

OPERATIONS RESEARCH APPROACHES FOR PROTEIN STRUCTURE PREDICTION IN LATTICE MODELS

by

Seyed Mojtaba Hosseini

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfilment of the Requirements for
the Degree of
Master of Science

in

Industrial Engineering



July, 2016

MSc Thesis

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Seyed Mojtaba Hosseini

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

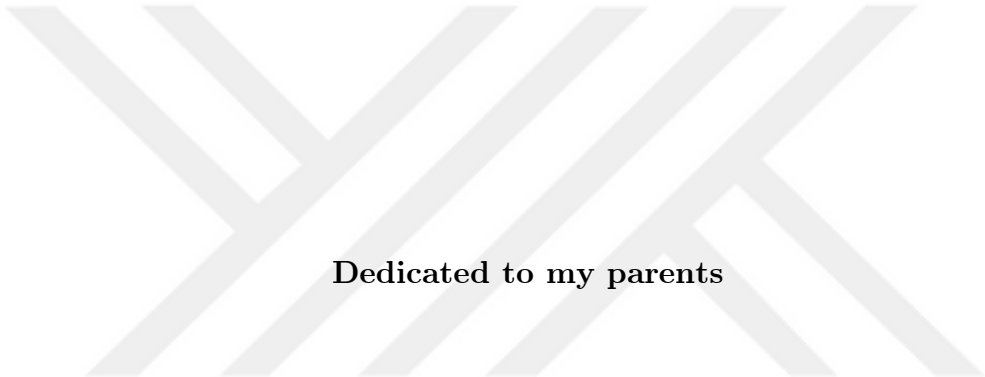
Committee Members:

Prof. Dr. Ceyda Oğuz

Prof. Dr. Emre Alper Yıldırım

Prof. Dr. Trkan Halilođlu

Date:



Dedicated to my parents

ABSTRACT

Protein structure prediction (PSP) consists of predicting the *native structure* of a protein from its sequence of *amino acids* by minimizing an *energy function*. The problem is of vital importance in medical science, molecular biology, biochemistry, and biophysics. PSP being *NP-hard*, even when abstracted to *lattice models*, is computationally challenging.

In this study, we develop several *mixed integer linear programming (MILP)* models for PSP problem under lattices, along with variety of *valid inequalities*, and two *symmetry breaking techniques*. We then propose three *metaheuristic* algorithms. While the focus of our study is on the *hydrophobic-polar (HP)* model under cubic and square lattices, next, we address some of the drawbacks of the HP model, by proposing extensions of our optimization methods to other more sophisticated PSP models. Finally, we evaluate the performance of these optimization methods with computational experiments.

We demonstrate that our MILP models outperform the state of the art models both in terms of running times in finding the optimal integer solutions, and in finding tight bounds on the objective value provided by linear programming (LP) relaxations of the models. We also show that the metaheuristic algorithms are able to find the optimal solutions for many benchmark instances. We then use the solution provided by these metaheuristic algorithms as initial solutions for the MILP models, and for constraining the feasible region. Integrated methods outperform significantly the state of the art models proposed for HP model in lattices. Finally, the computational results establish the robustness of our extended methods.

ÖZET

Bu çalışmada, enerji fonksiyonunun en aza indirgenerek, bir proteinin doğal yapısının, o proteinin amino asit dizilimi kullanılarak tahmin edilmesinden oluşan Protein Yapı Tahmini (PYT) problemi kafes model yapısında yöneylem araştırma bakış açısıyla ele alınmaktadır. Bu problem tıp, moleküler biyoloji, biyokimya ve biyofizik gibi bilim dalları için hayati önem taşımaktadır ve kafes model yapısına indirgendiğinde bile NP-Zor olup hesaplama açısından oldukça zorlayıcıdır.

Bu tezde, kafes model yapısındaki PYT problemi için çeşitli geçerli eşitsizlikler ve iki farklı simetri kırma yöntemiyle geliştirilen karışık tamsayılı doğrusal programlama (KTDP) modelleriyle birlikte, üç üstsezgisel algoritma sunulmuş ve bunların başarımları hesaplamalı deneylerle ölçülerek değerlendirilmiştir. Kübik ve kare kafes yapılarındaki hidrofobik-polar (HP) model, bu çalışmanın odak noktasını oluştururken, diğer karmaşık PYT modelleri için, bu tezde sunulan eniyileme yöntemlerinin daha kapsamlı halleri önerilerek, HP model ile ilgili bazı konular irdelenmiştir.

Önerilen KTDP modelleri, hem en iyi tamsayı çözümü bulmada, hem de bu modellerin doğrusal programlama (DP) gevşetmeleri için sıkı sınırlar bulmada, çalışma zamanı açısından, bilinen modellerden daha iyi sonuç vermiştir. Ayrıca, sunulan üstsezgisel algoritmaların, kıyaslama örneklerinin büyük bir kısmı için en iyi çözümleri bulabildiği gösterilmiştir. Daha sonra, üstsezgisel algoritmalarından elde edilen çözümler, matematiksel modeller için başlangıç çözümleri olarak kullanılmış; problemin özelliklerine göre, olurlu bölgeyi kısıtlamada bir yol olarak gösterilmiştir. Üstsezgisel algoritmalarla bütünleştirilerek sunulan matematiksel modeller, kare ve kübik kafes yapıdaki HP modeller için geliştirilen diğer matematiksel modellerden önemli ölçüde daha iyi sonuçlar vermiştir. Son olarak, diğer PYT modellerinin, sunulan eniyileme yöntemleri üzerindeki etkileri değerlendirilmiştir. Ayrıca, hesaplamalı sonuçlar sunulan yöntemlerin dayanıklılığını kanıtlamıştır.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks of gratitude to my advisor Prof. Ceyda Oğuz, without whom this research would not have been possible. I thank her for her patience, guidance, insights, and the strong support that helped me complete my thesis in the most complete way. I would like to extend my gratitude to the jury members Prof. Emre Alper Yıldırım and Prof. T rkan Halilođlu, and all of the professors who guided me through this new road and who gave me the opportunity to be part of the warm academic community of Ko University.

Secondly, I would like to thank all of my colleagues at Ko University, especially the past and present members of ENG 216. I would like to give special thanks to my friend Onur Dereli who generously translated the abstract to Turkish. Most importantly, I would like to thank my family and friends who helped me a lot in finalizing this thesis within the limited time frame.

TABLE OF CONTENTS

Chapter 1 Introduction.....	1
1.1 Protein and Native Structure.....	1
1.2 Protein Structure Prediction.....	3
1.3 Lattice Models.....	3
1.4 Hydrophobic Polar Model	5
1.5 Overview of the Thesis.....	6
Chapter 2 Literature Review	8
2.1 Exact Combinatorial Optimization Methods.....	8
2.2 Heuristic Methods	9
Chapter 3 Mathematical Models for Linear Chain HP Model in Cubic Lattice	12
3.1 Basic Geometric Model	19
3.2 Axis Assignment Model	25
3.3 Basic Location Assignment Model	28
3.4 Single Link Assignment Model.....	32
3.5 Double Link Assignment Model.....	36
3.6 Quadruple Neighborhood Assignment Model.....	42
Chapter 4 Metaheuristic Approaches for Linear Chain HP Model in Cubic Lattice	55
4.1 Tabu Search	56
4.2 Iterated Local Search	63
4.3 Scatter Search	66
4.4 Metaheuristic Approaches Combined with Mathematical Models.....	72
Chapter 5 Computational Results for Linear Chain HP Model in Cubic Lattice	74
5.1 Benchmark Instances	74
5.2 Computational Results for Mathematical Models.....	77
5.3 Computational Results for Metaheuristic Algorithms	95
5.1 Computational Results for Combination of Methods	97
Chapter 6 Drawbacks of Cubic HP Model and Solutions	101
6.1 Side-Chain HP Models under Cubic/Square Lattices.....	101
6.2 Linear Chain HP Models under Triangular Lattice.....	116
6.3 Extension to Other Energy Functions	119

Chapter 7 Conclusion And Future Work	127
Appendix I Proof of Lemma 1	129
Appendix II Column Generation for BLA	130
Appendix III Detailed Pseudocodes of Metaheuristic Algorithms	133
Appendix IV Partitioning a Set of Solutions	138
Appendix V Tuning Metaheuristic Algorithms	140
Appendix VI Most Distant Objects in a Cube	145
Bibliography	148



LIST OF TABLES

Table 1-1 Standard amino acid abbreviations and properties	2
Table 1-2 Properties of various regular lattices	4
Table 5-1 Standard benchmark instances for square lattice	75
Table 5-2 Standard benchmark instances for cubic lattice	75
Table 5-3 Samples of Z-Structure benchmarks	76
Table 5-4 Yue-Dill instances for cubic lattice	76
Table 5-5 Specifications of mathematical models in linear chain HP model under cubic and square lattices	77
Table 5-6 Performance of MILP models (square, half-length, central, S1-0)	80
Table 5-7 Performance of MILP models (square, half-length, central, S1-1)	81
Table 5-8 Performance of MILP models (square, half-length, central, S1-2)	81
Table 5-9 Performance of MILP models (square, half-length, central, S1-3)	82
Table 5-10 Performance of MILP models (square, half-length, central, S1-4)	83
Table 5-11 Performance of MILP models (square, half-length, central, S1-5)	83
Table 5-12 Performance of MILP models (square, quarter-length, central, S1-0)	84
Table 5-13 Performance of MILP models (square, quarter-length, central, S1-1)	85
Table 5-14 Performance of MILP models (square, quarter-length, central, S1-2)	85
Table 5-15 Performance of MILP models (square, quarter-length, central, S1-3)	86
Table 5-16 Performance of MILP models (square, quarter-length, central, S1-4)	87
Table 5-17 Performance of MILP models (square, quarter-length, central, S1-5)	87
Table 5-18 Comparative analysis of half- and quarter-length strategies in instance S1-4	88
Table 5-19 Sensitivity of LP relaxation, time, and number of variables of DLA and QNA models with the choice of m in square lattice	88
Table 5-20 Comparative analysis of performance of DLA and DLA-A under square lattice for five standard benchmark instances	89
Table 5-21 Comparison of MILP models with previous models in square lattice	90
Table 5-22 Performance of MILP models (cubic, half-length, central, S1-0)	91
Table 5-23 Performance of MILP models (cubic, quarter-length, central, S1-0)	91
Table 5-24 Performance of MILP models (cubic, half-length, terminal, S1-0)	91
Table 5-25 Performance of MILP models (cubic, quarter-length, terminal, S1-0)	91
Table 5-26 Performance of MILP models (cubic, half-length, central, S1-1)	92
Table 5-27 Performance of MILP models (cubic, quarter-length, central, S1-1)	92
Table 5-28 Performance of MILP models (cubic, half-length, terminal, S1-1)	92
Table 5-29 Performance of MILP models (cubic, quarter-length, terminal, S1-1)	92
Table 5-30 Performance of MILP models (cubic, half-length, central, S1-2)	92
Table 5-31 Performance of MILP models (cubic, quarter-length, central, S1-2)	93

Table 5-32 Performance of MILP models (cubic, half-length, terminal, S1-2).....	93
Table 5-33 Performance of MILP models (cubic, quarter-length, terminal, S1-2)	93
Table 5-34 Performance of MILP models (cubic, half-length, central, S1-3)	93
Table 5-35 Performance of MILP models (cubic, quarter-length, central, S1-3).....	93
Table 5-36 Performance of MILP models (cubic, half-length, terminal, S1-3).....	94
Table 5-37 Performance of MILP models (cubic, quarter-length, terminal, S1-3)	94
Table 5-38 Performance of MILP models (cubic, quarter-length, YD-1)	94
Table 5-39 Performance of MILP models (cubic, quarter-length, YD-2)	94
Table 5-40 Performance of MILP models (cubic, quarter-length, YD-3)	94
Table 5-41 Performance of metaheuristic algorithms on standard square benchmarks..	96
Table 5-42 Performance of metaheuristic algorithms on Z-structure instances.....	96
Table 5-43 Performance of metaheuristic algorithms on standard cubic benchmarks ...	97
Table 5-44 Performance of MILP models with initial solution (square, quarter-length)	98
Table 5-45 Performance of MILP models with initial solution (square, learning)	98
Table 5-46 Number of variables for different models under square lattice.....	99
Table 5-47 Performance of MILP models with initial solution (cubic, quarter-length, central)	99
Table 5-48 Performance of MILP models with initial solution (cubic, learning, central)	100
Table 6-1 Performance of MILP models for side-chain model (square,central,S1-0)....	111
Table 6-2 Performance of MILP models for side-chain model (square,terminal,S1-0) .	111
Table 6-3 Performance of MILP models for side-chain model (square,central,S1-1)....	112
Table 6-4 Performance of MILP models for side-chain model (square,terminal,S1-1) .	112
Table 6-5 Performance of MILP models for side-chain model (square,central,S1-2)....	113
Table 6-6 Performance of MILP models for side-chain model (square,terminal,S1-2) .	113
Table 6-7 Performance of MILP models for side-chain model (square,central,S1-3)....	114
Table 6-8 Performance of MILP models for side-chain model (square,terminal,S1-3) .	114
Table 6-9 Performance of MILP models for side-chain model (cubic,quarter,S1-0).....	115
Table 6-10 Performance of MILP models for side-chain model (cubic,quarter,S1-1) ...	115
Table 6-11 Performance of MILP models for side-chain model (cubic,quarter,S1-2) ...	115
Table 6-12 Performance of MILP models for side-chain model (cubic,quarter,S1-3) ...	116
Table 6-13 Performance of MILP models under triangular lattice.....	119
Table 6-14 Empirical contact energies for all pairs of standard amino acids as suggested by Berrera, Molinari & Fogolari (2003).....	122
Table 6-15 Instances of real proteins obtained from PDB	123
Table 6-16 Performance of DLAB on real proteins in square lattice (part 1).....	124
Table 6-17 Performance of DLAB on real proteins in square lattice (part 2).....	124
Table 6-18 Performance of ILS on real proteins in square and cubic lattices	124
Table 6-19 Performance of DLAB on real proteins in cubic lattice	125

Table 6-20 Distinguishable colors for amino acids obtained by maximizing the minimum similarity	125
Table 7-1 Performance of TS on S1-4, with <i>Pnoise</i> = 0	141
Table 7-2 Performance of TS on S1-4, with <i>Pnoise</i> = 0.1	142
Table 7-3 Performance of TS on S1-4, with <i>Pnoise</i> = 0.3	142
Table 7-4 Performance of TS on S1-4, with <i>Pnoise</i> = 0.5	142
Table 7-5 Performance of TS on S1-6, with <i>Pnoise</i> = 0	142
Table 7-6 Performance of TS on S1-6, with <i>Pnoise</i> = 0.1	143
Table 7-7 Performance of TS on S1-6, with <i>Pnoise</i> = 0.3	143
Table 7-8 Performance of TS on S1-6, with <i>Pnoise</i> = 0.5	143
Table 7-9 Performance of TS on S1-8, with <i>Pnoise</i> = 0	143
Table 7-10 Performance of TS on S1-8, with <i>Pnoise</i> = 0.1	144
Table 7-11 Performance of TS on S1-8, with <i>Pnoise</i> = 0.3	144

