5.

#### 4. Conclusion

In this paper, we have presented a genetic algorithm for backbone resonance assignment, called GANA, which is fully automated and can deal with noisy data. The performance of GANA on our test datasets is good for both the precision and the recall rates. In particular, GANA yields better recall rates than either PACES or MARS. Note that the recall rate represents the accuracy of an assignment method. The higher recall rates of GANA can be attributed to its two data structures: *candidate lists* and *adjacency lists*. GANA takes spin systems as input data, and uses the two data structures to assign the spin systems to the amino acids of a target protein. This design enables GANA to correctly map nearly all spin systems onto a target protein. Thus, the recall rates of GANA are generally high.

We have also proposed a scheme to generate a large dataset from BMRB to simulate real noisy data of false positives, false negatives, linking errors and a combination of these three error types. The synthetic datasets provide a good platform for compar-

ing different assignment systems.

### Acknowledgement

This work is supported in part by the thematic program of Academia Sinica under grant AS91IIS1PP and 94B003.

### References

- [1] Moseley, H. and Montelione, G. (1999) Automated analysis of NMR assignments and structures for proteins. *Curr. Opin. Struc. Biol.*, **9**, 635–642.
- [2] Bailey-Kellogg, C., Widge, A., Kelley, J., Berardi, M., Bushweller, J., and Donald, B. (2000) The noesy jigsaw: automated protein secondary structure and main-chain assignment from sparse, unassigned nmr data. In *The Fourth Annual International Conference on Research in Computational Molecular Biology (RECOMB 00')* pp. 33–44.
- [3] Bailey-Kellogg, C., Chainraj, S., and Pandurangan, G. (2004) A random graph approach to nmr sequential assignment. In *The Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB 04')* pp. 58–67.
- [4] Bartels, C., Guntert, P., Billeter, M., and Wuthrich, K. (1997) Garant—a general algorithm for resonance assignment of multidimensional nuclear magnetic resonance spectra. *J. Comput. Chem.*, **18**, 139–149.
- [5] Chen, Z., Jiang, T., Lin, G., Wen, J., Xu, D., and Xu, Y. (2002) Improved approximation algorithms for nmr spectral peak assignment. *Lect. Notes Comput. Sci.*, **2452**.
- [6] Chen, Z., Jiang, T., Lin, G., Wen, J., Xu, D., Xu, J., and Xu, Y. (2003) Approximation algorithms for nmr spectral peak assignment. *Theor Comput Sci*, **299**, 211–229.
- [7] Coggins, B. and Zhou, P. (2003) Paces: Protein sequential assignment by computer-assisted exhaustive search. *J. Biomol. Nmr.*, **26**, 93–111.
- [8] Guntert, P., Salzmann, M., Braun, D., and Wuthrich, K. (2000) Sequence-specific nmr assignment of proteins by global fragment mapping with the program mapper. *J. Biomol. Nmr.*, **18**, 129–137.
- [9] Hitchens, T., Lukin, J., Zhan, Y., McCallum, S., and Rule, G. (2003) Monte: An automated monte carlo based approach to nuclear magnetic resonance assignment of proteins. *J. Biomol. Nmr.*, **25**, 1–9.
- [10] Hyberts, S. and Wagner, G. (2003) Ibis—a tool for automated sequential assignment of protein spectra from triple resonance experiments. *J. Biomol. Nmr.*, **26**, 335–344.

