

# GENETIC ALGORITHMS IN DIFFERENT AREAS OF SCIENCE

by

GORDON CHALMERS

(Under the Direction of Shelby H. Funk and James H. Prestegard)

## ABSTRACT

Genetic algorithms, and other evolutionary mathematical algorithms, are important tools for finding approximate solutions to complex problems. These are in the category of NP-Hard problems, which can not be solved by direct searches. In this dissertation genetic algorithms are used to find (optimal, perhaps) solutions in different areas of science. These problems are explained in the introduction and in the subsequent chapters. Detailed use of the genetic algorithms is presented in several chapters, from real-time system scheduling analysis in sensitivity analysis, to nuclear magnetic spectral assignment, to a classic NP-Hard problem, the maximally spanning backbone  $k$ -tree problem.

The use of genetic algorithms is demonstrated to produce better results than earlier works in these fields. For example, in real-time systems the processor utilization is higher, in NMR an automated assignment package is presented in both large and in small proteins, and in the last project, to the maximally spanning  $k$ -tree problem, more and better solutions are found.

The presentation in this dissertation doesn't cover all of the work completed during the course of Ph.D. completion. However, additional work is described in an appendix.

GENETIC ALGORITHMS IN DIFFERENT AREAS OF SCIENCE

by

GORDON CHALMERS

B.S., M.S., University of California at Los Angeles, 1991

Ph.D., University of California at Los Angeles, 1995

A Dissertation Submitted to the Graduate Faculty  
of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2019

© 2019

Gordon Chalmers

All Rights Reserved

GENETIC ALGORITHMS IN DIFFERENT AREAS OF SCIENCE

by

GORDON CHALMERS

Electronic Version Approved:

Major Professors: Shelby H. Funk  
James H. Prestegard

Committee: Liming Cai  
Dan Hall

Electronic Version Approved:

Ron Walcott  
Interim Dean of the Graduate School  
The University of Georgia  
December 2019

## ACKNOWLEDGMENTS

I would like to thank Professors Shelby H. Funk, James H. Prestegard, and Robert J. Woods for continued encouragement, financial support, and collaboration. I have had very good experiences with these people. I also thank some of my collaborators for working with me over the years: Qi Gao, Alexander Eletsky, Rob Williams, David Theiker, Laura Morris, John Glushka, Lachele Foley, and numerous students. I have been financially funded by several grants over the years, including those from the National Institute of General Medical Sciences (Protein Structure Initiative grant U54GM094597, biotechnology resource grant P41GM103390, and R01GM033225).

# TABLE OF CONTENTS

|   | Page |
|---|------|
| ACKNOWLEDGMENTS . . . . .   | iv   |
| LIST OF FIGURES . . . . .   | vii  |
| LIST OF TABLES . . . . .  | x    |
| CHAPTER   |      |
| 1 INTRODUCTION . . . . .  | 1    |
| 1.1 GENETIC SENSITIVITY ANALYSIS OF REAL-TIME SYSTEMS . . . . .                                   | 1    |
| 1.2 SPARSE LABELING OF PROTEINS . . . . .   | 2    |
| 1.3 MAXIMALLY SPANNING $k$ -TREE PROBLEM AND RNA STRUCTURS .                                      | 4    |
| 1.4 SUMMARY . . . . .   | 5    |
| 2 GENETIC ALGORITHMS IN REAL-TIME SYSTEMS: SENSITIVITY ANALYSIS<br>AND MODE TRANSITIONS . . . . . | 6    |
| 2.1 INTRODUCTION TO SENSITIVITY ANALYSIS . . . . .  | 6    |
| 2.2 MODEL AND DEFINITIONS . . . . .   | 8    |
| 2.3 A MOTIVATING EXAMPLE . . . . .  | 12   |
| 2.4 RELATED WORK . . . . .  | 13   |
| 2.5 SCALEGA ALGORITHM . . . . .   | 14   |
| 2.6 EXPERIMENTS . . . . .   | 20   |
| 2.7 CONCLUSION AND FUTURE WORK . . . . .  | 34   |
| 3 GENETIC ALGORITHMS USED IN NMR . . . . .  | 35   |
| 3.1 ASSIGN SLP MD . . . . .   | 35   |

|     |   |    |
|-----|---|----|
| 3.2 | INTRODUCTION . . . . .  | 37 |
| 3.3 | RESULTS . . . . .   | 39 |
| 3.4 | APPLICATION WITH rST6Gal1 . . . . .   | 45 |
| 3.5 | VALIDATION . . . . .  | 48 |
| 3.6 | COMPARISON OF MD VERSUS SINGLE FRAME . . . . .  | 49 |
| 3.7 | DISCUSSION . . . . .  | 51 |
| 3.8 | MATERIALS AND METHODS . . . . .   | 54 |
| 3.9 | ASSIGN SLP SUPPLEMENT . . . . .   | 62 |
| 4   | RNA STRUCTURE AND MAXIMALLY SPANNING $k$ -TREES . . . . .   | 81 |
| 4.1 | EXAMPLE $k$ -TREE AND CHROMOSOME REPRESENTATION . . . . .   | 82 |
| 4.2 | GENETIC ALGORITHM MODEL . . . . .   | 85 |
| 4.3 | RESULTS AND COMPARISONS WITH EARLIER WORK . . . . .   | 89 |
| 4.4 | CONCLUSIONS . . . . .   | 90 |
| 5   | CONCLUSIONS OF DISSERTATION . . . . .   | 91 |
| 6   | APPENDIX : ADDITIONAL SOFTWARE . . . . .  | 92 |
| 7   | APPENDIX : PAPERS TO WHICH I HAVE CONTRIBUTED IN THE COURSE OF<br>DISSERTATION RESEARCH . . . . . | 95 |
|     | BIBLIOGRAPHY . . . . .  | 97 |

## LIST OF FIGURES

|      |  |    |
|------|--|----|
| 1.1  | This shows the flow of a generic genetic algorithm. . . . .  | 2  |
| 2.1  | Example of a 6 task system with 3 modes. . . . .   | 10 |
| 2.2  | Schedule of an example task with mixed priorities of incoming and outgoing tasks. . . . .  | 11 |
| 2.3  | Comparison of additive GA and TbTSA. . . . .   | 21 |
| 2.4  | The period increases of several methods are shown, each pseudo-harmonic task set has utilization near 100 percent. . . . .   | 26 |
| 2.5  | The time of calculation of scaleSA versus BiniSA. Each psuedo-harmonic task set utilization near 100 percent. . . . .  | 27 |
| 2.6  | Comparison of scale GA and scaleSA. . . . .  | 28 |
| 2.7  | The period increases of several methods are shown, each general task set has utilization near 100 percent. . . . .   | 29 |
| 2.8  | Box plots of fitness from scaleSA, scaleGA, and BiniSA in experiments 1. Utilization is 1.0 . Most are the first quartile, and there are many outliers due to the variation in the periods. There are outliers due to the large number of parameters and population of 1000. . . . . | 30 |
| 2.9  | Box plots of fitness from scaleSA and scaleGA in experiments set 2. Utilization is 1.5 . . . . .   | 31 |
| 2.10 | Histograms of fitnesses from scaleSA and scaleGA in experiments set 2. . . .   | 32 |



|     |  |    |
|-----|--|----|
| 3.1 | Heatmap showing ST6Gal1 phenylalanine assignments using simulated data. The numbers in each element are the fraction of times a crosspeak is assigned to a particular residue. Higher numbers are color-coded a darker blue and are taken to indicate a more confident assignment. All correct assignments should be on the diagonal in this case since the crosspeak order is was chosen to order with the residue order. . . . . | 44 |
| 3.2 | Heatmap showing ST6Gal1 phenylalanine assignments using experimental data. The 6 most confident assignments (fraction $\geq 0.5$ ) are shown in darker shades of blue. These are now scattered throughout since we don't know apriori how to order the crosspeaks. . . . .   | 46 |
| 3.3 | 800 MHz 2D $[^{15}\text{N}, ^1\text{H}]$ HSQC spectra of WT rST6Gal1 and single-point mutants F208Y, F356Y and F357Y. One crosspeak disappears in each of the mutant spectra (red circles) identifying the crosspeak belonging to the mutated site. . . . .  | 47 |
| 3.4 | Heatmap showing rST6Gal1 phenylalanine assignments using mutational constraints (F208 to 10, F356 to 14 and F357 to 7). . . . .  | 49 |
| 3.5 | Heatmap showing ST6Gal1 phenylalanine assignments using a single frame with a 1.10 Å RMSD of backbone atoms from those of the crystal structure, 4MPS. . . . .   | 50 |
| 3.6 | Model of rST6Gal1 with the donor analog, carboxy-TEMPO-CMP docked into the active site. Distances shown are those between the nitroxide oxygen of the TEMPO group and the amide protons of F240 (14.4), F357 (12.9), F208 (11.4) and F356 (12.2), respectively. . . . .  | 51 |
| 3.7 | Heatmaps comparing predicted and experimental values of each type of measurement (chemical shift, NOEs and RDCs) and total score contribution. Each number on both X and Y axes represent one labeled residue. The amino acid type is assumed known for the two sets of crosspeaks. . . . .  | 64 |

|      |   |    |
|------|---|----|
| 3.8  | Histogram showing the frequency with which each crosspeak (measurement) is assigned to each site (residue) for the protein with NMR structure 3CWI.   | 69 |
| 3.9  | Methyl-TROSY spectrum of $^{13}\text{C}$ - $^1\text{H}$ -alanine labeled perdeuterated HtpG. Superimposed on are blue circles representing chemical shifts of crosspeaks of the separately expressed N-terminal domain. The dotted ellipse centered at the single-domain shifts of A43 (radii 1.2 and 0.12 ppm) encloses 4 crosspeaks judged to be possible A43 crosspeaks in full length HtpG. . . . . | 73 |
| 3.10 | Crosspeak shifts on adding AMPPNP to apo-HtpG. Arrows show shifts, circles show peaks disappearing. . . . .   | 75 |
| 3.11 | Superimposed ribbon structures of the N-terminal (A) and middle plus C-terminal (B) domains of apo (green) and AMPNP (blue) forms of HtpG. $^{13}\text{C}$ - $^1\text{H}$ -labeled alanines with resonances that differ in chemical shift are shown in red. . . . .   | 76 |
| 3.12 | . Models of the apo-HtpG dimer. A) Domain orientations were obtained by RDC analysis. B) A model based on SAXS data obtained in solution at high pH (Krukenberg et al. 2008). NT, MD and CT domains are colored blue, cyan and green respectively. . . . .  | 79 |

## LIST OF TABLES

|     |   |    |
|-----|---|----|
| 2.1 | Example task set and comparison. . . . .  | 12 |
| 2.2 | Worst case response times of example task set. . . . .  | 13 |
| 2.3 | Parameters of the additive and scaling genetic algorithm. . . . .   | 25 |
| 2.4 | Parameters of the genetic algorithm in the mode transition problem. . . . .   | 29 |
| 2.5 | Utilization 100. Mean and Median of scaleSA, scaleGA, BiniSA, and TbTSA.<br>About 10 percent of TbTSA results are not included due to no solution of<br>period increases. . . . . | 33 |
| 2.6 | Utilization 150. Mean and Median of scaleSA, scaleGA. . . . .   | 34 |
| 3.1 | Non-NOE data used in rST6Gal1 <sup>15</sup> N-phenylalanine assignments. . . . .  | 59 |
| 3.2 | Non-NOE data used in rST6Gal1 <sup>15</sup> N-phenylalanine assignments. . . . .  | 59 |
| 3.3 | NOE peak list data used in rST6Gal1 <sup>15</sup> N-phenylalanine assignments. <sup>a</sup> . . . .   | 60 |
| 3.4 | NOE peak list data used in rST6Gal1 <sup>15</sup> N-phenylalanine assignments. <sup>a</sup> . . . .   | 61 |
| 3.5 | Top ranked solution for assignments of 3CWI-2K5P. . . . .   | 65 |
| 3.6 | Next top ranked solution for assignments of 3CWI-2K5P. . . . .  | 65 |
| 3.7 | Assignment summary of four test protein cases. . . . .  | 66 |
| 3.8 | Crosspeak Assignments for the NTD of full-length apo-HtpG. . . . .  | 74 |
| 4.1 | First cliques of a 4-tree from the genetic algorithm. The first 4 columns are<br>the nodes in the 4-tree, and total clique weights are in the last column. . . .                  | 83 |
| 4.2 | First cliques of a 4-tree from the dynamic program. The nodes of the cliques<br>are in the first four columns. Total clique weights are in the last column. . .                   | 83 |
| 4.3 | First 10 cliques of a 4-tree. Weights are in the last column. . . . .   | 84 |
| 4.4 | Chromosome of example $k$ -tree in Table 4.3 . . . . .  | 84 |
| 4.5 | Values in the chromosome and complexity. . . . .  | 86 |

|     |  |    |
|-----|--|----|
| 4.6 | Parameters used. . . . .   | 89 |
| 4.7 | Best individuals : dynamic program and genetic algorithm . . . . . | 90 |
| 4.8 | Best individuals : dynamic program and genetic algorithm . . . . . | 90 |

## CHAPTER 1

### INTRODUCTION

Genetic algorithms are used in three areas of science and their utility in solving NP-hard problems in these areas is demonstrated. The problems are real-time systems to find schedule changes, computational chemistry in NMR in spectral assignment of experimental measurements, and in bio-informatics to find better RNA structures.

Genetic algorithms use a population of chromosomes and evolve them iteratively using a chromosome crossover and mutation at each iteration, an objective function, and a constraint function. The model is based on the evolution of life. As the population improves it produces good individuals which achieve their ideal objective. The flow of a genetic algorithm is described in 1.1. Mathematically, this model will produce ideal solutions to difficult problems through their objective function, with few iterations. Exhaustive searches for these problems are impossible due to the number of calculations. This objective function models is used in a minimization or maximization procedure of the individual's fitness. The constraint function is used for limiting the different possible solutions to the ideal ones. Violations of the constraint function are however important in the search of the entire space of possible solutions, which could be disconnected. These genetic algorithms have found very good results in many NP-Hard problems. In these three types of genetic algorithm problems the search spaces are quite large, being on the order of  $10^{14}$  to  $10^{400}$ .

#### 1.1 GENETIC SENSITIVITY ANALYSIS OF REAL-TIME SYSTEMS

The first problem, sensitivity analysis and mode transitions, which are analyzed with genetic algorithms, is in Chapter 2. The goal of sensitivity analysis is to make an unschedulable task

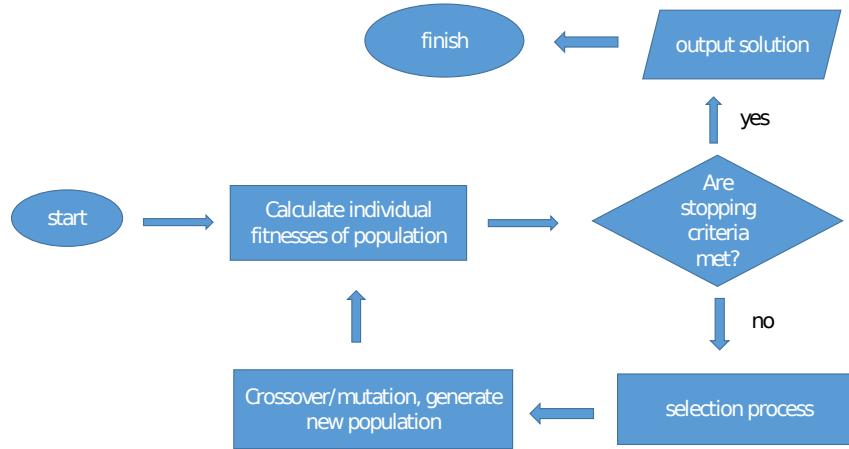


Figure 1.1: This shows the flow of a generic genetic algorithm.

set schedulable, or a mode transition task set schedulable. This is accomplished by changing task set parameters. The definitions of the task model and mode transitions are in Chapter 2. As explained and shown in this dissertation chapter, using a genetic algorithm has the advantage over earlier works in that the task set is treated as whole, by changing parameters in a global fashion and not task by task.

This problem is perhaps the most non-linear of the three problems in that it has to solve the time demand analysis equations in its objective function. This is both a non-linear and discontinuous function.

## 1.2 SPARSE LABELING OF PROTEINS

In NMR, one of the primary goals is structural determination, finding populations of conformations, ligand binding site determination, and determining the ligand epitope. Additionally,

more information can be found from the simulation of the NMR experimental information produced from a molecular model either using a crystal structure or a molecular dynamics trajectory study.

Software packages are presented and demonstrated in NMR sparse labeling in Chapter 3. Sparse labeling and sparse assignment means that a smaller set of tagged residues is used instead of a full set of residues in the protein. This is useful as it is less expensive for the construction of the experiment and has less computations in identifying the experimentally measured spectral intensities in the protein model and related proteins. Assignment, i.e. identifying which intensity is from which residue, is important in extracting useful information about the protein. An assignment, or partial assignment with reliability, can be used with different experiments to find the physical features of the protein mentioned in the previous paragraph. The disadvantage of an initial attempt, the AssignSLP package, is that a  $1/r^6$  approximation is used to find the predicted noe's; this approximation is very bad for large proteins. However, as demonstrations of the genetic algorithm software AssignSLP, we present the assignment of five small proteins and a larger protein HtpG. An improvement to the AssignSLP software is presented, which uses an Amber molecular dynamics trajectory to make better predictions from the molecular model including conformational sampling; the advantage of using a trajectory in AssignSLPMD (molecular dynamics) is that the genetic algorithm can make a reliable (partial assignment of most of the residues) assignment in much larger proteins, and for smaller proteins the assignment should be more trustworthy with this software. This software is used and demonstrated in providing a reliable assignment of a larger protein rST6Gal1.

NOE measurements, RDC measurements, and chemical shifts of the protons of different residue types are used; this is explained in the chapter. Software has been made and used in calculating spectral intensities (i.e strips) from Amber trajectories. This package MD2NOEProtein is not presented in this dissertation, as it does not use a genetic algorithm. The package AssignSLPMD uses a genetic algorithm with the inputs of predicted NOE's,

rdc's, and chemical shifts from the MD2NOEProtein package. There is also a statistical part in AssignSLPMD to give the reliability of the individual assignments of the measurements in the experimental spectrum.

The software has evolved to the point that all predicted observables use Amber trajectories, and molecular modeling. This is also a lengthy calculation in the context of the well known molecular dynamics Amber software. It can be used to simulate the motion of proteins over a microsecond range. The use of trajectory information from the modeling is important for an proper assignment on application of the genetic algorithm in AssignSLPMD to larger proteins.

The MD2NOEProtein and earlier MD2NOE packages do not use a genetic algorithm; these packages are referenced in the list of additional software in Appendix 6.

The characteristics of the objective function and constraints are different in these assignment genetic algorithms. The genetic algorithm is similar to the traveling salesman problem in that same permutation and crossover functions are used, but the NOE and RDC part of the objective function is very non-linear.

### 1.3 MAXIMALLY SPANNING $k$ -TREE PROBLEM AND RNA STRUCTURES

The third problem approximates the maximally spanning  $k$ -tree backbone problem. This is described in Chapter 4. The genetic algorithm is used to generalize and improve known results of approximate solutions from earlier studies using dynamic programming. The solutions are useful in a variety of contexts, including RNA structure determination and in a variety of other applications. An advantage in using the genetic algorithm presented in this chapter is that the genetic algorithm provides more optimal solutions than the unique solution from a dynamic program.

This problem uses a genetic algorithm which has an objective function that does not evolve the population infinitesimally small steps but rather in large discrete jumps. This is



unusual for a genetic algorithm, but the nature of the problem requires it as the chromosomes evolve in a global sense, and not gene by gene. This difference is due to the structure of the individual chromosomes in the population and how the  $k$ -trees are calculated.

## 1.4 SUMMARY

These chapters demonstrate how genetic algorithms can be used in improving the approximate solutions of NP-hard problems in these three areas of science.

Interestingly, the different problems have different features. These search spaces can be quite large, depending on the inputs to the problem in the genetic sensitivity problem, which can be of the order of  $10^{400}$  to  $10^{1000}$ . The second problem in the sparse labeling is also non-linear. In the third problem, maximally spanning  $k$ -tree, the evolution of the population is jumps and not by infinitesimal steps. This is unusual for a genetic algorithm. The unifying theme of these three works is the use of different genetic algorithms.

## CHAPTER 2

### GENETIC ALGORITHMS IN REAL-TIME SYSTEMS: SENSITIVITY ANALYSIS AND MODE TRANSITIONS

Key words: real-time scheduling, sensitivity analysis, mode transitions, fixed priority scheduling, genetic algorithms.<sup>1</sup>

#### 2.1 INTRODUCTION TO SENSITIVITY ANALYSIS

In real-time systems, temporal correctness is as important as logical correctness. In such systems, jobs have designated deadlines – if a job does not complete execution at or before its deadline, it is considered a system failure. These systems are used in applications where violating timing constraints could lead to catastrophic outcomes, such as airplane autopilot systems or anti-lock brake systems. In these systems, analysis must be performed before the system is implemented to be sure that jobs will meet their deadlines. The question arises: What should we do if the analysis finds that jobs will miss their deadlines? This is where sensitivity analysis is applied.

In sensitivity analysis, we examine how to change the parameters of a system that will miss deadlines to create a new system that will meet all deadlines. Ideally, we would like to change the parameters a little as possible, so that the final system has parameters as close to the original system as possible while meeting all deadlines.

In real-time systems, jobs are commonly executed repeatedly at periodic intervals. These repeated jobs, called *tasks*, are typically how we describe a real-time system. Utilization

---

<sup>1</sup>The work in this chapter is largely based on the papers, G. Chalmers, S.H. Funk, Genetic Algorithms in Real-Time Systems, submitted to Genetic Programming and Evolvable Machines, 2019; and G. Chalmers, S.H. Funk, Adjusting Real-Time Mode Transitions via Genetic Algorithms, 2017 16th IEEE International Conference on Machine Learning and Applications.

is defined as the resource usage over extended periods. One common scheduling paradigm, called fixed-priority scheduling, assigns priorities to tasks – all jobs generated by a task are executed at that task’s priority level. This work applies genetic algorithms to the problem of sensitivity analysis of fixed-priority task sets executing on a single processor. Unlike previous results on sensitivity analysis, which determine schedulability by changing task set parameters, such as execution times and periods, one at a time [3, 4, 10, 11, 13, 28, 59, 70, 88, 90–92, 92, 93], this work considers the parameters of all tasks as a whole by considering all the tasks in the task set simultaneously in a genetic algorithm. Earlier studies were not concerned with minimizing the changes in parameters while performing sensitivity analysis. As a result, the modified tasks can be significantly different than the original ones. We believe that creating new task sets with parameters similar to their original values could be important for a less flexible real-time system.

The earliest studies of sensitivity analysis changed the execution times to achieve schedulability, e.g. [13, 90]. Our work changes the task periods. Changing task deadlines is the same as changing execution times for real-valued periods=deadlines [13]. Genetic algorithms have been used in real-time scheduling problems in the past [1, 5, 44, 56, 61, 65, 71, 72, 94]. We compare our genetic algorithm work with these following types of sensitivity analysis.

- *Scaling SA* [88] scales all the periods or execution times by the same factor, until the task set is schedulable. Execution times are real-valued and periods equal deadlines. As the periods increase, the utilization decreases. This approach has a very quick runtime, especially if the scale changes are scanned with a larger graining, i.e. step.

- *TbTSA* [90] (task-by-task sensitivity analysis) examines the tasks from the highest priority to the lowest and increases the tasks’ periods one at a time until schedulability is achieved. This increases the periods of lower priority tasks disproportionately. This approach runs more slowly than scale SA.

- *Bini SA* [13] uses modified TDA equations and the idea of the “feasibility region” to find an efficient algorithm to schedule task sets that are not schedulable. This work also changes

task periods task by task. The algorithm uses a set of slack modified TDA equations and these equations are solved task-by-task. These equations are tested at “feasibility points” in order to find the task set changes. The number of these points is pseudo-polynomial in the task set parameters, but the TDA equations are more complicated.<sup>2</sup>

Although these approaches are different, none have had the emphasis of minimizing the aggregate changes of task set parameters and by considering the entire task set as a whole during sensitivity analysis. In general, our approach will take longer to revise task set parameters, but the revised task set will be much closer to the original one.

Furthermore, our approach is more flexible than the other approaches. We will demonstrate that the genetic algorithm approach to sensitivity analysis can be used to adjust periods or execution times with small changes to the task set. We can also adjust either additively by increasing or by scaling multiplicatively, as we will discuss in Section 2.4. An example of an important parameter which is used is the task period.

Section 2.2 describes the model and definitions. Section 2.3 gives a an example of the different sensitivity algorithms. Section 2.4 gives a discussion of genetic algorithms and its use in sensitivity analysis. Section 2.5 defines the genetic algorithm for both standard systems and multi-mode systems. Experiments are given and used to compare with earlier works in Section 2.6 for both the sensitivity of scheduling and for mode transitions. Finally, Section 2.7 concludes the sensitivity analysis and explores avenues of additional research.

## 2.2 MODEL AND DEFINITIONS

We first present the model of standard real-time task sets in Section 2.2.1. We show how to extend this to multi-mode systems in Section 2.2.2. Finally, we discuss changes in the system, such as different task models.

---

<sup>2</sup>Note: The Bini algorithm works only if the unschedulable task set has a utilization strictly less than 1.0.

### 2.2.1 STANDARD PERIODIC TASKS

We consider a task set  $\tau = \{\tau_1, \tau_2, \dots, \tau_N\}$  comprised of  $N$  periodic, independent, pre-emptible tasks. The parameters of each task  $\tau_i$  is:

- $T_i$ : **period of task**  $\tau_i$
- $e_i$ : **execution time of task**  $\tau_i$

The priorities of tasks are set in order of increasing periods [13]. These tasks are characterized by execution times  $e_i$  and periods  $T_i$ . These are ordered pairs  $\{e_i, T_i\}$ . The task  $\tau_i$  releases a sequence of jobs  $\tau_{i,0}, \tau_{i,1}, \dots$ , with  $\tau_{i,k}$  having release time  $kT_i$  and deadline  $(k+1)T_i$ . Each job  $\tau_{i,j}$  must be allowed to execute for  $e_i$  time units between its release time and deadline.

Tasks are assumed to be independent and scheduled using the rate monotonic (RM) fixed priority scheduling algorithm [54]. A task  $\tau_i$ 's worst-case response time ( $R_i$ ) is the maximum amount of time between the release time of a job  $\tau_{i,k}$  and its completion time. By the critical instant theorem [54], the worst-case response time occurs when all the tasks are simultaneously activated.

Each task  $\tau_i$  has an associated utilization  $u_i$ , which measures the fraction of time  $\tau_i$  executes over long intervals and is the ratio of the execution time to the period.

$$u_i = e_i/T_i. \quad (2.1)$$

The task set  $\tau$ 's utilization is the total utilization of all the tasks in  $\tau$

$$U = \sum_{i=1}^N u_i. \quad (2.2)$$

Offsets are a delay of the release of the jobs. The inclusion of offsets and also deadlines less than periods is a minor modification of the genetic algorithm.

|        | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 |
|--------|--------|--------|--------|--------|--------|--------|
| Mode 1 | X      | X      |        | X      |        |        |
| Mode 2 |        | X      | X      |        | X      | X      |
| Mode 3 |        | X      | X      | X      | X      |        |


  
mode independent task

Figure 2.1: Example of a 6 task system with 3 modes.