LIST OF FIGURES

Figure 1-1 Structure of an amino acid.....	3
Figure 1-2 Graph coloring problem only needs two colors for coloring cubic lattice	5
Figure 1-3 A solution in HP model for sequence $(HP)2PH2PHP2HPH2P2HPH$	6
Figure 3-1 Feasible region implied by terminal symmetry breaking in square lattice....	15
Figure 3-2 Feasible region implied by terminal symmetry breaking in cubic lattice.....	16
Figure 3-3 A sample solution in SLA for sequence $HP2H2PHP2HP2H3P2H$	33
Figure 3-4 Connectivity of amino acids of parity 0 and their predecessors in SLA	35
Figure 3-5 A sample solution in DLA for sequence $HP2H2PHP2HP2H3P2H$	37
Figure 3-6 Direction of contact between k and $k + 5$ is always the same as the forward link from $k + 2$ to $k + 3$ in square lattice	41
Figure 3-7 Feasible single contact combination, when $\psi_3 = -\psi_2 \wedge k - 1 - l_1 \geq 4$ (left), or $\psi_3 = -\psi_1 \wedge l_1 - k - 1 \geq 4$ (right)	47
Figure 3-8 Double contact combination with collinear (left) and perpendicular (right) forward backward links, when $l_1 > k + 1$	48
Figure 3-9 Double contact combination with collinear (left) and perpendicular (right) forward backward links, when $l_2 < k - 1$	49
Figure 3-10 Infeasible self-intersecting walk in double contact combinations, when $l_1 < k - 1 \wedge l_2 > k + 1 \wedge \psi_2 = -\psi_3$	49
Figure 3-11 Square double (left) and single (right) contact valid inequalities in QNA .	50
Figure 4-1 Possible moves used for linear chain lattice models in cubic lattice. Top left: corner move, top right: end move, bottom left: pull move, bottom right: crankshaft ...	60
Figure 4-2 A sample solution before and after perturbation in ILS	66
Figure 4-3 Limiting the size of problem by choosing m and L based on information provided by the initial solutions	73
Figure 5-1 Optimal solution for an instance of Z-structure class (Z8)	76
Figure 5-2 Sensitivity of LP relaxation of DLA and QNA to m in square lattice	79
Figure 6-1 A solution of side-chain HP PSP under cubic lattice for sequence S1-0.....	102
Figure 6-2 Square single contact valid inequalities in BSLA.....	106
Figure 6-3 Sample optimal solution in triangular lattice for sequence S1-0.....	117
Figure 6-4 Optimal solution obtained by ILS for proteins 1PSV (left) and 1LE0 (right) based on Berrera's energy matrix	125
Figure 7-1 Most distant objects in a cube of side length 255, analogous to most distinguishable colors.....	147

Chapter 1

INTRODUCTION

Proteins are fundamental elements to many cellular functions in all living beings. The biological function of a protein is in turn a function of the way it is folded into its three dimensional (3D) structure, therefore 3D structures of proteins are vital in *medical science, biology, biochemistry*, and studies of actions of drugs. Due to the interactions between proteins and drugs, determining the exact conformations of proteins helps in designing the drugs and therapies. Moreover, the 3D conformations help to detect the *misfolded* proteins, situations in which proteins are not folded into their correct 3D conformation. Misfolded proteins are known as the causes of variety of diseases such as Alzheimer's disease, Parkinson's disease, cystic fibrosis, mad cow disease, and even many cancers.

Many techniques exist for identifying the correct (or *native*) structures of proteins. One standard chemical method to determine the native structure of a protein consists of making a solution that only contains the protein, then crystallizing the protein, and finally an X-ray crystallography. A problem with this method is finding the suitable conditions under which the protein crystallizes properly. Another limit to this method is that some proteins (e.g. *membrane* proteins) do not crystallize. Finally, this procedure is too time consuming, and it might take years per protein. Another chemical method for obtaining information about the structure and dynamics of proteins is *Nuclear Magnetic Resonance (NMR)* spectroscopy. This method usually consists of several steps, each using a set of highly specialized techniques. Structure determination by NMR is also a very time consuming process, due to the need of interactive analysis of the data by highly trained scientists.

On the other hand, there are computer methods. For the abovementioned reasons, the problem of determining the structure of a protein using computer methods is one of the most important and challenging problems in *bioinformatics* and *computational biology*. On one hand, however, this problem has been shown to be *NP-hard* (Ngo et al., 1992), which indicates that finding an efficient algorithm for solving this problem is unlikely. On the other hand, general principles governing the folding of proteins into unique native structures are barely known. Therefore, simplified models have been proposed to help study the general principles of protein folding computationally.

1.1 Protein and Native Structure

Proteins are sequences of *amino acids*. Amino acids are chemical groups of the form in Figure 1-1, in which R, called *chain residue* (or *side-chain*), is a chemical group that

identifies the type of amino acid. The carbon atom connected to R is called *alpha-carbon* and is denoted by $C\alpha$. The NH_2 group connected to $C\alpha$ is called *amino group* or *N-terminus*, and the $COOH$ group connected to $C\alpha$ is called *carboxyl group* or *C-terminus*. In a protein, consecutive amino acids are connected to each other through *peptide bonds*, where amino group of one reacts with carboxyl group of the other. Amino acids are also called *monomers*, hence proteins are called *polymers*.

There are 20 standard types of amino acids, characterized by their chain residue. Amino acids may vary in size, side-chain charge, and *polarity/hydrophobicity*. An amino acid is *hydrophilic* or *polar*, if it tends to interact with the molecules of water, and is *hydrophobic* if it tends to avoid interaction with water. Amino acids are abbreviated in two ways: *3-Letter* and *1-Letter* abbreviations. The former is preferred for better readability, and the latter is used for encoding the proteins. Table 1-1 lists the 20 standard amino acids, along with their abbreviations (3-Letter and 1-Letter) and their polarity/hydrophobicity. We will later refer to this table when working with all 20 types of amino acids.

Table 1-1 Standard amino acid abbreviations and properties

Amino Acid	3-Letter	1-Letter	Hydrophobic (H)/Polar (P)
Alanine	Ala	A	H
Arginine	Arg	R	P
Asparagine	Asn	N	P
Aspartic acid	Asp	D	P
Cysteine	Cys	C	P
Glutamic acid	Glu	E	P
Glutamine	Gln	Q	P
Glycine	Gly	G	P
Histidine	His	H	P
Isoleucine	Ile	I	H
Leucine	Leu	L	H
Lysine	Lys	K	P
Methionine	Met	M	H
Phenylalanine	Phe	F	H
Proline	Pro	P	H
Serine	Ser	S	P
Threonine	Thr	T	P
Tryptophan	Trp	W	H
Tyrosine	Tyr	Y	P
Valine	Val	V	H

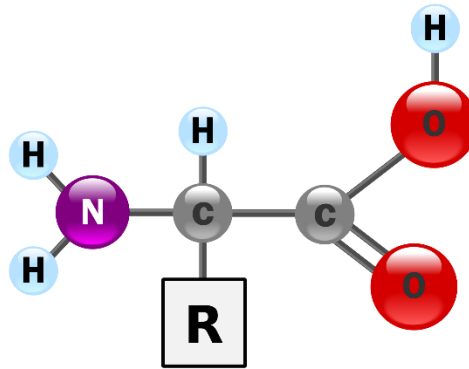


Figure 1-1 Structure of an amino acid

A special common class of proteins are the *globular* proteins. In globular proteins hydrophobic amino acids tend to be bounded towards the interior of the protein, while the polar (or hydrophilic) amino acids tend to bound towards the outer surface of the protein.

1.2 Protein Structure Prediction

Anfinsen (1973) introduces the *thermodynamic hypothesis* which states that the native 3D structure (called *tertiary structure*) that a protein folds into is the structure (also called *conformation*) with the lowest possible *free energy*. Anfinsen's principle states that, at least for the small globular proteins, the native structure of a protein can be determined by inter-atomic interactions, hence by knowing its sequence of amino acids (also known as its *primary structure*). Given the primary structure of a protein, the *Protein Structure Prediction (PSP)* is hence the problem of finding a 3D structure that minimizes the free energy of the protein.

1.3 Lattice Models

A common way of simplifying the PSP problem is to restrict the location of amino acids to *lattice* points of some lattice structure. There are two types of lattice models- *linear chain* models, and *side-chain* models. On one hand, linear chain models simplify the proteins by assuming that all amino acids are equally sized and shaped, hence represented as points, and their location in the 3D space is restricted to lattice points. Furthermore, in linear chain models the *bond length* (the bond between two consecutive C α atoms) is assumed to be unit, and it is assumed that the energy of a protein in a conformation can be measured as the sum of energies of all pairwise interactions (*contacts*) between amino acids in that conformation (Vendruscolo, Najmanovich & Domany, 2000). In these models, two nonconsecutive amino acids are said to be in contact, if they occupy adjacent lattice points. PSP in linear chain lattice models is therefore the problem of finding a

“self-avoiding walk” (SAW) into the lattice, which minimizes the total pairwise contact energies.

In the side-chain model, however, amino acids are represented by *backbone* nodes as the backbone of the protein, connected to a side-chain node. In other words, each monomer occupies two adjacent nodes in the lattice, one for backbone and the other for side-chain. Similar to linear chain model, the bond length between backbones, and between backbones and side-chains are assumed to be the same and set to unit. A conformation of the protein in this model is represented by a “self-avoiding manner” (SAM) of the backbones on the lattice points with side-chain nodes mapped to adjacent lattice points, such that no lattice point is occupied by more than one backbone node or side-chain node (Istrail & Lam, 2009). As before, the energy of a conformation is the total pairwise contact energies between amino acids.

Lattice models are studied under different regular lattices, both in two dimensional (2D) and 3D, or free form general graphs. Each regular lattice is characterized by a set of *basic vectors*, i.e. the vectors that determine the adjacent nodes of an arbitrary node in the lattice. Table 1-2 derived from Godzik, Kolinski, & Skolnick (1993) shows some properties of *Square*, *Triangular*, *Cubic*, *Face-Centered Cubic (FCC)*, *Body-Centered Cubic (BCC)*, and *Knight Walk (KW)* lattices. It is important to note that some lattices are *parity restricted* (or *bipartite*), i.e. walks of length $2q + 1$ cannot start from and return to the same location, for $q \in \mathbb{N}$. In other words, starting from and returning to a point is possible only if the length of the walk is even. Lemma 1 states this property for cubic lattice and is proven in Appendix I. Similar proofs can be given for other lattices. As illustrated in Figure 1-2, parity restriction of cubic lattice implies that the graph coloring problem only requires two colors for coloring this lattice.

Table 1-2 Properties of various regular lattices

Space	Name	Basic vectors	Number of basic vectors	Parity restricted
2D	Square	$(\pm 1, 0), (0, \pm 1)$	4	Yes
	Triangular	$(\pm 1, 0), \left(\pm \frac{1}{2}, \pm \frac{\sqrt{3}}{2}\right)$	6	No
3D	Cubic	$(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)$	6	Yes
	FCC	$(\pm 1, 0, \pm 1), (0, \pm 1, \pm 1), (\pm 1, \pm 1, 0)$	12	No
	BCC	$(\pm 1, \pm 1, \pm 1)$	8	Yes
	Knight Walk	$(\pm 2, 0, \pm 1), (\pm 2, \pm 1, 0)$	24	Yes
		$(0, \pm 2, \pm 1), (\pm 1, \pm 2, 0)$ $(0, \pm 1, \pm 2), (\pm 1, 0, \pm 2)$		

Lemma 1: Cubic lattice is parity restricted (proof is given in Appendix I)

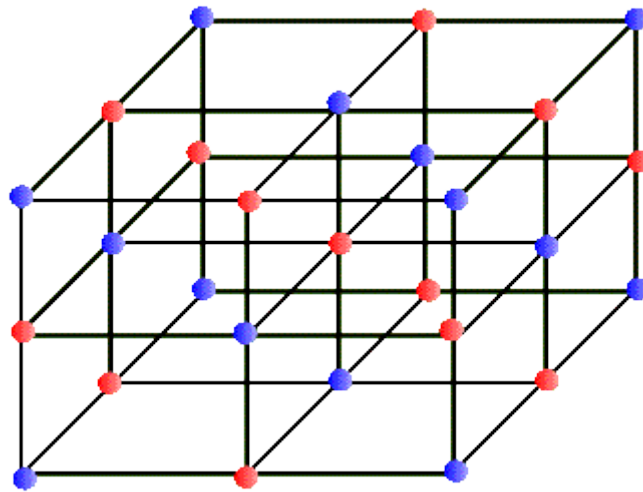


Figure 1-2 Graph coloring problem only needs two colors for coloring cubic lattice

1.4 Hydrophobic Polar Model

The simplest, and yet most studied, lattice model proposed for reducing the computational complexity of PSP problem is the *hydrophobic-polar (HP)* model introduced by Dill (1985) and further explained by Lau & Dill (1989) and Dill et al. (1995). This model is based on the observation that interactions between hydrophobic amino acids is the key element that contributes to the free energy of proteins, and hydrophobic force is the major force determining the unique native conformation of small globular proteins. While it might not be the case in general proteins, in globular proteins it is assumed that native conformations of globular proteins tend to form compact hydrophobic cores (Dill et al., 1995). In these proteins hydrophobic amino acids are covered by polar (or *hydrophilic*) amino acids, to avoid interactions with the solvent.

HP model reduces the primary 20-letter amino acid sequence of a protein to a two-letter sequence of hydrophobic (H) and polar (P) amino acids. Furthermore, in HP model the space is mainly reduced to a cubic (or square) lattice, and each contact (as defined in lattice models) between two hydrophobic amino acids (*H-H* contacts) contributes a constant value $-\epsilon$ to the energy function, while the contribution of other contacts (i.e. *H-P* and *P-P* contacts) to the energy function is zero. Therefore, the problem of finding a SAW/SAM that minimizes the total energy, becomes the problem of finding a SAW/SAM that maximizes the total number of H-H contacts. For this reason, throughout this thesis we interchangeably refer to the number of contacts as *absolute energy*. Despite the heavily simplified nature of HP model, it is still able to capture the essence of the important components of protein folding (Dill, 1990). Figure 1-3 shows an optimal SAW for the sequence $(HP)2PH2PHP2HPH2P2HPH$ with energy -11 (or

absolute energy 11), where the solid lines show the backbone bonds, and dashed lines represent the H-H contacts. Moreover, in this figure hydrophobic and polar amino acids are respectively represented by blue and red nodes.