- [11] Langmead, C., Yan, A., Lilien, R., Wang, L., and Donald, B. (2003) Large a polynomialtime nuclear vector replacement algorithm for automated nmr resonance assignments. In *The Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB'03)* pp. 176–187.
- [12] Lin, G., Xu, D., Chen, Z., Jiang, T., Wen, J., and Xu, Y. (2003) An efficient branch-and-bound algorithm for the assignment of protein backbone nmr peaks. In *Proceedings of the First IEEE Computer Society Bioinformatics Conference (CSB'02)* pp. 165–174.
- [13] Malmodin, D., Papavoine, C., and Billeter, M. (2003) Fully automated sequence-specific resonance assignments of heteronuclear protein spectra. *J. Biomol. Nmr.*, **27**, 69–79.
- [14] Ou, H., Lai, H., Serber, Z., and Dotsch, V. (2001) Efficient identification of amino acid types for fast protein backbone assignments. *J. Biomol. Nmr.*, **21**, 269–273.
- [15] Slupsky, C., Boyko, R., Booth, V., and Sykes, B. (2003) Smartnotebook: A semi-automated approach to protein sequential nmr resonance assignments. *J. Biomol. Nmr.*, **27**, 313–321.
- [16] Wang, X., Xu, D., Slupsky, C., and Lin, G. (2003) Automated protein nmr resonance assignments. In *Proceedings of the Second IEEE Computer Society Bioinformatics Conference (CSB'03)* pp. 197–208.
- [17] Xu, Y., Xu, D., Kim, D., Olman, V., Razumovskaya, J., and Jiang, T. (2002) Automated assignment of backbone nmr peaks using constrained bipartite matching. *Comput. Sci. Eng.*, **4**, 50–62.
- [18] Atreya, H., Sahu, S., Chary, K., and Govil, G. (2000) A tracked approach for automated nmr assignments in proteins (tatapro). *J. Biomol. Nmr.*, **17**, 125–136.
- [19] Atreya, H., Chary, K., and Govil, G. (2002) Automated nmr assignments of proteins for high throughput structure determination: Tatapro ii. *Curr. Sci. India.*, **83**, 1372–1376.
- [20] Buchler, N., Zuiderweg, E., Wang, H., and Goldstein, R. (1997) Protein heteronuclear nmr assignments using mean-field simulated annealing. *J. Magn. Reson.*, **125**, 34–42.
- [21] Leutner, M., Gschwind, R., Liermann, J., Schwarz, C., Gemmecker, G., and Kessler, H. (1998) Automated backbone assignment of labeled proteins using the threshold accepting algorithm. *J. Biomol. Nmr.*, **11**, 31–43.
- [22] Li, K. and Sanctuary, B. (1997) Automated resonance assignment of proteins using heteronuclear 3d nmr. backbone spin systems extraction and creation of polypeptides. *J. Chem. Inf. Comp. Sci.*, **37**, 359–366.
- [23] Lukin, J., Gove, A., Talukdar, S., and Ho, C. (1997) Automated probabilistic

- method for assigning backbone resonances of (c-13,n-15)-labeled proteins. *J. Biomol. Nmr.*, **9**, 151–166.
- [24] Zimmerman, D., Kulikowski, C., Huang, Y., Feng, W., Tashiro, M., Shimotakahara, S., Chien, C., Powers, R., and Montelione, G. (1997) Automated analysis of protein nmr assignments using methods from artificial intelligence. *J. Mol. Biol.*, **269**, 592–610.
- [25] Chang, C., Chou, H., Chuang, J., Chuang, D., and Huang, T. (2002) Solution structure and dynamics of the lipoic acid-bearing domain of human mitochondrial branched-chain alpha-keto acid dehydrogenase complex. *J. Biol. Chem.*, **277**, 15865–15873.
- [26] Holland, J. (1975) Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press, Ann Arbor.
- [27] Mitchell, M. (1996) An introduction to genetic algorithms, MIT Press, Cambridge, Mass.
- [28] Jung, Y.S. and Zweckstetter, M. (2004) Mars - robust automatic backbone assignment of proteins. *J Biomol Nmr*, **30**, 11-23.

Table 1. Amino acid types based on their carbon chemical shift characteristics [19]

Carbon chemical shift	Amino acid
Absence of C <sup>β</sup>	Gly
14 < C <sup>β</sup> < 24	Ala
56 < C <sup>β</sup> < 67	Ser
C <sup>α</sup> < 64 and 24 < C <sup>β</sup> < 36	Lys, Arg, Gln, Glu, His, Trp, Cys <sup>red</sup> , Val and Met
C <sup>α</sup> ≥ 64 and 24 < C <sup>β</sup> < 36	Val
C <sup>α</sup> < 64 and 36 < C <sup>β</sup> < 52	Asp, Asn, Phe, Tyr, Cys <sup>oxd</sup> , Ile and Leu
C <sup>α</sup> ≥ 64 and 36 < C <sup>β</sup> < 52	Ile
–	Pro
C <sup>β</sup> > 67	Thr

Since Glycine has only a proton on its side chain, it has no C<sup>β</sup> chemical shift. Proline intrinsically has no peaks appearing in NMR spectra, so it has neither C<sup>α</sup> nor C<sup>β</sup> chemical shifts. Cys<sup>red</sup> and Cys<sup>oxd</sup> represent reduced Cystein and oxidized Cystein, respectively.

Table 3. Detailed attributes of the hbSBD and hbLBD datasets.

Dataset	hbSBD	hbLBD
# of amino acids (A.A.)	53	85
# of A.A. manually assigned by biologists	42	80

# of HSQC peaks	58	78
# of CBCA(CO)NH peaks	258	271
# of CBCANH peaks	224	620
False positives (CBCA(CO)NH)	67.4%	41.0%
False positives (CBCANH)	25.0%	48.4%

Table 4. Experimental results of GANA for different datasets and rounds.