### 2.2.2 MODE TRANSITION MODEL

The mode transition problem requires more detail in the description of the model. There are different modes, denoted by  $\{M_1, M_2, \dots, M_{\max}\}$ . There are a total set of tasks and each mode  $M$  uses a subset of the total set of tasks. This is for the model we use in which the incoming tasks release their jobs at the mode change request instant. Not all tasks are in the different modes; those in all the modes are mode independent.

During the mode transition there are two sets of modes sharing the resource. The incoming mode has the incoming tasks and the outgoing mode has the outgoing tasks.

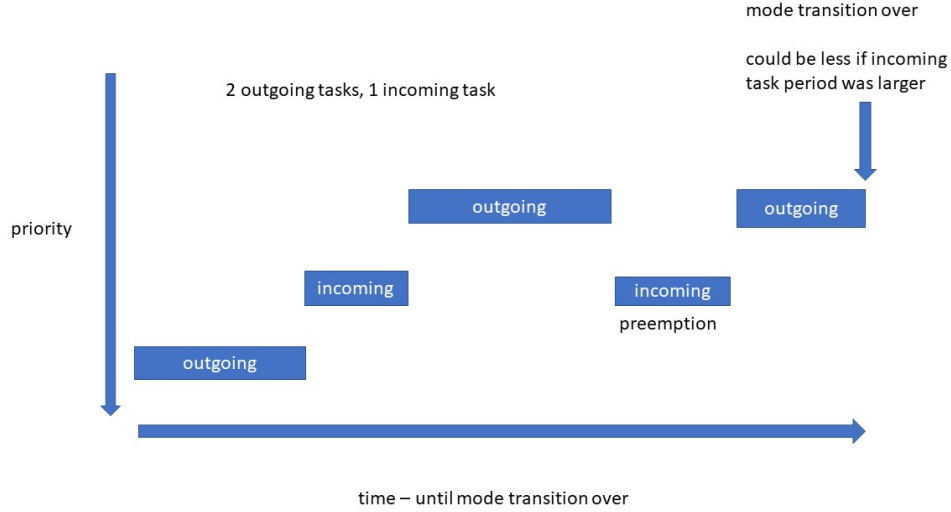


Figure 2.2: Schedule of an example task with mixed priorities of incoming and outgoing tasks.

If there are mixed priorities, then the mode transition time depends on the periods of the incoming tasks; this time can be lessened by increasing the periods of the incoming tasks, which in turn reduces the interference with the outgoing tasks. Figure 2.1 illustrates a simple mode transition and how the interference of the incoming tasks has an effect on the mode transition time. By increasing the periods of the incoming tasks, the mode transition time is decreased.

Figure 2.2 shows the effect of changing the periods of the incoming tasks on the mode transition time. If the incoming tasks periods are increased then there is less interference with the outgoing tasks and the mode transition time will decrease.

There are different types of mode transitions. The first is defined by outgoing tasks executing once after the mode change request. After the outgoing tasks completely execute

the mode transition is over. The second type of mode transition is if the outgoing tasks can execute a finite set of times.

### 2.3 A MOTIVATING EXAMPLE

Consider a system of four tasks  $\tau_1, \tau_2, \tau_3$ , and  $\tau_4$  with periods ( $T_i$ ) and execution times ( $e_i$ ) shown in the first columns of Table 2.1. The utilization of the example task set is 2.26, which is clearly infeasible.

Table 2.1: Example task set and comparison.

|          | Period<br>$T_i$ | Execution time<br>$e_i$ | $T'_i$   |       |     | $\delta_i$  |       |    |
|----------|-----------------|-------------------------|----------|-------|-----|-------------|-------|----|
|          |                 |                         | scaleSA  | TbTSA | GA  | scaleSA     | TbTSA | GA |
| $\tau_1$ | 4               | 1                       | 11       | 4     | 10  | 7           | 0     | 6  |
| $\tau_2$ | 10              | 6                       | 28       | 10    | 29  | 18          | 0     | 19 |
| $\tau_3$ | 11              | 10                      | 31       | 70    | 29  | 20          | 59    | 18 |
| $\tau_4$ | 20              | 10                      | 56       | 1400  | 29  | 36          | 1380  | 9  |
|          |                 |                         | $U'=100$ | 80    | 100 | $\Delta=81$ | 1439  | 52 |

The remaining columns of Table 2.1 show the revised periods ( $T'_i$ ) and the increase of periods ( $\delta_i = T_i - T'_i$ ) derived using scaleSA, TbTSA, and the genetic algorithm. The revised utilization values of the task set found by scaleSA is 80%. Both TbTSA and the genetic algorithm have final utilizations of 100%. On the other hand, TbTSA increases the task periods by 1439, whereas the genetic algorithm only increases the periods by 52. This is a factor of approximately 30. We observe that scaleSA makes smaller changes to the periods than TbTSA, but it also creates tasks with smaller utilization values. The genetic algorithm finds a solution with smaller parameter changes than the other approaches and is also able to create a task set with a high utilization value.

Based on these periods, we can examine tasks' worst case response times. Table 2.2 illustrates the response times of the tasks created by the 3 sensitivity analysis approaches. We see, that scaleSA and the genetic algorithm have the same response times and that TbTSA has large response times for the lower priority tasks. This is because the task-by-task



approach tends to allocate most of the processing time to the high priority tasks, resulting in large response times for the low priority tasks. In fact, the TbTSA type of sensitivity analysis can even fail. If the workload at some priority level is 100% then there is no time available for the lower priority tasks to execute, regardless of how much the tasks' periods are increased. This is more likely to occur for large task sets.

Table 2.2: Worst case response times of example task set.

| WCRT  | scaleSA | TbTSA | GA |
|-------|---------|-------|----|
| $R_1$ | 1       | 1     | 1  |
| $R_2$ | 7       | 8     | 7  |
| $R_3$ | 18      | 70    | 18 |
| $R_4$ | 29      | 1400  | 29 |

The genetic algorithm searches for a global minimum of period increases. Therefore, it will always be able to find a modified task set that will meet all deadlines. ScaleSA will also be able to find feasible task sets, but it applies the same scaling factor to all tasks. As we have seen in the example above, this can result in unnecessarily low utilization values. The genetic algorithm has the advantage of considering all tasks at once, but applying different increases to different tasks. This is more flexible than either of the other two approaches. It contains the advantages of both approaches without having the disadvantages. As we will see in Section 2.6.1, this advantage comes at a cost – the runtime of the genetic algorithm tends to be longer than that of the other approaches, but not unduly so.

## 2.4 RELATED WORK

The genetic algorithm starts with a population of possible solutions and repeatedly improves the population until finding a solution to the problem. The GA is formulated as an objective function along with some constraints. The optimal solution is a result that satisfies the constraints and minimizes the objective function.

Improving the population usually requires two operations, crossover and mutation. The genetic algorithm finds new solutions until a specified stopping criterion is met. The flow of a

genetic algorithm is described in Chapter 1. First the initial population is created. Particular initializations of the population are useful to improve the performance of the search. Then the algorithm iterates until the stopping criteria is met. In this way, the desired solution is found within a given threshold of accuracy. The specific behavior of the genetic algorithm is also affected by adjusting various parameters such as initialization of the population, and the crossover and mutation rates, as mentioned above.

Genetic algorithms have influenced many areas of science which involve NP-hard problems including real-time systems. With regards to sensitivity analysis, the use of genetic algorithms have indirectly influenced this area of research through different types of scheduling, such as task mapping on network-on-chips (NoCs) or multiprocessors [65,71], job shop scheduling [18, 19], and project management [17]. These works are concerned with job allocation, i.e. which job goes to which resource, which is computationally difficult. The TDA equations are used as a constraint in finding the global minimum of the task set parameter changing. This makes our work different from, for example, task mapping of a task to a resource in a multi-processor system, which use static job allocations to resources.

## 2.5 SCALEGA ALGORITHM

The tools of evolutionary programming, genetic programming (GA), etc., are applicable to scheduling problems, and the approach using the solution to the TDA equations in as a constraint enables the GA to find solutions of task parameter changes of unschedulable task sets. These changes are more optimal than earlier sensitivity analysis', in the sense of less task changes of task set parameters to make an unschedulable task set schedulable. Previous approaches used task-by-task modifications within the sensitivity analysis algorithm, and the genetic algorithm approach uses a global search of task parameter change values for a minimal task set change. This approach does not guarantee a minimal set of task parameter changes, due to the fact that the genetic algorithm could give local minima solutions. However, it is better at total change in task parameter changes than prior sensitivity analysis

techniques if the goal is to minimize task set parameter changes, as experiments show. In general this problem of finding task set parameter changes to make an unschedulable task set schedulable is more complex than the scheduling problem (pseudo-polynomial) because task set parameter changes have to be examined, especially if the changes of task periods are taken into account, each independently. This makes the genetic algorithm approach well-suited for finding a minimum of task set parameters, without this complexity.

Each individual of the population is a set of period increases and worst case response times. Each of these parameters are genes of the individual; for a task set with  $N$  tasks, there are  $2N$  genes – namely, the revised periods  $T'_i$  and the worst-case response times  $R_i$ , for  $1 \leq i \leq N$ . There is a fitness given to each of the individuals which involves both the objective function and the constraints.

The population is improved until the sum of the period increases is minimized. The user must specify when the algorithm will stop by providing a desired accuracy of the solution. The comparison is done between the different known algorithms, at different utilizations, and with different types of task sets (general and pseudo-harmonic). Also, periods are increased by adding or by scaling. Genetic algorithms are a well-known method for optimizing complex problems using evolutionary computation.

The additive genetic algorithm uses an objective function of the sum of additive increases to the periods of the task set. Because our model assumes all of the variables are integers  $\delta_i \in (1 \leq i \leq N)$ . The additive genetic algorithm increases the periods  $T_i$  by an increment  $\delta_i$  is  $T'_i = T_i + \delta_i$ , and  $\delta_i \geq 0$ , for  $1 \leq i \leq N$ . Finally, the additive genetic algorithm minimizes the sum of the period increases, i.e. minimize  $\sum_{i=1}^N \delta_i$ . We call this function the *fitness function*, which is defined in the model section.

The multiplicative genetic algorithm is similar to the additive genetic algorithm. Instead of incrementing the periods, though, we scale the periods up by a factor  $\lambda_i$ , which can be a real-value no smaller than 1.  $T'_i = \lambda_i \cdot T_i$ , and  $\lambda_i \geq 1$ , for  $1 \leq i \leq N$ . In this case the fitness

function is the sum of the scaling factors, i.e., minimize  $\sum_{i=1}^N \lambda_i$ . We examined several fitness functions, such as a product of scaling factors. The sum of scale factors was the best form.

### 2.5.1 SENSITIVITY ANALYSIS OF UNSCHEDULABLE TASK SETS

We use 2 approaches for sensitivity analysis – additive and multiplicative. For each task  $\tau_i$ , the genetic algorithms for these approaches use the parameters  $\delta_i$  and  $\lambda_i$ , respectively.

- $\delta_i$ : **additive GA: increase of period**  $T_i$  ( $\delta_i = T_i - T'_i$ )
- $\lambda_i$ : **scale GA: scale factor of period** ( $\lambda_i = T'_i/T_i$ )

Our approach to modifying unschedulable task sets to make them schedulable is based on time demand analysis (TDA) [52], which provides the *level- $i$  time-demand function*  $w_i(t)$ :

$$w_i(t) = e_i + \sum_{j < i} \left\lceil \frac{t}{T_j} \right\rceil e_j. \quad (2.3)$$

Task  $\tau_i$ 's worst-case response time is the minimum value  $t$  such that  $t = w_i(t)$ . In this section, we show how to use a genetic algorithm to find new periods  $T'_1, T'_2, \dots, T'_n$  and worst-case response times  $R_1, R_2, \dots, R_n$ , which are the smallest values of the level  $k$  time demand function  $w_i^k$  such  $w_i^k = w_i^{k+1}$ . so that for each task  $\tau_i$ , we have a solution to the fixed-point equation,

$$R_i = e_i + \sum_{j < i} \left\lceil \frac{R_i}{T'_j} \right\rceil e_j \quad (2.4)$$

and the tasks' periods are changed as little as possible.

Because Equation 2.4 is a discontinuous non-linear equation over the variables  $T'_i$  and  $R_i$  ( $1 \leq i \leq N$ ), we must use complex optimization strategies, such as a genetic algorithm to find an optimal (or close to optimal solution).

We now present our genetic algorithm for finding period increases of unschedulable task sets. The first constraint of our genetic algorithm involves the level- $i$  demand. Specifically, we need to be sure that for each task  $\tau_i$ , the level- $i$  demand over an interval of length  $w_i$

cannot be longer than the interval length:

$$R_i \leq e_i + \sum_{j < i} \left\lceil \frac{R_i}{T'_j} \right\rceil e_j. \quad (2.5)$$

The next constraint is used to ensure that all tasks will meet their deadlines. Specifically, the worst case response time  $R_i$  cannot be larger than the period,

$$R_i \leq T'_i. \quad (2.6)$$

for all  $i$ .

Finally, we ensure that worst-case response times are monotonically increasing. Thus, we add the constraint for  $1 \leq i < N$

$$R_{i+1} \geq R_i, \text{ for } 1 \leq i < N. \quad (2.7)$$

Technically, this constraint is not necessary, as lower priority tasks always have larger worst-case response times than higher priority tasks. We add this constraint because it reduces the size of the genetic algorithm's search space, which has the effect of allowing the algorithm to find better solutions more quickly.

### 2.5.2 MODE TRANSITIONS AND GENETIC ALGORITHMS

In this work, the first type of mode transition is examined in which the outgoing tasks execute once; it is straightforward to generalize the formalism to the second type.

In the case of different modes, the genetic algorithm sensitivity formulation has to have the added detail of outgoing and incoming tasks. The constraints and objective function are slightly different. Define the outgoing tasks by  $O$  and the incoming tasks by  $I$ .  $\{I_i\}$  is the set of incoming tasks with higher priority than task  $\tau_i$ , and  $O_i$  is the set of outgoing tasks with higher priority.

The priorities of the incoming and outgoing tasks are, in general, different. This is important, as interferences from the incoming tasks could delay the outgoing tasks' worst-case response times and thus increase the mode transition time. The mode transition time is the

worst case response time  $R_l$  of the lowest priority outgoing task  $\tau_l$ , which defines the outgoing task set  $O$  completion. By the TDA equation,  $R_l$  is the smallest value  $w_l^{(k)}$  such that,

$$w_l^{(k+1)} = \sum_{\tau_j \in O_i} e_j + \sum_{\tau_i \in I_l} \left\lceil \frac{w_l^{(k)}}{T_i} \right\rceil e_i, \quad (2.8)$$

where all the outgoing task execution times are included for the two modes  $m$  and  $n$ , and  $I_i$  is the set of incoming tasks with priority higher than  $\tau_i$ .

This equation finds the worst case response time of the lowest priority outgoing task. If all the incoming tasks have lower priorities than the outgoing tasks, then the mode transition time is truly minimal since there is no interference from the incoming tasks. In this case the mode transition completion is just the sum of the execution times of the outgoing tasks.

The genetic algorithm tries to find the minimal change in the task parameters such that the mode transition is over before a specified mode transition deadline. The periods of the incoming tasks are scaled. The algorithm is compared with earlier approaches.

### 2.5.3 OTHER CONSIDERATIONS

We note that this genetic algorithm approach is very flexible – changing assumptions in the model can be easily reflected by simple changes in the constraints or the fitness functions presented above. Below we present several modifications.

#### BLOCKING GROUPS OF TASKS

The scaling genetic algorithm could scale task periods in blocks of tasks. In this approach, the scaling factor is the same for each block of tasks. For example, a set of 128 tasks could be divided into 16 blocks. In this way, each set of 8 tasks uses the same scaling factor, in a fixed priority manner. In this modification, the search space can be reduced by an amount due to the reduction in the number of genes in the chromosomes.

## FITNESS FUNCTION AND CONSTRAINT MODIFICATION

We might want to limit the changes on some of the task parameter changes. We could set hard limits on these changes by setting bounds on  $\delta_i$  or  $\lambda_i$  (depending on which approach we are considering). For example, if  $\tau_1$ 's period cannot be increased by more than 20%, we could add the following constraint:

$$T'_1 \leq 1.2 \cdot T_1 \quad (2.9)$$

Alternatively, we could discourage increasing certain periods without setting hard bounds on them by altering the fitness function. For example, if we want to force the genetic algorithm to try to increase all tasks other than  $\tau_1$ , the additive fitness function might be restated as

$$100\delta_1 + \sum_{i=2}^N \delta_i. \quad (2.10)$$

In this manner,  $T_1$  would only be increased if not increasing  $T_1$  would cause other periods to be increased by more than 100.

## ALTERNATE CHROMOSOMES

We also could reduce execution times rather than increasing periods. In this case, the periods would remain constant, but the execution times  $e_i$  would be revised to  $e'_i$ . The addition or scaling equations would then be replaced with  $e'_i = e_i - \delta_i$  or  $e'_i = e_i/\lambda_i$ , respectively, in the genetic algorithm, and the chromosome would contain the execution time scalings. Reducing execution and increasing periods are equivalent in the equation,  $U = \sum_{\text{tasks}} e_i/T_i$ , but not in the implementation of the TDA equations. In practice, execution time would be reduced by simplifying and optimizing the code.

## DEADLINE LIMITATIONS

Deadlines less than periods and offsets could also be included. The generalized TDA is more complex than the usual TDA with the use of deadlines greater than periods, but for deadlines less than periods the change in the TDA analysis is very minor. Positive offsets have been examined but not included in the experiments section.

## SEARCH SPACE

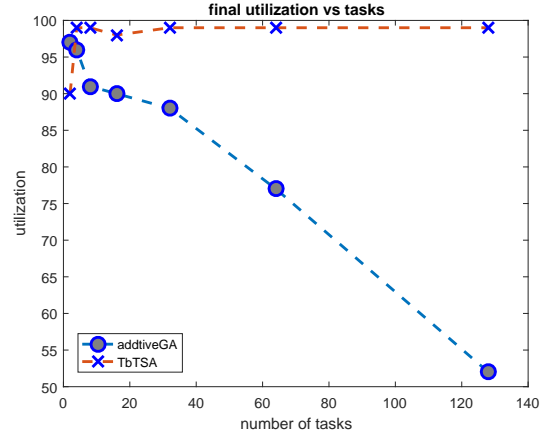
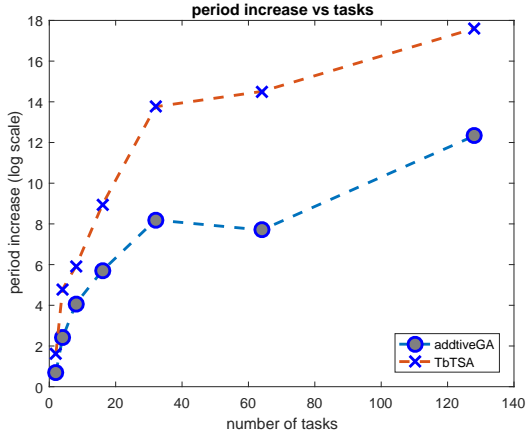
The genetic algorithm's search space could be quite large. For example, if period increases range from 0 to 100 and there are  $N$  tasks, then the search space is  $100^{2N} = 10^{4N}$  if the period increases are integer. This search space is even larger if the period increases is non-integer. This follows from the fact that there are  $2N$  genes. Each of the periods  $T_i$  and worst-case response times  $R_i$  could have values from 0 to 100; if the worst case response times are larger than 100 then the search space increases. For problems of this size, the genetic algorithm could be slow, though tuning the parameters carefully can have a dramatic impact on the runtime. Initializing the population near a possible solution also increases the efficiency of the genetic algorithm.

From the discussion above, we see that the genetic algorithm approach is very flexible. Changes in the model under consideration can be easily accommodated by changes in the genetic algorithm and would not increase its complexity.

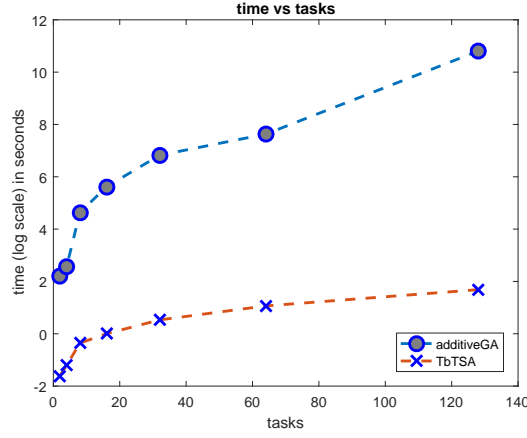
## 2.6 EXPERIMENTS

The experiments in this section will illustrate the use and effectiveness of the genetic algorithm as compared to other types of sensitivity analysis.





(a) Fitness, i.e. sum of period increases, after (b) Utilization after using the additive GA and using the additive GA and TbTSA.



(c) Time of additive GA and TbTSA.

Figure 2.3: Comparison of additive GA and TbTSA.

### 2.6.1 SCHEDULE SENSITIVITY

Three types of experiments are done to evaluate the genetic algorithm approach. These experiments tested the different types of sensitivity analysis for task sets with utilization near 100 percent and greater than 100 percent.

For most experiments, we generated 1000 task sets for each of the different task set sizes ( $N$ ). Due to time constraints, we were unable to perform 1000 simulations for a simulation

of 128 tasks with the additive genetic algorithm. For that scenario, we simulated 200 task sets.

In all the experiments, the scaleGA outperformed previous approaches in the minimization of the period/deadline changes. The genetic algorithm can also offer different solutions to the change in the task set, whereas the previous solutions can not. There is a population of solutions with a single most fit individual. Thus, the GA improves upon previous techniques in two important ways: modified task sets conform more closely to original task sets; and multiple possible solutions are provided giving designers options in selecting the final task sets. The downside of the genetic algorithm is that it can take significantly more time to find all of these solutions. The trade-off is clear: minimal parameter change versus time to find the solution.

The number  $N$  of tasks used in all of these experiments was

$$N \in \{2, 4, 8, 16, 32, 64, 128\} \quad (2.11)$$

and the task utilizations were randomly generated using UUnifast [12].

#### FIRST SET OF EXPERIMENTS

The first set of experiments used unschedulable task sets with utilization below or near 100 percent and compares the genetic algorithm with the approaches of scaleSA and BiniSA. In these experiments, pseudo-harmonic task sets were used. Pseudo-harmonic task sets are those in which all the periods are “almost” multiples of each other. The period increases of these different types of schedule changes that make unschedulable task sets into schedulable task sets are calculated for these different types of sensitivity analysis. The time of calculation is also reported for the scaleSA and BiniSA algorithms; the calculation time of the genetic algorithm is much larger.

In the first set of experiments the periods were chosen following the third experiment of [13]. Task periods were chosen to make an almost harmonic task set, i.e. periods of

5,10,100,200,500,1000 and the task utilizations were randomly chosen with the UUnifast algorithm such that the utilization was close and less than 100 percent and unschedulable.

The results of the first experiments using psuedo-harmonic task sets are shown in Figure 2.4. Note that the total period increases are smaller with the genetic algorithm, and ScaleSA does well also. Also, there are discontinuities as the number of tasks increases; this is not explained but the algorithm is nonlinear and recursive, and fixed points are common in these types of recursive equations. The genetic algorithm scaleGA has difficulty with 128 tasks or more. This could be addressed by reducing the number of scale factors in the algorithm, i.e. by blocking tasks with a single scale factor each group of tasks.

The time of calculation of scaleSA and BiniSA is compared and shown in Figure 2.5. The time it takes to create a schedulable task set using scaleSA depends upon the search; this search used a scale factor in increments of .05, (i.e. a scale factor 1, 1.05, ...). At each stage of scaling the task periods and the TDA equations were checked until a schedulable set of scaled task deadlines was found. ScaleSA seems to be faster with this graining then scaleBini. The time it takes to find the period increase of the scaleGA is large due to the stopping criteria used, which is 1000 iterations.

## SECOND SET OF EXPERIMENTS

The second set of experiments uses task sets with utilization greater than 100 percent. The task sets are general and non-pseudo-harmonic. The scaleSA, tbtsSA, and both types of genetic algorithm (scaled and additivie) are used. The BiniSA can not be compared due to the fact that the BiniSA algorithm can not be used with utilization greater than 100 percent. In these experiments, both the deadlines and execution times are integer valued and the utilization of the initial task sets are between 100 and 200 percent.

The second set of experiments used randomly selected periods in the range  $[1, N * 10]$ , where  $N$  is the number of tasks.

Two types of GA simulations were performed in the second set of experiments, which use unschedulable task sets. The first experiments use the additive increase of the periods (additiveGA) and the second type of experiments in this set used the scaling of the periods by real numbers with the genetic algorithm (scaleGA). The scaling of periods in scaleGA performed better. This choice was used to compare the genetic algorithm of the additive deadline increases of TbTSA and scaled deadline increases of scaleSA. This means that the population was randomly generated “close” to the output of the scaleSA algorithm, plus or minus 10 percent in the period increases from the output of scaleSA; the time to find the output of scaleSA is minimal compared to the time of scaleGA. Nevertheless, the initialization is important for the performance of the scaleGA. The initialization of the population of chromosomes can be important, as is typical in genetic algorithms. The parameters used in the genetic algorithm are shown in Table 2.3. In addition to the scaleGA approach, the periods of the tasks in the scaleGA experiments each had their own scaling factor.

The second set of experiments use high utilizations of unschedulable tasks. Unschedulable task sets with high utilization are important for sensitivity analysis of generally over-loaded task sets and also mode-transitions, in which processor utilization can easily go beyond 100 percent. In order to compare with the known sensitivity analyses for general task sets with utilization greater than 100 percent, we did simulations using scaleGA, additiveGA, scaleSA and TbTSA. Unfortunately, BiniSA can not be used with unschedulable task sets of utilization greater than 100 percent, which is the reason for not including it in the second set of experiments.

The two additive approaches of TbTSA and additiveGA are compared, and the results are in the next three figures. Figure 2.3a presents the average change in periods using the additive algorithms and TbTSA. Figure 2.3b presents the final utilization. Figure 2.3c presents the time of calculation of these two methods. We observe that the additive GA changes task periods less than the TbTSA. While TbTSA always creates task sets with utilization near

100%, scale GA utilizations are lower – particularly for larger task sets. This is perhaps due to the type of changes which are being made to the initial task set.

Next, we present the comparison between the scaling approaches, the scaleGA and scaleSA for general task sets with utilizations greater than 100 percent. The average period increases are shown in Figure 2.6a. These experiments use an unschedulable task set with utilization greater than 100 percent and change the periods to make these schedulable. The average final utilizations are shown in Figure 2.6b. The time of sensitivity calculations are shown in Figure 2.6c. The average period increases of the scaling GA is typically at least 50 % less and can be almost 90% of the scaleSA period increases. The utilization is high for both scale algorithms, even for 128 tasks, but is particularly high for scale GA.

In general, the scale genetic algorithm is able to find task sets whose periods are closer to the original task periods. We see that the scale GA approach finds the best solutions to the sensitivity analysis problem. This approach was able to find high-utilization task sets whose parameters were closer to the original parameters than any of the other approaches. The runtime of the scaleGA, while longer than scaleSA, is still quite reasonable.

### THIRD SET OF EXPERIMENTS

The third type of experiments used arbitrary task sets of utilization close to 100 percent, with utilization at most 100 percent. For these experiments we use a variation of BiniSA, which only considers a subset of all possible feasibility points.