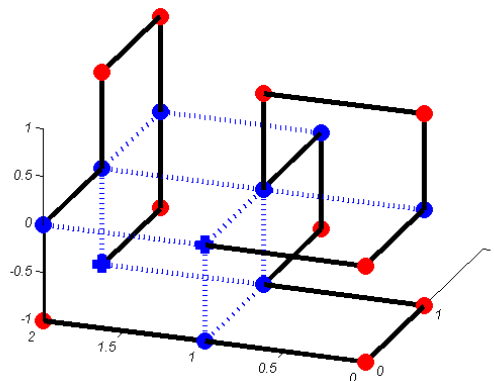


Figure 1-3 A solution in HP model for sequence $(HP)2PH2PHP2HPH2P2HPH$

1.5 Overview of the Thesis

In this thesis we analyze and introduce several operations research approaches to the PSP problem under lattice structures. Our focus is on the HP model under the cubic and square lattices. Our primary aim is to find a method that is able to find the global optimal solutions for the problem in relatively long proteins (around 60 amino acids), and near optimal solutions for longer proteins. Our computational results indicate that this goal is achievable by the methods proposed in this thesis. The thesis is organized as follows.

In Chapter 2, we present some of the well-known optimization techniques proposed for PSP problem in lattice models. The chapter is divided into two sections. In the first section, we review a variety of solution methods that aim to solve the PSP problem exactly. These methods include *total enumeration*, *mixed integer linear programming (MILP)* models, and *constraint programming (CP)* models. In the second section, we review some of the most important heuristic algorithms proposed for this problem.

Chapter 3, consists of our mathematical models for the conventional linear chain HP model under square and cubic lattices. For each model, we introduce *valid inequalities* and *symmetry breaking techniques* to improve their performance. We show by the computational experiments that these models, integrated with valid inequalities and symmetry breaking techniques provide the tightest bounds compared to other models previously proposed in the literature.

In Chapter 4, we introduce three metaheuristic approaches for finding near optimal solutions. First we adapt *Tabu Search (TS)* to the conventional linear chain HP model in both square and cubic lattices. We show how this algorithm can gain experience during the search and how this experience can be used inside other metaheuristic algorithms

such as *Iterated Local Search (ILS)* and *Scatter Search (SS)*, which use TS as a local search method. Followed by MILP models and metaheuristic methods, we propose methods for integrating these two approaches, so that the metaheuristic algorithms help MILP models to find the global optimal solutions in shorter time. It is observed that the solutions obtained by the metaheuristic methods provide tight lower bounds for the PSP problem. Due to the strong formulations of our models, the combined method starts with a small optimality gap, and is able to find the global solution (or verify optimality of the initial solutions) in a short time.

Chapter 3 and Chapter 4 are concluded by computational experiments presented in Chapter 5. Our computational results suggest the superior performance of our models, both in terms of running time and the upper bound on the number of contacts, to the previously proposed models. Moreover, metaheuristic approaches are able to find the optimal solutions for many of the standard benchmark instances, and are able to provide good solutions to the mathematical models. Our computational experiments also indicate that while solving sequences as short as 40 amino acids in the square lattice is challenging for the mathematical models, the integrated methods is able to find the optimal solution for sequences with up to 64 amino acids in a matter of seconds. Similarly, the integrated method shows superior performance to the plain mathematical models.

In Chapter 6, we point out three drawbacks of the conventional HP model, and extend our methods to overcome each of these drawbacks. First, we extend our mathematical models to the side-chain model. Next, we consider the HP model under triangular lattice, and extend one of our mathematical models to this lattice. Finally, we adapt the models and metaheuristics to the general energy functions, in which interactions between all 20 types of amino acids contribute to the energy function. In each of these sections, we conduct computational experiments to study the effect of each of these modifications on the performance of the methods proposed for the conventional HP model. Our computational results show the robustness of the extensions of our methods.

The thesis is concluded in Chapter 7, by presenting our conclusions and suggestion of directions for future studies. In conclusion, two of our MILP models strengthened using valid inequalities and symmetry breaking techniques outperform previously proposed mathematical models in finding tight upper bounds on the absolute energy using LP relaxations of the models. In some of the instances, our models are strong enough to find the integer solutions using the LP relaxations of the models. Integrated with metaheuristic algorithms, mathematical models outperform significantly the state of the art mathematical models proposed for HP model under square and cubic lattices. Therefore, this integrated method can be used as an alternative for finding the optimality guaranteed solutions for moderate sized instances of the problem.

Chapter 2

LITERATURE REVIEW

In this chapter we review some of the most important and state of the art methods proposed for the PSP problem in lattices. These methods can be viewed from two perspectives; exact combinatorial optimization methods, and heuristic algorithms. A third class of combinatorial methods is the approximation algorithms, which guarantee near optimal solutions in a polynomial time for all instances of the problem. In this thesis we focus our attention on the first two categories. This chapter is divided into two sections; first we introduce the exact combinatorial optimization methods, followed by heuristic approaches to this problem.

2.1 Exact Combinatorial Optimization Methods

The most trivial exact way of finding the global optimal solution is to enumerate over all SAWs, and evaluate the objective function. In the context of HP model, this is to count the number of H-H contacts. Jensen (2008) describes an algorithm for the enumeration of SAWs on the square lattice. Using 128 processors, the algorithm enumerates all SAWs of length up to 71. However, due to implementation issues and high cost of computation, even for the shortest sequences, complete enumeration methods are evidently inapplicable to the HP model under square lattice, let alone cubic and more complex lattices, with higher number of feasible SAWs. Kloczkowski et al. (2003) have adapted a similar concept to the HP model under square and cubic lattices.

Constraint-based Hydrophobic Core Construction (CHCC) is the first exact method proposed to the HP PSP problem by Dill et al. (1995). CHCC enumerates all possible conformations of the hydrophobic amino acids under the cubic lattice, called *hydrophobic cores*, and then tries to find location of amino acids such that they fit into the conformation. It is worth mentioning that this method has been shown to be incomplete by Backofen & Will (2006).

Integer programming (IP) formulations of the HP model have been suggested by Carr, Hart & Newman (2002) and extended by Greenberg, Hart & Lancia (2004). These formulations, which we will refer to as the *Basic Location Assignment (BLA)* models, are the cornerstones of some other formulations, as well as many of our methods. A similar formulation has been proposed by Yanev et al. (2011), in which they try to find an upper bound on the number of contacts between hydrophobic amino acids, by relaxing the integrality constraints of the model. As we will later discuss in the computational results of mathematical models, this model yields nothing more than the upper bound that can be obtained by simply counting the number of amino acids in the odd and even positions of the sequence. Another formulation inspired by BLA is proposed by Dittel,

Fügenschuh & Martin (2011), in which they rigorously study the polyhedral aspects of SAWs and propose a set of valid inequalities. They also test their results on some benchmark instances of HP sequences. Recently, an extension of BLA to the side-chain model has been proposed by Nunes et al. (2016).

Different formulations have been proposed as well. Yoon (2006) provides some optimization approaches to the protein folding problem. He introduces two IP models along with five *constraint programming* (CP) models. He also develops techniques to reduce the running time of the CP models. The IP formulations proposed by Yoon (2006) use fewer number of variables, but these models show poor performances due to the weakness of the formulations. Namsu & Park (2010) also propose a different model, which uses fewer number of variables, aiming to find an upper bound on the number of H-H contacts. Their model is yet very slow in solving both integer and fractional models due to the same reason.

A *Constraint Logic Programming* (CLP) approach to the problem under FCC lattice and based on empirical contact energies suggested by Berrera, Molinari & Fogolari (2003) is proposed by Dal Palu, Dovier & Fogolari (2004). In this method they consider all of 20×20 contacts between the amino acids. It is worth mentioning that they study this energy function under linear chain model. Mostly inspired by CHCC of Dill et al. (1995), Backofen & Will (2006) have proposed a CP method called *Constraint-based Protein Structure Prediction* (CPSP). CPSP is able to solve PSP under several lattice models, as well as more sophisticated energy functions. However, since their method highly depends on estimating the number of contacts and the hydrophobic core, it only works if a good estimation of the shape of the core is known. The method is therefore inapplicable to the variations of the problem in which all pairs of amino acids contribute to the energy function, such as the empirical contact energies suggested by Berrera, Molinari & Fogolari (2003).

2.2 Heuristic Methods

Second class of methods proposed for PSP under lattice models is heuristic methods. Due to NP-completeness of the simplest lattice model, i.e. HP model (Berger & Leighton, 1998), this area covers majority of the methods proposed for this problem. During the last twenty years, variety of heuristic algorithms both stochastic and deterministic have aimed to tackle this problem. In the scope of lattice models, these methods mainly address the HP model under square or cubic lattices.

Stochastic sampling methods such as *Monte Carlo* methods with *simulated annealing*, *genetic algorithms*, and *ant colony algorithm* are among the best known algorithms

proposed. One of the primary and most important metaheuristic approaches proposed for HP PSP under square lattice is the early *Genetic Algorithm* proposed by Uger & Moulton (1993). Along with the algorithm, they propose a set of benchmark instances that has been extensively used in the later proposed algorithms. They test their algorithm with these instances and show its superior performance over conventional Monte Carlo methods.

A heuristic method called *hydrophobic zipper (HZ)* is proposed by Dill et al. (1993), which is based on the assumption that hydrophobic contacts cannot be broken once they are formed, and the existing hydrophobic contacts constrain the formation of other parts. This method is combined with Monte Carlo procedure by Toma & Toma (1996) in the *Contact Interaction (CI)* algorithm, in which they treat each amino acid differently by assigning different conformational freedoms to each amino acid. Contrasted with HZ in which hydrophobic contacts become immobile after being formed, different freedoms allow CI to change previously formed contacts based on the assigned mobilities.

The *Core-directed chain Growth* method (*CdG*), proposed by Beutler & Dill (1996), is a heuristic method that uses a bias function to grow a chain towards forming a hydrophobic core. Another chain growth algorithm is proposed by Bauer (1997), in which they enumerate various SAWs using a “look-ahead” strategy that chooses only promising positions while growing the chain. This method extensively prunes the search space to decrease the computational time. However, this limited search space localizes the optimal solution obtained by this method.

Several Monte Carlo heuristics have been proposed for this problem, and have been shown to be effective in solving the HP model under square and cubic lattices. Among these methods are *pruned-enriched Rosenbluth method (PERM)* proposed by Grassberger (1997) and *replica exchange Monte Carlo* algorithm (*REMC*) proposed by Thachuk et al. (2007). PERM is a chain growth method that iteratively constructs solutions, by pruning the unpromising partial conformations. REMC on the other hand is a population based improvement method that maintains a number of independent replicas of a potential solution. A temperature is assigned to each of these replicas, and each replica performs a Monte Carlo search at the associated temperature.

An *Ant Colony Optimization (ACO-HPPFP-3)* metaheuristic is proposed by Shmygelska & Hoos (2005). This algorithm and REMC use a set of moves called *pull moves* and the so called *VHSD* moves. For more information on these moves the reader is referred to Section 4.1 of Chapter 4. While both ACO-HPPFP-3 and REMC are able to find the optimal solutions for standard benchmark instances of square and cubic lattices, REMC

outperforms both PERM and ACO-HPPFP-3 in terms of running time and robustness in finding the optimal solution of other benchmark instances.

A set of effective neighborhood search moves for the square lattice, called pull moves, is proposed by Lesh et al. (2003). They also propose a *Tabu Search (GTabu)* algorithm for the square lattice HP model. They show that pull moves are complete in the sense that any solution under the square lattice can be converted to any other solution using a series of these moves. They also localize three of long benchmarks instances.

Another Tabu Search algorithm for the HP model under square lattice has been proposed by Blazewicz et al. (2004), in which they alter angles between amino acids for defining neighborhoods and moves. Their method, however, fails to find good solutions for the long standard instances. A relatively new Tabu Search based algorithm called *Filter and Fan (F&F)* has been proposed by Rego et al. (2011) for the square lattice HP model. The method explores the solution space using pull moves in a dynamic way. The computational results obtained by this algorithm show the robustness and efficiency of this algorithm under the square lattice. The main factor in the short computational time of this algorithm is that unlike REMC and other evolutionary algorithms such as ACO-HPPFP-3, F&F requires only a single trial solution.

Chapter 3

MATHEMATICAL MODELS FOR LINEAR CHAIN HP MODEL IN CUBIC LATTICE

In this chapter we present HP PSP problem under cubic and square lattices using *mixed integer linear programming* (MILP) models. In order for a solution in lattice models to be a SAW, it is sufficient to satisfy three sets of constraints: 1) each amino acid must occupy exactly one lattice node (*assignment*), 2) A lattice node can be occupied by at most one amino acid (*non-overlapping*), and 3) consecutive amino acids must occupy adjacent nodes in the lattice (*connectivity*).

In our models, we will satisfy these constraints from three perspectives. First, in the so called *Basic Geometric Model* (BGM) we will view the problem from its geometric perspective; we introduce integer variables for the coordinates of the location of amino acids, as well as variables and constraints to define distance and contact between amino acids, and to maintain connectivity and self-avoiding. In the second model, which we refer to as the *Axis Assignment Model* (AAM), we “binarize” the variables associated with the coordinates. In the third and final class of models which is primarily inspired by the model first proposed by Carr, Hart & Newman (2002), we view the problem as a pure binary location assignment problem. In this class we further introduce several variations to make the models stronger.

Symmetries Associated with SAWs in Cubic Lattice

In many combinatorial optimization problems there exist *symmetries* inherent in the problems that, if not eliminated, can mire a *branch-and-bound* solver by allowing it to explore such alternative *symmetric solutions*, hence burdening it with unnecessary running time. In the PSP problem, three types of symmetries associated with SAWs in the lattices exist. The first type of symmetry is *translation*, i.e. adding a constant vector to the location of all amino acids does not change the actual conformation. In our models, we eliminate this symmetry by fixing the location of one of the amino acids at origin. We refer to this amino acid as the *middle amino acid*, and denote it by m . The second type of symmetry is *rotation*, i.e. rotating the location of all of amino acids around any of the axes x , y , or z does not change the actual conformation. The last type of symmetry is caused by *reflection*, i.e. reflecting the location of amino acids with respect to xy , yz , and zx planes does not change the conformation.

We propose two strategies for eliminating rotation and reflection symmetries. The first strategy, which we refer to it as the *central symmetry breaking* technique, is developed

for the first time in this thesis. The second technique, which we refer to as the *terminal symmetry breaking*, is introduced by Greenberg, Hart & Lancia (2004) for the square lattice, and we extend and generalize it to the cubic lattice.