Test Dataset	1 round		10 rounds	
	P	R	P	R
Original	99.61	99.26	99.67	99.34
False Positives	99.55	99.19	99.64	99.32
False Negatives	99.34	98.85	99.53	99.10
Linking Errors	99.35	98.87	99.55	99.18
Combined errors	98.60	97.78	98.96	98.28

Table 5: Experimental results of GANA and PACES for 21 proteins with linking errors

Protein ID	$n$	GANA		PACES	
		P	R	P	R
4354	370	100.00	100.00	100.00	94.22
5316	288	99.59	97.98	100.00	99.62
5468	266	100.00	100.00	100.00	98.33
4384	262	100.00	100.00	100.00	93.64
4022	260	100.00	100.00	100.00	92.95
4102	232	100.00	98.90	100.00	94.81
4844	221	100.00	100.00	100.00	98.98
4836	217	100.00	100.00	100.00	96.59
4834	189	100.00	100.00	100.00	99.39
4094	133	100.00	100.00	100.00	100.00
5142	130	100.00	100.00	100.00	100.00
4444	128	100.00	100.00	100.00	100.00
4032	124	100.00	100.00	100.00	100.00
4152	214	100.00	100.00	100.00	96.94
4402	210	98.44	98.44	0.00	0.00
4082	139	100.00	100.00	100.00	99.24
4722	168	100.00	100.00	100.00	97.26
4769	76	100.00	100.00	100.00	87.88
4457	227	100.00	100.00	100.00	17.39
4341	192	100.00	100.00	100.00	51.43
4136	110	95.24	93.02	100.00	81.94
Average	197.90	99.68	99.44	95.24	85.74
Average (exclude #4402)	197.3	99.74	99.50	100	90.03

$n$  = the number of residues in the protein.

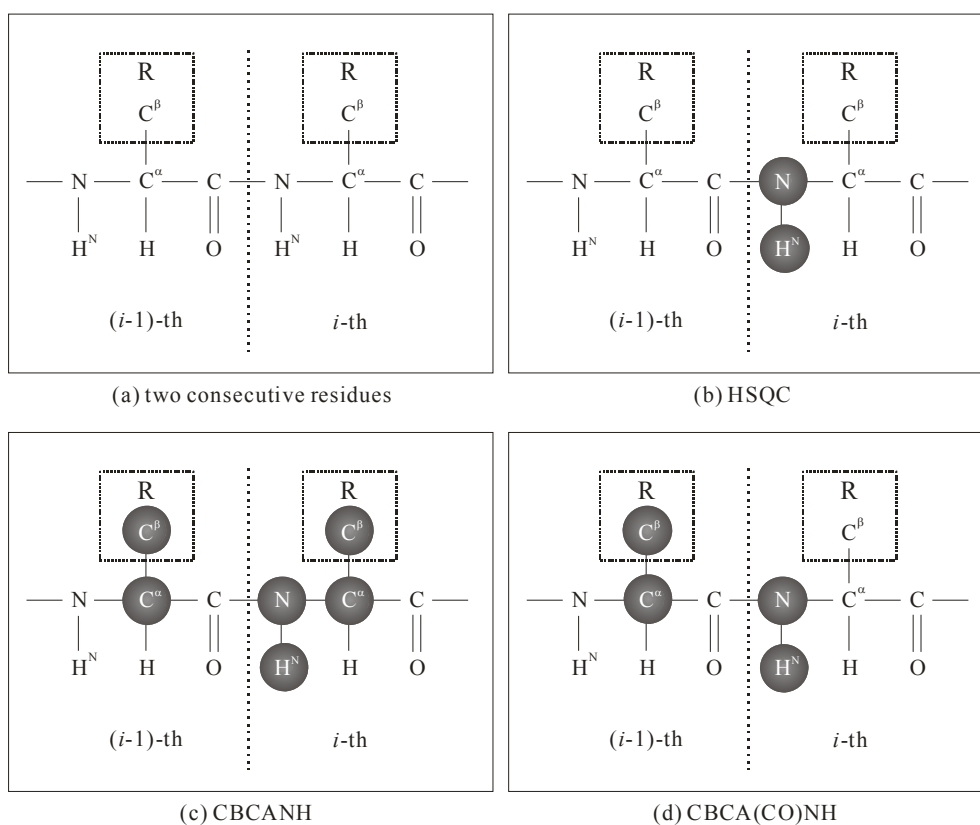


Figure 1. Different NMR experiments on two consecutive residues. The detected atoms with available chemical shifts are marked in black.