Table 2.3: Parameters of the additive and scaling genetic algorithm.

| Parameter            | Value        |      |
|----------------------|--------------|------|
| Population size      | 100          | 100  |
| Crossover fraction   | .9           | .9   |
| Mutation             | Gaussian, .9 | .9   |
| Stall parameter      | 500          | 50   |
| Tolerance constraint | .01          | .01  |
| Tolerance objective  | 1e-6         | 1e-6 |
| Elite fraction       | .2           | .1   |

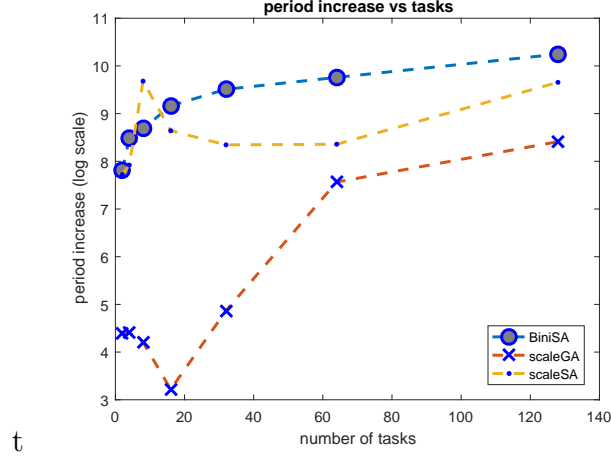


Figure 2.4: The period increases of several methods are shown, each pseudo-harmonic task set has utilization near 100 percent.

In the last set of experiments, the third set, we use general task sets of utilization close to 100 percent and general task sets. This is to show how the algorithms, including the GA approach, compare with the known approaches for general task sets of utilizations near 100 percent, without psuedo-harmonicity. The task sets were chosen with periods of  $[1, N * 10]$ , where  $N$  is the number of tasks. The task utilizations were computed using the UUnifast algorithm [12]. The integer task deadline times are scaled to make the initial utilization as close to 100 percent as possible. The point of these experiments is to show the difference of the sensitivity analysis of these different techniques at 100 percent for general task sets, which are not psuedo-harmonic. In these experiments, scaleSA, scaleBini, and scaleGA are compared. Due to the lack of known feasibility points, the BiniSA algorithm is approximated by using only periods as feasibility points. The total period increase is shown for these methods in Figure 2.7. Note that the scaleGA outperforms the previous techniques in minimizing the sum of the period increases. Task-by-task algorithms, such as scaleSA, TbTSA, and BiniSA are not as optimal in minimizing the task set parameter changes, such as periods.

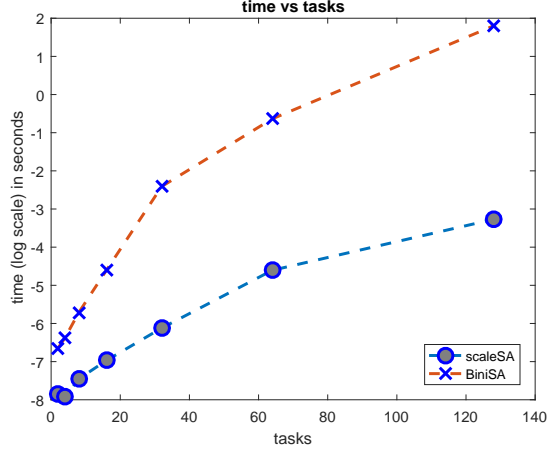


Figure 2.5: The time of calculation of scaleSA versus BiniSA. Each psuedo-harmonic task set utilization near 100 percent.

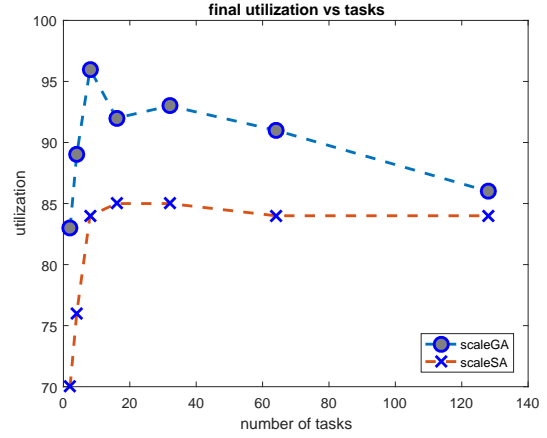
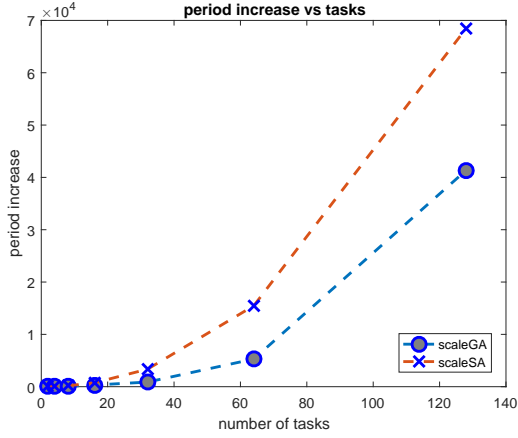
### 2.6.2 MODE TRANSITIONS

In this section, 1000 experiments are done for different numbers of tasks to show the effectiveness of the genetic algorithm in treating the mode transition. Each experiment uses a different set of task periods and execution times for the incoming and outgoing modes. These experiments are performed using 2 modes and a total set of tasks of

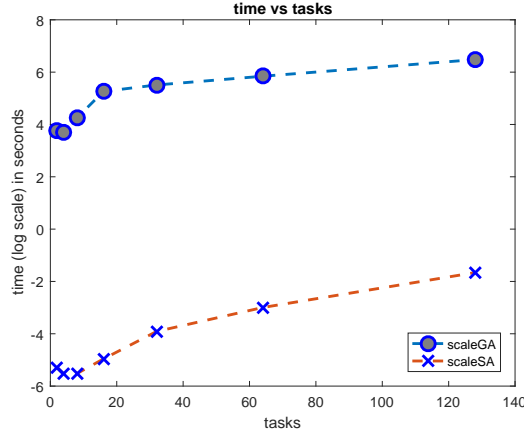
$$N = \{2, 4, 8, 16, 32, 64, 128\}, \quad (2.12)$$

for each  $N$ . Task set utilizations are created using the UUniFast algorithm [12], and task periods randomly chosen from zero to  $10 * N$ . Fixed priority scheduling is used in that higher priority is given to smaller periods.

For the mode switching experiments, each initial set of  $N$  tasks is divided into two modes with the same number of tasks, e.g. 32 tasks is two modes of 16 task sets in each mode. These two modes define the incoming and outgoing task sets. The periods of the incoming tasks can increase, reducing the interference with the outgoing tasks, thus making the system schedulable. The outgoing task parameters do not change. The “mode time



(a) Fitness, i.e. sum of period increases, after (b) Utilization after using the scaleGA and using the scaleGA and scaleSA.



(c) Time of scaleGA and scaleSA.

Figure 2.6: Comparison of scale GA and scaleSA.

multiplier” equation is used as a constraint in the genetic algorithm. This constraint forces outgoing tasks to complete their last jobs before the mode time deadline. In these experiments the mode time multiplier, defined in the model section, is 2.

Two sets of mode switching experiments were completed. In combining the total utilization of incoming and outgoing tasks is 100 percent and tested the scaleSA, scaleGA, BiniSA, and TbTSA at this utilization. The second set used a combined utilization of 150 percent.



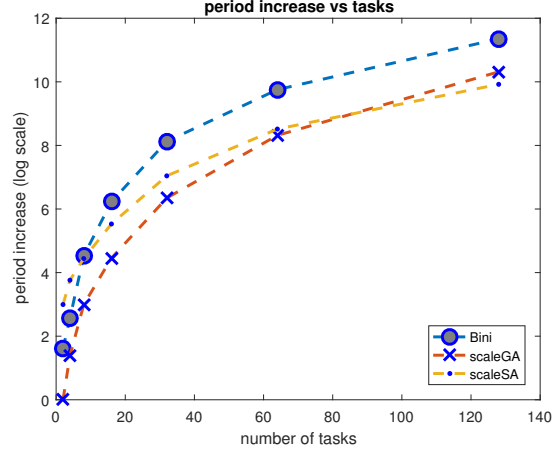


Figure 2.7: The period increases of several methods are shown, each general task set has utilization near 100 percent.

Table 2.4: Parameters of the genetic algorithm in the mode transition problem.

|                |      |
|----------------|------|
| Population     | 1000 |
| Crossover rate | .95  |
| Generations    | 2000 |
| TolFun         | .02  |
| TolCon         | .02  |
| EliteFraction  | .20  |
| StallGenLim    | 100  |

Due to the utilization values, these task sets would not be schedulable unless changes are made to the schedule.

The parameters of the genetic algorithm shown in Table 4.6.

The population in the genetic algorithm was initialized close to the output of the scaleSA algorithm; “close” means that the initial scale factors are between 1 and 10, i.e. if scaleSA gives a factor of  $\lambda$ , each task in the system has its scaling gene initialized to a value between  $\lambda$  and  $10\lambda$ . The initial worst case response times are plus or minus 30 percent from scaleSA,

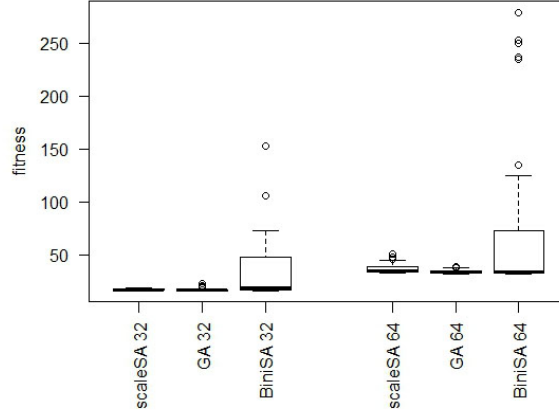


Figure 2.8: Box plots of fitness from scaleSA, scaleGA, and BiniSA in experiments 1. Utilization is 1.0 . Most are the first quartile, and there are many outliers due to the variation in the periods. There are outliers due to the large number of parameters and population of 1000.

i.e. between .7R and 1.3R. The scaleSA algorithm seems to be the next best algorithm for minimizing task set parameter changes. Using its output to initialize the population improved the performance of scaleGA. This initialization of the population improves the fitness performance of the GA. The runtime of scaleGA wasn't affected, however, the total change in the periods was less than what is found from a random population.

Comparisons and contrasts of scaleGA are made with earlier sensitivity analysis' in the mode transition problem - scaleSA, scaleBini, and scaleTbTSA. Two sets of experiments are done. In the first set of experiments, the utilization is fixed to 100 percent and the task parameters are non-integer. This is due to the utilization constraint in scaleBini of less than 100 percent. In the second set of experiments, the utilization is greater than 100, i.e. utilization of 150 is presented. The task set parameters are integer; in general the integer or non-integer nature makes little difference in the comparison with the different algorithms.

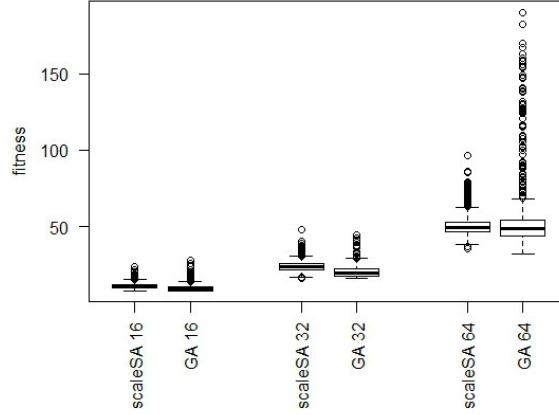


Figure 2.9: Box plots of fitness from scaleSA and scaleGA in experiments set 2. Utilization is 1.5 .

Box plots of the distribution of the total fitness of the different task sets is shown in Figure 2.8. These box plots are for utilization of 100 percent of the combined two modes before the parameter changes, and for  $N = 32$  and  $N = 64$ . The minimum total fitness is the sum of the scaling factors of half of these tasks since there are half in each mode, and outgoing task's parameters don't change.

The GA generally gives a lower total fitness than scaleSA or BiniSA, although sometimes performs worse than scaleSA, which appears to be the case with 32 tasks. BiniSA has a larger spread of the total fitnesses in the distribution; this is due to increasing task periods from high priority to low priority. At 100 percent utilization the total fitness is close to the minimum, due to the fact that half utilization is 50. In the experiments, the incoming mode was unschedulable. TbTSA was not included due to the large general increase in task periods; this algorithm, unlike the task-by-task BiniSA algorithm does not use a set of modified TDA equations to find minimal task set changes. The mean and medians for these experiments are in Table 2.5.

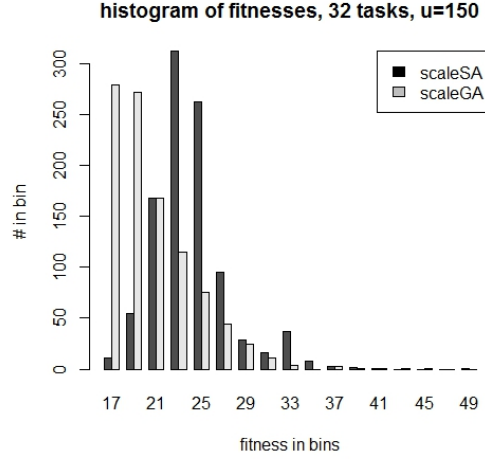


Figure 2.10: Histograms of fitnesses from scaleSA and scaleGA in experiments set 2.

In the first set of experiments the scaleGA and scaleSA gave similar results. The incoming mode task parameters don't have to change too much to achieve schedulability of the mode transition. If the tolerances of the constraints and objective function were decreased, the GA could possibly improve at the cost of additional computation.

Box plots are also shown for the second set of experiments, Figure 2.9. In these experiments, the total initial utilization was approximately 150%, and the cases of  $N = 16, 32, 64$  are shown. The GA had lower fitnesses on average for the 1000 task sets. ScaleSA is the second best algorithm for minimizing total task set changes. BiniSA was not included because the utilization was greater than 100%. The TbTSA algorithm produces large task changes and the results are instead included in the tables 2.6. Also, TbTSA in about 10 percent of the task sets does not produce a solution; this is because there could be an intermediate utilization of 100 percent of the partial task set, forcing the lower priority task period increases to infinity. The GA fitnesses were less than the TbTSA fitnesses by 10 to 30 percent generally. The mean, median, and the inter-quartile ranges are shown in the box plots. Outliers are present

Table 2.5: Utilization 100. Mean and Median of scaleSA, scaleGA, BiniSA, and TbTSA. About 10 percent of TbTSA results are not included due to no solution of period increases.

|                     | scaleSA | scaleGA | BiniSA |
|---------------------|---------|---------|--------|
| Mean ( $N = 32$ )   | 17.56   | 16.97   | 36.58  |
| Median ( $N = 32$ ) | 17.28   | 16.68   | 18.95  |
| Mean ( $N = 64$ )   | 36.46   | 34.24   | 70.84  |
| Median ( $N = 64$ ) | 35.2    | 34.13   | 33.79  |

due to the large number of parameters and population of 1000; there is more variation than with utilization 1.0 due to a larger range in the variation of the task periods.

It is interesting to note that the median of the scaleGA distribution always seems to be greater than the 75 percent quartile of scaleSA, which is the closest algorithm in effectiveness of task set changes to scaleGA. This shows that statistically the scaleGA algorithm performs better on average than scaleSA. BiniSA does in fact produce much worse total fitnesses than scaleSA or scaleGA by at least on average a factor of 4.

Histograms are presented for the second set of experiments in Figure 2.10. Note the distributions of the different algorithms. ScaleSA does not have a minimization goal of task set changes in the scaling of the task periods.<sup>3</sup>

Two tables, 2.5 and 2.6, have the basic statistical information (mean and median) of the histograms.

Mode independent tasks were also used in the set of incoming task parameter changes in a set of different runs. The mode independent task periods are not changed. These tasks are recurring during the mode change and the results are not changed using the incoming tasks and the total parameter change. The total utilization, however, has increased with these tasks.

---

<sup>3</sup>The histograms for TbTSA are quite skew and in a large fraction of the task sets there is no solution to schedulability. Its' histograms are not included.

Table 2.6: Utilization 150. Mean and Median of scaleSA, scaleGA.

|                     | scaleSA | scaleGA |
|---------------------|---------|---------|
| Mean ( $N = 16$ )   | 11.4    | 9.88    |
| Median ( $N = 16$ ) | 11.36   | 9.08    |
| Mean ( $N = 32$ )   | 24.18   | 20.55   |
| Median ( $N = 32$ ) | 23.68   | 19.66   |
| Mean ( $N = 64$ )   | 50.8    | 54.31   |
| Median ( $N = 64$ ) | 49.28   | 48.49   |

## 2.7 CONCLUSION AND FUTURE WORK

This work formulated the problem of scheduling fixed-priority real-time systems in terms of optimization problems. Genetic algorithms were used to examine the sensitivity of scheduling task sets by scaling periods for both scheduling of unschedulable task sets and also mode transitions. Earlier sensitivity analysis approaches were compared with these genetic algorithms. From this work we see that genetic algorithms are an effective technique for turning unschedulable task sets into schedulable task sets with as little change as possible. Experiments demonstrated that the GA can produce schedulable task sets with smaller parameter changes than existing techniques.

There are several avenues of additional research that be explored based on this work. Different optimization schemes could be used, such as particle swarm. The sensitivity analysis problem can also be formulated as a convex quadratic scheduling problem, which has special techniques to find the global minimum of task set parameters.

## CHAPTER 3

### GENETIC ALGORITHMS USED IN NMR

**Key Words:** mammalian cell culture, molecular dynamics, ligand docking, genetic algorithm, sialyltransferase, resonance assignments, assignment program, sparse labeling, perdeuteration, heat-shock protein, Hsp90, HtpG, protein structure,  $^1\text{H}$ - $^{13}\text{C}$  methyl RDCs.<sup>1</sup>

#### 3.1 ASSIGN SLP MD

A genetic algorithm has been created to assign protein NMR resonance. This means that the crosspeaks of an HSQC (heteronuclear single quantum coherence) spectrum can be attached to particular residues of the protein. There are many ingredients involved. The measurements are potentially rdc's, measured chemical shifts of the  $^{15}\text{N}$ - $^1\text{H}$  or  $^{13}\text{C}$ - $^1\text{H}$  pairs in the residues of interest. Sparsely labeling is used in this line of research; only a set of residues such as all phenylalanines are used in the measurements and the experiment is done with a labeled protein in which these residues are  $^{13}\text{C}$  or  $^{15}\text{N}$  enriched. In addition to these measurements, predictions of the same measurements using molecular dynamics (MD simulations are used). Amber is a well known tool that creates a molecular trajectory and from this the predicted spectral parameters are calculated. The predicted chemical shifts are found using ShiftX2

---

<sup>1</sup>The work in this chapter is based on the papers, G. Chalmers, A. Eletsky, L. Morris, J. Yang, F. Tian, R.J. Woods, K.W. Moremen, J.H. Prestegard, NMR Resonance Assignment Strategy: Characterizing Large Sparsely Labeled Glycoproteins, *Journal of Molecular Biology*. v. 431, pp. 2369-2382, issue 12; K. Pederson, G. Chalmers, Q. Gao, D. Elnatan, T.A. Ramelot, L. Ma, G.T. Montelione, M.A. Kennedy, D.A. Agard, J.H. Prestegard, NMR characterization of HtpG, the *E. coli* Hsp90, using sparse labeling with  $^{13}\text{C}$ -methyl alanine, *Journal of Biomolecular NMR*, vol 68, issue 3, pp. 225-236, July 2017; Q. Gao, G.R. Chalmers, K.W. Moremen, J.H. Prestegard, NMR assignments of sparsely labeled proteins using a genetic algorithm, *Journal of Biomolecular NMR* (2017) 67:283-294; G. Chalmers, J.N. Glushka, B.L. Foley, R.J. Woods, J.H. Prestegard. Direct NOE simulation from long MD trajectories. *Journal of Magnetic Resonance*. 2016;265:1-9.

or ppm1 software and the trajectory. The rdc's can be found using the average coordinates of the N-H's or C-H's and order parameters describing any fast internal reorientation. As mentioned in the introduction, another software package, MD2NOEProtein, that we created, calculates these. This is not in the dissertation due to the fact that it does not use a genetic algorithm; it is referenced.

AssignSLPMD started with AssignSLP. The latter also requires the input of predictions from its associated MD2NOE software; but a single frame was used for the coordinates of the N-H or C-H bond and in the chemical shift predictions. This single frame does limit the use of AssignSLP. AssignSLP and MD2NOE are not appropriate for molecules with substantial internal motion. Larger proteins often have substantial internal motion and longer MD trajectories are required in order to sample these motions. Conformational changes in motion of the protein can give misleading predictions unless these changes are included in the MD trajectory. The AssignSLP genetic algorithm package was successfully used in the assignment of 5 small proteins (small meaning  $< 10$  kiloDaltons) and a larger 3-domain dimer [33] [66].

AssignSLP and MD2NOE, were generalized to AssignSLPMD and MD2NOEProtein. A 'sphere' approximation was included in the MD2NOEProtein software, which effectively extends the MD trajectory by adding in an artificial tumbling of the protein. These predicted spectral data are then used in AssignSLPMD. AssignSLPMD is very different from AssignSLP due to the fact that every set of predictions to measurements uses the full trajectory. The chemical shifts of the N-H's and C-H's are found by averaging the chemical shift predictions at every frame of the trajectory (every 2 picoseconds of 1000 nanoseconds). The rdc's include a order parameter correction due to the motional wobbling of the bond of the N-H's or C-H's. These order parameters are calculated in MD2NOEProtein. The inclusion of the sphere approximation, and the full trajectory extend the use of the AssignSLPMD genetic algorithm software to large proteins. The software was tested in the assignment of rST6Gal1, a protein of 36 kD [15].



This chapter discusses the AssignSLPMD genetic algorithm software in detail, and not the AssignSLP software. AssignSLPMD supersedes AssignSLP and can calculate everything that AssignSLP can. Both packages have statistics programs that give the likelihood of correctness of each spectral intensity to residue. The results of AssignSLP and the 6 proteins are also given in this chapter in the supplement to demonstrate its use [33,66]; these results can also be found from AssignSLPMD and in the latter principle would be more accurate due to the use of an MD trajectory. The description of AssignSLPMD and its use and text in glycoproteins such as ST6Gal1 is largely taken from the work in our recent publication [15].

## 3.2 INTRODUCTION

NMR structural studies of uniformly  $^{13}\text{C}/^{15}\text{N}$  labeled proteins larger than 40-60 kDa are challenging even when perdeuteration is used to enhance resolution and sensitivity [66]. For glycosylated proteins, which are often expressed in mammalian cell culture to produce native-like glycosylation, perdeuteration is not possible; even structural studies of 20-30 kDa proteins are then challenging. Moreover, uniform isotopic labeling in mammalian cells with  $^{13}\text{C}$  and  $^{15}\text{N}$  can be costly as a mix of isotopically labeled amino acids, as opposed to isotopically labeled metabolic substrates, such as glucose and ammonium chloride, must be supplied. An economically viable alternative exists, namely sparse labeling using a single or small subset of isotopically labeled amino acids [62]. Sparse labels can provide long range structural constraints through paramagnetic perturbations of resonance positions and intensities, as well as orientational constraints from residual dipolar couplings (RDCs) [45], [69], [64]. These constraints, along with chemical shift perturbation on interaction with other entities, can often be used to position ligands in binding sites and assemble proteins in multi-protein complexes [38] [8]. However, resonances must still be assigned to specific sites in proteins, and this must now be done without the aid of the triple resonance experiments usually applied to uniformly labeled proteins [32].

We recently introduced a strategy for resonance assignment of sparsely-labeled proteins that relies on acquisition of nuclear Overhauser effects (NOEs), RDCs and chemical shifts; all parameters measured directly from, or through modulation of, crosspeaks seen in basic two-dimensional heteronuclear single quantum coherence (HSQC) or multiple quantum coherence (HMQC) spectra. The strategy was implemented in a program package, ASSIGN\_SLP, that employed a genetic algorithm to optimize pairing of specific spectral crosspeaks with specific protein sites using scores that compare experimental measurements of these parameters to predictions based on prior structural information, primarily from a single X-ray structure. The package was tested on a set of four small non-glycosylated proteins having known structures and crosspeak assignments, as well as a small glycoprotein [34]. It was subsequently applied to a larger non-glycosylated and perdeuterated protein, for which only the structure of isolated domains was known [67]. While the general approach showed success with smaller systems, it became clear that for larger systems, factors in addition to the technical aspects of associating predictions with experimental measurement would have to be considered. These include degeneracies in data that increase with the number of labeled sites, the greater probability of internal motion affecting observables and the more extensive spin-spin interactions that occur in larger proteins. Here, we introduce an approach that predicts parameters from molecular dynamics (MD) trajectories, as opposed to single structural snapshots from X-ray structures, to better account for effects of internal motion and spin-spin interactions on predicted parameters. It also uses an improved procedure for identification of high-confidence assignments in the presence of data degeneracy. This approach, now embodied in a software package entitled ASSIGN\_SLP\_MD, proves useful in providing key assignments for a challenging 36 kDa glycoprotein, the luminal domain of rST6Gal1 (hereafter just rST6Gal1).

ST6Gal1 is a sialyltransferase that adds a sialic acid to the terminal galactose of N-linked glycans of many glycoproteins, and is therefore of importance in mammalian physiology [86]. The bond it forms is from the 2-carbon of sialic acid to the 6-oxygen of galactose, as opposed to the 3-oxygen of galactose. The specificity of the hemagglutinin of the avian influenza

virus for the 2-3 linkage, found on glycans in the human gut, but seldom in the upper respiratory tract, is what restricts the transmission of bird flu to humans [75], [43]. Levels of 2-6 linked sialic acid also rise in certain types of cancer and there is significant effort devoted to understanding the possible role of sialylation in this disease [20], [9]. A decade ago we began an NMR-based structural study of rST6Gal1 [55]. At the time there were no crystal structures of ST6Gal1, or any of a close structural homolog. Using a sparse labeling approach in which all phenylalanines were labeled with  $^{15}\text{N}$  we demonstrated adequate resolution and sensitivity to detect HSQC crosspeaks from all 16 phenylalanine amide protons in the construct. Using a paramagnetic analog of the sialic acid donor (CMP-sialic acid), in which carboxy-TEMPO replaced the carboxyl-carrying sialic acid, we also showed that four of the 16 crosspeaks lost significant intensity. Based on an expected  $1/r^6$  distance dependence of intensity loss, this number was deemed consistent with the number of phenylalanines in peptide segments believed to form the active site. However, in the absence of assignments we were unable to use the paramagnetic constraints to dock the donor analog in the active site of a homology model. In 2013 two X-ray structures appeared [50], [58], one of the rat enzyme on which our NMR work had been done [58]. With this structure in hand, along with previously collected RDC data, newly collected  $^1\text{H}$ - $^1\text{H}$  NOE data, and our new sparse label assignment strategy, we have proceeded with assignments of a new construct of rST6Gal1, isotopically labeled with  $^{15}\text{N}$  in all phenylalanines. A subset of the assignments are validated using a limited set of mutants in which single phenylalanines are changed to tyrosines, and then the assignments are used to place a sugar donor analog in the active site of rST6Gal1 in a manner consistent with paramagnetic perturbation data.