Central symmetry breaking

In this strategy, after fixing the middle amino acid at origin, we further fix amino acid $m + 1$ at lattice point $(1,0,0)$. This simple restriction eliminates rotation symmetry around y and z axes. To eliminate the symmetry around x axis, we force the first amino acid on the left side of m that is not placed on the x axis to acquire $+1$ for the y -coordinate and 0 for the z -coordinate. Note that amino acid $k < m$ lies on the x axis, if and only if it is placed at the point $(k - m, 0, 0)$. Therefore, if all amino acids $l \in \{k + 1, \dots, m - 1\}$ lie on the x axis, according to this symmetry breaking technique, amino acid k must be located at either $(k - m, 0, 0)$ or $(k - m + 1, 1, 0)$. Therefore, excluding lattice points $Y_k = \{(k - m + 1, -1, 0), (k - m + 1, 0, \pm 1)\}$ from feasible lattice points of amino acid $k < m$ implies breaking this symmetry. Note that in rare cases in which all amino acids $k < m$ lie on the x axis, i.e. amino acid 0 is located at $(-m, 0, 0)$, there still exists symmetry around x axis. We will deal with this symmetry by adding further constraints to the models.

The only symmetry that we need to eliminate is reflection with respect to xy plane. Note that this symmetry is only possible in the cubic lattice. Before introducing techniques to eliminate this symmetry we first prove two rather simple properties of cubic lattice.

Proposition 1: If amino acid $k < m$ is located at lattice point $(x, y, 0)$ such that $|x| + |y| = m - k$, then all amino acids $l \in \{k + 1, \dots, m - 1\}$ must lie on the xy plane.

Proof: Note that amino acid $m - 1$ is restricted to xy plane. Due to connectivity of the chain, amino acid $k + 1$ can be located at one of the points $(x \pm 1, y, 0)$, $(x, y \pm 1, 0)$, or $(x, y, \pm 1)$. Note that distance of lattice point $(x, y, \pm 1)$ to the origin is $|x| + |y| + 1 = m - k + 1$. Since distance of amino acid $k + 1 < m$ to the m cannot exceed $m - k - 1$, it is impossible to place amino acid $k + 1$ at point $(x, y, \pm 1)$, hence the only feasible points for $k + 1$ are $(x \pm 1, y, 0)$ and $(x, y \pm 1, 0)$ which lie on the xy plane. ■

Corollary 1: If amino acid $k < m$ is located at $(x, y, -1)$, for which $|x| + |y| = m - k - 1$, then either amino acid $k + 1$ lies on the plane $z = -1$, or all amino acids $l \in \{k + 1, \dots, m - 1\}$ lie on the plane $z = 0$.

Proof: Without loss of generality assume that $x \geq 0$ and $y \geq 0$. Due to connectivity of the chain, amino acid $k + 1$ can be located at one of the points $(x \pm 1, y, -1)$, $(x, y \pm 1, -1)$, $(x, y, 0)$ or $(x, y, -2)$. Note that distance of lattice points $(x + 1, y, -)$, $(x, y + 1, -1)$ and $(x, y, -2)$ to the origin is $x + y + 2 = m - k + 1$, which makes these points infeasible for $k + 1$. If $k + 1$ is located at $(x - 1, y, -1)$ or $(x, y - 1, -1)$ then proof is complete, otherwise according to Proposition 1, all of amino acids $l \in \{k + 1, \dots, m - 1\}$ must lie inside the xy plane. ■

Now, to eliminate this symmetry, we require the first amino acid on the left of $m - 1$ that does not lie inside the xy plane to have +1 as the z-coordinate. Corollary 1 implies that if amino acid $k < m$ is located at $(x_k, y_k, -1)$, for which $|x_k| + |y_k| = m - k - 1$, none of amino acids $l \in \{k + 1, \dots, m - 1\}$ can obtain +1 value for z-coordinate. Therefore, one simple way for partially eliminating this symmetry is to exclude $\Phi_k = \{(x, y, -1) \in \mathbb{Z}^3: |x| + |y| = m - k - 1\}$ from feasible lattice points of amino acid $k < m - 1$. This, however, is not sufficient for fully eliminating reflection with respect to the xy plane for two reasons. First, if all amino acids $l \in \{k + 1, \dots, m - 1\}$ lie on the xy plane, but amino acid $k + 1$ is located at some point $(x_{k+1}, y_{k+1}, 0)$ for which $|x_{k+1}| + |y_{k+1}| < m - k - 1$, then amino acid k may obtain -1 as the z-coordinate. Second, in rare cases, if all amino acids $k < m$ lie on the xy plane, this symmetry may still occur. We will deal with these symmetries by adding further constraints and variables to the models.

Terminal symmetry breaking

First we introduce this technique in the square lattice, and then we explain how it can be extended to the cubic lattice. After fixing amino acid m at the origin, in order to eliminate rotational symmetries, we can limit the feasible lattice points of the last (or first) amino acid to one of the quadrants, say quadrant I, i.e. $x_{N-1} \geq 0$ and $y_{N-1} \geq 0$. Moreover, in order to eliminate the reflection with respect to line $x = y$, we can further limit the feasible points to one of the half-quadrants in quadrant I, say $x \geq y$. Although not stated in Greenberg, Hart & Lancia (2004), this restriction imposes restrictions on the feasible points of other amino acids as well. Let us denote by t_k , the distance of amino acid k to the amino acid $N - 1$ along the sequence, i.e. $t_k = N - 1 - k$. In general for amino acid k , this restriction along with the connectivity of chain imply that $x_k \geq -t_k$, $y_k \geq -t_k$, and $x_k \geq y_k - t_k$. Also note that if amino acid k is placed in the third quadrant, then its distance to origin cannot be greater than t_k , therefore $-x_k - y_k \leq t_k$, or equivalently $x_k \geq -y_k - t_k$. As shown in Figure 3-1, since $x_k \geq y_k - t_k$ and $x_k \geq$

$-y_k - t_k$ imply that $x_k \geq -t_k$, we can define the general form of the constraints as $x_k \geq y_k - t_k$, $x_k \geq -y_k - t_k$ and $y_k \geq -t_k$.

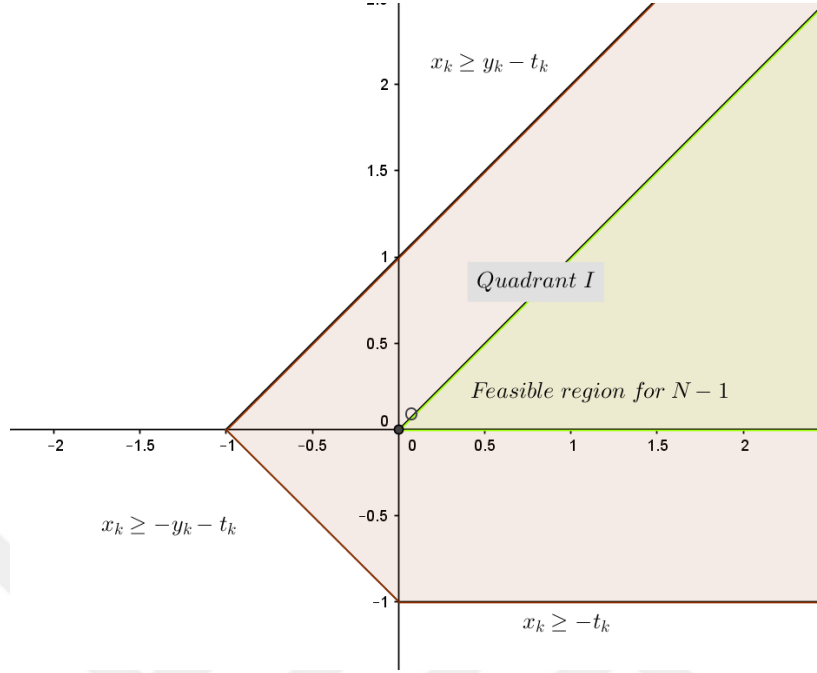


Figure 3-1 Feasible region implied by terminal symmetry breaking in square lattice

Similar to square lattice, to eliminate rotational symmetries, we can limit the feasible lattice points of the last (or first) amino acid to one of the octants, say octant I (top-front-right), i.e. $x_{N-1} \geq 0$, $y_{N-1} \geq 0$, and $z_{N-1} \geq 0$. Now we need to further eliminate the reflection with respect to planes $x = y$, $x = z$, and $y = z$. Note that eliminating reflection with respect to two of these planes also eliminates reflection with respect to the other plane. In fact, eliminating these symmetries is equivalent to finding an order in x , y , and z coordinates of the terminus. Without loss of generality, we choose the order $x \geq y \geq z$. Figure 3-2 illustrates the feasible region for terminus amino acid in cubic lattice. Similar to square lattice, we can generalize this restriction to any amino acid k as $x_k \geq -t_k$, $y_k \geq -t_k$, $z_k \geq -t_k$, $x_k \geq y_k - t_k$, $y_k \geq z_k - t_k$, and $x_k \geq z_k - t_k$. Furthermore, note that if k is placed in octant VII, then its distance to the origin (point A in Figure 3-2) cannot exceed t_k , i.e. $x_k + y_k + z_k \geq -t_k$.

Due to fixing the middle amino acid at the origin, the feasible lattice points for other amino acids resemble diamonds centered at the origin, henceforth we also refer to this solution space as the *diamond solution space*, contrasted with the *cube* and *square solution spaces* as presented in works like Nunes et al. (2016) and Yanev et al. (2011), in which they limit the solution space to a cube and square of predefined side length, respectively.

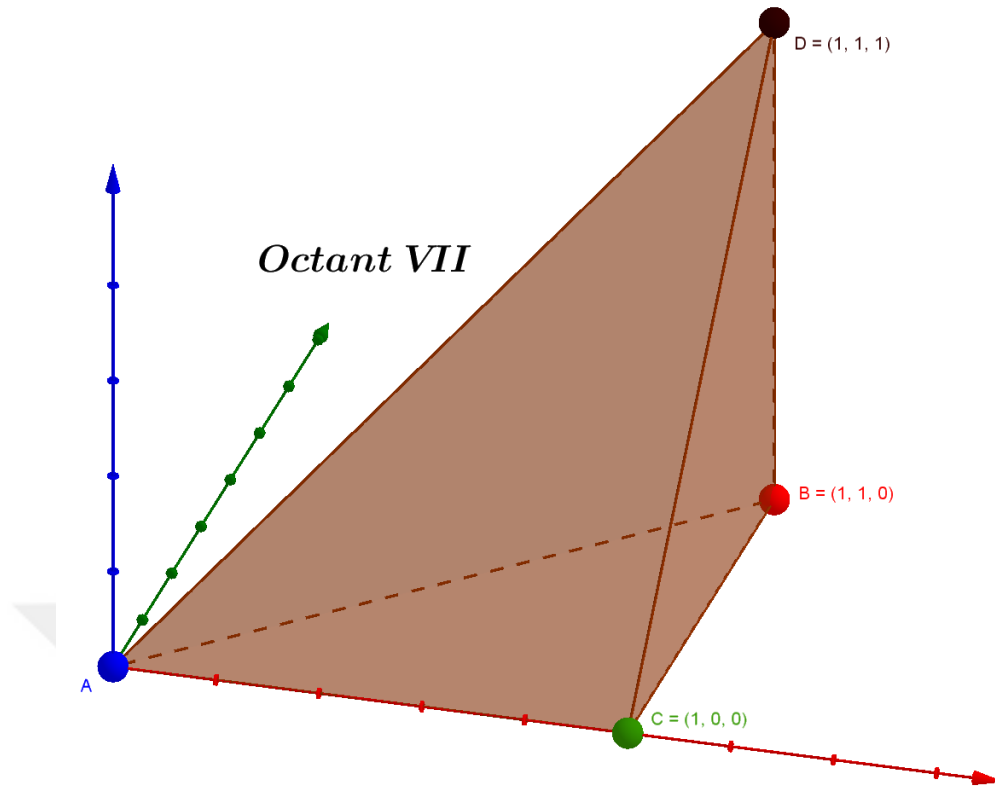


Figure 3-2 Feasible region implied by terminal symmetry breaking in cubic lattice

Notation

We use N and m to denote the number of amino acids in a protein, and the middle amino acid, respectively. In all of the models we set $m = \left\lfloor \frac{N-1}{2} \right\rfloor$, unless otherwise explicitly stated. By $\mathcal{R}$ we denote the set of 0-based indices of amino acids in the sequence, i.e. $\mathcal{R} = \{0, 1, \dots, N-1\}$. By t_k we denote the distance of amino acid k to the terminal amino acid $N-1$, i.e. $t_k = N-1-k$. We choose a limit for the *maximum distance* that amino acids can get from m , and denote it by L . In all of the models we set $L = \max\{m, t_m\}$, unless explicitly stated otherwise. ρ_k denotes hydrophobicity of amino acid k , and is equal to 1 if it is hydrophobic, and 0 if polar, for $k \in \mathcal{R}$. μ_k denotes parity of amino acid k , i.e. $\mu_k \equiv |k - m| \bmod 2$, $k \in \mathcal{R}$. By $\mathcal{R}^\theta$ we denote the set of amino acids of parity θ , i.e. $\mathcal{R}^\theta = \{k \in \mathcal{R} : \mu_k = \theta\}$. Note that $\mathcal{R} = \mathcal{R}^0 \cup \mathcal{R}^1$. Similarly, by $\mathcal{H}^\theta$ we denote the hydrophobic amino acids of parity $\theta \in \{0, 1\}$. By ∂_k we denote the maximum number of contacts that amino acid k can form in a feasible conformation. In cubic lattice, this value is 5 for terminal amino acids and 4 otherwise. For the square lattice, this value is 3 and 2, respectively.

By δ we denote the set of all *basic vectors* of cubic lattice, i.e. $\delta = \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$, and by δ_d we denote the d -th basic direction in δ , for $d = 1, \dots, |\delta|$. By δ^+ we denote the three basic vectors of nonnegative elements, i.e. $\delta^+ = \{\delta_1, \delta_2, \delta_3\}$, and we refer to them as the *standard basic vectors*. We use $\Delta_{p,q}$ to denote the *Manhattan distance* between points p and q , i.e. $\Delta_{p,q} = \sum_{d \in \delta^+} |(p - q) \cdot d|$, for $p, q \in \mathbb{Z}^3$. We use Δ with one index as the distance to the origin, i.e. $\Delta_p = \Delta_{p,(0,0,0)}$. We use μ_p with index of type point to denote the remainder of Δ_p divided by 2, i.e. $\mu_p \equiv \Delta_p \bmod 2$, for $p \in \mathbb{Z}^3$. We denote the set of all possibly feasible points by $\mathcal{F}$, i.e. $\mathcal{F} = \{p \in \mathbb{Z}^3 : \Delta_p \leq L\}$. Similar to $\mathcal{R}^\theta$, by $\mathcal{F}^\theta$ we denote the set of all possibly feasible points of parity θ , i.e. $\mathcal{F}^\theta = \{p \in \mathcal{F} : \mu_p = \theta\}$, for $\theta \in \{0, 1\}$. By F_k we denote the set of all feasible lattice points for amino acid k .