<i>SSGroup<sub>i</sub></i>		
116.50	8.25	
51.9	19.4	0
57.5	63.3	1

Figure 2. An example of an *SSGroup* headed by 116.5 (chemical shift of  $^{15}\text{N}$ ) and 8.25 (chemical shift of  $^1\text{H}^{\text{N}}$ ). The *SSGroup* contains two spin systems: an inter-spin system, indicated by 0 with  $\text{C}^{\alpha}$  chemical shift 51.9 and  $\text{C}^{\beta}$  chemical shift 19.4; and an intra-spin system, indicated by 1 with  $\text{C}^{\alpha}$  chemical shift 57.5 and  $\text{C}^{\beta}$  chemical shift 63.3.

6 (G):	5 8 16 22 25 66 67 68 69
7 (Y):	58 82 118 160
8 (D):	4 7 9 12 13 17 42 56 61 71 72
9 (D):	4 7 9 12 13 17 42 56 61 71 72

Figure 3: An example of candidate lists of four residues. The candidate list of the 7th residue (Y, Tyrosine) is 58, 82, 118, and 160, which means that the inter- and intra-spin systems in *SSGroups* 58, 82, 118, and 160 are within  $Range(G)$  and  $Range(Y)$ , respectively. The 7th residue will only be assigned one of *SSGroups* 58, 82, 118, and 160. Note that, if *SSGroups* 58, 82, 118, and 160 are already assigned to other residues, the 7th residue will be assigned  $-1$ , which means an empty *SSGroup*; thus, the chemical shift of  $C^{\alpha}_7$ ,  $C^{\beta}_7$  will be determined by the inter-spin system of the *SSGroup* of the 8th residue.

<i>SSGroup</i> <sub>1</sub> :
$AL_L(1)$ : 23, 11, 99
$AL_R(1)$ : 38, 47, 65, 41, 71
<i>SSGroup</i> <sub>2</sub> :
$AL_L(2)$ : 7, 20
$AL_R(2)$ : 35, 28, 93, 81
.....
<i>SSGroup</i> <sub>n</sub> :
$AL_L(n)$ : 12, 29, 17
$AL_R(n)$ : 22, 101, 43, 57, 68.

Figure 4: An example of adjacency lists with  $n$  *SSGroups*. (Note that the *SSGroup* numbers in each adjacency list are sorted according to the left-hand side of Inequality (1) and (2).) The inter-spin system of *SSGroup* 1 is connected with the intra-spin systems of *SSGroups* 23, 11, and 99; and the intra-spin system of *SSGroup* 1 is connected with the inter-spin systems of *SSGroups* 38, 47, 65, 41, and 71.

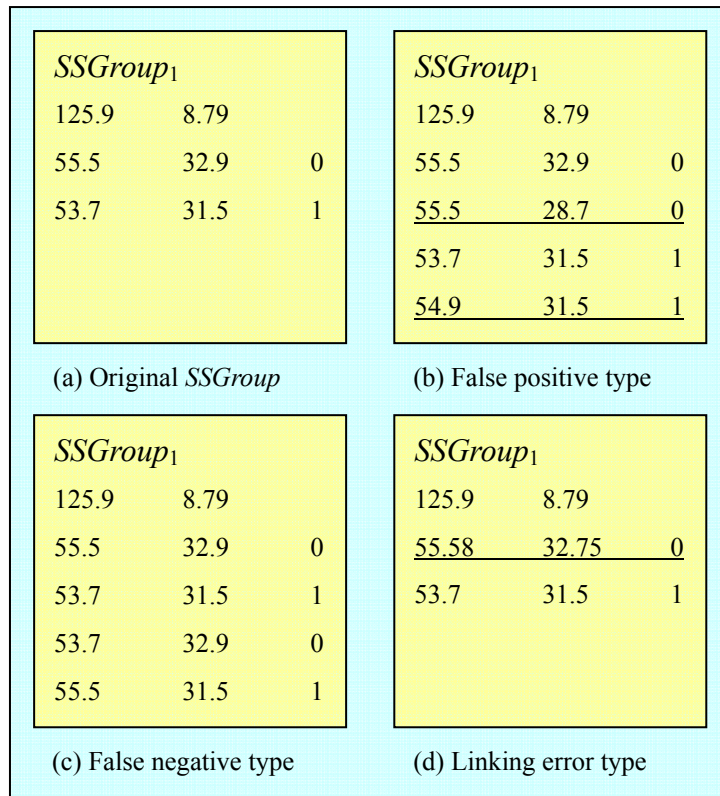


Figure 5: Examples of modified data with different error types.