### 3.3 RESULTS

The ASSIGN\_SLP\_MD package is a collection of programs, primarily MATLAB scripts, that accepts as input a user-supplied MD trajectory, one or more files with experimental NOE peak lists (or NOE vectors derived from NOE strip plots), a file with  $^1\text{H}$  chemical shifts for

labeled sites, a file with  $^{15}\text{N}$  or  $^{13}\text{C}$  chemical shifts for labeled sites and one or more files with RDC lists. Each of the files ends with a list of error estimates modified by weights for the specific data type. As success is very dependent on having adequate amounts of experimental data, it is recommended that at least one NOE file or one RDC file, in addition to chemical shifts, be present. Predicted data are appended to experimental files by scripts that call other programs to make these predictions. PPMONE [53] or SHIFTX2 [42] are used to predict chemical shifts averaged over frames of the trajectory. In the case of NOEs, a new version of our MD2NOE program, MD2NOE\_Protein, is called; it uses the trajectory directly to make NOE predictions, taking into account the effects of internal motion and the extended interactions among multiple proton spins [16], [76]. In the case of RDCs, trajectories are used to calculate order parameters, which measure the amplitude of rapid variations in  $^1\text{H}$ - $^{15}\text{N}$  or  $^1\text{H}$ - $^{13}\text{C}$  bond orientations relative to the molecular frame, and produce coordinates for an average bond orientation; these in turn are used to adjust motionally-averaged experimental RDCs to a rigid equivalent and back-calculate predicted RDCs for each trial assignment using an algorithm similar to that in the REDCAT program [85]. A master script then calls a genetic algorithm that begins with a randomly generated set of assignments (each “gene” being a list of 16 crosspeaks assigned to 16 different sites in our case). It calculates scores for each list based on an objective function that compares predicted and measured data, and it uses a series of runs with different crossover and mutation rates to mix assignments among the best scoring lists (genes) in an attempt to find an optimal assignment. Solutions with scores below a user-specified maximum are saved and latter analyzed by scripts that order output in terms of increasing scores and generate a heatmap showing the frequency of assignment of each crosspeak to each residue.

At the heart of the program is the objective function used in the genetic algorithm search for an optimal assignment. Initially this was defined as the sum of root-mean-square deviations (RMSDs) between measured and predicted values, divided by estimated errors (predicted plus observed standard deviations), for all data types except NOEs. The RMSDs

minimize as agreement between measurements and predictions improves, as required for a well-behaved objective function. NOEs were treated differently because they are not represented by a single number, but by a series of intensities at the chemical shifts of NOE donating protons (actually a vector representation of a strip-plot from a 3D-NOESY spectrum). A Pearson correlation coefficient (R-value), which is a common way of assessing the similarity of two vectors was used to compare predicted and measured NOE vectors. The total NOE score, considering NOE vectors emanating from all crosspeaks, was then given as  $(1-R)^2$ , as opposed to  $R^2$ , divided by an estimated error, since R would go from 1 for perfect correlation to -1 for complete anti-correlation.

### 3.3.1 NEW ADDITIONS

The primary improvement in ASSIGN\_SLP\_MD comes from using, not just a single snapshot of a protein structure as typically exists in a crystal structure, but from using long MD simulations to capture some of the effects of conformational averaging. This is not new in principle; MD simulations have been used previously to improve chemical shift prediction [53] and to provide order parameters which aid in interpretation of spin relaxation data [40], but they have not been used routinely. Until a few years ago a 1 s MD run on a fully solvated protein, the size of rST6Gal1 would have been considered impractical. However, advances in computational hardware are now putting this timescale within reach of many laboratories. Our simulation of ST6Gal1 began with a crystal structure of the rat enzyme under conditions where neither donor nor acceptor was present (PDB ID 4MPS) [58]; these conditions match the conditions under which experimental data were collected. Unfortunately, this structure is missing a loop from 354-362 that contains two of the 16 phenylalanines. This loop was added directly from a structure of the homologous human protein in which the nucleoside portion of the donor was present (PDB ID 4JS1) [50]. The run required about two weeks on two GPUs running the PMEMD module of AMBER 14 [14]. Additional details are included in Materials and Methods.

The effect of using an MD simulation to improve prediction of RDCs is substantial. RDCs provide information on bond vector orientations ( $^1\text{H}$ - $^{15}\text{N}$  bonds in our case) relative to a molecular alignment frame, but in the presence of internal motions that rapidly reorient these vectors, measured RDCs are reduced from their rigid limit (scaling by 1.0) to values scaled by the same order parameters that affect spin relaxation measurements. Dividing experimental RDCs by MD-derived order parameters scales values up to rigid equivalents that can easily be compared to predictions made during the genetic algorithm search. Order parameters for two of the residues within rST6Gal1, F132 and F356, are particularly small with values of 0.51 and 0.59 respectively.

While the use of the MD trajectory to better approximate RDC data proves valuable, the potential impact of using MD-based predictions is most dramatic in the case of NOEs. Our initial application to small proteins used an assumed  $1/r^6$  distance dependence and distances extracted from crystal structures to predict NOE intensity contributions from each potential donating proton for each crosspeak. Chemical shifts of donating protons were predicted by the software PPMONE [53] or SHIFTX2 [42] and predicted intensities were centered on these shifts, but spread over a region reflecting the uncertainty in prediction, to generate predicted NOE vectors [34]. For larger proteins and proteins having more internal motion, the  $1/r^6$  assumption breaks down for two reasons. First, for internal motions that are fast compared to molecular tumbling, motional averaging depends on  $1/r^3$  (plus an angular term), not  $1/r^6$ . Second, spin-diffusion effects, which are particularly prevalent in large proteins, make long-distance transfers by indirect mechanisms important. Direct calculation of correlation functions from an MD trajectory takes care of the former problem [41]. Use of a “complete” relaxation matrix takes care of the latter problem [63]. As a case in point, assuming a  $1/r^6$  dependence, the ratio of NOEs for the amide proton of F208 in rST6Gal1 on inversion of neighboring proton HD1 on its phenyl ring and inversion of neighboring proton HB3 on its  $\beta$ -carbon would be 4.7. Using the program MD2NOE-Protein [16], that incorporates both correlation function calculations and a “complete” relaxation matrix approach, one obtains a

ratio of NOEs at a 40 ms mixing time of 0.9. We use this program to generate predicted NOE vectors in our new assignment strategy. For consistency with other terms in our objective function, NOEs are now scored as the RMSD of (1-R).

Weighting of various data types in our objective function have also changed. In our initial application various data types were simply weighted by the inverse of an estimated error. Dividing by estimated errors makes contributions of individual terms approach 1 when deviations approach estimated error. With standard estimates of error, a total score equal to the number of data types then provides a cutoff below which any total assignment should be considered acceptable. Because we are usually more interested in the confidence that can be placed in the assignment of a particular site to a crosspeak of interest (one perturbed on ligand binding, for example) than the assignment of all crosspeaks, we had suggested use of a confidence score based on the frequency of assignment of a crosspeak to one particular site within the set of all complete assignments deemed acceptable. We plan to keep this means of confidence assessment. However, there are factors, other than precision of measurements and predictions, that should be included in the weights used in the course of our genetic algorithm search. Factors that are not well represented in estimates of error include what we call “information content”. For example, degeneracies in RDC data may arise in certain proteins (an alpha-helical bundle) because the vectors connecting spin pairs ( $^{15}\text{N}$ - $^1\text{H}$  amides) may be nearly parallel. This would reduce information content of the RDCs. Also, missing data allows interchange of assignments regardless of the precision of measurement. In this new version we have introduced an option that allows weighting by information content in addition to the inverse of an error estimate. In practice we define this as the variance in score relative to the square of the range of scores for each data type and provide MATLAB scripts that calculate weights for each data type. In addition, scaling by the ratio of the number of independent data points (measured values in most cases, but measured -5 in the case of RDCs where 5 order tensor elements must be determined from the measured values) to the number of sites to be assigned is included automatically as a part of the inverse of

error estimates. This decreases the importance of data types when experimental data for particular sites are missing.

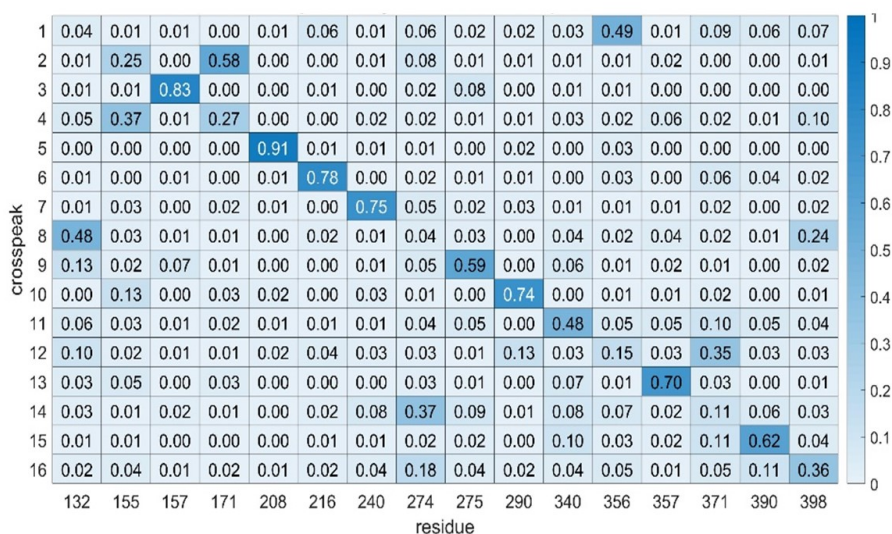


Figure 3.1: Heatmap showing ST6Gal1 phenylalanine assignments using simulated data. The numbers in each element are the fraction of times a crosspeak is assigned to a particular residue. Higher numbers are color-coded a darker blue and are taken to indicate a more confident assignment. All correct assignments should be on the diagonal in this case since the crosspeak order is was chosen to order with the residue order.

**Establishing a confidence cutoff.** In using any assignment program, it is important to attach a level of confidence to the assignments made. This can be done by first testing the program on simulated experimental data that has been generated by adding random errors of a known magnitude to a predicted set. This procedure is documented in 'Weighting Assignment Score' as part of the documentation in the download at [http :  
//tesla.ccruc.uga.edu/software/AssignSLPMD/](http://tesla.ccruc.uga.edu/software/AssignSLPMD/); the documentation also explains the confidence cutoff. A predicted set appropriate for our eventual application to rST6Gal1 was generated using chemical shifts from a combination of PPM and SHIFTX2 calculations averaged over rST6Gal1 trajectory frames, using two RDC sets from application of the REDCAT program to a single frame from the trajectory, and using NOEs from the MD2NOE\_Protein program as described above. A simulated experimental set was then generated by adding



random errors to chemical shifts and RDCs, within limits that proved applicable to the actual experimental data (see section on application to experimental data below). For RDCs, 2 and 4 pieces of data, respectively, were also deleted from the two sets to mimic missing data in the actual experiments. NOE intensities were randomly varied within a 25% limit and peak positions were varied within errors for shifts. Application of the program ASSIGN\_SLP\_MD gave a best solution with an unweighted best score of 4.1. Using an unweighted score cutoff of 5.0 (an average contribution of 1.0 for each of the 5 data types), the heatmap presented in Fig. 1 was produced. The numbers displayed in the heatmap are the fraction of time an assignment of a crosspeak to the same residue is made in a set of total assignments having scores between 4.1 and 5.0. The residues and calculated data were not scrambled, so the correct solutions occur on the diagonal. Note that most of the high fractions (darker blue) occur along the diagonal. If we choose a cutoff level of 0.50, we would identify 9 assignments as highly confident and there would be only one false positive. Hence, this cutoff can be associated with approximately a 90% confidence level. Using a less conservative approach in which errors are scaled down by 2/3, all 16 peaks have highly confident assignments and all 16 are assigned correctly.

This confidence level of 90% can be found by taking a simulated set of experimental data in which all peaks are sequentially assigned in the diagonal, adding 20% random noise and checking the assignment from re-running the algorithm. The confidence level is found by counting the number of correct assignments to the predicted level. .5 gives 9 assignments as correct. This procedure gives an estimated level of confidence in the fractions of the heatmap. Until a more rigorous statistical interpretation of the fraction of assigning is developed, this gives a rough interpretation of these fractions.

### 3.4 APPLICATION WITH RST6GAL1

Experimental data on ST6Gal1 consisted of chemical shifts from an 800 MHz  $^1\text{H}$ - $^{15}\text{N}$  HSQC spectrum, NOEs from an 800 MHz NOESY-HSQC spectrum and two sets of 900 MHz RDC

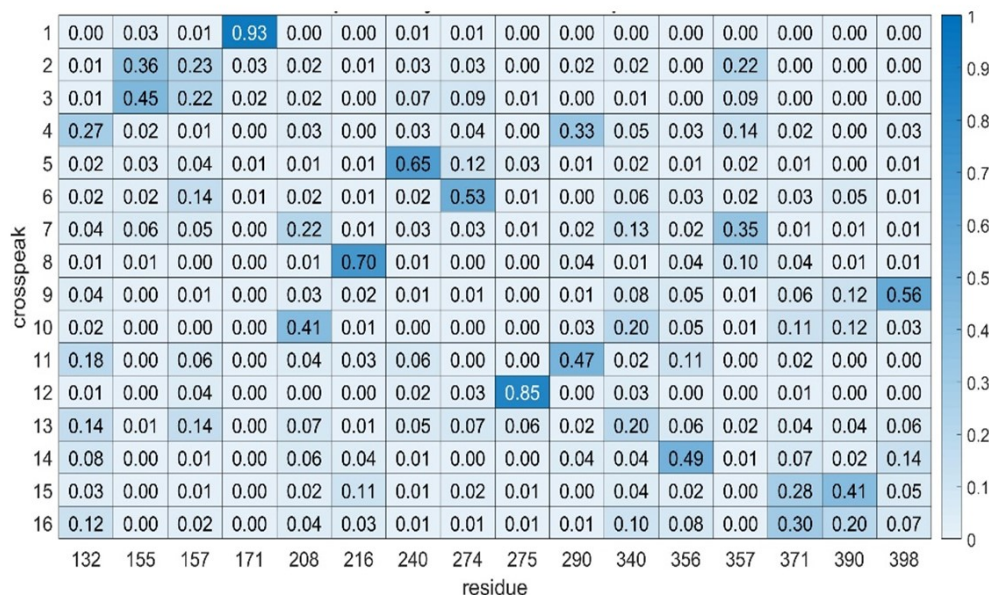


Figure 3.2: Heatmap showing ST6Gal1 phenylalanine assignments using experimental data. The 6 most confident assignments (fraction  $\geq 0.5$ ) are shown in darker shades of blue. These are now scattered throughout since we don't know apriori how to order the crosspeaks.

data, one using bacteriophage, and one using alkyl-ethylene-glycol (C12E5, PEG) bicelles to orient the protein. Errors for the chemical shifts are dominated by errors in predictions; these were initially set to two times the errors suggested by the authors of SHIFTX2. Errors in NOEs were taken from the noise level in experimental spectra and errors in RDCs were estimated based on line widths of spectra. Both the data and error estimates are detailed in Materials and Methods. Using these errors, the initial run of ASSIGN\_SLP\_MD failed to give any solutions with a score below the expected error-derived limit of 5.0. This is likely due to error contributions to data or simulations that are difficult to predict (for example, truncation of NOEs by exchange phenomena or failure to sample all conformers in the 1 s trajectory). We therefore raised all errors by 50% and repeated the run. The minimum solution then had a raw score of 4.4. A heatmap generated using all solutions below 5.0 is

shown in Fig. 2. If we use the confidence cutoff of 0.50 suggested by our simulated data, we would confidently assign 6 of the 16 crosspeaks.

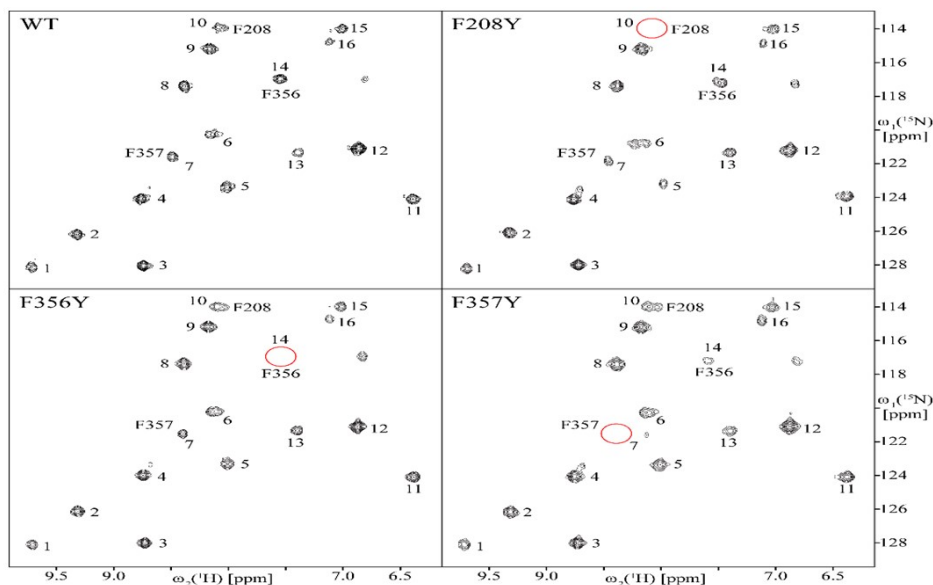


Figure 3.3: 800 MHz 2D  $[^{15}\text{N}, ^1\text{H}]$  HSQC spectra of WT rST6Gal1 and single-point mutants F208Y, F356Y and F357Y. One crosspeak disappears in each of the mutant spectra (red circles) identifying the crosspeak belonging to the mutated site.

Data not included in the objective function can in general be used to assess accuracy. Our prior work on rST6Gal1 had shown that addition of an analog of rST6Gal1's nucleotide sugar donor that carries a paramagnetic TEMPO group causes paramagnetic relaxation enhancement (PRE) and intensity loss for four crosspeaks (6,7,10 and 14) [55]. Manually docking this donor analog into the active site of the 4MPS crystal structure and having the missing residues 356 and 357 modeled in from the 4JS1 structure results in a position for the TEMPO nitroxide group with the four closest phenylalanine amide protons (those of F208, F240, F356 and F357) at distances of 12, 16, 11 and 10 Å. Of this group, one assignment is at the edge of our confidence limit, F356 to crosspeak 14; this is in agreement with PRE data. While below our confidence limit, both F208 and F357 have their highest fraction of assignments to crosspeaks 10 and 7 respectively. This too is in agreement with PRE data.

Based on frequency of assignment, peak 6 would be incorrectly assigned to F274, a residue far removed from the active site.

In addition to this work, we also made an additional trajectory CYYH by changing a residue to a histidine. This trajectory is not in our published work but gave a slightly more reliable assignment. There was different conformational structure and the assignment gave a better fit.

### 3.5 VALIDATION

A more robust validation can be carried out by mutating phenylalanine residues to tyrosines, resulting in elimination of crosspeaks for the mutated residues. This was done for the three phenylalanines closest to the TEMPO group in the ST6Gal1 model, F208, F356, and F357. HSQC spectra for the 3 mutated proteins are shown in Fig. 3. Along with the HSQC spectrum of the wild type (WT) protein. In each of the spectra for mutated proteins one crosspeak is missing (red circles). This clearly assigns crosspeak 10 to F208 crosspeak 14 to F356 and crosspeak 7 to F357. The assignment made by ASSIGN\_SLP\_MD is therefore correct for our near-confident assignment (F356) as well as the highest fraction assignments of F357 and F208. While the mutational validation does not strictly overlap with our confident assignments (only the assignment of crosspeak 7 to F357 is close with a fraction of 0.49 as opposed to 0.50), the correlation of mutational assignments with the highest scoring assignment in each case adds confidence to our procedure.

Mutational assignments can, of course be regarded as additional experimental data. These are easily incorporated into our assignment strategy through a penalty matrix (residue by crosspeak) that adds a zero score to our objective function for any assignment known to be correct and a high score ( $\sim 10$ ) for all other assignments. The results of applying this procedure using the 3 mutational assignments are shown in Fig. 4. There are now 9 assignments that we would regard as confident. There has been one notable removal of an assignment from the confident assignment list, that of crosspeak 6 to F274. Since crosspeak 6 is one of the

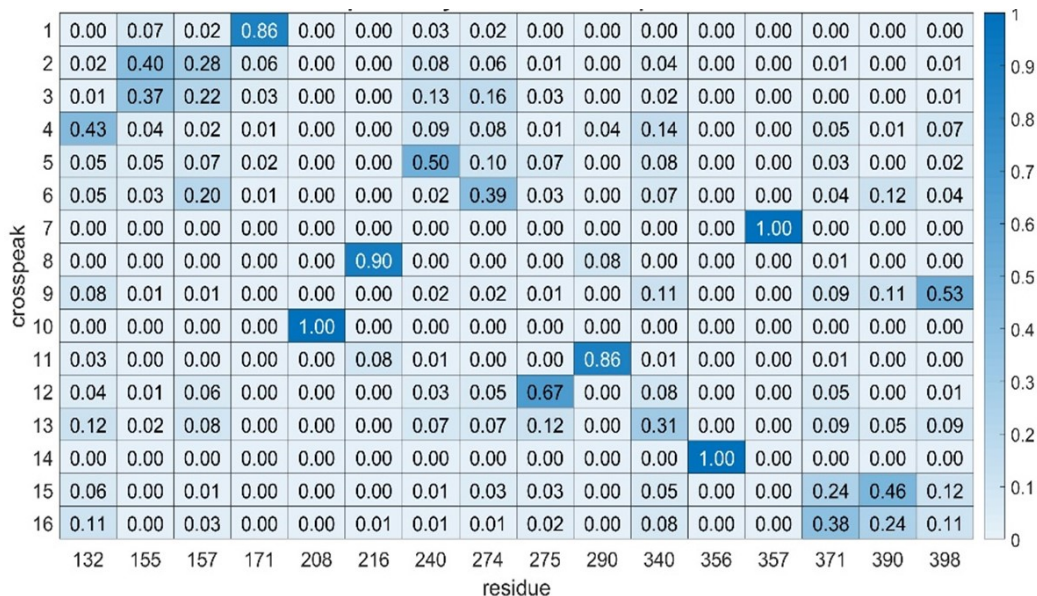


Figure 3.4: Heatmap showing rST6Gal1 phenylalanine assignments using mutational constraints (F208 to 10, F356 to 14 and F357 to 7).

crosspeaks showing intensity loss in the presence of a paramagnetically tagged donor analog, and F274 is not among the list of nearby residues, removal from this list is reassuring.

### 3.6 COMPARISON OF MD VERSUS SINGLE FRAME

A remaining question is whether the use of an MD trajectory to simulate NMR data has made a significant difference in the quality of crosspeak assignments. To examine this, we used a single frame version of the ASSIGN\_SLP\_MD program which assumes a  $1/r^6$  distance dependence to derive relative NOE intensities. Two single frames having the smallest RMSDs of backbone atom positions from the crystal structure (1.11 and 1.16 Å) were chosen from 500 samplings of the trajectory (the crystal structure could not be used directly because of the absence of the 354-362 loop). The procedures and errors used were identical to those used

with the MD version of the program. The raw scores for the best solutions in the two frames were 4.3 and 4.2 respectively, not very different from those using the MD derived predictions. However, the fractions assigned to any particular pairing are generally lower and the number of high confidence scores are lower (4 and 6 in the 1.11Å and 1.16Å frames respectively). A heatmap for the 1.11Å frame produced using assignments with scores below 5.0 is presented in Fig. 5. There is some similarity to the MD-based assignment in Figs. 2 and 4. For example, crosspeak 1 is confidently assigned to F171, crosspeak 5 has its highest fraction of assignments to F240 and crosspeak 15 has its highest fraction of assignments to F390. However, neither of the single frame runs makes a highest fraction assignment consistent with any of the three mutationally validated assignments. Clearly, there is a substantial advantage in using MD simulations to improve predictions.

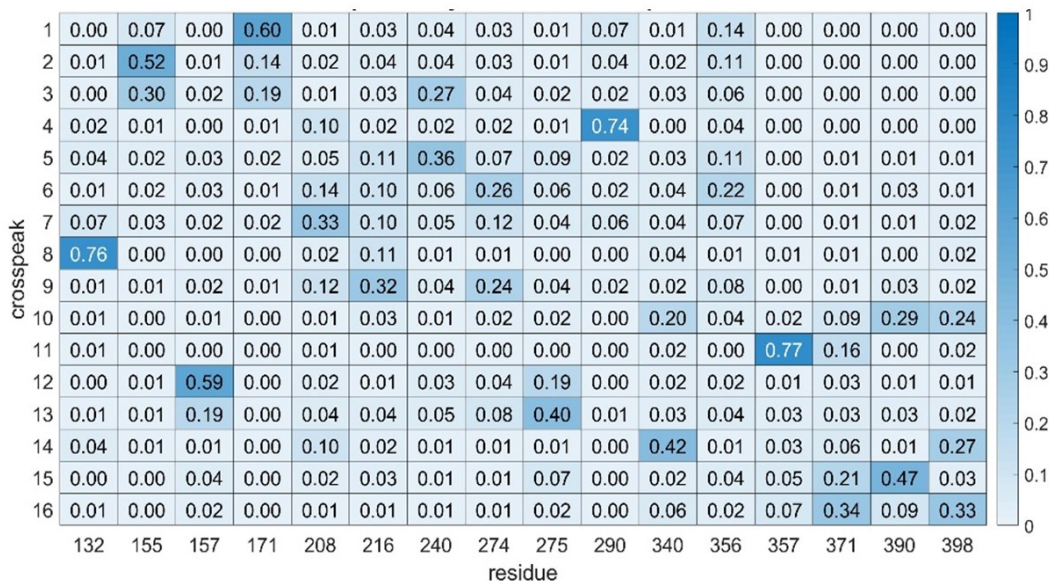


Figure 3.5: Heatmap showing ST6Gal1 phenylalanine assignments using a single frame with a 1.10 Å RMSD of backbone atoms from those of the crystal structure, 4MPS.



### 3.7 DISCUSSION

The data presented on the assignment of  $^1\text{H}$ - $^{15}\text{N}$  crosspeaks in HSQC spectra of the sparsely-labeled glycoprotein, rST6Gal1, suggests that similar assignments will be possible on a host of biomedically relevant proteins that are best expressed in mammalian, or other eukaryotic cells. Validation of assignments has confirmed an ability to set reasonable confidence limits on assignment so that, even when total assignments are not possible, a subset can be identified as trusted assignments. In many cases, some of these crosspeaks will be perturbed by ligand binding, leading to identification of residues involved in active sites of enzymes or binding pockets of receptors. For ST6Gal1, the peak at the edge of our confidence limit, (peak 14 assigned to F356) is perturbed by addition of a reaction product, cytidine monophosphate (CMP) that is known to inhibit sialylation activity [55]. In other cases, a sufficient number of trusted peaks may be perturbed by paramagnetic moieties, to allow use as constraints in ligand docking or refinement of protein structure.

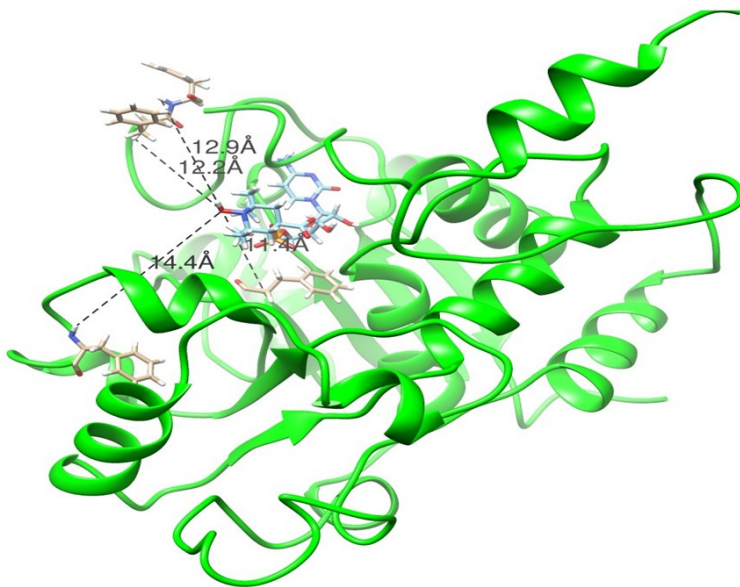


Figure 3.6: Model of rST6Gal1 with the donor analog, carboxy-TEMPO-CMP docked into the active site. Distances shown are those between the nitroxide oxygen of the TEMPO group and the amide protons of F240 (14.4), F357 (12.9), F208 (11.4) and F356 (12.2), respectively.