Fixing m at origin, along with the connectivity of the chain and parity restriction property of cubic lattice, implies that $F_k \subseteq \{p \in \mathcal{F}^{\mu_k} : \Delta_p \leq |k - m|\}$. Note that each of the symmetry breaking techniques further restrict F_k as follows:

i) **For central symmetry breaking technique:**

$$F_k = \begin{cases} \{(0,0,0)\}, & k = m; \\ \{(1,0,0)\}, & k = m + 1; \\ \{(0,1,0), (-1,0,0)\}, & k = m - 1; \\ \{p \in \mathcal{F}^{\mu_k} \setminus \{F_m \cup F_{m+1} \cup Y_k \cup \Phi_k\} : \Delta_p \leq m - k\}, & k < m - 1; \\ \{p \in \mathcal{F}^{\mu_k} \setminus \{F_m \cup F_{m+1}\} : \Delta_p \leq k - m\}, & k > m + 1. \end{cases}$$

In this symmetry breaking techniques, since we restrict the position of $m - 1$, some other marginal points are also cut from the feasible positions of amino acids preceding $m - 1$ in the sequence.

ii) **For terminal symmetry breaking technique:**

Fixing m at the origin defines $F_m = \{(0,0,0)\}$. According to the restrictions enforced by terminal symmetry breaking technique, for amino acid $k \neq m$, $F_k = \{p \in \mathcal{F}^{\mu_k} : \Delta_p \leq |m - k|, p_x \geq -t_k, p_y \geq -t_k, p_z \geq -t_k, p_x \geq p_y - t_k, p_y \geq p_z - t_k, p_x \geq p_z - t_k, p_x + p_y + p_z \geq -t_k\}$.

Strategies for Choosing m and L

For a fixed value of m , smaller value of maximum allowable distance (L) results in smaller feasible region for each amino acid. This in turn may reduce the computational time and

increase the quality of obtained solutions. Setting $L = \max\{m, t_m\}$ guarantees that the feasible region contains all possibly feasible conformations. Note that $m = \frac{N}{2}$ results in the minimum value for L , which is $\max\{\frac{N}{2}, N - 1 - \frac{N}{2}\} = \frac{N}{2}$. We refer to this choice of (m, L) as the “half-length” strategy. However, in many cases this choice of L makes the feasible region unnecessarily large, as some of the feasible solutions may never be optimal. Special cases of this observation are the globular proteins, as they tend to form compact conformations, containing hydrophobic amino acids at the core of the conformation. That is why in the presence of good initial solutions, it is reasonable to choose one of the amino acids in the center of the conformation as m , and its maximum distance to other amino acids plus a guarantee value (e.g. 2) as L (*learning strategy*).

This observation, however, cannot be extended to all imaginable sequences, especially sequences that do not represent globular proteins. As we will discuss in Chapter 5, there is a class of artificial sequences called *Z-Structure* sequences, which possesses unique *diagonal* optimal solutions under the square lattice. As shown in Figure 5-1, in these instances, if we set $m = \frac{N}{2}$, then any value for L less than $\max\{m, t_m\} = \frac{N}{2}$ would exclude the optimal conformation from the feasible region. Inspired by the optimal solution of *Z-Structure*, we note that setting $m = \frac{N}{4}$ and $L = \max\{m, \frac{N}{2} - m\}$, which we refer to as the “quarter-length” strategy, does not exclude the optimal solution of this class.

Conjecture 1: Choosing (m, L) based on the “quarter-length” strategy preserves at least one optimal solution for any sequence.

Note that this choice of m and L includes feasible solutions of objective function equal to 0 for any sequence, which implies that this reduced feasible region is large enough to avoid any contact between amino acids. Although it is not mathematically proven, this insight and our tests on several instances confirm that this choice of m and L includes at least one of the optimal conformations.

As we will discuss in Chapter 5, smaller feasible region does not necessarily result in a stronger model. In fact, as for benchmark S1-4, in spite of a more limited feasible region, quarter-length strategy results in worse results for some of the models, which is due to the accumulation of hydrophobic amino acids at the center of the sequence. Inspired by this observation, another strategy for selecting m and L can be introduced based on the sequence, which we refer to as *sequence-based* strategy. Since the location of hydrophobic amino acids of parity 0 mainly determines the quality of the models, in this strategy, we

should find m and L such that total number of possible lattice points for hydrophobic amino acids of parity 0 is minimized.

Let L_k denote the maximum distance that amino acid k can get from the middle amino acid, i.e. $L_k = \min\{L, |m - k|\}$. It is not hard to see that in the square lattice $|F_k| = \mathcal{O}(L_k^2)$ and in cubic lattice $|F_k| = \mathcal{O}(L_k^3)$. Therefore, one can choose m and L such that $\sum_{k \in \mathcal{H}^0} L_k^2$ in 2D and $\sum_{k \in \mathcal{H}^0} L_k^3$ in 3D is minimized. In this strategy, inspired by the evidence behind quarter-length strategy, we set L to $\max\{m, \frac{N}{2} - m\}$ if $m < \frac{N}{2}$ and to $\max\{N - 1 - m, m - \frac{N}{2}\}$ if $m > \frac{N}{2}$.

In the following section we present the Basic Geometric Model (BGM), which is the simplest and the closest model to the definition of SAWs in terms of definition of variables and constraints.

3.1 Basic Geometric Model

In this model we present the coordinates of locations of amino acids with integer variables, along with constraints and auxiliary variables to define distance and contacts between amino acids. Before representing the model, let us further investigate the maximum number of contacts that a sequence of amino acids can form under the cubic and square lattices.

Upper Bound on the Number of Contacts

Due to the parity restriction property of cubic and square lattices, we can form a *bipartite* graph G in which the first part of G contains hydrophobic amino acids of parity 0 ($\mathcal{H}^0$) and the other part consists of hydrophobic amino acids of parity 1 ($\mathcal{H}^1$). Let $G = (\mathcal{H}^0, \mathcal{H}^1, C)$, where $C \subset \mathcal{H}^0 \times \mathcal{H}^1$ is the set of edges. C consists of all pairs of nodes $k \in \mathcal{H}^0$ and $l \in \mathcal{H}^1$, unless k and l are consecutive in the sequence, i.e. $|k - l| = 1$. With this definition of G , finding the maximum number of contacts in the feasible solutions is equivalent to finding maximal $C_{HH} \subseteq C$ such that the degree of each node $k \in \mathcal{H}^0 \cup \mathcal{H}^1$ is at most ∂_k , that is the maximum number of contacts that amino acid k can form.

This problem can be represented as the following linear programming model:

$$\max \sum_{(k,l) \in C} Z_{kl} \quad (\text{BGM-1})$$

Subject to:

$$\sum_{(k,l) \in \mathcal{C}} Z_{kl} \leq \partial_k, \quad \forall k \in \mathcal{H}^0 \quad (\text{BGM-2})$$

$$\sum_{(k,l) \in \mathcal{C}} Z_{kl} \leq \partial_l, \quad \forall l \in \mathcal{H}^1 \quad (\text{BGM-3})$$

$$0 \leq Z_{kl} \leq 1, \quad \forall (k,l) \in \mathcal{C} \quad (\text{BGM-4})$$

We will use this model in the main model in the form of valid inequalities.

Decision Variables

The first set of decision variables, which we refer to as *coordinate variables*, includes integer variables that define the location of amino acids:

$$X_k^d: \text{location of amino acid } k \text{ along direction } d, \quad k \in \mathcal{R}, d \in \delta^+ \quad (\text{BGM-5})$$

The second set of decision variables, which we refer to as *sign indicators*, consists of binary variables that determine relative side of amino acid k with respect to amino acid l along the given direction:

$$S_{kl}^d = \begin{cases} 1, & X_k^d \geq X_l^d \\ 0, & X_k^d \leq X_l^d \end{cases}, \quad k < l \in \mathcal{R}, d \in \delta^+ \quad (\text{BGM-6})$$

Next sets of variables are binary variables, which we refer to as *distance indicators*, and respectively determine if distance of two amino acids k and l along the given direction is 0, 1, or more than 2:

$$D_{kl}^{0,d} = \begin{cases} 1, & X_k^d = X_l^d \\ 0, & \text{otherwise} \end{cases}, \quad k < l \in \mathcal{R}, d \in \delta^+ \quad (\text{BGM-7})$$

$$D_{kl}^{1,d} = \begin{cases} 1, & |X_k^d - X_l^d| = 1 \\ 0, & \text{otherwise} \end{cases}, \quad k < l \in \mathcal{R}, d \in \delta^+ \quad (\text{BGM-8})$$

$$D_{kl}^{2,d} = \begin{cases} 1, & |X_k^d - X_l^d| \geq 2 \\ 0, & \text{otherwise} \end{cases}, \quad k + 1 < l \in \mathcal{R}, d \in \delta^+ \quad (\text{BGM-9})$$

Finally, the last set of variables, which we refer to as *contact variables*, includes binary variables that determine if there exists a contact between two hydrophobic amino acids of different parities along a given direction:

$$H_{kl}^d = \begin{cases} 1, & D_{kl}^{1,d} = 1 \wedge D_{kl}^{0,q_1} = 1 \wedge D_{kl}^{0,q_2} = 1 \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\} \quad (\text{BGM-10})$$

where q_1 and q_2 are respectively the first and the second directions in $\delta^+ - \{d\}$. Note that in the cubic lattice, a contact is formed between two amino acids if they share the same values along two of the basic directions and adjacent values along the remaining direction.

Model (BGM)

$$\max \sum_{k \in \mathcal{H}^0} \sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} \sum_{d \in \delta^+} H_{kl}^d \quad (\text{BGM-11})$$

Subject to:

$$X_m^d = 0, \quad \forall d \in \delta^+, \quad (\text{BGM-12})$$

$$X_k^{\delta_1} + s_2 X_k^{\delta_2} + s_3 X_k^{\delta_3} \leq \min\{L, |m - k|\}, \quad \forall k \in \mathcal{R} - \{m \pm 1, m\}, s_2, s_3 \in \{-1, 1\}, \quad (\text{BGM-13})$$

$$X_k^{\delta_1} + s_2 X_k^{\delta_2} + s_3 X_k^{\delta_3} \geq \min\{L, |m - k|\}, \quad \forall k \in \mathcal{R} - \{m \pm 1, m\}, s_2, s_3 \in \{-1, 1\}, \quad (\text{BGM-14})$$

$$H_{kl}^d \leq D_{kl}^{1,d}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{BGM-15})$$

$$H_{kl}^d \leq D_{kl}^{0,q_1}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{BGM-16})$$

$$H_{kl}^d \leq D_{kl}^{0,q_2}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{BGM-17})$$

$$\sum_{i=0}^2 D_{kl}^{i,d} = 1, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-18})$$

$$\sum_{d \in \delta^+} D_{kl}^{0,d} \leq 2, \quad \forall k < l \in \mathcal{R}, \mu_k \neq \mu_l, \quad (\text{BGM-19})$$

$$D_{kl}^{0,q_1} + D_{kl}^{0,q_2} - 1 \leq D_{kl}^{2,d}, \quad \forall k < l \in \mathcal{R}, \mu_k = \mu_l, d \in \delta^+, \quad (\text{BGM-20})$$

$$\sum_{d \in \delta^+} D_{k-1,k}^{1,d} = 1, \quad \forall k \in \mathcal{R} - \{0\}, \quad (\text{BGM-21})$$

$$X_k^d - X_l^d \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d}, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-22})$$

$$X_l^d - X_k^d \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d}, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-23})$$

$$X_k^d - X_l^d \geq D_{kl}^{1,d} + 2D_{kl}^{2,d} - M_{kl}^{2,d}(1 - S_{kl}^d), \quad \forall k < l \in \mathcal{R}, \quad (\text{BGM-24})$$

$$X_l^d - X_k^d \geq D_{kl}^{1,d} + 2D_{kl}^{2,d} - M_{kl}^{2,d} S_{kl}^d, \quad \forall k < l \in \mathcal{R}, \quad (\text{BGM-25})$$

$$\sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} \sum_{d \in \delta^+} H_{kl}^d \leq \partial_k, \quad \forall k \in \mathcal{H}^0, \quad (\text{BGM-26})$$

$$\sum_{k \in \mathcal{H}^0 - \{l \pm 1\}} \sum_{d \in \delta^+} H_{kl}^d \leq \partial_l, \quad \forall l \in \mathcal{H}^1, \quad (\text{BGM-27})$$

$$\sum_{d \in \delta^+} H_{kl}^d \leq 1, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, \quad (\text{BGM-28})$$

$$\sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} H_{kl}^d \leq 2, \quad \forall k \in \mathcal{H}^0, d \in \delta^+, \quad (\text{BGM-29})$$

$$\sum_{k \in \mathcal{H}^0 - \{l \pm 1\}} H_{kl}^d \leq 2, \quad \forall l \in \mathcal{H}^1, d \in \delta^+, \quad (\text{BGM-30})$$

$$S_{jk}^d + S_{kl}^d - S_{jl}^d \leq 1, \quad \forall j < k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-31})$$

$$S_{jk}^d + S_{kl}^d \geq S_{jl}^d, \quad \forall j < k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-32})$$

$$D_{kl}^{0,d}, D_{kl}^{1,d}, S_{kl}^d \in \{0,1\}, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-33})$$

$$D_{kl}^{2,d} \in \{0,1\}, \quad \forall k+1 < l \in \mathcal{R}, d \in \delta^+, \quad (\text{BGM-34})$$

$$H_{kl}^d \in \{0,1\}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, \quad (\text{BGM-35})$$

$$X_k^d \in \mathbb{Z}, \quad \forall k \in \mathcal{R}, d \in \delta^+. \quad (\text{BGM-36})$$

Constraint set (BGM – 12) requires that the middle amino acid is located at lattice point (0,0,0). Constraint sets (BGM – 13) and (BGM – 14) guarantee that the maximum distance of location of amino acid k to (0,0,0) is at most $\min\{L, |m - k|\}$. Constraint sets (BGM – 15) to (BGM – 17) define the contact between two hydrophobic amino acids along standard basic direction d . That is, in order for two amino acids k and l to be adjacent along direction d , $|X_k^d - X_l^d|$ must be 1, and their values must be the same along two other directions. Note that due to the parity property of cubic lattice, only amino acids of opposite parities can form contacts. Furthermore, since consecutive hydrophobic amino acids cannot form a contact, the contact decision variables and constraints are not defined for such pairs. Note that in these constraints, the subscripts of the right hand side variables are swapped if $k > l$.

Constraint set (BGM – 18) indicates that amino acids k and l must take exactly one of the following values as the distance along the standard basic direction d : 0, 1, or more than 1. Constraint set (BGM – 19) requires that amino acids of different parities do not share the same location. Moreover, again due to the parity restriction property of cubic lattice, constraint set (BGM – 20) not only guarantees that two amino acids of equal parities do not share the same location, but also indicates that if they share the same coordinates along two basic direction q_1 and q_2 , then their distance along the other direction (d) must be at least 2. Constraint set (BGM – 21) ensures that consecutive amino acids are adjacent along exactly one of the standard basic directions. Constraint sets (BGM – 22) to (BGM – 25) define the relations between coordinate variables (X), distance indicators (D) and sign indicators (S). Proposition 2 proves correctness of this statement. In these constraints $M_{kl}^{1,d}$ and $M_{kl}^{2,d}$ are large enough positive numbers. Corollary 2 and Corollary 3 respectively present the least values of these numbers.