We can illustrate this latter case by using our previously published perturbations of cross-peaks by an analog of rST6Gal1’s sugar donor, sialylated cytidine monophosphate (NeuAc-CMP) [55]. The analog replaces the sialic acid with a carboxy-TEMPO group that retains the carboxyl group of sialic acid but replaces the six-membered ring of sialic acid with that of TEMPO. The TEMPO group carries a nitroxide oxygen with an unpaired electron distal from the phosphate ester connection to CMP. This oxygen is taken as the origin of paramagnetic perturbations. Prior estimates of distances between the oxygen and amide protons at sites associated with crosspeaks 6, 7, 10 and 14 were 14.3-15Å, 12.6-14.4Å, 10.6-12.9Å, and <17.0Å. The latter number is only an upper limit due to the low intensity of crosspeak 14 which is broadened significantly in the presence of CMP-TEMPO as well as CMP itself. To generate a model consistent with these distances, a structure taken from frame 100,000 of a rST6Gal1 trajectory (a stable point about 200 ns into the 1 s simulation) was superimposed with the structure of hST6Gal1 determined with the reaction product (CMP) in place (4SJ2). Then the CMP moiety of our donor analog was superimposed with the CMP of 4SJ2, and the torsions of the two phosphate ester bonds plus the phosphate oxygen to TEMPO bond were adjusted to place the nitroxide oxygen within the above distance limits without introducing van der Waals clashes. The resulting structure is shown in Figure 6. The distances between nitroxide oxygen and the amide protons of F240 assigned to peak 6, F357 assigned to peak 7, F208 assigned to peak 10, and F356 assigned to peak 14 are 14.4Å, 12.9Å, 11.4Å and 12.2Å, respectively. The structure is chemically reasonable and places the carboxylated carbon of the analog in a position where SN2 attack by the O6 oxygen of a galactose-containing acceptor can approach.

The use of an MD trajectory to improve prediction of NMR data has proven particularly useful. rST6Gal1 may not be representative of all proteins in the extent of improvement. It has a loop containing two of the labeled phenylalanines that is not visible in the rat crystal structure. This loop is near the active site and clearly undergoes motion as evidenced by motional broadening of the phenylalanine resonances in the presence of CMP [55]. However,



many enzymes share a tendency to have flexible regions as a part of their active site. Hence, the advantages shown for ST6Gal1 may apply to certain subsets of proteins having high internal mobilities.

As for future applications, increasing the fraction of confident assignments is clearly important. More precise experimental measurements and longer MD simulations will likely help. However, addition of other data types may be more important. We have already illustrated the impact of adding constraints from mutational studies. Other data types can also be added. Pseudo contact shifts (PCSs) share a functional form with RDCs [64] [36], as do PREs with NOEs, making addition straightforward. Applications to larger proteins are also of interest. Resolution of  $^1\text{H}$ - $^{15}\text{N}$  crosspeaks in the HSQC spectra shown here is certainly adequate to target proteins twice the size of rST6Gal1. However, sensitivity can be an issue. This will drop steeply for fully protonated glycoproteins of larger size. One encouraging prospect is the possibility of labeling with  $^{13}\text{C}$  methyl groups. Labeling all methyls in isoleucine, leucine and valine (ILV labeling) has provided a route to NMR characterization of large perdeuterated proteins expressed in bacterial cell cultures [39]. Assignment of methyl resonances in these instances presents challenges that parallel those for sparsely labeled glycoproteins. Alternative assignment strategies, similar in some respects to that described here, have been introduced recently [47], [74], [60]. Reliance on NOE data is one common aspect, but reduction in numbers of protonated sites by deuteration has allowed interpretation in terms of constraints on a very qualitative level. ASSIGN\_SLP\_MD is certainly applicable to data on ILV-labeled and perdeuterated proteins, and its use of MD trajectories to make interpretation of NOE data more quantitative may be particularly useful. A current limitation is the availability of appropriate crystal structures. This could be relaxed if an appropriate homology model could be selected. The minimum scores reached in making an assignment with ASSIGN\_SLP\_MD in many ways reflects the quality of the structural model used, and it may be possible to simultaneously obtain an assignment and select the best among several homology models. Comparison between predicted and measured chemical

shifts from  $^{13}\text{C}$ - $^{13}\text{C}$  correlation spectra acquired by solids NMR have already been used to screen homology models [21], and with addition of more data types this may be possible with sparsely labeled samples as well.

## 3.8 MATERIALS AND METHODS

### 3.8.1 PROTEIN EXPRESSION, MATAGENESIS, AND PURIFICATION

Protein sample preparations used in collection of RDC data and PRE data were analogous to those described in a previous publication [55]. New samples were prepared for the collection of NOE data and validation by mutagenesis using modified methods for expression, labeling, and purification as described in the literature [35], [57]. Briefly, expression constructs encoding the luminal domain of rat ST6Gal1 (UniProt P13721, residues 103 to 403) in the pGEn2 vector were transiently transfected into HEK293S (GnTI $^{-}$ ) cells [57] and metabolic labeling with  $^{15}\text{N}$ -Phe was initiated 16 h after transfection by exchange of the culture medium for custom FreeStyle 293 expression medium (Thermo Fisher Scientific) depleted in Phe and supplemented with 150 mg/L  $^{15}\text{N}$ -Phe 98% (Cambridge Isotope Laboratories, Andover, MA) and 2.2 mM valproic acid. The recombinant protein was harvested from the culture supernatant after 6 days of growth, purified by  $\text{Ni}^{2+}$ -NTA chromatography, and concentrated to  $\sim 1$  mg/mL. The resulting protein preparation was digested with recombinant TEV protease to cleave between ST6Gal1 and GFP, and recombinant endoglycosidase F1 (EndoF1) was used to cleave the glycans to single GlcNAc residues [57]. The preparation was then subjected to  $\text{Ni}^{2+}$ -NTA chromatography a second time to remove the GFP fusion tag, TEV protease, and EndoF1, each of which contain a His tag [57]. The samples were further purified by Superdex 75 chromatography (GE Healthcare Life Sciences) using a 20mM HEPES, pH 7.5, 250mM NaCl, and 60mM imidazole buffer. Peak fractions of ST6Gal1 were collected and concentrated to 20 mg/ml using an ultrafiltration pressure cell membrane. Exchange to NMR buffers (20mM Sodium Phosphate, pH 6.5, and 100mM NaCl for NOE

and mutational studies) was accomplished using Centricon centrifugal filtration units with a 10kDa cutoff. Site directed mutations of ST6Gal1 (F208Y, F357Y, and F357Y) were performed using the Q5 site-directed mutagenesis kit (New England Biolabs, Ipswich, MA) in the pGEn2-rST6Gal expression vector.

### 3.8.2 NMR DATA

One bond  $^{15}\text{N}$ - $^1\text{H}$  RDCs were measured using the interleaved fHSQC and fHSQC-TROSY experiments collected at 25 C on a Varian Inova 900 MHz spectrometer equipped with a cryogenic triple resonance probe. Data were collected over a 24 h period with acquisition times of 30 and 80 ms for  $t_1$  and  $t_2$ , respectively, and a 1.5 s recycle delay. NMR data were processed and analyzed using FELIX software. The rST6Gal1 samples were in 10 mM phosphate, 200 mM NaCl, pH 6.8, with 10%  $^2\text{H}_2\text{O}$ ; partial alignment was obtained using PEG (3% C12E5) and pf1 phage (10mg/mL) media as previously described [69], giving deuterium splittings of the water resonance of 13 and 21 Hz respectively. Protein concentrations were at 350 and 400 M for phage and PEG media respectively.

A 3D  $^{15}\text{N}$ -edited [ $^1\text{H}$ ,  $^1\text{H}$ ] NOESY-HSQC spectrum of a  $^{15}\text{N}$ -Phe labeled WT rST6Gal1 sample was recorded on an 800 MHz Bruker AVANCE NEO spectrometer equipped with a 5mm cryogenic triple-resonance probe. The NMR sample contained 270 ul of 630 uM  $^{15}\text{N}$ -Phe WT rST6Gal1, 4 uM DSS, 0.02% sodium azide and 10%  $^2\text{H}_2\text{O}$  in a Shigemi tube. NOE mixing time was set to 60 ms, and acquisition times  $t_{3,max}(^1\text{H})$ ,  $t_{2,max}(^1\text{H})$  and  $t_{3,max}(^{15}\text{N})$  were set to 46 ms, 10ms and 10ms, respectively. Total acquisition time was 40 h, with a 1.1s recycle delay. A 2D [ $^{15}\text{N}$ ,  $^1\text{H}$ ] HSQC was also recorded in 20 m with 1.0 s recycle delay and acquisition times  $t_{2,max}(^1\text{H})$  and  $t_{1,max}(^{15}\text{N})$  of 106 ms and 39 ms, respectively. Spectra were processed with TopSpin v3.5 (Bruker BioSpin) and analyzed with CARA v1.9.1.7.

Experimental NOE vectors were produced by averaging spectral intensity over an ellipse in the HSQC plane of NOE strip plots with dimensions 0.03 ppm ( $^1\text{H}$ ) and 0.65 ppm ( $^{15}\text{N}$ ).

Diagonal peaks in all vectors, as well as the H<sub>2</sub>O resonance (4.79 ppm) in vectors 1 and 3, were removed by setting intensity within 0.17 ppm of the corresponding signal to zero.

2D [<sup>15</sup>N, <sup>1</sup>H] HSQC spectra of single-point rST6Gal1 tyrosine mutant samples were recorded on the same 800 MHz Bruker AVANCE NEO spectrometer, but equipped with a 1.7 mm cryogenic triple-resonance probe. Samples consisted of 40 ul solutions of 330 uM F208Y, 580 uM F256Y, or 220 uM F357Y rST6Gal1 with 7.5 uM DSS and 0.09% sodium azide in 10% <sup>2</sup>H<sub>2</sub>O. Acquisition parameters were the same as for WT rST6Gal1, only the number of transients was adjusted.

**MD Simulation and Docking.** The starting point for the MD simulation was the 4MPS crystal structure; the missing 354-362 segment was modeled in using the corresponding segment from the 4JS1 structure and minimized. The simulation was then carried out using the PMEMD module of the AMBER 14 package [14]. The ff14SB force field was used for protein residues and the GLYCAM\_06j-1 force field [48] was used for the two GlcNAc residues attached to Asn residues at sites 146 and 158. A cubic box of TIP3 water extending a minimum of 8Å from the protein surface was used to solvate the protein. The system was first energy minimized by 50000 steps of minimization, then heated to 300 K in 2 fs steps over 1 ns. The 1 s MD simulation was carried out using 2 NVIDIA GeForce GTX TITAN Black GPUs on a 4 GPU laboratory computer and required about 2 weeks. For use in NOE simulations, frames of the trajectory were aligned by minimizing deviations of backbone  $\alpha$ -carbons using tools in cpptraj, an AMBER 14 utility [73]. For chemical shift predictions by PPM and SHIFTX2 every 200<sup>th</sup> frame was extracted and saved as a model in PDB format, again using tools in cpptraj [73]. Graphic depictions of structures and docking of ligands was preformed using tools in the Chimera package [68].

### 3.8.3 ASSIGN\_SLP MD PACKAGE.

The ASSIGN\_SLP\_MD package [15], as implemented in this study, contained several modules that prepared input for the search module, executed the search and assembled

output for presentation to the user. All are designed to operate under a LINUX operating system and execution with different input files can be facilitated by using bash scripts. Efforts are underway to integrate the separate modules of the program and develop a user interface. Up-to-date versions, as well as additional documentation, are available at <http://tesla.ccruc.uga.edu/software/>.

$^{15}\text{N}$  and  $^1\text{H}$  predicted chemical shifts for amide groups of the selected amino acid type were extracted from output of PPM [53] and SHIFTX2 [42] run on the PDB format trajectory by a MATLAB script called “Procedure for Spectra Generation”. The shifts were then appended to lists of experimental shifts, and a list of estimated error, as modified by weights, was added in separate input text files for  $^{15}\text{N}$  and  $^1\text{H}$  shifts.

NOE predicted peak lists for the 16 phenylalanine amide protons were prepared by a new version of the program, MD2NOE [16], written in C++ and called MD2NOE-Protein. Both predicted and experimental peak lists were converted to 512 point vectors containing gaussian lines of a user specified width (0.2 Hz in this study) at the predicted or experimental chemical shift of donating protons and intensity as specified in the peak lists, again using the MATLAB script, “Procedure for Spectra Generation”. Autopeaks were not included, but a pseudo autopeak of intensity equal to the maximum NOE peak intensity averaged over all 16 vectors was added at the end of each vector. These were output as spreadsheets in csv format. Experimental NOE columns from 3D- $^{15}\text{N}$ -edited NOESY-HSQC spectra can also be converted to vectors with gaussian-broadened peaks with the same program. As in the case of  $^{15}\text{N}$  and  $^1\text{H}$  shifts, weighted errors are added to the end of each vector.

A MATLAB script entitled “Order Parameters” was used to calculate order parameters and average  $^1\text{H}$ - $^{15}\text{N}$  bond vectors for use in predicting RDCs for each trial assignment within the search module. Output was in the form of a text file with a line for each residue containing the six coordinate entries for the bonded pair and an order parameter. Experimental RDCs were provided in separate text files for each medium with an ordered list of weighted errors following the RDCs. Weights added to account for information content were calculated

with separate MATLAB scripts for chemical shifts, RDCs, and NOEs; these generated distributions of scores by comparing each entry to every other entry and extracting a variance for the distribution, divided by the range of scores.

The search module, called ASSIGN\_SLP\_MD, is based on a genetic algorithm function call (ga) available as a part of the optimization toolbox of the MATLAB package. It reads in output from the various preparation modules and functions as described in our previous publication [34] except for the changes in score contributions to the objective function as described in the main text of this manuscript. Searches are repeated with 16 different combinations of mutation and crossover rates (2, 4, 6, 8 For each) to maximize the adequacy of the search. Every trial assignment with a score below a user specified level (raw score of 5 in the application presented) is saved in a text file along with the total score, and individual contributions from the various data types. Each search ceases when no improvement in score beyond the tolerance of  $1e-4$  is achieved or a maximum number of 500 iterations is reached.

The analysis module retrieves the output of the genetic algorithm search, orders the output by total score and eliminates all duplicates. A distribution of randomly generated scores is then calculated so that a mean and variance can be extracted, and Z-scores appended to each assignment in the ordered list. Heatmaps are then generated by considering the fraction of times the same crosspeak is assigned to the same residue within a set of all assignments having a score below a user specified limit. In the example presented, the limit selected to be an unweighted raw score equal to the number of experimental data types (5 in our example) or one unit above the minimum unweighted raw score when the this was greater than the number of data types; this resulted in inclusion of 1000 to 10,000 solutions in the sets discussed here.

#### 3.8.4 DATA SUMMARY

Data used in the application to rST6Gal1 are summarized in the following tables.

Table 3.1: Non-NOE data used in rST6Gal1<sup>15</sup>N-phenylalanine assignments.

| Data               | Crosspeaks |     |     |     |     |     |     |     |     |
|--------------------|------------|-----|-----|-----|-----|-----|-----|-----|-----|
|                    | 1          | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
| <sup>1</sup> H cs  | 9.7        | 9.3 | 8.7 | 8.8 | 8.0 | 8.1 | 8.5 | 8.4 | 8.2 |
| error              | 0.4        | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| <sup>15</sup> N cs | 128        | 126 | 128 | 124 | 123 | 120 | 121 | 117 | 115 |
| error              | 4          | 4   | 4   | 4   | 4   | 4   | 4   | 4   | 4   |
| RDCpeg             | -16        | 7   | -5  | 2   | -13 | 999 | 2   | 29  | -1  |
| error              | 10         | 10  | 10  | 10  | 10  |     | 20  | 10  | 20  |
| RDCpfl             | 0          | -6  | -18 | -9  | -6  | -27 | -4  | 21  | -11 |
| error              | 5          | 5   | 5   | 5   | 5   | 5   | 5   | 5   | 5   |

Table 3.2: Non-NOE data used in rST6Gal1<sup>15</sup>N-phenylalanine assignments.

| Data               | Crosspeaks |     |     |     |     |     |     | Wt <sup>a</sup> |
|--------------------|------------|-----|-----|-----|-----|-----|-----|-----------------|
|                    | 10         | 11  | 12  | 13  | 14  | 15  | 16  |                 |
| <sup>1</sup> H cs  | 8.1        | 6.4 | 6.9 | 7.4 | 7.5 | 7.0 | 7.1 | 0.35            |
| error              | 0.4        | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |                 |
| <sup>15</sup> N cs | 114        | 124 | 121 | 121 | 117 | 114 | 114 | 0.55            |
| error              | 4          | 4   | 4   | 4   | 4   | 4   | 4   |                 |
| RDCpeg             | 999        | 999 | -23 | -2  | 28  | 999 | 9   | 0.42            |
| error              |            |     | 10  | 10  | 20  |     | 10  |                 |
| RDCpfl             | 999        | 30  | -2  | -9  | 18  | -19 | 999 | 0.37            |
| error              |            | 5   | 5   | 5   | 5   | 5   |     |                 |

<sup>a</sup> Weights (Wt) include an estimate of information content (variance/range<sup>2</sup>) and a penalty for missing data (#data/#sites for chemical shifts and ((#data-5)/#sites for RDCs). Errors for chemical shifts are 2x ShiftX2 estimates; RDC errors are approximately 20% of line widths.

Table 3.3: NOE peak list data used in rST6Gal1<sup>15</sup>N-phenylalanine assignments.<sup>a</sup>

| Data      | Crosspeaks |     |     |     |     |     |     |     |     |
|-----------|------------|-----|-----|-----|-----|-----|-----|-----|-----|
|           | 1          | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
| 1H cs     | 2.4        | 2.8 | 2.5 | 1.5 | 2.1 | 0.8 | 3.0 | 1.8 | 1.6 |
| intensity | 17         | 21  | 18  | 18  | 18  | 12  | 22  | 12  | 17  |
| 1H cs     | 2.8        | 3.1 | 2.9 | 1.7 | 2.3 | 2.7 | 4.7 | 2.4 | 3.0 |
| intensity | 13         | 16  | 42  | 50  | 12  | 14  | 25  | 24  | 16  |
| 1H cs     | 3.7        | 4.7 | 3.3 | 2.8 | 2.9 | 4.5 |     | 3.5 | 3.2 |
| intensity | 13         | 10  | 41  | 25  | 23  | 12  |     | 11  | 12  |
| 1H cs     |            | 6.1 | 3.6 | 3.3 | 3.0 | 7.7 |     | 4.2 | 3.6 |
| intensity |            | 28  | 10  | 18  | 14  | 12  |     | 39  | 14  |
| 1H cs     |            | 8.0 | 7.4 | 4.4 | 4.1 |     |     | 5.1 | 3.8 |
| intensity |            | 22  | 30  | 120 | 20  |     |     | 20  | 23  |
| 1H cs     |            |     |     | 4.4 | 4.3 |     |     | 7.3 | 4.6 |
| intensity |            |     |     | 120 | 12  |     |     | 22  | 35  |
| 1H cs     |            |     |     | 4.8 | 6.3 |     |     | 8.1 | 7.0 |
| intensity |            |     |     | 30  | 12  |     |     | 26  | 19  |
| 1H cs     |            |     |     | 7.1 | 7.8 |     |     | 9.4 | 8.4 |
| intensity |            |     |     | 19  | 11  |     |     | 40  | 24  |
| 1H cs     |            |     |     | 8.3 | 9.5 |     |     |     |     |
| intensity |            |     |     | 16  | 12  |     |     |     |     |

<sup>a</sup> NOE vectors used were a sum of gaussian peaks of width 0.4 ppm placed at chemical shifts and having intensities taken from the vectors emanating from crosspeaks in NOESY-HSQC spectra. Diagonal peaks and water peaks were removed and a peak of intensity equal to the average of the maximum peak in each vector was added to the end of experimental and predicted vectors to retain intensity sensitivity in R-factor calculations. Only points above 2 x noise = 5 are listed. Error was estimated comparing peak 10 to a vector having only an autpeak: (1-R) = 0.14. The NOE weight was 0.5.



Table 3.4: NOE peak list data used in rST6Gal1<sup>15</sup>N-phenylalanine assignments.<sup>a</sup>

| Data      | Crosspeaks |     |     |     |     |     |     |
|-----------|------------|-----|-----|-----|-----|-----|-----|
|           | 10         | 11  | 12  | 13  | 14  | 15  | 16  |
| 1H cs     | 9.4        | 1.1 | 2.1 | 4.0 | 1.6 | 4.0 | 3.2 |
| intensity | 4          | 12  | 10  | 23  | 16  | 13  | 11  |
| 1H cs     |            | 1.6 | 2.0 | 6.8 | 3.0 | 8.3 | 4.2 |
| intensity |            | 12  | 43  | 14  | 21  | 16  | 10  |
| 1H cs     |            | 3.4 | 3.2 |     | 3.2 |     |     |
| intensity |            | 10  | 30  | 15  |     |     |     |
| 1H cs     |            | 6.1 | 3.6 | 3.3 | 3.0 | 7.7 |     |
| intensity |            | 47  | 55  |     | 17  |     |     |
| 1H cs     |            | 6.1 | 4.6 |     | 4.3 |     |     |
| intensity |            | 10  | 20  |     | 22  |     |     |
| 1H cs     |            |     | 6.6 |     | 4.7 |     |     |
| intensity |            |     | 43  |     | 25  |     |     |
| 1H cs     |            |     | 7.0 |     | 7.7 |     |     |
| intensity | 19         |     |     | 17  |     | 18  |     |
| 1H cs     |            |     | 7.4 |     |     |     |     |
| intensity |            |     | 44  |     |     |     |     |
| 1H cs     |            |     | 8.9 |     |     |     |     |
| intensity |            |     | 32  |     |     |     |     |

### 3.9 ASSIGN SLP SUPPLEMENT

In this supplement the assignment results from the genetic algorithm are shown for six proteins. The first five proteins have already been reliably assigned and the assignments can be found in the pdb downloads from the protein data bank. The genetic algorithm assignment of these first five proteins agreed mostly with the previously known manual assignments [33]. These assignments were used as a validity test of the progressing genetic algorithm assignment software. In addition, this was the first work in which the algorithm was used to assign proteins, and the output solutions of assignment were studied for better interpretation of the output and improvements in the software. Better interpretation means better understanding of how to find the most reliable assignment from a set of possible assignments and statistical interpretation.

A sixth protein, the large 3 domain 145 kD dimer, has not been previously assigned (for reference, the previous four proteins are  $\leq 10$  kDa). The genetic algorithm software was used to assign this [15]. The software is designed to be used with bad or missing measurements, and one of the domains of HtpG has many missing. However, a partial assignment of the protein was obtained. This assignment demonstrates what a reliable assignment of measurements can be used for; it was used to find regions of structural change between the apo and AMPNP forms of HtpG. Description of this assignment tests has been published [66] and much of the text that follows is derived from those publications.

The description of the software package AssignSLP is not described in this supplement; only the results for these six protein assignments are given.

#### **Initial five proteins: 3C4S, 3CWI, 3LMO, 3FIA, ROBO1**

Four test proteins were chosen from the 40 pairs of NMR-X-ray structures produced by the Northeast Structural Genomics group, imposing the additional requirements that the resolution of the X-ray structures are below 2 Å and that NOE peak lists with crosspeak

intensities are available [31]. The NMR data for Robo1-Ig1-2 are from the work reporting its interaction with heparan sulfate [37]. There are several X-ray structures for the Ig1-2 construct, but these show significant differences in inter-domain orientation. For the purpose of this application domain motions were simulated in a long MD trajectory (1 microsecond) [37] and an x-ray structure (PDB 2V9R) selected that closely approximated the domain orientation in the most highly populated state of this trajectory.

Assignments for our four uniformly labeled test proteins and one glycoprotein have been produced using the programs introduced above. The four uniformly labeled test proteins range in size from 55 to 212 amino acids. Different mixes of secondary structures are represented, including those rich in alpha-helix, rich in beta-sheet and a combination of both. There are instances of missing data and different levels of internal motion. For the two domain construct from Robo1, an example of a sparsely labeled glycoprotein,  $^{15}\text{N}$ - $^1\text{H}$  HSQC spectra for the lysine and phenylalanine labeled versions are shown in Supplementary Materials Figure S1 [37]. These spectra give examples of the resolution that can be expected for a sparsely labeled, non-deuterated 23 kDa protein.

Working with the first four proteins, for which assignments are well documented by traditional methods, provides an opportunity to evaluate the degree to which each measurement type contributes to the assignment process. Their contributions can be visualized in heatmaps similar to those generated by one of the auxiliary analysis scripts. The examples shown in Figure 2 use the X-ray structure, 2K5P, for prediction and the deposited information for the NMR structure, 3CWI, for experimental data. Experimental assignments are listed on the y axis and predicted assignments are listed on the x axis, both ordered with respect to increasing residue numbers. Correct assignments fall on the diagonal. The contributions to the total score from each data type have been generated using an in-house MATLAB script (available at the ASSIGN\_SLP download site). The values are represented on the plots in gray-scale, with black representing zero (best score) and white a normalized score of 1. The amino acids represented are 7 alanines and 9 valines. Since we do not allow cross-assignments

between the amino acid types, white regions exist for coordinates 1-7, 8-16 and 8-16, 1-7. The first 3 panels are heatmaps of the scores for individual data types; NOEs, chemical shifts, and RDCs. From these heatmaps, it is clear that NOEs are the most informative since the darkest spots for most possible assignments fall on the diagonal. However, it is also obvious that there are cases with little distinction between pairs of possible assignments (scores for peaks 1, 6, and 13), and an incorrect assignment would be indicated for peaks 9 and 12. Data for RDCs and chemical shifts are typically less definitive, but still useful. Adding all the scores together produces a plot in which the diagonal box is darkest for all but one possible site. The heatmaps have already been used to extract an error estimate for NOEs, but they could be used to evaluate the proper weighting for all data types. We will examine this possibility in the future.

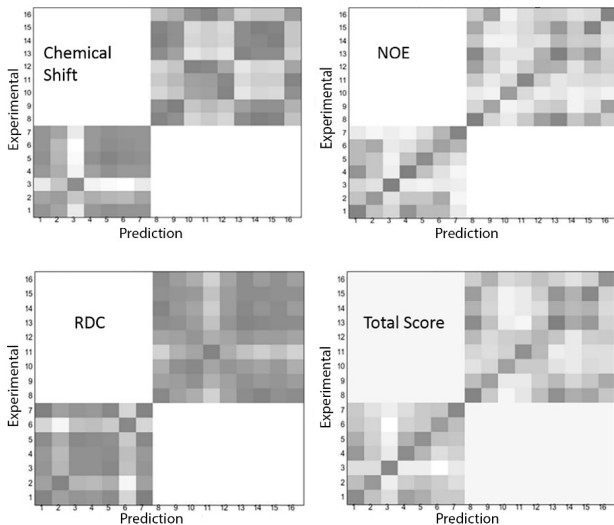


Figure 3.7: Heatmaps comparing predicted and experimental values of each type of measurement (chemical shift, NOEs and RDCs) and total score contribution. Each number on both X and Y axes represent one labeled residue. The amino acid type is assumed known for the two sets of crosspeaks.

An example of the output of our assignment program for the 3CWI-2K5P protein is shown in Table 2. The output contains not only all the possible assignments but also the solution rank and the score contributions from each type of measurement. There is a comparison of

experimental and predicted RDC data for each site. For chemical shifts the experimental data are given for each site; the predicted data are given in the output header. For NOEs individual score contributions in terms of  $(1-R)^2$  are given. In the example presented, the first rank solution is a single-swap of two residue assignments (peak 3 should assigned to 48 and peak 4 should assigned to 32); peak 3 has no RDC data, making assignments for this pair somewhat ambiguous.

Table 3.5: Top ranked solution for assignments of 3CWI-2K5P.

|                |       |       |       |       |       |       |       |        |
|----------------|-------|-------|-------|-------|-------|-------|-------|--------|
| Solution       |       |       |       |       |       |       |       |        |
| Rank 1         |       |       |       |       |       |       |       |        |
| Peak number    | 2     | 5     | 1     | 3     | 4     | 7     | 6     | 12     |
| Residue number | 15    | 26    | 30    | 32    | 48    | 51    | 59    | 5      |
| Exp.RDC        | 2.69  | 7.24  | 0.96  | 999   | 0.74  | -9.87 | 3.33  | -3.24  |
| Calculated RDC | 3.04  | 8.41  | -0.14 | 0     | 0.42  | -8.94 | 1.1   | -1.92  |
| Exp. shift (N) | 121.2 | 126.1 | 131.1 | 120.2 | 119.9 | 119.5 | 121.1 | 226    |
| Exp. shift (H) | 7.36  | 8.71  | 8.54  | 8.01  | 8.03  | 7.54  | 8.33  | 108.56 |
| NOE score      | 0.01  | 0.2   | 0     | 0.11  | 0.23  | 0.1   | 0.01  | .05    |

Table 3.6: Next top ranked solution for assignments of 3CWI-2K5P.

|                |         |         |         |         |           |         |         |         |
|----------------|---------|---------|---------|---------|-----------|---------|---------|---------|
| Solution       |         |         |         |         |           |         |         |         |
| Rank 1         |         |         |         |         |           |         |         |         |
| Peak number    | 15      | 9       | 14      | 11      | 8         | 13      | 16      | 10      |
| Residue number | 12      | 20      | 29      | 35      | 37        | 43      | 54      | 60      |
| Exp.RDC        | 999     | 3.2     | 9.5     | 999     | -0.17     | 999     | -1.28   | 999     |
| Calculated RDC | 0       | 3.5     | 9.6     | 0       | 0.15      | 0       | 0.11    | 0       |
| Exp. shift (N) | 225*    | 217.9*  | 219.2*  | 221.2*  | 227.6*    | 226.8*  | 226.2*  | 221.8*  |
| Exp. shift (H) | 108.56* | 107.81* | 107.58* | 107.33* | 109.62*   | 108.87* | 109.22* | 109.06* |
| NOE score      | 0.05    | 0.15    | 0.03    | 0.2     | 0         | 0.43    | 0       | 0       |
| Data type      | RDC     | N       | H       | NOE     | Sum/Score |         |         |         |
| Total score    | 1.48    | 0.99    | 1.19    | 1.52    | 5.19      |         |         |         |

\*100 is automatically added to the chemical shift for the second type of amino acid so that different types of amino acid will not be cross-assigned. 999 is used to indicate data that are not available. The incorrect assignment is colored in gray; 3 and 4 should be interchanged.

\*100 is automatically added to the chemical shift for the second type of amino acid so that different types of amino acid will not be cross-assigned. 999 is used to indicate data that are not available. The incorrect assignment is colored in gray; 3 and 4 should be interchanged.

The results of application of our assignment program to all four uniformly labeled test cases are summarized in Table 3. In all cases  $^1\text{H}$  and  $^{15}\text{N}$  chemical shifts, NOE peak lists for HSQC crosspeaks, and a single set of RDCs were available. The top score solutions contain at least 70% of correct assignments (case 3FIA) and can reach 100% of correct assignments (case 3C4S). The correct solution is always found near the top of the list; the worst case is number 11 out of 7493 solutions for 3FIA which has 6 missing RDCs.’

Table 3.7: Assignment summary of four test protein cases.

| PDB  | 3C4S         | 3CWI         | 3LMO          | 3FIA         |
|--|--------------|--------------|---------------|--------------|
| Labeled Sites and Number                         | Ala 4, Val 8 | Ala 7, Val 9 | Ala 12, Lys 6 | Ala 8, Lys 6 |
| Number of Acceptable Solutions*                  | 260          | 1376         | 14006         | 7493         |
| Top Score Solution (correct/total)               | 12/12        | 14/16        | 16/18         | 10/14        |
| Correct Solution Rank                            | 1            | 4            | 2             | 11           |
| Consistently Assigned Crosspeaks (correct/total) | 7/12         | 10/16        | 15/18         | 10/14        |
| Missing Data                                     | 0            | 5 RDCs       | 1 RDC         | 6 RDCs       |

\*If the lowest score for an application is below 4, all the solutions with a score under 5 are collected. If the lowest score is above four, all the solutions with a score more than the lowest score plus 1.0 are collected.

The application to Robo1-Ig1-2 deserves a separate discussion. Robo1-Ig1-2 is both larger than the other test proteins, (212 residues), it has the potential complication of internal motion between domains, and it is a glycoprotein where sparse labeling with individual amino acids is necessary. The top ranked assignment from an initial run (having a score of 4.09) contains 10 correct assignments and the completely correct solution was solution number 366. However, the Robo1 protein is a good example of using some intelligence in changing the weights of the contributions in the objective function to improve performance. The

calculation using initial error estimates had high chemical shift contributions to the scores and several of the RDC's did not agree with the back calculation of individual contributions. By increasing the errors of the chemical shift terms to lessen their importance in the objective function, the correct solution moved from a rank of 366 to a rank of 18 in a list of 354 acceptable solutions with scores less than 5.09. The top ranked solution still had 10 correct assignments.

For Robo1-Ig1-2 it is possible to see some of the reason for the four missed assignments in the top ranked solution. The RDC degeneracy makes it hard to distinguish peak 4 from 9, peak 6 from 7 and peak 13 from 14. Therefore, swaps between assignments for these pairs might have been expected. 10 correct out of 14 is in fact not a bad result. We might also have expected the RDC data to be compromised in the Robo1-Ig1-2 case by the existence of inter-domain motion. This could have led to different alignment tensors for the two domains and completely incorrect RDC predictions when assuming a rigid structure and extracting a single set of alignment parameters. The fact that RDCs fit reasonably well may mean that motions are fairly restricted in the presence of the large attached glycan. The crystal structures showing large variations in inter-domain geometry were all produced on non-glycosylated material.

It may seem convenient to focus on top-ranked solutions, however, this is not particularly valuable for a protein for which there is not prior knowledge of the correct assignment. If we were to assume the top ranked solution in each case to be correct, we would have assigned 14 of the 74 sites in these five proteins incorrectly (19%), and we would not have known which of the 74 assignments was incorrect. Identifying sites which are assigned with high confidence is actually more important than obtaining a complete assignment. For example, in applications to ligand binding by chemical shift perturbation, one only needs to know the assignment of the perturbed peak, and for domain orientation using RDC measurements, one only needs an adequate number of confident assignments to use RDCs.

One approach to assessing the probability of a correct assignment is to look at the frequency with which a crosspeak is assigned to the same site in solutions which fall within a standard deviation or so of satisfying experimental data. Since we have tried to scale scores relative to estimated error for each data type, the cut-off for solutions to examine should be roughly equal to the number of data types used. Our test cases had 4 data types. We would expect to see a significant number of solutions with scores below 4. Two of the test proteins fall in this class, 3CWI had a best solution score of 3.66 and 3FIA had a best solution score of 3.87. The other 3 had best scores of 4.45, 5.38, and 4.09. The higher scores could represent an underestimate of error, a systematic deviation in some data due to internal motion, or minor differences in structure between solution and crystal. To get an adequate sampling of solutions we will examine solutions with scores less than 5.0 if the minimum score is less than 4 and one plus the minimum when the minimum score is larger or equal to 4. We consider these to be acceptable assignments.

A visual way of presenting this analysis is in the form of a histogram. Figure 3 uses protein 3CWI as an example. Other examples are contained in Supplemental Materials Figure S2. Histograms show the number of times a crosspeak is assigned to each site. If we take consistency to be assignment of the same residue to a given crosspeak in more than 50% of the acceptable assignments, we find the following: of the 60 assignments which we can compare to the results of traditional triple resonance assignments, we would assign with confidence 35 peaks or about 60% of them. We find that among these 35 we would make one mistake. This would correspond to being correct 97% of the time, something close to a 95% confidence limit. The Robo1 system is a little different because we have good reason to believe that the structural model may be inadequate. Nevertheless, applying the same criteria we find that we can assign 7 of the 14 peaks with confidence and all of these agree with our manual assignment.

## HtpG



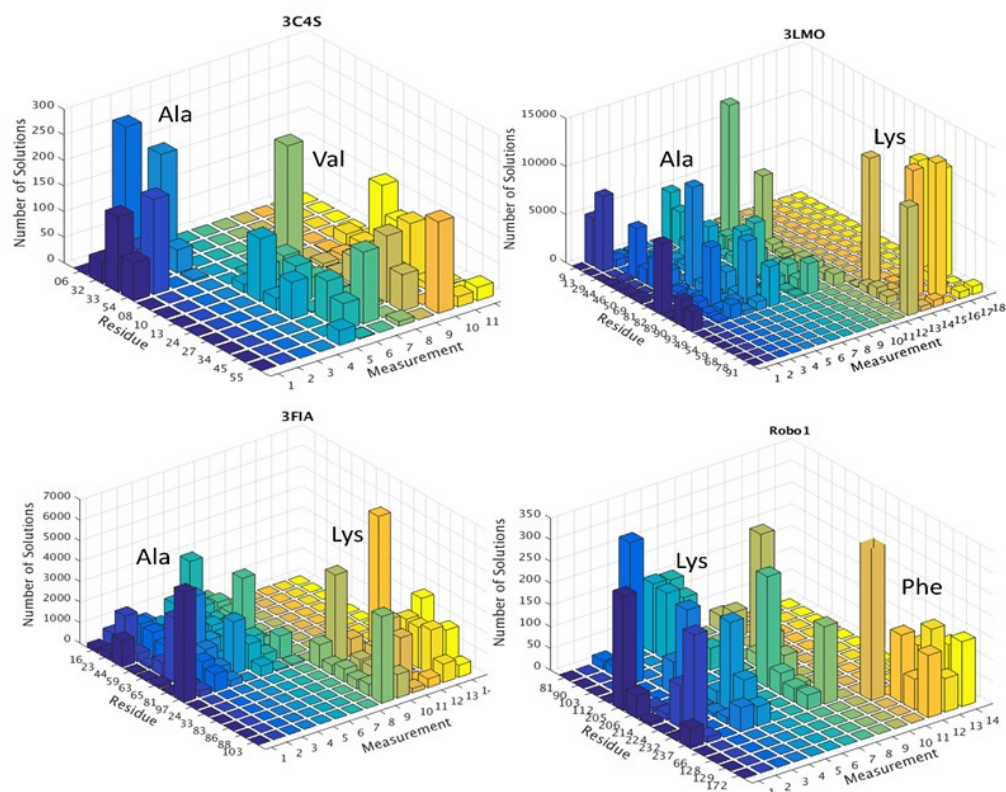


Figure 3.8: Histogram showing the frequency with which each crosspeak (measurement) is assigned to each site (residue) for the protein with NMR structure 3CWI.

In this part of the supplement the results of using the genetic algorithm assignment software is shown in the case of a large 3 domain protein, HtpG. In addition to explaining the assignment, the use of a reliable assignment is shown in applications. This demonstrates that the genetic algorithm assignment for a large protein, which would be hard to do manually, is useful for finding applications.

**Assignments of sparsely-labeled HtpG.** The program ASSIGN\_SLP\_1.1.2, which is based on a genetic algorithm search for the assignments best matching measured to predicted NMR data (chemical shifts, RDCs, NOEs), was modified to achieve assignments of full length HtpG. This modified version is available at the website: <http://tesla.ccrc.uga.edu/software/>. The numbers of observable NOEs are naturally reduced in a perdeuterated protein and few NOEs between the well-dispersed  $^{13}\text{C}$ - $^1\text{H}$ -labeled alanine methyls were observed. Hence, only RDCs and chemical shifts were used. The lack of NOEs, combined with the increased protein size, made it necessary to use data coming from partial assignments of crosspeaks from individual domains. Normally one would expect to transfer assignments based on an exact match of crosspeak positions in domain and full-length spectra, but because of changes in chemical shifts that arise from domain-domain interactions and pH differences, overlap of crosspeaks is not exact. Therefore, a list of possible matches was generated by considering all assigned domain crosspeaks within a generous chemical shift radius of a full length crosspeak (0.12 ppm  $^1\text{H}$  shift, 1.2 ppm  $^{13}\text{C}$  shift). These lists were transformed to a user input constraint matrix in which a zero indicated an acceptable assignment and a one indicated an unacceptable assignment. Penalties were assigned based on the occurrence of assignments carrying a one or zero (~10 for ones and 0 for zeros) and added to an overall assignment score. As in the original description of the program [29], score contributions for agreement of measured and predicted chemical shifts (calculated using the program PPM.ONE) were represented as root-mean-square-deviations (RMSDs), normalized to 1 for deviations equal to estimated errors. Similarly, score contributions for RDCs came from RMSDs of measured versus predicted values, normalized and adjusted for information content. The predicted

RDCs, however, must be recalculated for each assignment. Therefore, procedures paralleling those in the REDCAT program [78] are, incorporated directly in ASSIGN\_SLP\_1.1.2. To facilitate use of the program, raw RDC values were corrected for methyl rotation and use of a  $C\alpha$  to  $C\beta$  vector as opposed to a C-H vector in the back-calculation.

Assignments were done domain by domain. Because there were typically multiple full length crosspeaks associated with each domain crosspeak, it was possible to have more crosspeaks than domain sites in an assignment task. Because crosspeaks are sometimes missing due to motional broadening or overlap, it was also possible to have more alanine sites than crosspeaks in certain assignment tasks. Our implementation of the genetic algorithm requires an equal number of sites and crosspeaks. Therefore, data for extra sites or crosspeaks were designated with 999, as well as in the case of missing data, and contributions to scores were omitted whenever a 999 occurred.

ASSIGN\_SLP\_1.1.2 outputs a list possible assignments that include those with a total score less than a user-entered cut-off. It is recommended that this be set to approximately 1.5 times the number of data types (in our case there are four data types, two chemical shifts ( $^1\text{H}$  and  $^{13}\text{C}$ ) and two types of RDCs). Because the normalized scores would be one at the limit of estimated error for each data type, a score of 6 would correspond to solutions with all observables deviating from predictions by approximately 1.5 times standard error. The completely correct assignment is nearly always in this output, but it is not necessarily the one with the top score. However, interest is really in which sites can be assigned to a particular crosspeak with high confidence as opposed to identifying a completely correct of assignment. Therefore, we have devised a criterion for selecting these high confidence sites [79]. Based on test cases with known assignments, a site that is assigned to the same crosspeak more than 50% of the time in the list using an appropriate cut-off, is an assignment made with high confidence ( $\sim 95\%$  confidence limit). Additional sites, with a particular assignment simply being the most frequent (usually 2 times the next most frequent), are considered to be ones made with moderate confidence.

**Homology modeling of HtpG-AMPNP.** A homology model for the HtpG dimer in the presence of AMPNP was constructed using the UCSF Chimera program [80]. The zebra fish mitochondrial Hsp90, Trap1, (PDB code 4IPE) was selected as a template [51]. The 36% identical sequences were aligned using Clustal Omega [81] using default parameters as in the Chimera interface. Modeling was done via the web version of MODELLER [82], again using default parameters as provided in the Chimera interface.

**Modeling of the solution apo-HtpG structure.** Using RDC data from each domain, the program REDCAT [83] was used to extract a set of principal order parameters and Euler angles that relate the principal alignment frame to the original coordinate frame. In each case, the highest resolution domain structure for the apo form was used (2IOQ, 2GQ0 and 1SF8 for the NTD, MD and CTD, respectively). The Euler angels were used to rotate each domain into its principal alignment frame and the domains were assembled by translating the domains (and their three 180 rotational equivalents) to find the best option for covalent linkage.

**Assignment of HtpG.** Figure 3 shows a Methyl-TROSY spectrum of full length HtpG  $^{13}\text{C}$ - $^1\text{H}$  labeled in all alanine methyl groups. The sensitivity is quite high for a system of this size ( $\sim 145$  kDa as a dimer). The spectrum was acquired in approximately 1 hr on a 100 L 216 M sample. There are approximately 40 resolvable crosspeaks in the region where alanine methyl crosspeaks fall. This is 85% of the possible 47 peaks expected based on the expression construct. While it is possible to have alanine methyls scramble to valine and leucine [84], under conditions of our expression protocol the level of scrambling appears to be small. The appearance of crosspeaks from individual domains assigned to alanine methyls in similar regions, and in similar numbers, supports this contention. RDCs were collected for each of the resolved peaks. There were a few cases where the fit to a single modulated and decaying exponential was unacceptable due to low signal to noise or possibly multiple modulations in the case of overlapping peaks. In all, 30 and 37 RDCs were found acceptable for the phage and peg alignments, respectively. These, along with errors estimated from fitting, are

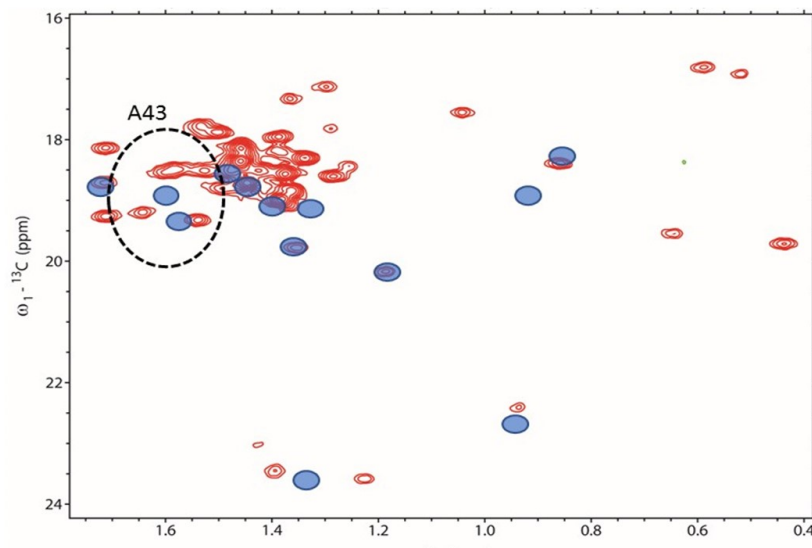


Figure 3.9: Methyl-TROSY spectrum of  $^{13}\text{C}$ - $^1\text{H}$ -alanine labeled perdeuterated HtpG. Superimposed on are blue circles representing chemical shifts of crosspeaks of the separately expressed N-terminal domain. The dotted ellipse centered at the single-domain shifts of A43 (radii 1.2 and 0.12 ppm) encloses 4 crosspeaks judged to be possible A43 crosspeaks in full length HtpG.

included in Supplemental Table 1. The chemical shifts of crosspeaks and the  $^{13}\text{C}$ - $^1\text{H}$  methyl RDCs were compared to predictions from PPM\_ONE [87] and REDCAT, respectively, using the highest-resolution crystal structure available for each HtpG domain, i.e., 2IOR for NTD, 1SF8 for CTD and 2GQ0 for MD [30] [89].

We started the search for assignments using the domain with the highest number of triple resonance assignments (NTD). Also shown in Figure 3 are symbols at the chemical shifts of alanine methyls taken from HNCACB and  $^{13}\text{C}$ -HSQC spectra of the isolated NTD. An ellipse is drawn around the point corresponding to A43 at a radii corresponding to estimated uncertainties in position due to separation of domains and differences in pH (0.12 ppm in the proton dimension and 1.2 ppm in the  $^{15}\text{N}$  dimension). This shows that four crosspeaks seen in the full-length protein can potentially be assigned to the methyl resonance of residue A43.

A similar analysis was done for each of the other Ala methyl resonances assigned in the NT domain. In some cases the number of possible crosspeaks is reduced from the number found in the ellipse due to unique assignments of a crosspeak to other domains. Table 1 shows the final list of crosspeaks in the full-length protein that could possibly be assigned to each of the residues in the NTD. It also shows the isolated domain chemical shifts, final assignments and a confidence estimate. Initially, 5 residues were uniquely assigned to a crosspeak and the others had degeneracies ranging from 2 to 7. The program ASSIGN\_SLP\_1.1.2 was then used to search for the best assignment of the 13 sites in the N terminus to the 22 crosspeaks appearing in the various lists of degeneracies. Using the program twelve crosspeaks were assigned with high confidence and one with moderate confidence.

Table 3.8: Crosspeak Assignments for the NTD of full-length apo-HtpG.

| Possible crosspeaks  | Residue | $^{13}\text{C}$ shift | $^1\text{H}$ shift | Assignment | Confidence |
|----------------------|---------|-----------------------|--------------------|------------|------------|
| 14,15,20,21,24,28,29 | 39      | 18.7                  | 1.45               | 20         | moderate   |
| 16                   | 42      | 18.9                  | 1.74               | 16         | high       |
| 9,11,19,22           | 43      | 19.1                  | 1.63               | 19         | high       |
| 26,35                | 50      | 18.4                  | 0.86               | 26         | high       |
| 12,13,18             | 98      | 19.2                  | 1.36               | 13         | high       |
| 14,15,20,21,24,27,28 | 114     | 18.9                  | 1.43               | 24         | high       |
| 26,35                | 130     | 18.9                  | 0.93               | 35         | high       |
| 4                    | 134     | 23.0                  | 0.95               | 4          | high       |
| 5                    | 143     | 20.3                  | 1.21               | 5          | high       |
| 6                    | 144     | 19.9                  | 1.37               | 6          | high       |
| 9,11                 | 157     | 19.5                  | 1.60               | 11         | high       |
| 2                    | 165     | 23.8                  | 1.38               | 2          | high       |
| 12,13,15,18,21       | 205     | 19.2                  | 1.40               | 12         | high       |

We then attempted assignments for the next most highly assigned domain, the CTD. Examining overlap of crosspeaks between the isolated domain and the full length protein in a manner similar to that described above, only 2 sites could be uniquely assigned. The other 10 sites had numbers of possible cross peaks ranging from 2 to 7. Three of the crosspeaks in the list of possible assignments were then eliminated based on their high confidence assignment

to sites in the NT. Application of the program resulted in 6 additional high confidence assignments of crosspeaks to sites in the CTD and four moderate confidence assignments.

The middle domain had only one definitive assignment, and the remaining crosspeaks had lists of 1 to 7 crosspeaks associated with each of the remaining twelve sites. ASSIGN\_SLP\_1.1.2 yielded six high confidence assignments and four probable assignments. The complete list of  $^{13}\text{C}$ - $^1\text{H}$ -methyl assignments for single domain and full-length proteins is included in Supplemental Table 1. While the assignments are far from complete, they allow identification of some residues undergoing shifts on conversion from apo to AMPPNP forms of HtpG and a limited analysis of the changes in domain-domain orientations on this conversion.

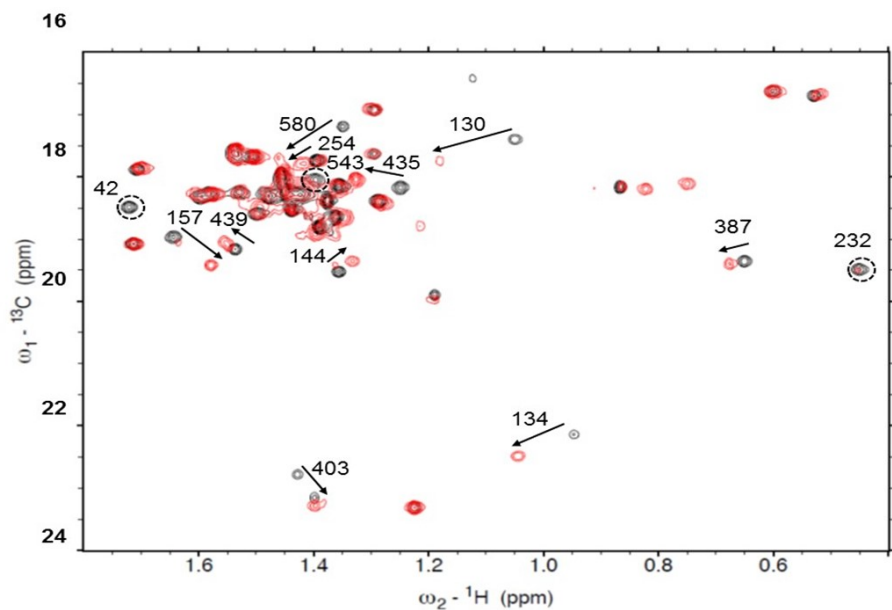


Figure 3.10: Crosspeak shifts on adding AMPPNP to apo-HtpG. Arrows show shifts, circles show peaks disappearing.

**Chemical shift perturbations on AMPPNP addition.** Figure 4 shows a superposition of Methyl-TROSY spectra of apo and AMPPNP forms of HtpG. AMPPNP was added at 5 mM concentration and heated at 37 C for 1 hour to assure complete conversion to the nucleotide bound form [49]. Assigned peaks that shift or disappear are labeled with residue

numbers. It is useful to look at these residues in the context of where they lie in the respective structures. There is no structure of the AMPPNP form of HtpG. However, there is a crystal structure of the AMPPNP form of the mitochondrial Hsp90, Trap1 [51], and SAXS data suggest that the overall conformations of AMPPNP forms of Trap1 and HtpG are similar. We have made a homology model of the AMPPNP form of HtpG using the Trap1 structure as a template.

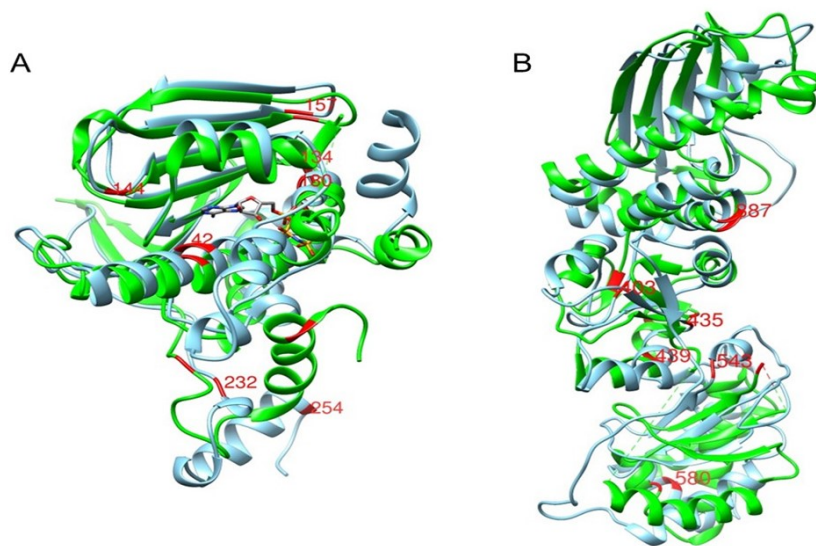


Figure 3.11: Superimposed ribbon structures of the N-terminal (A) and middle plus C-terminal (B) domains of apo (green) and AMPPNP (blue) forms of HtpG.  $^{13}\text{C}$ - $^1\text{H}$ -labeled alanines with resonances that differ in chemical shift are shown in red.