Constraint sets (BGM – 26) to (BGM – 30) are valid inequalities limiting the number of H-H contacts, derived from model (BGM – 1) – (BGM – 4). Constraints (BGM – 26) and (BGM – 27) state that each nonterminal hydrophobic amino acid can be in contact with at most ∂_k other nonconsecutive hydrophobic amino acids. Constraint set (BGM –

28) indicates that each pair of amino acids can form a contact along at most one of the standard basic directions. Constraint sets (BGM – 29) and (BGM – 30) indicate that each amino acid can form at most two contacts along each of the standard basic directions (one along positive and one along negative directions). Constraint sets (BGM – 31) and (BGM – 32) are simple valid inequalities derived from the transitive property of inequalities, i.e. if $X_k^d \geq X_l^d$ ($X_k^d \leq X_l^d$) and $X_j^d \geq X_k^d$ ($X_j^d \leq X_k^d$) then $X_j^d \geq X_l^d$ ($X_j^d \leq X_l^d$). We can also add the symmetry breaking techniques as additional constraints.

Proposition 2: Constraint sets (BGM – 22) to (BGM – 25) correctly define the relationship between coordinate variables (X), distance indicators (D) and sign indicators (S).

Proof: Let us define $\Delta_{kl}^d = X_k^d - X_l^d$, hence $-\Delta_{kl}^d = X_l^d - X_k^d$. With this notation (BGM – 22) to (BGM – 25) read:

$$\Delta_{kl}^d \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d} \quad (\text{BGM-37})$$

$$-\Delta_{kl}^d \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d} \quad (\text{BGM-38})$$

$$\Delta_{kl}^d \geq D_{kl}^{1,d} + 2D_{kl}^{2,d} - M_{kl}^{2,d} (1 - S_{kl}^d) \quad (\text{BGM-39})$$

$$-\Delta_{kl}^d \geq D_{kl}^{1,d} + 2D_{kl}^{2,d} - M_{kl}^{2,d} (1 - S_{kl}^d) \quad (\text{BGM-40})$$

First, we show that for given Δ_{kl}^d , distance and sign indicators obtain consistent values.

Case 1: If $\Delta_{kl}^d = 0$, then $D_{kl}^{0,d} = 0$ and S_{kl}^d can be either 0 or 1. If $S_{kl}^d = 1$, from (BGM – 39), and if $S_{kl}^d = 0$, from (BGM – 40) we have $0 \geq D_{kl}^{1,d} + 2D_{kl}^{2,d}$, hence $D_{kl}^{1,d} = D_{kl}^{2,d} = 0$. Therefore, from (BGM – 18) we have $D_{kl}^{0,d} = 1$. On the other hand, if $D_{kl}^{0,d} = 1$, from (BGM – 18) we have $D_{kl}^{1,d} = D_{kl}^{2,d} = 0$, therefore (BGM – 39) and (BGM – 40) are consistent with both values 0 and 1 for S_{kl}^d .

Case 2: If $\Delta_{kl}^d = 1$, then $D_{kl}^{1,d} = 1$ and $S_{kl}^d = 1$. Note that having $S_{kl}^d = 0$, (BGM – 39) would imply that $-1 \geq D_{kl}^{1,d} + 2D_{kl}^{2,d}$ which cannot be true, hence S_{kl}^d can only be 1. From (BGM – 37) we have $1 \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d}$, which implies that at least one of $D_{kl}^{1,d}$ or $D_{kl}^{2,d}$ must be nonzero. Note that since $S_{kl}^d = 1$, (BGM – 39) implies that $1 \geq D_{kl}^{1,d} + 2D_{kl}^{2,d}$, which cannot be true unless $D_{kl}^{2,d} = 0$, therefore $D_{kl}^{1,d} = 1$.

Case 3: If $\Delta_{kl}^d = -1$, then $D_{kl}^{1,d} = 1$ and $S_{kl}^d = 0$. Similar to case 2, having $S_{kl}^d = 1$, (BGM – 40) would imply that $-1 \geq D_{kl}^{1,d} + 2D_{kl}^{2,d}$ which cannot be true, hence S_{kl}^d can only be 0.

From (BGM – 38) we have $1 \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d}$, which implies that at least one of $D_{kl}^{1,d}$ or $D_{kl}^{2,d}$ must be nonzero. Note that since $S_{kl}^d = 0$, (BGM – 40) implies that $1 \geq D_{kl}^{1,d} + 2D_{kl}^{2,d}$, which cannot be true unless $D_{kl}^{2,d} = 0$, therefore $D_{kl}^{1,d} = 1$.

Case 4: If $\Delta_{kl}^d \geq 2$, then $D_{kl}^{2,d} = 1$ and $S_{kl}^d = 1$. From (BGM – 37) we have $2 \leq \Delta_{kl}^d \leq D_{kl}^{1,d} + M_1^d D_{kl}^{2,d}$, which cannot be true unless $D_{kl}^{2,d} = 1$, hence from (BGM – 21) we have $D_{kl}^{1,d} = 0$. As shown in case 2, (BGM – 39) implies that $S_{kl}^d = 1$.

Case 5: If $\Delta_{kl}^d \leq -2$, then $D_{kl}^{2,d} = 1$ and $S_{kl}^d = 0$. From (BGM – 38) we have $2 \leq -\Delta_{kl}^d \leq D_{kl}^{1,d} + M_{kl}^{1,d} D_{kl}^{2,d}$, which cannot be true unless $D_{kl}^{2,d} = 1$, hence from (BGM – 18) we have $D_{kl}^{1,d} = 0$. As shown in case 3, (BGM – 40) implies that $S_{kl}^d = 1$.

Now, we show that for given distance and sign indicators, Δ_{kl}^d obtains consistent value.

Case 6: If $D_{kl}^{0,d} = 1$, then $\Delta_{kl}^d = 0$. Since $D_{kl}^{0,d} = 1$, from (BGM – 21) we have $D_{kl}^{1,d} = D_{kl}^{2,d} = 0$, hence from (BGM – 37) and (BGM – 38) we have $0 \leq \Delta_{kl}^d \leq 0$, i.e. $\Delta_{kl}^d = 0$.

Case 7: If $D_{kl}^{1,d} = 1$ and $S_{kl}^d = 1$, then $\Delta_{kl}^d = 1$. Since $D_{kl}^{1,d} = 1$, from (BGM – 18) we have $D_{kl}^{2,d} = 0$, hence from (BGM – 37) and (BGM – 38) we have $-1 \leq \Delta_{kl}^d \leq 1$. From cases 1, 2, and 3, it follows that Δ_{kl}^d can only be 1.

Case 8: If $D_{kl}^{1,d} = 1$ and $S_{kl}^d = 0$, then $\Delta_{kl}^d = -1$. Similar to case 7 we have $-1 \leq \Delta_{kl}^d \leq 1$. Again, from cases 1, 2, and 3, it is deduced that Δ_{kl}^d can only be -1 .

Case 9: If $D_{kl}^{2,d} = 1$ and $S_{kl}^d = 1$, then $\Delta_{kl}^d \geq 2$. Since $D_{kl}^{2,d} = 1$, from (BGM – 18) we have $D_{kl}^{1,d} = 0$, hence from (BGM – 39) $\Delta_{kl}^d \geq 2$.

Case 10: If $D_{kl}^{2,d} = 1$ and $S_{kl}^d = 0$, then $\Delta_{kl}^d \leq -2$. Similar to case 9, since $D_{kl}^{2,d} = 1$, we have $D_{kl}^{1,d} = 0$, hence from (BGM – 40) $-\Delta_{kl}^d \geq 2$, or equivalently $\Delta_{kl}^d \leq -2$.

Note that any combination of values for these variables other than these cases will result in violating constraints, hence cases 1 to 10 prove this proposition ■

Now we introduce and prove how big $M_{kl}^{1,d}$ and $M_{kl}^{2,d}$ should be.

Corollary 2: In constraint sets (BGM – 22) and (BGM – 23), $M_{kl}^{1,d} = \min \left\{ l - k, \max_{p_k \in F_k, p_l \in F_l} |(p_k - p_l) \cdot d| \right\}$ is large enough.

Proof: Note that when $|\Delta_{kl}^d| \geq 2$, from Proposition 2 (case 4) (BGM – 37) and (BGM – 38) imply that $|\Delta_{kl}^d| \leq M_{kl}^{1,d}$. $M_{kl}^{1,d} \leq l - k$ directly follows from the fact that distance of two amino acids in the lattice cannot be greater than their distance in the sequence. Since $k < l$, this distance is $l - k$. On the other hand, distance of two amino acids must be less than the allowable distance according to their feasible lattice points, hence $M_{kl}^{1,d} \leq \max_{\substack{p_k \in F_k \\ p_l \in F_k}} |(p_k - p_l) \cdot d|$. Note that $p \cdot d$ is the inner product of lattice point p and direction d , which yields coordinate of point p along direction d .

Corollary 3: In constraint sets (BGM – 24) and (BGM – 25), $M_{kl}^{2,d} = M_{kl}^{1,d} + 2$.

Proof: Similar to Corollary 1, if $\Delta_{kl}^d \geq 2$, Proposition 2 (case 4) yields $D_{kl}^{2,d} = 1$, $D_{kl}^{1,d} = 0$, and $S_{kl}^d = 1$. Therefore, from (BGM – 40) we have $-\Delta_{kl}^d \geq 2 - M_{kl}^{2,d}$. On the other hand if $\Delta_{kl}^d \leq -2$, from (BGM – 39) we have $\Delta_{kl}^d \geq 2 - M_{kl}^{2,d}$. Therefore in any case it is necessary to have $M_{kl}^{2,d} \geq 2 + |\Delta_{kl}^d|$. Since the least upper bound on $|\Delta_{kl}^d|$ is $M_{kl}^{1,d}$, $M_{kl}^{2,d} = 2 + M_{kl}^{1,d}$ satisfies this requirement. ■

3.2 Axis Assignment Model

In *Axis Assignment Model* (AAM), we assign each amino acid to a value on each of axes (x, y, and z). Our aim is to strengthen BGM, while maintaining the low number of variables. The main difference between AAM and BGM is that location of amino acids along each basic direction is now represented using binary variables. The contact between two amino acids is defined the same way as in BGM. In this model, we define another set A_k^d for each amino acid k , which is the set of possible values it can acquire along the basic directions d , and in the most general case is equal to $A_k^d = \{x \in \mathbb{Z}: |x| \leq \min\{|k - m|, L\}\}$. Note that A_k^d might be different under different settings of the problem, and different symmetry breaking techniques.

Decision Variables

The first set of variables, which we refer to as the *axis assignment variables*, includes binary variables determining if amino acid k is assigned to given values along given basic direction:

$$X_k^{d,x} = \begin{cases} 1, & k \text{ is assigned to } x \text{ along direction } d \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{R}, d \in \delta^+, x \in A_k^d \quad (\text{AAM-1})$$

The next three sets of decision variables given in (AAM – 2) to (AAM – 4) are *distance indicators*, first of which is a set of binary variables that determines if two amino acids k and l share the same location along the given direction:

$$D_{kl}^{0,d} = \begin{cases} 1, & X_k^{d,x} = X_l^{d,x} = 1 \text{ for some } x \in A_k^d \cap A_l^d, \\ 0, & \text{otherwise} \end{cases}, \quad k < l \in \mathcal{R}, d \in \delta^+. \quad (\text{AAM-2})$$

The binary variables given as in (AAM – 3) are defined for all amino acids, except the first amino acid in the sequence, and determine whether the amino acid is adjacent to its successor in the sequence along the given direction:

$$D_k^{1,d} = \begin{cases} 1, & X_k^{d,x} = X_{k-1}^{d,x+1} + X_{k-1}^{d,x-1} = 1 \text{ for some } x \in A_k^d, \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{R} - \{0\}, d \in \delta^+. \quad (\text{AAM-3})$$

The last set of distance indicators determines if two nonconsecutive hydrophobic amino acids k and l are adjacent along the given direction:

$$DH_{kl}^{1,d} = \begin{cases} 1, & X_k^{d,x} = X_l^{d,x+1} + X_l^{d,x-1} = 1 \text{ for some } x \text{ such that} \\ & x \in A_k^d \wedge (x+1 \in A_l^d \vee x-1 \in A_l^d) \\ 0, & \text{otherwise} \end{cases}, \quad (\text{AAM-4})$$

$$k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+.$$

Finally, the last set of binary variables determines whether there exists a contact between two nonconsecutive hydrophobic amino acids k and l :

$$H_{kl}^d = \begin{cases} 1, & DH_{kl}^{1,d} = 1 \wedge D_{kl}^{0,q_1} = 1 \wedge D_{kl}^{0,q_2} = 1, \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+. \quad (\text{AAM-5})$$

Model (AAM)

$$\max \sum_{k \in \mathcal{H}^0} \sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} \sum_{d \in \delta^+} H_{kl}^d \quad (\text{AAM-6})$$

Subject to:

$$X_m^{d,0} = 1, \quad \forall d \in \delta^+, \quad (\text{AAM-7})$$

$$\sum_{x \in A_k} X_k^{x,d} = 1, \quad \forall k \in \mathcal{R}, d \in \delta^+, \quad (\text{AAM-8})$$

$$\sum_{d \in \delta^+} D_{kl}^{0,d} \leq 2, \quad \forall k < l \in \mathcal{R}, \quad (\text{AAM-9})$$

$$X_k^{d,x} \leq X_{k-1}^{d,x} + X_{k-1}^{d,x-1} + X_{k-1}^{d,x+1}, \quad \forall k \in \mathcal{R} - \{0\}, d \in \delta^+, x \in A_k^d, \quad (\text{AAM-10})$$

$$X_k^{d,x} + X_l^{d,x} - D_{kl}^{0,d} \leq 1, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, x \in A_k^d \cap A_l^d, \quad (\text{AAM-11})$$

$$X_k^{d,x} - X_l^{d,x} + D_{kl}^{0,d} \leq 1, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, x \in A_k^d \cap A_l^d, \quad (\text{AAM-12})$$

$$-X_k^{d,x} + X_l^{d,x} + D_{kl}^{0,d} \leq 1, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, x \in A_k^d \cap A_l^d, \quad (\text{AAM-13})$$

$$X_k^{d,x} + X_l^{d,x-1} + X_l^{d,x+1} - DH_{kl}^{1,d} \leq 1, \quad (\text{AAM-14})$$

$$\forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, x \in A_k^d, x \pm 1 \in A_l^d,$$

$$X_k^{d,x} - X_l^{d,x-1} - X_l^{d,x+1} + DH_{kl}^{1,d} \leq 1, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, x \in A_k^d, \quad (\text{AAM-15})$$

$$-X_k^{d,x-1} - X_k^{d,x+1} + X_l^{d,x} + DH_{kl}^{1,d} \leq 1, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, x \in A_l^d, \quad (\text{AAM-16})$$

$$X_k^{d,x} + X_{k-1}^{d,x-1} + X_{k-1}^{d,x+1} - D_k^{1,d} \leq 1, \quad \forall k \in \mathcal{R} - \{0\}, d \in \delta^+, x \in A_k^d, x \pm 1 \in A_{k-1}^d, \quad (\text{AAM-17})$$

$$X_k^{d,x} - X_{k-1}^{d,x-1} - X_{k-1}^{d,x+1} + D_k^{1,d} \leq 1, \quad \forall k \in \mathcal{R} - \{0\}, d \in \delta^+, x \in A_k^d, \quad (\text{AAM-18})$$

$$-X_k^{d,x-1} - X_k^{d,x+1} + X_{k-1}^{d,x} + D_k^{1,d} \leq 1, \quad \forall k \in \mathcal{R} - \{0\}, d \in \delta^+, x \in A_{k-1}^d, \quad (\text{AAM-19})$$

$$\sum_{d \in \delta^+} D_k^{1,d} = 1, \quad \forall k \in \mathcal{R} - \{0\}, \quad (\text{AAM-20})$$

$$D_{k-1,k}^{0,d} + D_k^{1,d} = 1, \quad \forall k \in \mathcal{R} - \{0\}, d \in \delta^+, \quad (\text{AAM-21})$$

$$X_k^{\delta_1,x} + X_k^{\delta_2,y} + \sum_{z \in A_k: (x,y,z) \notin F_k} X_k^{\delta_3,z} \leq 2, \quad \forall k \in \mathcal{R}, x \in A_k^{\delta_1}, y \in A_k^{\delta_2}, \quad (\text{AAM-22})$$

$$X_k^{\delta_3,z} + X_k^{\delta_1,x} + \sum_{y \in A_k: (x,y,z) \notin F_k} X_k^{\delta_2,y} \leq 2, \quad \forall k \in \mathcal{R}, x \in A_k^{\delta_1}, z \in A_k^{\delta_3}, \quad (\text{AAM-23})$$

$$X_k^{\delta_2,y} + X_k^{\delta_3,z} + \sum_{x \in A_k: (x,y,z) \notin F_k} X_k^{\delta_1,x} \leq 2, \quad \forall k \in \mathcal{R}, z \in A_k^{\delta_3}, y \in A_k^{\delta_2}, \quad (\text{AAM-24})$$

$$H_{kl}^d \leq DH_{kl}^{1,d}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{AAM-25})$$

$$H_{kl}^d \leq D_{kl}^{0,q_1}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{AAM-26})$$

$$H_{kl}^d \leq D_{kl}^{0,q_2}, \quad \forall k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{AAM-27})$$

$$\sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} \sum_{d \in \delta^+} H_{kl}^d \leq \partial_k, \quad \forall k \in \mathcal{H}^0, \quad (\text{AAM-28})$$

$$\sum_{k \in \mathcal{H}^0 - \{l \pm 1\}} \sum_{d \in \delta^+} H_{kl}^d \leq \partial_l, \quad \forall l \in \mathcal{H}^1, \quad (\text{AAM-29})$$

$$\sum_{d \in \delta^+} H_{kl}^d \leq 1, \quad k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, \quad (\text{AAM-30})$$

$$\sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} H_{kl}^d \leq 2, \quad \forall k \in \mathcal{H}^0, d \in \delta^+, \quad (\text{AAM-31})$$

$$\sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} H_{kl}^d \leq 2, \quad \forall l \in \mathcal{H}^1, d \in \delta^+, \quad (\text{AAM-32})$$

$$X_k^{d,x} \in \{0,1\}, \quad \forall k \in \mathcal{R}, d \in \delta^+, x \in A_k^d, \quad (\text{AAM-33})$$

$$D_{kl}^{0,d} \in \{0,1\}, \quad \forall k < l \in \mathcal{R}, d \in \delta^+, \quad (\text{AAM-34})$$

$$D_k^{1,d} \in \{0,1\}, \quad \forall k \in \mathcal{R} - \{0\}, d \in \delta^+, \quad (\text{AAM-35})$$

$$DH_{kl}^{1,d}, H_{kl}^d \in \{0,1\}, \quad k \in \mathcal{H}^0, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta^+, \quad (\text{AAM-36})$$

Many of the constraints in AAM have equivalent constraints in BGM, and here we only explain different constraints. Constraint set (AAM – 8) ensures that exactly one of the values along each basic direction is chosen for each amino acid. Constraint sets (AAM – 11) to (AAM – 19) define the relation between the axis assignment variables and the distance indicators. Constraint set (AAM – 10) together with constraint sets (AAM – 20) and (AAM – 21) maintain the connectivity of the sequence. Constraint sets (AAM – 22) to (AAM – 24) are valid inequalities preventing amino acids to be assigned to lattice points out of their set of feasible points defined using symmetry breaking techniques.

3.3 Basic Location Assignment Model

In this model, we further discretize the positions, this time assigning each amino acid to a lattice point. *Basic Location Assignment (BLA)* is primarily introduced in the work of Carr, Hart & Newman (2002) and then further explained in Greenberg, Hart & Lancia (2004). Ever since, models of location assignment class have been introduced in further works such as in Dittel, Fügenschuh & Martin (2011) and Yanev et al. (2011). Since this model is a cornerstone to our work, in this section we first introduce the basic model using our notations and feasible sets obtained by our symmetry breaking techniques. Then we introduce the valid inequalities and techniques that further strengthen the model. Finally, we will introduce more sophisticated models along with the valid inequalities associated with them in *Single Link Assignment (SLA)*, *Double Link Assignment (DLA)* and *Quadruple Neighborhood Assignment (QNA)* models.

Decision Variables

The first set of decision variables, which we refer to as *location assignment variables*, contains binary variables determining whether amino acid k is assigned to lattice point p in its feasible set of lattice points:

$$x_p^k = \begin{cases} 1, & k \text{ occupies } p \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{R}, p \in F_k. \quad (\text{BLA-1})$$

The second set of binary variables determines if a contact exists between a feasible lattice point p of parity 0, and a feasible lattice point of parity 1, which is adjacent to p along basic direction d .

$$W_p^d = \begin{cases} 1, & (p, p+d) \text{ are occupied by H amino acids} \\ 0, & \text{otherwise} \end{cases}, \quad (\text{BLA-2})$$

$$p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1.$$

Model (BLA)

$$\max \sum_{p \in \mathcal{F}^0} \sum_{d \in \delta: p+d \in \mathcal{F}^1} W_p^d \quad (\text{BLA-3})$$

Subject to:

$$\sum_{p \in F_k} X_p^k = 1, \quad \forall k \in \mathcal{R}, \quad (\text{BLA-4})$$

$$\sum_{\substack{k \in \mathcal{R}: \\ p \in F_k}} X_p^k \leq 1, \quad \forall p \in \mathcal{F}, \quad (\text{BLA-5})$$

$$X_p^{k+1} + X_p^{k-1} \leq \sum_{d \in \delta: p+d \in F_k} X_{p+d}^k, \quad k \in \mathcal{R}, p \in F_{k+1} \cup F_{k-1}^1, \quad (\text{BLA-6})$$

$$W_p^d \leq \sum_{k \in \mathcal{H}^0: p \in F_k} X_p^k, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{BLA-7})$$

$$W_p^d \leq \sum_{k \in \mathcal{H}^1: p+d \in F_k} X_{p+d}^k, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{BLA-8})$$

$$X_p^k \in \{0,1\}, \quad k \in \mathcal{R}, p \in F_k, \quad (\text{BLA-9})$$

$$W_p^d \in \{0,1\}, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1. \quad (\text{BLA-10})$$

Objective function (BLA – 3) counts the number of pairs of adjacent lattice points that are both occupied by hydrophobic amino acids. Constraint set (BLA – 4) ensures that exactly one of the feasible points of amino acids is chosen. Constraint set (BLA – 5) prevents assigning more than one amino acid to each lattice point. Constraint set (BLA – 6) which we refer to as the strong connectivity constraints, guarantees connectivity of consecutive amino acids. These constraints are strong in the sense that the left side of the constraints can be any of X_p^{k+1} or X_p^{k-1} (weak connectivity constraints), without losing connectivity. Constraint sets (BLA – 7) and (BLA – 8) indicate that in order to have an H-H contact (or link) on the edge $(p, p+d)$, there must be hydrophobic amino acids located at each of its ends.

¹ If any of the variables are not defined in these domains, we replace them with 0 in the constraint.

With these sets of variables, we try to maximize the number of edges whose both ends are occupied by hydrophobic amino acids. We note that $(BLA - 3)$ does not distinguish between consecutive and nonconsecutive hydrophobic pairs. However, this does not affect the optimal solution, as the optimal value will be equal to the absolute value of energy plus total number of sequential H-H bonds. Let S_{fb}^{HH} denote the number of sequential H-H bonds. Since this value is constant for a given sequence, maximizing number of H-H contacts and bonds also maximizes the absolute value of energy, hence:

$$|E^*| = \max \sum_{p \in F^0} \sum_{d \in \delta: p+d \in F^1} W_p^d - S_{fb}^{HH}. \quad (BLA-11)$$

Symmetry Breaking Variables and Constraints

As explained at the beginning of this chapter, we introduce techniques to eliminate some of the symmetric solutions by limiting the feasible region. In the central technique, not all of the symmetries can be eliminated by confining the feasible lattice points of amino acids. Here we try to eliminate other symmetries by introducing constraints and auxiliary variables. In order to eliminate rotation around the x axis, starting from amino acid $m - 1$ through amino acid 0 , we need the first nonzero y-coordinate to be $+1$. We guaranteed this by excluding lattice points $(k - m + 1, -1, 0)$ and $(k - m + 1, 0, \pm 1)$ from feasible lattice points of amino acids $k < m$. If all amino acids $k < m$ lie on the x axis, i.e. $X_{(-m,0,0)}^0 = 1$, then the first amino acid starting from $m + 2$ to $N - 1$ that does not lie on the x axis, must be assigned to lattice point $(k - m - 1, 1, 0)$. In other words, $X_{(-m,0,0)}^0 = 1$ and $X_{(k-m-1,0,0)}^{k-1} = 1$ imply that either $X_{(k-m-1,1,0)}^k = 1$ or $X_{(k-m,0,0)}^k = 1$. Hence,

$$X_{(k-m-1,1,0)}^k + X_{(k-m,0,0)}^k \geq X_{(-m,0,0)}^0 + X_{(k-m-1,0,0)}^{k-1} - 1, \quad \forall k > m + 1. \quad (BLA-12)$$

Similarly, we can eliminate reflection with respect to xy plane, for amino acids $m - 2$ through 0 by further defining new variables. We define binary variable Z_k which is 1, if amino acid $k \in \{0, \dots, m - 2\}$ lies on the xy plane, and 0 otherwise.

$$Z_k = \sum_{(x,y,0) \in F_k} X_{x,y,0}^k, \quad \forall k \in \{0, \dots, m - 2\}. \quad (BLA-13)$$

We also define S_k for $k \in \{0, \dots, m - 3\}$, which is 1 if all amino acids $k < l \leq m - 2$ lie on the xy plane, and 0 otherwise. Note that in a recursive manner, this definition requires that:

$$S_k \leq S_{k+1}, \quad S_k \leq Z_{k+1}, \quad S_k \geq S_{k+1} + Z_{k+1} - 1, \quad \forall k \in \{0, \dots, m - 4\} \quad (BLA-14)$$

$$S_{m-3} = Z_{m-2}. \quad (BLA-15)$$

The symmetry breaking constraints are hence:

$$Z_k + \sum_{(x,y,1) \in F_k} X_{x,y,1}^k \geq S_k, \quad \forall k \in \{0, \dots, m-3\}. \quad (\text{BLA-16})$$

Strengthening the Model

Distance based connectivity valid inequalities

This set of valid inequalities extends the connectivity constraints (BLA-6). In these valid inequalities, we take advantage of the fact that amino acids k and $k \pm l$ should be assigned to lattice points no further than l apart, for any positive value l .

$$\sum_{\substack{i=-l: \\ \mu_{k+i}=\mu_{k+l}}}^l X_p^{k+i} \leq \sum_{\substack{q \in F_k: \\ \Delta_{p,q} \leq l}} X_q^k, \quad l \in \mathbb{N}, k \in \mathcal{R}, p \in \bigcap_{\substack{i=-l: \\ \mu_{k+i}=\mu_{k+l}}}^l F_{k+i}. \quad (\text{BLA-17})$$

Note that (BLA-17) is equivalent to (BLA-6) when $l = 1$. Furthermore, note that due to parity restriction property of cubic lattice, from $k-l$ to $k+l$, only amino acids with parity of point p can be assigned to this point. That is why we restrict the domain of lattice point p to parity of $k+l$ (μ_{k+l}), to prevent adding redundant constraints. Due to this restriction, note that the left hand side can consist of at most $2 \left\lfloor \frac{l+1}{2} \right\rfloor$ terms. In order to prevent imposing too many constraints to the model, we will only add the constraints in which point p is a feasible lattice point for all amino acids of parity μ_{k+l} from $k-l$ to $k+l$, hence having all $2 \left\lfloor \frac{l+1}{2} \right\rfloor$ terms present in the left hand side of the inequality.

Backbone variables and constraints

Carr, Hart & Newman (2002) propose additional variables to make the optimal LP relaxation value closer to the optimal integer value of the problem. The variables, which they call *backbone variables*, are defined as:

$$E_{kpd}^+ = \begin{cases} 1, & X_p^k = X_{p+d}^{k+1} = 1 \\ 0, & \text{otherwise} \end{cases}, \quad (\text{BLA-18})$$

$$E_{kpd}^- = \begin{cases} 1, & X_p^k = X_{p+d}^{k-1} = 1 \\ 0, & \text{otherwise} \end{cases}. \quad (\text{BLA-19})$$

which are defined for lattice point p and direction d . In their implementation, they define these variables only for hydrophobic amino acids of parity 0, or polar amino acids of parity 0 whose successor or predecessor amino acids are hydrophobic. They further define the relation between location assignment variables and backbone variables by introducing *backbone constraints*:

$$\sum_{d \in \delta} E_{kpd}^+ = X_p^k, \quad k \in \mathcal{H}^0, p \in F_k, \quad (\text{BLA-20})$$

$$\sum_{d \in \delta} E_{kpd}^- = X_p^k, \quad k \in \mathcal{H}^0, p \in F_k, \quad (\text{BLA-21})$$

$$\sum_{d \in \delta} E_{k+1,p-d,d}^- = X_p^k, \quad k \in \mathcal{H}^1, p \in F_k, \quad (\text{BLA-22})$$

$$\sum_{d \in \delta} E_{k-1,p-d,d}^+ = X_p^k, \quad k \in \mathcal{H}^1, p \in F_k. \quad (\text{BLA-23})$$

They show that adding the backbone variables and constraints implies backbone connectivity of consecutive HH, HP and PH amino acids. Note in their model, they reduce the model to hydrophobic amino acids only, for the sole purpose of finding an upper bound on the number of contacts, hence their *improved* model can no longer be used for finding the complete optimal integer solutions.