Figure 5A shows ribbon diagrams for the superposition of the NTDs for the crystal structure of the apo form (2IOQ) and for the homology model. The perturbed alanine residues are colored in red. There are significant structural differences throughout this domain, several perturbed residues are close to the nucleotide binding site (residues A42, A130, A134). A232 and A254 are in the linker between the NTD and the MD, where major differences in the modeled structure are seen.



Figure 5B shows ribbon diagrams for a superposition of the MD and CTD with perturbed alanines in red. A435, A439 and A543 are close to the junction between the MD and CTD where substantial structural differences are predicted. A387 is in a region of less structural difference, but this is near a predicted area of contact between monomers in the AMPNP dimer structure. A580, in the CTD, is near the dimerization interface of both structures and in a region that shows some structural variation, as predicted by our model. A403, in the MD, is isolated from inter-domain contacts, but still shows some predicted structural variation. Hence, most perturbations of chemical shifts can be rationalized based on a structural comparison of apo crystal structures and a homology model of the AMPNP structure. This supports the validity of the assignments, and demonstrates the ability of chemical shift to report on regions of structural change in proteins.

**Inter-domain structure of apo-HtpG.** Hsp90's clearly sample a range of conformations as evidenced by different crystal, SAXS and EM structures. Internal structures of individual domains might be expected to be better preserved, and there is even some data to suggest that certain inter-domain contacts may be preserved. In the pair of structures for apo and ADP forms of HtpG CTD and MD, orientations are nearly identical [77]. Matching C $\alpha$  carbons in the MD plus CTD, (residues 233-624) the overall alignment is 2.2Å. They align even closer in the GRP94 apo and AMPPNP structures [27]. RDCs provide one means of assessing the conservation of domain-domain orientations. When sufficient data are available ( $>5$ ) for a rigid segment, a best set of order parameters can be determined and used to back-calculate RDCs from a trial structure and compare those to measured RDCs for all sites. A Q factor [7] gives a measure of how well the crystal structure compares to that in solution. The amount of data we have for some of the domains is marginal, so we initially tried fitting a 2 domain segment. Using 18 pieces of alanine methyl RDCs from the phage alignment, spread over the MD and CTD of apo-HtpG, a Q factor of 0.81 was obtained. This is not very good agreement. Some assessment of the level of agreement can be obtained by comparing Q factors obtained for individual domains to that for combined domains, but

this must be done using the same number of RDCs. Using 8 randomly picked alanine methyl RDCs, 4 from each domain, we found an average Q factor of 0.4. This can be compared to Q factors of the individual domains using similar numbers of RDCs (0.2 and 0.2). The larger Q for the combined domains clearly suggests that even the MD - CTD orientations of the apo form of HtpG seen in the crystal structure are not well maintained in solution.

In principle, the principal order parameters obtained for each domain, or their combination in a generalized degree of order (GDO), can be used as a direct indicator of internal motion; if the structure was flexible, and ordering occurred principally through interactions with one domain, GDOs would be smaller for non-interacting domains. The GDOs determined for phage alignment are 0.0018, 0.0020 and 0.0023 for the NT, middle, and CTD, respectively, showing no significant indication of preferential alignment and allowing no clear conclusion about internal domain motions. However, the GDOs for PEG alignment are 0.0039, 0.0036 and 0.0007 indicating possible preferential alignment by the NTD in this medium, and the existence of some internal motion reducing average alignment of the other domains. [2] [6]

It is also possible to use RDC data on a domain by domain basis to determine an average structure of the apo form in solution. Order parameters determined for each domain can be converted to a set of principle order parameters and Euler angles that relate the original coordinate frame to coordinates in a principal alignment frame. For a rigid molecule all domains would share this alignment frame. Hence, once in the principal alignment frame, multiple domains can be assembled by translation to a position where inter-domain connectivities can be made. Because RDCs are insensitive to rotation about any of the principal frame axes by 180, four possible orientations of each added domain need to be examined in this assembly process, but usually only one of these will allow reasonable linkages between domains. A similar procedure can be used when there is inter-domain motion, but domain positions must then be viewed as a representation of an average structure. A structure of the apo form assembled in this way from PEG data is shown in Figure 6A. For the NTD

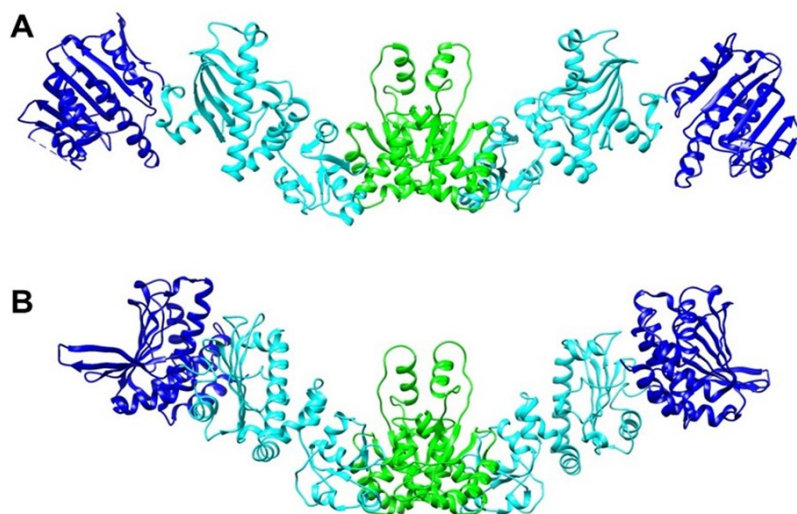


Figure 3.12: . Models of the apo-HtpG dimer. A) Domain orientations were obtained by RDC analysis. B) A model based on SAXS data obtained in solution at high pH (Krukenberg et al. 2008). NT, MD and CT domains are colored blue, cyan and green respectively.

to MD connection, only a 180 z rotation of the MD produces an acceptable linkage. For the MD to CTD connections with both zero and 180 x rotations produce acceptable structures. Because of the symmetry of the CTD dimer and its orientation in the principal alignment frame, these rotations actually produce the pair of MDs as seen in the dimer structure. The relative orientations for the domains are represented well in Figure 6A. However, their translational positions depend entirely on acceptable positions for covalent connection between domains. This is not a serious problem for the NTD to MD connection because there are no missing residues between the structures used to model this pair of domains. For the MD to CTD connection >15 residues are missing and there is a significant lack of translational definition.

There is a SAXS model for apo-HtpG as it exists in solution at high pH [46]. It is depicted in Figure 6B. To aid comparison we have oriented CTDs similarly and translated the NTD-MD pair of the RDC model to match as best as possible the MD in the SAXS model. It is apparent that the MD domains are oriented similarly in the two models, while the orientation of the NTD differs and a more extended structure results. One must use caution in interpreting these models. We do see evidence of inter-domain motion, so these represent structures subject to averaging processes that are quite dependent on the source of data. Also, RDC data available in this case are quite minimal (8 or 9 RDCs per domain), and therefore, results are more prone to error than typical applications. However, the observation of a structure that is more extended in solution than in crystal structures of apo and nucleotide-bound forms is certainly well-supported by these data.

## CHAPTER 4

### RNA STRUCTURE AND MAXIMALLY SPANNING $k$ -TREES

$k$  trees are a particular type of a clique cover. This type of covering a graph is relevant to RNA structure determination, neural networks, and other mathematical problems. The  $k$  tree has a type of cover which has a well defined tree width [25]; this makes the cover useful in modeling RNA structure. A backbone  $k$ -tree is one in which nodes are sequentially connected; that is, node  $i$  is connected to node  $i + 1$ . These types of trees are considered in this work.

Previous works [24, 26] have used maximally spanning backbone constrained  $k$ -trees in RNA structure calculations. Partly based on a detailed evaluation of many RNA examples, it is fitting to use a weighted  $k$ -tree in the possible structural characterization. It was found that most RNA molecules do have a bounded tree width, which means their structure are mostly tree-like with bounded tree width, at least in some portions of the RNA, such as binding of different residues. The weighting of the  $k$ -tree is used to constrain in the RNA model nucleotide interactions. This aspect of the possible and likelihood of the binding of nucleotides can be translated into a set of edge weights of the  $k$ -tree graph. The highest weighted  $k$ -tree is used to find the highest probability of correct interactions between nucleotides using a set of clique weights in the  $k$ -tree graph model.

It is of interest to find the best fit of the clique weights to that of a  $k$ -tree. The question is: given a set of edge weights to model the nucleotide interactions, what is the highest weighted  $k$ -tree for a given set of nodes. This is particularly important for  $k = 3$  due to the limited tree width structure of an RNA molecule, but it is also important for larger  $k$ . The genetic algorithm is used to find best possible solutions for a weighted  $k$ -tree graph given

the constraints of the edge weights and backbone; best means to maximize the sum of edge weights in a graph construction. The backbone constraint is due to the fact that the RNA is linearly constructed and the nodes are connected in the sense that node  $i$  is connected to node  $i + 1$  for all  $i$ . The advantage of using the genetic algorithm is that better solutions in fitness are found. In contrast to earlier work [22–26], which used a dynamic program, the genetic algorithm produces multiple solutions with high fitness.

It is noteworthy to point out that although the dynamic program described in [22–26] is an exhaustive search technique, it won’t give the optimal solution for a given  $k$ -tree due to an approximation in using 3-trees at the initial stage [22, 25, 26]. There is no guarantee that the genetic algorithm will give an optimal solution. However, initializing the population with a good solution from the dynamic program improves the result. It is always a very proper step to initialize the population near a good solution. This reduces the complexity of finding better solutions.<sup>1</sup>

As an example output of the dynamic program and genetic algorithm, 4-trees from a 100 node graph is shown for the first few cliques. The solutions are different; the first 10 cliques are listed in Tables 4.1 and 4.2. There is a minor difference in the ninth clique, and there forward. These differences are small, but the genetic algorithm does give a better  $k$ -tree in total clique weight than the dynamic program, and for  $k = 3$  to  $k = 20$ .

#### 4.1 EXAMPLE $k$ -TREE AND CHROMOSOME REPRESENTATION

Before the genetic algorithm model is presented, an example  $k$ -tree is presented. This should clarify the model and chromosome representation in the next section. An example maximally spanning  $k$ -tree is given in Table 4.3. In this case  $N = 100$  and  $k = 4$ . The order of the nodes in the chromosome is in the fifth column. The 5-cliques of the 4-tree are the rows in

---

<sup>1</sup>The results of these calculations are available upon request, both from the dynamic program and genetic algorithm from a particular set of edge weights.

Table 4.1: First cliques of a 4-tree from the genetic algorithm. The first 4 columns are the nodes in the 4-tree, and total clique weights are in the last column.

|    |    |    |    |    |        |
|----|----|----|----|----|--------|
| 39 | 40 | 41 | 42 | 36 | 5.6603 |
| 36 | 39 | 40 | 42 | 68 | 2.6396 |
| 36 | 39 | 42 | 68 | 43 | 2.7885 |
| 36 | 39 | 43 | 68 | 35 | 2.4445 |
| 35 | 36 | 43 | 68 | 55 | 2.7784 |
| 35 | 43 | 55 | 68 | 33 | 3.1212 |
| 35 | 43 | 55 | 68 | 52 | 3.3013 |
| 43 | 52 | 55 | 68 | 51 | 2.2908 |
| 51 | 52 | 55 | 68 | 53 | 2.3108 |
| 51 | 53 | 55 | 68 | 54 | 2.8092 |

Table 4.2: First cliques of a 4-tree from the dynamic program. The nodes of the cliques are in the first four columns. Total clique weights are in the last column.

|    |    |    |    |    |        |
|----|----|----|----|----|--------|
| 39 | 40 | 41 | 42 | 36 | 5.6603 |
| 36 | 39 | 40 | 42 | 68 | 2.6396 |
| 36 | 39 | 42 | 68 | 43 | 2.7885 |
| 36 | 39 | 43 | 68 | 35 | 2.4445 |
| 35 | 36 | 43 | 68 | 55 | 2.7784 |
| 35 | 43 | 55 | 68 | 33 | 3.1212 |
| 35 | 43 | 55 | 68 | 52 | 3.3013 |
| 43 | 52 | 55 | 68 | 51 | 2.2908 |
| 51 | 52 | 55 | 68 | 54 | 2.0000 |
| 51 | 53 | 55 | 68 | 54 | 2.9585 |

the first 5 columns. The first 10 cliques are shown. The clique construction is first examined and explained, and then the chromosome of this  $k$ -tree is shown and explained.

The order of the 100 nodes is 36, 68, 43, 35, 55, 33, 52, 51, 53, 54,  $\dots$ . The base clique is the first, i.e. (39, 40, 41, 42, 36). This chromosome is found with the dynamic program. To illustrate the construction the next 2 cliques the construction are explained. The second clique is going to insert the next node, which is node 68, into the tree. There is only one previous clique to use, and that is clique 1. Node 68 has replaced node 41. Note that the

first four entries are always ordered from smallest to largest node number; this is for easier bookkeeping in the program.

The node 43 is the next node to be used in the  $k$ -tree. There are now 2 earlier cliques that can be used in the creation of the 3rd clique. In this case the second clique is used. Node 43 replaces node 40. It gets more complicated as there are more choices in the clique construction for high clique number.

However, the question remains as to why these choices of clique construction of node insertion were made. The construction is going to attempt to find the highest weight spanning  $k$ -tree. The genetic algorithm searches these possibilities with iteration, crossover, and mutation. It isn't correct to find the maximal weight clique at each clique number construction in a direct calculation. It can be done, but it leads to non-maximally weighted  $k$ -trees. The chromosome is given in Table 4.4.

Table 4.3: First 10 cliques of a 4-tree. Weights are in the last column.

|    |    |    |    |    |        |
|----|----|----|----|----|--------|
| 39 | 40 | 41 | 42 | 36 | 5.6603 |
| 36 | 39 | 40 | 42 | 68 | 2.6396 |
| 36 | 39 | 42 | 68 | 43 | 2.7885 |
| 36 | 39 | 43 | 68 | 35 | 2.4445 |
| 35 | 36 | 43 | 68 | 55 | 2.7784 |
| 35 | 43 | 55 | 68 | 33 | 3.1212 |
| 35 | 43 | 55 | 68 | 52 | 3.3013 |
| 43 | 52 | 55 | 68 | 51 | 2.2908 |
| 51 | 52 | 55 | 68 | 53 | 2.3108 |
| 51 | 53 | 55 | 68 | 54 | 2.8092 |

Table 4.4: Chromosome of example  $k$ -tree in Table 4.3

|              |    |    |    |    |    |    |    |    |    |    |     |
|--------------|----|----|----|----|----|----|----|----|----|----|-----|
| S first part | 36 | 68 | 43 | 35 | 55 | 33 | 52 | 51 | 53 | 54 | ... |
| second part  | 0  | 1  | 2  | 3  | 4  | 5  | 5  | 7  | 8  | 9  | ... |
| third part   | 0  | 3  | 3  | 3  | 2  | 2  | 2  | 1  | 1  | 2  | ... |

The 3 parts of the chromosome are made from the information in Section 4.4, after including all 100 nodes, only 10 cliques are described in Section 4.4. This formatting of the



$k$ -tree is used in the genetic algorithm. This is used in a  $(N - k + 1) \times k$  matrix, described in Section 4.2 which is then used to make the sequence of cliques in the  $k$ -tree.

## 4.2 GENETIC ALGORITHM MODEL

The genetic algorithms is an example of an evolutionary algorithm which is efficient in finding optimal or near-optimal solutions to complex NP-Hard problems. It uses a possible set of solutions, defined as ‘chromosomes’ which evolve towards the best solution. Optimal means that the overall best solution is the maximum or minimum of fitness of the solutions. Fitness, as defined by the user, could mean the maximum or minimum; this could be, in the case of the Traveling Salesperson problem, the minimum distance of someone visiting a set of cities. Of course, the evolution towards the near-optimal may lead to near optimal solutions. These are usually local minima of the fitness function which is a measure of the accuracy of the solution.

At each iteration of the evolution, crossover and mutation of these ‘chromosomes’ is used to improve the possible solutions towards the best fitness. The flow of the genetic algorithm is described in the introductory Chapter 1. There are various aspects that have to be considered in using a genetic algorithm, and in this case, the initialization of the population of chromosomes, or possible solutions to the problem, is important. Tuning of the parameters is partially solved by scanning over a range of crossover and mutation rates.

This program uses a genetic algorithm to find the maximal spanning  $k$ -tree of a complete graph. The graph is weighted and the maximum weight  $k$ -tree is searched for at a given  $k$ .

The input and output of the algorithm is :

Input : number of nodes  $N$ , size  $k$  of clique, weights  $w(m, n)$  where  $m$  and  $n$  are nodes, initial population.

Output : an ordered list of all weighted backbone  $k$ -trees with total weight greater than some value. The total weight of a  $k$ -tree is the sum of all the edge weights from the edges in the  $k$ -tree.

The chromosome of the genetic algorithm contains information which can be used to construct the spanning  $k$ -tree; the spanning  $k$ -tree has  $N - k + 1$ -cliques. The chromosome has the minimal information to reconstruct the set of  $(k + 1)$ - cliques. It has can be used easily with mutation and crossover functions. There are 3 components to the chromosome. The first part of the chromosome has genes representing an ordered set of the numbers from 1 to  $N$  with no repetition; this is the node order used in the construction of the spanning  $k$ -tree. In the construction of the spanning  $k$ -tree each clique is made by taking an earlier clique and changing one of the nodes. For example, the tenth clique could be made by taking the fifth clique and replacing the third node in the clique with node ten. The second part of the chromosome is a set of integers which say which earlier clique is to be used in making the new clique. Each  $k + 1$ -clique has  $k + 1$  nodes in its definition. These nodes are attached to each other by edges, and the weight of the clique is the sum of the weights of these edges. The third part of the chromosome is a collection of numbers that specify which node in the chosen clique is to be replaced by the new node.

|                  | node $(1, \dots, N)$ | clique<br>$(N + 1, \dots, N + N - k + 1)$ | node insertion<br>$(1 + (N - k + 1) + 1, \dots)$ |
|------------------|----------------------|---|--|
| Number of genes: | $N$                  | $N - k + 1$                               | $N - k + 1$                                      |
| Values of genes: | 1 to $N$             | 1, 1 to 2, 1 to 3, ...                    | 1 to $k + 1$ , 1 to $k + 1$                      |
| Total            | $N!$                 | $1/2 \times (N - k + 1)^2$                | $(N - k + 1) \times (k + 1)$                     |

Table 4.5: Values in the chromosome and complexity.

The objective function is the sum of all the weights of the edges of the spanning  $k$ -tree; these edge weights are counted only once even if the edge is shared by two different cliques. For  $N$  nodes there is a spanning  $k$ -tree with maximal weight. The solution of the maximal spanning  $k$ -tree problem is the  $k$ -tree with maximal weight.

Next are the constraints used in the genetic algorithm. Each number in the first part of the chromosome is unique and made from numbers 1 to  $N$ . There is also a penalty to the fitness of the  $k$ -tree if it is not of backbone type. For every node that does not appear to be in a backbone there is a penalty. Using a penalty instead of a constraint keeps these chromosomes

in the population possibly. The inclusion of non-backbone spanning trees is necessary in the evolution to find backbone trees; a non-backbone spanning tree may evolve into one that is backbone type. The output file does include an ordered list of all  $k$ -trees, ordered with from the maximal backbone, which in the case of 100 nodes is 99, total backbone.

Mutation of the population is done in 3 steps. First, two chromosomes are selected from the population. Second, a random number is chosen from 0 to the total number of nodes; this will limit the operation in the 1st segment of the chromosome. Then the 2 segments of these chromosomes are interchanged between the 1st and 2nd genes of these chromosomes. This permutation exchanges the node addition in the construction of the  $(k + 1)$ - clique, without changing anything else in the chromosomes. Third, in one of the chromosomes the node insertion point is randomly interchanged with an earlier node insertion; this is the 2nd part of the chromosome. The mutation is essentially both a type of crossover and a typical mutation.

The crossover function has 2 parts in it. The first part selects a segment in the region of the chromosome as mutation and sorts it numerically. The second part selects a segment in the same section of the chromosome and flips the order. These operations are done several times. This is possibly done for all the chromosomes with probability of crossover.

The creation of the population uses the output of the dynamic algorithm. Initializing the population can be very important for an effective genetic algorithm. The population is created by taking the best  $k$ -tree from the dynamic program. If there are  $N$  chromosomes in the genetic algorithm then  $N$  identical chromosomes, from this  $k$ -tree, are in the initial population.

The program works as follows :

- as mentioned earlier 'clique' is a 2-d array of dimension  $(N - k + 1) \times k$ , where  $N$  is the total number of nodes. The clique size is  $k + 1$ , since there are  $k + 1$  nodes in a  $(k + 1)$ -clique)

- input file of edge weights. This file has 3 columns. The first two numbers in each row are the nodes of the edges. The third is the edge weight. The program will parse it and turn it into a 2d array.

- The program is object oriented.

The translation of a chromosome to a  $k$ -tree and a calculation of the total weight is given next in pseudo-code. This algorithm is used in the objective function.

1. begin – Take first  $k + 1$  integers from array chromosome and store in clique(1,:). This is the base clique.

2. Define the clique counter, totalclique = 1;

3. This loop constructs all cliques in the  $k$ -tree. While  $i \leq N - k + 1$ , starting at 1,

4. Take the 3 parts corresponding to the  $(k + i + 1)$ -clique. These are chromosome  $(k + 1 + i)$  as the node number, chromosome  $(k + 1 + i + N)$  as the previous clique, chromosome  $(k + 1 + i + N + (N - k + 1))$  as the node insertion. The  $i$ 'th clique is constructed as described previously. Take chromosome  $(k + 1 + i + N + (N - k + 1))$  constructed before and replace at the node insertion the node chromosome  $(k + 1 + N + i)$ .

6. Add new edge weights to totalweight.

7. totalclique++;

8. clique(totalclique,:)=newclique(:);

9. end

For each chromosome the program generates a spanning  $k$ -tree with its weight. Each iteration has a population of chromosomes. At each iteration the entire population is stored in a Matlab .dat cell array file. After the genetic algorithm stops, the post-processing program eliminates duplicate chromosomes, sorts, and converts the information into a user friendly text file. The user specifies the lowest fitness, and all unique spanning  $k$ -trees are given in the output with total weight greater than this lowest fitness. We are interested in the highest weight, which would be the first in the list.

There are several parameters that go into the genetic algorithm. These parameters are listed in Table 4.6. There are two parts to making a working genetic algorithm. The first is the genetic algorithm. The second is the tuning of the parameters, i.e. finding correct parameters with experimentation. There are general guidelines such as high crossover and low mutation, and not too large a population.

Table 4.6: Parameters used.

|                   |             |
|-------------------|-------------|
| population        | 500         |
| crossover rate    | .2,.4,.6,.8 |
| mutation rate     | .2,.4,.6,.8 |
| max iterations    | 500         |
| elite count       | .1          |
| stopping criteria | 500         |

All possible combinations of the crossover and mutation rates are used in the algorithm. The chromosomes are described in this Section and an example of a chromosome is given in the next section.

### 4.3 RESULTS AND COMPARISONS WITH EARLIER WORK

Results are presented next of the calculation of individuals in the genetic algorithm population. The  $k$ -trees from  $k = 3$  to  $k = 20$  for  $k$ -trees having 100 nodes. The unique solutions with fitness greater than the maximal fitness solution of the dynamic programming are stored in files for each  $k$ . Unlike the dynamic program results, which is a single  $k$ -tree for each  $k$ , the genetic algorithm gives thousands of solutions with better fitness.

Typically, the maximal fitness found from the genetic algorithm for each  $k$  is less than 1 better. However, the number of unique solutions is large. Table 4.8 gives the maximal fitness from the dynamic program and the genetic algorithm for these  $k$  values. The table also gives the number of solutions between the maximal fitness of the dynamic program and the genetic algorithm. The time of the program is less than a second for each iteration; as the program continues to run iterations more solutions are found.

Table 4.7: Best individuals : dynamic program and genetic algorithm

| $k$ -tree        | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     | 11     |
|------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| dynamic fitness  | 192.82 | 254.31 | 313.52 | 369.45 | 423.99 | 476.78 | 527.95 | 578.28 | 628.20 |
| genetic fitness  | 193.74 | 255.48 | 315.36 | 359.01 | 425.87 | 478.22 | 530.39 | 580.72 | 630.73 |
| no. of solutions | 188    | 1353   | 4477   | 3052   | 4063   | 3076   | 9493   | 10194  | 10254  |

Table 4.8: Best individuals : dynamic program and genetic algorithm

| $k$ -tree        | 12     | 13     | 14     | 15     | 16     | 17     | 18     | 19     | 20      |
|------------------|--------|--------|--------|--------|--------|--------|--------|--------|---------|
| dynamic fitness  | 628.20 | 723.72 | 769.78 | 769.78 | 859.24 | 885.40 | 946.23 | 988.63 | 1030.00 |
| genetic fitness  | 630.80 | 725.23 | 771.60 | 816.81 | 861.80 | 887.80 | 947.87 | 990.59 | 1033.04 |
| no. of solutions | 6044   | 6396   | 6564   | 7454   | 12721  | 12648  | 11386  | 8583   | 15106   |

#### 4.4 CONCLUSIONS

A genetic algorithm has been used find thousands of solutions to the maximally spanning backbone  $k$ -tree problem for any  $k$ . Results are compared with earlier work using a dynamic program. These results are generally useful for many areas of science and are generalized to partial backbone  $k$ -trees with any number of backbone edges.

## CHAPTER 5

### CONCLUSIONS OF DISSERTATION

This dissertation shows that the use of genetic algorithms is important in solving or approximately solving complex problems. These problems are in the context of NP-Hard. The same calculations in an exhaustive search would be infeasible. These problems are described in the introductory Chapter 1.

Three types of problems are considered. The first is that of sensitivity analysis and mode transition sensitivity to make schedules and computation feasible. The resource utilization, i.e. processor(s), is higher if the genetic algorithm described in Chapter 2 is used to make a schedule feasible. The second problem is about the NMR spectral assignment of HSQC from sparsely labeled residues of large and small proteins. The third problem, in Chapter 4 is about the problem of finding maximally spanning  $k$ -trees. There are other contexts of these solutions which are not discussed. The genetic algorithms are shown to generate improved results in all problems over previous techniques. The results are shown in the different Chapters.

The assignment genetic algorithm AssignSLPMD and MD2NOEProtein software will be used in a larger context due to an R01 National Institute of Health grant awarded to several professors who worked on the development and use of the software. The use of these software packages are currently being simplified and will be available in a GUI download. These software packages, including the unsimplified versions, will be developed.

## CHAPTER 6

### APPENDIX : ADDITIONAL SOFTWARE

While this dissertation is about the use of genetic algorithms, there has been work in programs developing programs that use additional software that uses molecular dynamics trajectories to simulate NMR observables, such as NOE build-up curves, relaxation rates, 3J couplings, and more. Most NMR observables can be calculated with trajectory molecular dynamics information. Comparing experimentally observed parameters to predicted calculations from molecular dynamics trajectories is important in improving molecular dynamics modeling. After comparing the trajectory calculations with experimental information, the force fields underlying the molecular dynamics modeling can be improved. The comparison is important in molecular dynamics simulation.

There are several software packages that calculate NMR observables using the Amber molecular dynamics software. These packages are available at the Professor Prestegard and Professor Woods software sites,

<http://tesla.ccruc.uga.edu/software/>

<http://glycam.org/docs/othertoolsservice/publication-related-materials/browse-publication-related-materials/>

or

<https://dev.glycam.org/> .

- MD2NOEProtein is a software package that calculates an NOE peak list, NOE build-up curves, and other relaxation parameters of large proteins from a molecular dynamics Amber trajectory. This package is large, and the output is used in the genetic algorithm



package AssignSLPMD. It is appropriate to use the MD2NOEProtein package in calculating NOE build-up curves and other relaxation parameters of larger proteins from a known known relaxation molecular dynamics structure and trajectory. The package can use an arbitrary set of initial and final spin states of protons of large protons, which means that any type of NMR spin relaxation; the experiment is simulated by an Amber trajectory with calculations of the observed measurements. It's default is sparse labeling. It is designed to model the sampling of motion in larger proteins, for which isotope labeling of only a sparse set on amino acids is practical. A simplified version is currently in development, and the output has only the NOE peak list in Sparky format. A simplified version of the AssignSLPMD package described in Chapter 4 is also in development. Both MD2NOEProtein and AssignSLPMD require a large number of inputs, and the simplified versions are being utilized in a GUI (graphical user interface), which is in development.

- MD2NOE is the earlier version of MD2NOEProtein. It was successfully used in calculating NOE build-up curves and NOE's of small molecules, including carbohydrates, using a trajectory. The carbohydrate results are not in this dissertation. The use can be found in a paper listed in Appendix 6.

- AssignSLP is a genetic algorithm software package that does not use a molecular dynamics trajectory to find an assignment of proteins. AssignSLPMD generalized this software package very much by using the trajectory in all aspects of the calculation. The latter package, for example, uses the output of MD2NOEProtein, includes order parameters to improve RDC calculations, and uses a trajectory for the calculation of average chemical shifts. This difference is described in Chapter 4.

- 3J coupling distribution calculates 3J couplings from a molecular dynamics Amber trajectory and generates histograms for phi-psi plots of the conformations of molecules. The results could be used for structure calculations of carbohydrates.

- ParticularRelaxationRate calculates R1 and R2 relaxation rates of large and small molecules. These trajectory calculations were used in the identification of the glycosola-

tion site of a pentasacharide and its epitope, after comparing experiments with predictions from several molecular dynamics trajectories, of the large skp1 protein. The paper and its results are listed in Appendix 7.

- SingleFrameNMR calculates NMR observables: NOE's, their build-up curves, spectral density functions, 3J-couplings, and saturation difference transfers. These calculations use a pdb file or single frame of an MD trajectory. This package is being developed.

## CHAPTER 7

### APPENDIX : PAPERS TO WHICH I HAVE CONTRIBUTED IN THE COURSE OF DISSERTATION RESEARCH

G. Chalmers, S.H. Funk, Genetic Algorithms in Real-Time Systems, submitted to Genetic Programming and Evolvable Machines, 2019.

G. Chalmers, A. Eletsky, L. Morris, J. Yang, F. Tian, R.J. Woods, K.W. Moremenm, J.H. Prestegard, NMR Resonance Assignment Strategy: Characterizing Large Sparsely Labeled Glycoproteins, Journal of Molecular Biology. 2019, v. 431: pp. 2369-2382, issue 12.

G. Chalmers, S.H. Funk, Adjusting Real-Time Mode Transitions via Genetic Algorithms, 2017, 16th IEEE International Conference on Machine Learning and Applications, Cancun, paper ID 169.

M.O. Sheikh, D. Thieker, G.R. Chalmers, C.M. Schafer, M. Ishihara, P. Azadi, R.J. Woods, J.N. Glushka, B. Bendiak, J.H. Prestegard, C.M. West, O<sub>2</sub> sensing-associated glycosylation exposes the F-box-combining site of the Dictyostelium Skp1 subunit in E3 ubiquitin ligases. J. Biol. Chem. 2017, v. 292: pp. 18897-18915.

K. Pederson, G. Chalmers, Q. Gao, D. Elnatan, T.A. Ramelot, L. Ma, G.T. Montelione, M.A. Kennedy, D.A. Agard, J.H. Prestegard, NMR characterization of HtpG, the E. coli Hsp90, using sparse labeling with <sup>13</sup>C-methyl alanine, Journal of Biomolecular NMR. July 2017, issue 3, v. 68: pp. 225-236.

Q. Gao, G.R. Chalmers, K.W. Moremen, J.H. Prestegard, NMR assignments of sparsely labeled proteins using a genetic algorithm, *Journal of Biomolecular NMR*. 2017, v. 67:pp. 283–294.

G. Chalmers, J.N. Glushka, B.L. Foley, R.J. Woods, J.H. Prestegard. Direct NOE simulation from long MD trajectories. *Journal of Magnetic Resonance*. 2016, v. 265: pp. 1-9.

## BIBLIOGRAPHY

- [1] M. Ababneh, S. Hassan, and S. Bani-Ahmad. On static scheduling of tasks in real time multiprocessor systems: an improved ga-based approach. *The International Arab Journal of Information Technology*, 101(6), 2014.
- [2] A. Bahrami, A. Assadi, J.L. Markley, and H. Eghbalnia. Probabilistic interaction network of evidence algorithm and its application to complete labeling of peak lists from protein nmr spectroscopy. *PLoS Computational Biology*, 5, 2009.
- [3] P. Balbastre, I. Ripoll, and A. Crespo. Minimum deadline calculation for periodic real-time tasks in dynamic priority systems. *Computers, IEEE Transactions*, 57(1):96 – 109, 2008.
- [4] P. Balbastre, I. Ripoll, and A. Crespo. Period sensitivity analysis and dp domain feasibility region in dynamic priority systems. *Journal of systems and software*, 82(7):1098–1111, 2009.
- [5] M. Bartshi. *A Genetic Algorithm for Resource-Constrained Scheduling*. PhD thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1996.
- [6] J.L. Battiste and G. Wagner. Utilization of site-directed spin labeling and high resolution heteronuclear nuclear magnetic resonance for global fold determination of large proteins with limited nuclear overhauser effect data. *Biochemistry*, 39:5355–5365, 2000.
- [7] A. Bax. Weak alignment offers new nmr opportunities to study protein structure and dynamics. *Protein Science*, 12:1–16 doi:10.1110/ps.0233303, 2003.

- [8] W. Becker, K.C. Bhattiprolu, N. Gubensak, and K. Zangger. Investigating protein-ligand interactions by solution nuclear magnetic resonance spectroscopy. *Chemphyschem*, 19:895–906, 2018.
- [9] G.P. Bhide and K.J. Colley. Sialylation of n-glycans: mechanism, cellular compartmentalization and function. *Histochemistry and Cell Biology*, 147:149–74, 2017.
- [10] E. Bini and G. Buttazzo. The space of edf feasible deadlines. In *Real-Time Systems, 2007. ECRTS '07. 19th Euromicro Conference*, pages 19–28, July 2007.
- [11] E. Bini and G. Buttazzo. The space of edf deadlines: the exact region and a convex approximation. *Real-Time Systems*, 41(1):27–51, 2009.
- [12] E. Bini and G.C. Buttazzo. Biasing effects in schedulability measures. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS 2004)*, pages 196–203, July 2004.
- [13] E. Bini, M. Di Natale, and G. Buttazzo. Sensitivity analysis for fixed-priority real-time systems. *Real-Time Systems*, 39(1):5–38, 2008.
- [14] D.A. Case, J.T. Berryman, R.M. Betz, Q. Cai, D.S. Cerutti, T.E. Cheatham III, T.A. Darden, H.G. Duke HG, A.W. Goetz, S. Gusarov, N. Homeyer, P. Janowski, J. Kaus, I. Kolossváry, A. Kovalenko, T.S. Lee, T. Luchko, R. Luo, B. Madej, K.M. Merz, F. Paesani, D.R. Roe, A. Roitberg, C. Sagui, R. Salomon-Ferrer, C.L. Simmerling, W. Smith, J. Swails, R.C. Walker, J. Wang, R.M. Wolf, X. Wu, and P.A. Kollman. Amber 14. *University of California, San Francisco*, 2014.
- [15] G. Chalmers, A. Eletsky, L. Morris, J. Yang, F. Tian, R.J. Woods, K.W. Moremen, and J.H. Prestegard. Nmr resonance assignment strategy: Characterizing large sparsely labeled glycoproteins. *Journal of Molecular Biology*, 431:2369–2382, 2019.
- [16] G. Chalmers, J.N. Glushka, B.L. Foley, R.J. Woods, and J.H. Prestegard. Direct noe simulation from long md trajectories. *Journal of Magnetic Resonance*, 265:1–9, 2016.

- [17] C.K. Chang, M.J. Christensen, and T. Zhang. Genetic algorithms for project management. *Annals of Software Engineering*, 11(1):107–139, 2001.
- [18] R. Cheng, M. Gen, and Y. Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms, part 1. representation. *Computers and industrial engineering*, 30(4):983–997, 1996.
- [19] R. Cheng, M. Gen, and Y. Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms, part 2: hybrid genetic search strategies. *Computers and industrial engineering*, 36(2):343–364, 1999.
- [20] M.N. Christiansen, J. Chik, L. Lee, M. Anugraham, J.L. Abrahams, and N.H. Packer. Cell surface protein glycosylation in cancer. *Proteomics*, 14:525–46, 2014.
- [21] J.M. Courtney, Q. Ye, A.E. Nesbitt, M. Tang, M.D. Tuttle, E.D. Watt, and et al. Experimental protein structure verification by scoring with a single, unassigned nmr spectrum. *Structure*, pages 1958–66, 2015.
- [22] L. Ding, A. Samad, G. Li, R.W. Robinson, X. Xue, R. Malmberg, and L. Cai. Polynomial-time algorithms for maximum spanning k-tree constrained by hamiltonian path. pages 1164–1173, 2016.
- [23] L. Ding, A. Samad, G. Li, R.W. Robinson, X. Xue, R.L. Malmberg, and L. Cai. Finding maximum spanning k-trees on backbone graphs in polynomial time. *test*, 2016.
- [24] L. Ding, X. Xue, S. LaMarca, R. L. Malmberg, C. Momany, and L. Cai. Accurate prediction of rna 3d structure based on backbone k-tree model. 2016.
- [25] L. Ding, X. Xue, S. LaMarca, M. Mohebbi, A. Samad, R. Malmberg, and L. Cai. Ab initio prediction of rna nucleotide interactions with backbone k-tree model. In *ECCB’14 Workshop on Computational Methods for Structural RNAs*, pages 25–42, 2014.

- [26] L. Ding, X. Xue, S. LaMarca, M. Mohebbi, A. Samad, and R.L. Malmberg. Accurate prediction of rna nucleotide interactions with backbone k-tree model. *Bioinformatics*, 10, 2015.
- [27] D.E. Dollins, J.J. Warren, R.M. Immormino, and D.T. Gewirth. Structures of grp94-nucleotide complexes reveal mechanistic differences between the hsp90 chaperones. *Mol Cell*, 2007.
- [28] F. Dorin, J. Goossens, and P. Richard. Slack-based sensitivity analysis for edf. In *Proceedings Work-In-Progress Session of the 14th Real-Time and Embedded Technology and Applications Symposium, (RTAS 2008)*, pages 33–36, April 2008.
- [29] F. Sievers et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Mol Syst Biol*, 7:doi:10.1038/msb.2011.75, 2011.
- [30] R. Xiao et al. The high-throughput protein sample production platform of the northeast structural genomics consortium. *Journal of Structural Biology*, 172:21–33 doi:10.1016/j.jsb.2010.07.011, 2010.
- [31] J.K. Everett, R. Tejero, S.B. Murthy, T.B. Acton, J.M. Aramini, M.C. Baran, J. Benach, J.R. Cort, A. Eletsky, F. Forouhar, R. Guan, A.P. Kuzin, H.W. Lee, G. Liu, R. Mani, B. Mao, J.L. Mills, A.F. Montelione, K. Pederson, R. Powers, T. Ramelot, P. Rossi, J. Seetharaman, D. Snyder, G.V. Swapna, S.M. Vorobiev, Y. Wu, R. Xiao, Y. Yang, C.H. Arrowsmith, J.F. Hunt, M.A. Kennedy, J.H. Prestegard, T. Szyperski, L. Tong, and G.T. Montelione. A community resource of experimental data for nmr / x-ray crystal structure pairs. *Protein Science*, 25:30–45, 2016.
- [32] D.P. Frueh. Practical aspects of nmr signal assignment in larger and challenging proteins. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 78:47–75, 2014.



- [33] Q. Gao, G.R. Chalmers, K.W. Moremen, and J.H. Prestegard. Nmr assignments of sparsely labeled proteins using a genetic algorithm. *Journal of Biomolecular NMR*, 67:283–294, 2017.
- [34] Q. Gao, G.R. Chalmers, K.W. Moremen, and J.H. Prestegard. Nmr assignments of sparsely labeled proteins using a genetic algorithm. *Journal of Biomolecular Nmr*, 67:283–94, 2017.
- [35] Q. Gao, C.Y. Chen, C. Zong, S. Wang, A. Ramiah, P. Prabhakar, and et al. Structural aspects of heparan sulfate binding to robo1-ig1-2. *ACS Chemical Biology*.
- [36] Q. Gao, C.Y. Chen, C. Zong, S. Wang, A. Ramiah, P. Prabhakar, and et al. Structural aspects of heparan sulfate binding to robo1-ig1-2. *Acs Chemical Biology*, 11:3106–13, 2016.
- [37] Q. Gao, C.Y. Chen, C. Zong, S. Wang, A. Ramiah, P. Prabhakar, L.C. Morris, G.J. Boons, K.W. Moremen, and J.H. Prestegard. Structural aspects of heparan sulfate binding to robo1-ig1-2. *ACS Chemical Biology*, page (DOI: 10.1021/acschembio.6b00692), 2016.
- [38] C. Gobl, T. Madl, H. Simon, and M. Sattler. Nmr approaches for structural analysis of multidomain proteins and complexes in solution. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 80:26–63, 2014.
- [39] N.K. Goto, K.H. Gardner, G.A. Mueller, R.C. Willis, and L.E. Kay. A robust and cost-effective method for the production of val, leu, ile ( $\delta$  1) methyl-protonated n-15-, c-13-, h-2-labeled proteins. *Journal of Biomolecular Nmr*, 13:369–74, 1999.
- [40] Y. Gu, D.W. Li, and R. Bruschweiler. Nmr order parameter determination from long molecular dynamics trajectories for objective comparison with experiment. *Journal of Chemical Theory and Computation*, 10:2599–607, 2014.

- [41] Y.N. Gu, D.W. Li, and R. BruschweilerS. Decoding the mobility and time scales of protein loops. *Journal of Chemical Theory and Computation*, 11:1308–14, 2015.
- [42] B. Han, Y.F. Liu, S.W. Ginzing, and D.S. Wishart. Shiftx2: significantly improved protein chemical shift prediction. *Journal of Biomolecular Nmr*, 50:43–57, 2011.
- [43] Y. Ji, Y.J.B. White, J.A. Hadden, O.C. Grant, and R.J. Woods. New insights into influenza a specificity: an evolution of paradigms. *Current Opinion in Structural Biology*, 44:219–31, 2017.
- [44] Q. Jiang. *A genetic algorithm for multiple resource-constrained project scheduling*. PhD thesis, University of Wollongong, 2004.
- [45] N. Tjandra K. Chen. The use of residual dipolar coupling in studying proteins by nmr. *In: Zhu G, editor. Nmr of Proteins and Small Biomolecules*, pages 47–67, 2012.
- [46] K.A. Krukenberg, F. Forster, L.M. Rice, A. Sali, and D.A. Agard. Multiple conformations of e-coli hsp90 in solution: Insights into the conformational dynamics of hsp90. *Structure*, 16:755–765 doi:10.1016/j.str.2008.01.021, 2008.
- [47] J. Kim, Y.J. Wang, G. Li, and G. Veglia. A semiautomated assignment protocol for methyl group side chains in large proteins. 2016.
- [48] K.N. Kirschner, A.B. Yongye, S.M. Tschampel, J. Gonzalez-Outeirino, C.R. Daniels, L.B. Foley, and et al. Glycam06: A generalizable biomolecular force field. carbohydrates. *Journal of Computational Chemistry*, 29:622–55, 2008.
- [49] K.A. Krukenberg, U.M.K. Bottcher, D.R. Southworth, and D.A. Agard. Grp94, the endoplasmic reticulum hsp90, has a similar solution conformation to cytosolic hsp90 in the absence of nucleotide. *Protein Science*, 18:1815–1827 doi:10.1002/pro.191, 2009.

- [50] B. Kuhn, J. Benz, M. Greif, A.M. Engel, H. Sobek, and M.G. Rudolph. The structure of human alpha-2,6-sialyltransferase reveals the binding mode of complex glycans. *Acta Crystallographica Section D-Biological Crystallography*, 69:1826–38, 2013.
- [51] L.A. Lavery, J.R. Partridge, T.A. Ramelot, D. Elnatan, M.A. Kennedy, and D.A. Agard. Structural asymmetry in the closed state of mitochondrial hsp90 (trap1) supports a two-step atp hydrolysis mechanism. *Mol Cell*, 53, 2014.
- [52] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the Real-Time Systems Symposium - 1989*, pages 166–171, Santa Monica, California, USA, 1989. IEEE Computer Society Press.
- [53] D.W. Li and R. Bruschweiler. Ppm: a side-chain and backbone chemical shift predictor for the assessment of protein conformational ensembles. *Journal of Biomolecular Nmr*, 54:257–65, 2012.
- [54] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.
- [55] S. Liu, A. Venot, L. Meng, F. Tian, K.W. Moremen, G.J. Boons, and et al. Spin-labeled analogs of cmp-neuac as nmr probes of the alpha-2,6-sialyltransferase st6gal i. *Chemistry Biology*, 14:409–18, 2007.
- [56] U. ManChon, C. Ho, S. Funk, and K. Rasheed. Gart: A genetic algorithm based real-time system scheduler. *Evolutionary Computation (CEC), 2011 IEEE Congress*, pages 886–893.
- [57] L. Meng, F. Forouhar, D. Thieker, Z. Gao, A. Ramiah, H. Moniz, and et al. Enzymatic basis for n-glycan sialylation: structure of rat alpha2,6-sialyltransferase (st6gal1) reveals conserved and unique features for glycan sialylation. *Journal Biological Chemistry*, 288:34680–98, 2013.

- [58] L. Meng, F. Forouhar, D. Thieker, Z.W. Gao, A. Ramiah, H. Moniz, and et al. Enzymatic basis for n-glycan sialylation structure of rat alpha 2,6-sialyltransferase (st6gal1) reveals conserved and unique features for glycan sialylation. *Journal of Biological Chemistry*, 288:34680–98, 2013.
- [59] N. Min-Allah, S.U. Khan, and W. Yongji. Optimal task execution times for periodic tasks using nonlinear constrained optimization. *The Journal of Supercomputing*, 59(3):1120–1130, 2010.
- [60] Y.R. Monneau, P. Rossi, A. Bhaumik, C.D. Huang, Y.J. Jiang, T. Saleh, and et al. Automatic methyl assignment in large proteins by the magic algorithm. *Journal of Biomolecular Nmr*, 69:215–27, 2017.
- [61] D. Montana, M. Brinn, and G. Bidwell. Genetic algorithms for complex, real-time scheduling. In *Systems, Man, and Cybernetics, International Conference*, volume 3, pages 2213–2218. IEEE, 1998.
- [62] K.W. Moremen, A. Ramiah, M. Stuart, J. Steel, L. Meng, F. Forouhar, and et al. Expression system for structural and functional studies of human glycosylation enzymes. *Nature Chemical Biology*, 14:156, 2018.
- [63] H.N.B. Moseley, E.V. Curto, and N.R. Krishna NR. Complete relaxation and conformational exchange matrix (corcema) analysis of noesy spectra of interacting systems - 2-dimensional transferred noesy. *Journal of Magnetic Resonance Series B*, 108:243–61, 1995.
- [64] C. Nitsche and G. Otting. Pseudocontact shifts in biomolecular nmr using paramagnetic metal tags. *Progress in Nuclear Magnetic Resonance Spectroscopy*, pages 20–49, 2017.
- [65] M. Norazizi, S.M. Sayuti, and L.S. Indrusiak. A function for hard real-time system search-based task mapping optimization. In *2015 IEEE 18th International Symposium on Real-Time Distributed Computing*, pages 66–73, April 2015.

- [66] K. Pederson, G. Chalmers, Q. Gao, D. Elnatan, T.A. Ramelot, L. Ma, G.T. Montelione, M.A. Kennedy, D.A. Agard, and J.H. Prestegard. Nmr characterization of htpg, the e. coli hsp90, using sparse labeling with  $^{13}\text{C}$ -methyl alanine. *Journal of Biomolecular NMR*, 68:225–236, July 2017.
- [67] K. Pederson, G.R. Chalmers, Q. Gao, D. Elnatan, T.A. Ramelot, L.C. Ma, and et al. Nmr characterization of htpg, the e-coli hsp90, using sparse labeling with c- $^{13}$ -methyl alanine. *Journal of Biomolecular Nmr*, 68:225–36, 2017.
- [68] E.F. Pettersen, T.D. Goddard, C.C. Huang, G.S. Couch, D.M. Greenblatt, E.C. Meng, and et al. Ucsf chimera - a visualization system for exploratory research and analysis. *Journal of Computational Chemistry*, 25:1605–12, 2004.
- [69] J.H. Prestegard, C.M. Bougault, and A.I. Kishore. Residual dipolar couplings in structure determination of biomolecules. *Chemical Reviews*, 104, 2004.
- [70] S. Punnekkat, R. Davis, and A. Burns. Sensitivity analysis of real-time task sets. *Advances in computing science - ASIAN97*, 1345 of the series Lecture Notes in Computer Science:72–82, 1997.
- [71] A. Racu and L.S. Indrusiak. Using genetic algorithms to map hard real-time on noc-based systems. In *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–8. IEEE, July 2012.
- [72] R. Racu, A. Hamann, and R. Ernst. A formal approach to multi-dimensional sensitivity analysis of embedded real-time systems. In *Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS 2006)*, pages 3–12, July 2006.
- [73] D.R. Roe and T.E. Cheatham. Ptraj and cpptraj: Software for processing and analysis of molecular dynamics trajectory data. *Journal of Chemical Theory and Computation*, 9:3084–95, 2013.

- [74] E. Schmidt and P. Guntert. A new algorithm for reliable and general nmr resonance assignment. *Journal of the American Chemical Society*, 134:12817–29, 2012.
- [75] E.K. Schneider, J. Li, and T. Velkov. A portrait of the sialyl glycan receptor specificity of the h10 influenza virus hemagglutinin-a picture of an avian virus on the verge of becoming a pandemic? *Vaccines*, 5, 2017.
- [76] M.O. Sheikh, D. Thieker, G. Chalmers, C.M. Schafer, M. Ishihara, P. Azadi, and et al. O-2 sensing-associated glycosylation exposes the f-box-combining site of the dictyostelium skp1 subunit in e3 ubiquitin ligases. *Journal of Biological Chemistry*, 292:18897–915, 2017.
- [77] A.K. Shiau, S.F. Harris, D.R. Southworth, and D.A. Agard. Structural analysis of e-coli hsp90 reveals dramatic nucleotide-dependent conformational rearrangements, a. *Cell*, 127:329–340 doi:10.1016/j.cell.2006.09.027, 2006.
- [78] D.R. Southworth and D.A. Agard. Species-dependent ensembles of conserved conformational states define the hsp90 chaperone atpase cycle. *Mol Cell*, 32:631–640 doi:10.1016/j.molcel.2008.10.024, 2008.
- [79] D.R. Southworth and D.A. Agard. Client-loading conformation of the hsp90 molecular chaperone revealed in the cryo-em structure of the human hsp90:hop complex. *Mol Cell*, 42:771–781 doi:10.1016/j.molcel.2011.04.023, 2011.
- [80] A. Stechmann and T. Cavalier-Smith. Evolutionary origins of hsp90 chaperones and a deep paralogy in their bacterial ancestors. *Journal Eukaryotic Microbiology*, 51:364–373, 2004.
- [81] T.O. Street, L.A. Lavery, K.A. Verba, C.T. Lee, M.P. Mayer, and D.A. Agard. Cross-monomer substrate contacts reposition the hsp90 n-terminal domain and prime the chaperone activity. *Journal Molecular Biology*, 415:3–15 doi:10.1016/j.jmb.2011.10.038, 2012.

- [82] V. Tugarinov, P.M. Hwang, J.E. Ollerenshaw, and L.E. Kay. Cross-correlated relaxation enhanced h-1-c-13 nmr spectroscopy of methyl groups in very high molecular weight proteins and protein complexes. *Journal American Chemical Society*, 125:10420–10428 doi:10.1021/ja030153x, 2003.
- [83] V. Tugarinov, V. Kanelis, and L.E. Kay. Isotope labeling strategies for the study of high-molecular-weight proteins by solution nmr spectroscopy. *Nature Protocols*, 1:749–754 doi:10.1038/nprot.2006.101, 2006.
- [84] V. Tugarinov and L.E. Kay. Ile, leu, and val methyl assignments of the 723-residue malate synthase g using a new labeling strategy and novel nmr methods. *Journal American Chemical Society*, 125:13868–13878 doi:10.1021/ja030345s, 2003.
- [85] H. Valafar and J.H. Prestegard. Redcat: a residual dipolar coupling analysis tool. *Journal of Magnetic Resonance*, 167:228–41, 2004.
- [86] A. Varki. Biological roles of glycans. *Glycobiology*, 27:3–49, 2017.
- [87] K.A. Verba, R.Y.R. Wang, A. Arakawa, Y.X. Liu, M. Shirouzu, S. Yokoyama, and D.A. Agard. Structural biology atomic structure of hsp90-cdc37-cdk4 reveals that hsp90 traps and stabilizes an unfolded kinase. *Science*, pages 1542–1547 doi:10.1126/science.aaf5023, 2016.
- [88] S. Vestal. Fixed-priority sensitivity analysis for linear compute time models. *Software Engineering, IEEE Transactions*, 20(4):308–317, 1994.
- [89] H. Yagi, K.B. Pilla, A. Maleckis, B. Graham, T. Huber, and G. Otting. Three dimensional protein fold determination from backbone amide pseudocontact shifts generated by lanthanide tags at multiple sites. *Structure*, 21:883–890, 2013.
- [90] R. Yerraballi, R. Mukkamala, K. Maly, and H.A. Wahab. Issues in schedulability analysis of real-time systems. In *Real-Time Systems, 1995. Proceedings., Seventh Euromicro Workshop*, pages 87–92, June 1995.

- [91] F. Zhang. Sensitivity analysis of scaling factor for a subset of real-time tasks. In *Computer Technology and Development (ICCTD), 2010 2nd International Conference*, pages 22–26, November 2010.
- [92] F. Zhang, A. Burns, and S. Baruah. Sensitivity analysis for edf scheduled arbitrary deadline real-time systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2010 IEEE 16th International Conference*, pages 61–70, August 2010.
- [93] F. Zhang, A. Burns, and S. Baruah. Sensitivity analysis of the minimum task period for arbitrary deadline real-time systems. In *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium*, pages 101–108, December 2010.
- [94] W. Zhang, H. Xie, B. Cao, and A.M.K. Cheng. Energy-aware real-time task scheduling for heterogeneous multiprocessors with particle swarm optimization algorithm. *Mathematical problems in engineering*, 2014, article ID 287475.