Contiguity valid inequalities

A set of valid inequalities, which similar to (BLA – 17) extends the connectivity constraints (BLA – 6) to more general forms, is introduced by Dittel, Fügenschuh & Martin (2011), in their rigorous study of polyhedral aspects of self-avoiding walks. Let L be an arbitrary subset of $\mathcal{F}$, and A_L denote the set of points in $\mathcal{F}$ which are adjacent to the points in L . Then, from connectivity of the chain it follows that:

$$\sum_{p \in L} X_p^{k-1} \leq \sum_{q \in A_L \cap F_k} X_q^k, \quad \forall k \in \mathcal{R}, L \subset F_{k-1}, \quad (\text{BLA-24})$$

$$\sum_{p \in L} X_p^{k+1} \leq \sum_{q \in A_L \cap F_k} X_q^k, \quad \forall k \in \mathcal{R}, L \subset F_{k+1}. \quad (\text{BLA-25})$$

Note that if L is singleton, then these valid inequalities result in the (weak) connectivity constraints. Dittel, Fügenschuh & Martin (2011) further study the cases in which these valid inequalities become facet defining constraints, i.e. contain feasible integer points within the hyper planes defined by these constraints.

3.4 Single Link Assignment Model

Single Link Assignment model (SLA) is a refined form of BLA. In this model, we define variables to determine the location of each amino acid of parity 0 and its successor in the sequence at the same time. Note that because of the parity restriction property of cubic lattice, this can be done by assigning each amino acid of parity 0 to points of the same parity, along with a basic direction.

In SLA we set $m = 2 \left\lfloor \frac{N}{4} \right\rfloor$, to guarantee that amino acid 0 is always of parity 0. Figure 3-3 demonstrates a sample solution in this model for cubic lattice, in which red arrows represent the *forward links*, while the solid black lines show the backward links. Similar to Figure 1-3, blue and red nodes respectively represent hydrophobic and polar amino acids and the dashed blue lines represent H-H contacts. Note that in this example $N = 18$, hence $m = 8$.

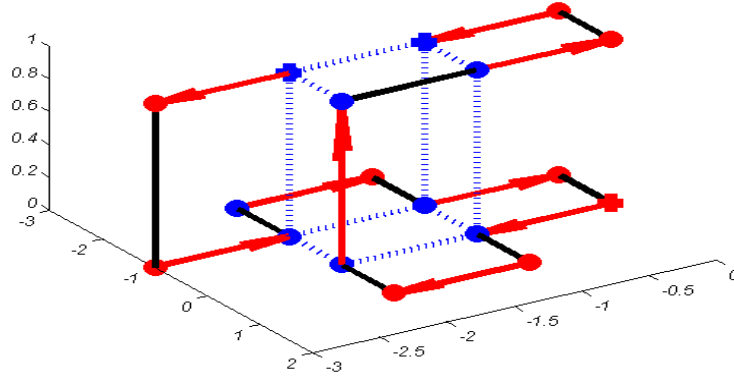


Figure 3-3 A sample solution in SLA for sequence HP2H2PHP2HP2H3P2H

Decision Variables

The first set of binary variables, which we refer to as the *link assignment variables*, determines if amino acid k of parity 0 is assigned to a feasible lattice point p , while amino acid $k + 1$ is assigned to an adjacent point along direction f , which we refer to as the forward direction.

$$X_{p,f}^k = \begin{cases} 1, & (k, k+1) \text{ occupy } (p, p+f) \\ 0, & \text{otherwise} \end{cases}, k \in \mathcal{R}^0, f \in \delta, p \in F_k: p+f \in F_{k+1}. \quad (\text{SLA-1})$$

The second set of decision variables is the same as (BLA – 2)

$$W_p^d = \begin{cases} 1, & p, p+d \text{ are occupied by H amino acids} \\ 0, & \text{otherwise} \end{cases}, \quad (\text{SLA-2})$$

$$p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1.$$

Model (SLA)

$$\max \sum_{p \in \mathcal{F}^0} \sum_{d \in \delta: p+d \in \mathcal{F}^1} W_p^d \quad (\text{SLA-3})$$

Subject to:

$$\sum_{p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} X_{p,f}^k = 1, \quad \forall k \in \mathcal{R}^0, \quad (\text{SLA-4})$$

$$\sum_{k \in \mathcal{R}^0: p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} X_{p,f}^k \leq 1, \quad \forall p \in \mathcal{F}^0, \quad (\text{SLA-5})$$

$$\sum_{k \in \mathcal{R}^0: p \in F_{k+1}} \sum_{f \in \delta: p-f \in F_k} X_{p-f,f}^k \leq 1, \quad \forall p \in \mathcal{F}^1, \quad (\text{SLA-6})$$

$$X_{p,f}^k \leq \sum_{\substack{b \in \delta - \{f\}: \\ p+f-b \in F_{k+2}}} \sum_{\substack{g \in \delta - \{b\}: \\ p+f-b+g \in F_{k+3}}} X_{p+f-b,g}^{k+2}, \quad \forall k \in \mathcal{R}^0, f \in \delta, p \in F_k: p+f \in F_{k+1}, \quad (\text{SLA-7})$$

$$W_p^d \leq \sum_{\substack{k \in \mathcal{H}^0: \\ p \in F_k}} \sum_{\substack{f \in \delta - \{d\}: \\ p+f \in F_{k+1}}} X_{p,f}^k, \quad \forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{SLA-8})$$

$$W_p^d \leq \sum_{\substack{k \in \mathcal{H}^1: \\ p+d \in F_k}} \sum_{\substack{f \in \delta - \{d\}: \\ p+d-f \in F_{k-1}}} X_{p+d-f,f}^{k-1}, \quad \forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{SLA-9})$$

$$X_{p,f}^k \in \{0,1\}, \quad k \in \mathcal{R}^0, f \in \delta, p \in F_k: p+f \in F_{k+1}, \quad (\text{SLA-10})$$

$$W_p^d \in \{0,1\}, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1. \quad (\text{SLA-11})$$

Constraint set (SLA – 4) requires that each amino acid of parity 0 is assigned to one of its feasible points, and its successor is assigned to one of adjacent points of that point. Constraint set (SLA – 5) prevents assigning more than one amino acid of parity 0 to a point of parity 0. In a similar manner, constraint set (SLA – 6) prevents assigning more than one amino acid of parity 1 to a point of parity 1. Note that in this model, $X_p^k = \sum_{f \in \delta} X_{p,f}^k$ if $k \in \mathcal{R}^0$, and $X_p^k = \sum_{f \in \delta} X_{p-f,f}^{k-1}$ if $k \in \mathcal{R}^1$, and constraint sets (SLA – 4) to (SLA – 6) are equivalent to constraint sets (BLA – 4) and (BLA – 5) in the BLA model.

Constraint set (SLA – 7), which we refer to as *single link connectivity constraints*, ensures the chain connectivity. As shown in Figure 3-4, since in SLA, connectivity of amino acids of parity 0 and their successors in the sequence is guaranteed through link assignment variables, we only require connectivity of amino acids of parity 0 and their predecessors in the sequence. In this figure the red arrows represent the forward links, while the dashed blue arrow denotes the connectivity that we should guarantee via constraints. Note that we can still use the connectivity constraints used in BLA model as valid inequalities, as explained in the valid inequalities subsection.

Constraint sets (SLA – 8) and (SLA – 9) guarantee that two ends of the edge $(p, p+d)$ are occupied by hydrophobic amino acids. However, unlike BLA, in this model the link between hydrophobic amino acids of parity 0 and their successors is excluded from these constraints, making the model stronger. Note that the model is written for the case in which every amino acid of parity 0 has a successor in the chain, i.e. N is even. In case N

is odd, we should define a single location assignment variable for the last amino acid and update the formulations respectively.

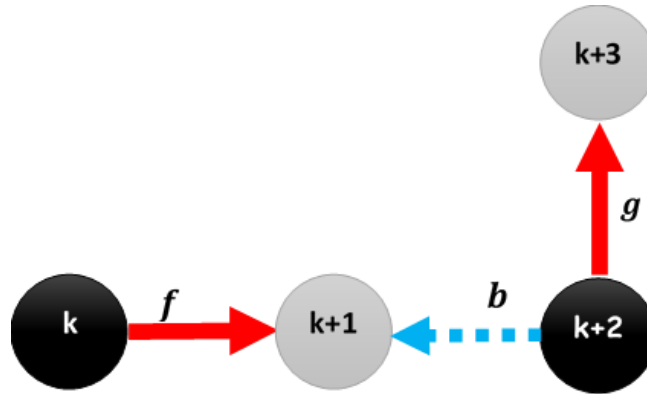


Figure 3-4 Connectivity of amino acids of parity 0 and their predecessors in SLA

Similar to BLA, we try to maximize the number of edges whose both ends are occupied by hydrophobic amino acids. Note that with (SLA – 8) and (SLA – 9) we are excluding the consecutive bonds between hydrophobic amino acids of parity 0 and their successor in the chain. However, (SLA – 3) does not exclude consecutive bonds between hydrophobic amino acids of parity 0 and predecessor in the chain. Similar to BLA, this does not affect the optimal solution, as the optimal value will be equal to absolute value of energy plus total number of sequential H-H bonds between hydrophobic amino acids of parity 0 and their hydrophobic predecessor amino acids. Let S_b^{HH} denote the number of such H-H bonds. Since this value is constant for a given sequence, maximizing the number of H-H contacts and bonds also maximizes the absolute value of energy:

$$|E^*| = \max \sum_{p \in \mathcal{F}^0} \sum_{\substack{d \in \delta: \\ p+d \in \mathcal{F}^1}} W_p^d - S_b^{HH}. \quad (\text{SLA-12})$$

Valid Inequalities

Aggregate connectivity constraints in SLA

We can add the strong connectivity constraints (BLA – 6) to SLA using link assignment variables. Note that we can write location assignment variables in BLA using link assignments as:

$$X_p^k = \sum_{f \in \delta} X_{p,f}^k, \quad k \in \mathcal{R}^0, \quad (\text{SLA-13})$$

$$X_p^k = \sum_{f \in \delta} X_{p-f,f}^{k-1}, \quad k \in \mathcal{R}^1. \quad (\text{SLA-14})$$

Hence (BLA – 6) must be rewritten for two cases: 1) $k \in \mathcal{R}^0$ and 2) $k \in \mathcal{R}^1$. In the first case we have:

$$X_p^{k+1} = \sum_{f \in \delta} X_{p-f,f}^k, \quad (\text{SLA-15})$$

$$X_p^{k-1} = \sum_{f \in \delta} X_{p-f,f}^{k-2}, \quad (\text{SLA-16})$$

$$\sum_{\substack{d \in \delta: \\ p+d \in F_k}} X_{p+d}^k = \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{f \in \delta} X_{p+d,f}^k = \sum_{f \in \delta} X_{p-f,f}^k + \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{\substack{f \in \delta: \\ f \neq -d}} X_{p+d,f}^k. \quad (\text{SLA-17})$$

Therefore, using (SLA – 15) to (SLA – 17) we can rewrite (BLA – 6) as:

$$\begin{aligned} \sum_{f \in \delta} X_{p-f,f}^k + \sum_{f \in \delta} X_{p-f,f}^{k-2} &\leq \sum_{f \in \delta} X_{p-f,f}^k + \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{\substack{f \in \delta: \\ f \neq -d}} X_{p+d,f}^k \Rightarrow \\ \sum_{f \in \delta} X_{p-f,f}^{k-2} &\leq \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{\substack{f \in \delta: \\ f \neq -d}} X_{p+d,f}^k. \end{aligned} \quad (\text{SLA-18})$$

Similarly, in the second case where $k \in \mathcal{R}^1$, we have

$$X_p^{k+1} = \sum_{f \in \delta} X_{p,f}^{k+1}, \quad (\text{SLA-19})$$

$$X_p^{k-1} = \sum_{f \in \delta} X_{p,f}^{k-1}, \quad (\text{SLA-20})$$

$$\sum_{\substack{d \in \delta: \\ p+d \in F_k}} X_{p+d}^k = \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{f \in \delta} X_{p+d-f,f}^{k-1} = \sum_{\substack{f \in \delta: \\ p+f \in F_{k-1}}} X_{p,f}^{k-1} + \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{\substack{f \in \delta: \\ f \neq d \\ p+d-f \in F_{k-1}}} X_{p+d-f,f}^{k-1}. \quad (\text{SLA-21})$$

Therefore, using (SLA – 19) to (SLA – 21) we can rewrite (BLA – 6) as:

$$\begin{aligned} \sum_{f \in \delta} X_{p,f}^{k+1} + \sum_{f \in \delta} X_{p,f}^{k-1} &\leq \sum_{f \in \delta} X_{p,f}^{k-1} + \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{\substack{f \in \delta: \\ f \neq d \\ p+d-f \in F_{k-1}}} X_{p+d-f,f}^{k-1} \Rightarrow \\ \sum_{f \in \delta} X_{p,f}^{k+1} &\leq \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \sum_{\substack{f \in \delta: \\ f \neq d \\ p+d-f \in F_{k-1}}} X_{p+d-f,f}^{k-1}. \end{aligned} \quad (\text{SLA-22})$$

3.5 Double Link Assignment Model

Now in the last general model for both square and cubic lattice, we introduce *Double Link Assignment* model (DLA), in which we determine the location of each amino acid of parity 0, along with location of its successor and predecessor amino acids at the same time, using two distinct basic directions. Here, in order to maintain the backbone

connectivity, it is enough to make sure that the location of each amino acid of parity 1 determined by its predecessor is the same as the one determined by its successor. We further show how this new model strengthens the previous models.

In this model, we set $m = 2 \left(\left\lfloor \frac{N}{2} \right\rfloor - \left\lfloor \frac{N}{4} \right\rfloor \right) - 1$, to guarantee that amino acid 1 is always of parity 0. Note that the following model is written for the case in which every amino acid of parity 0 has a successor and a predecessor in the chain, i.e. N is odd. In case N is even, we should define a single backward link assignment variable for the last amino acid and update the formulations respectively. Figure 3-5 illustrates an example of a solution in DLA model, in which blue and red arrows respectively represent the *forward* and *backward* links. Contrasted with Figure 3-3, in Figure 3-5 the bonds are replaced with backward links. Note that since in this example we have even number of amino acids, the last amino acid is of parity 0 but is represented by a backward link only.

For maximizing the absolute energy, we can count the number of contacts in two ways: 1) count the number of pairs of adjacent lattice points that are occupied by hydrophobic amino acids, or 2) count the number of contacts between pairs of nonconsecutive hydrophobic amino acids of opposite parities. Each of these ways requires different variables and constraints. In the following, we first introduce the model concerning the first measure of absolute energy, which we refer to its objective function as the *location-wise* objective function, and denote the associated model by *DLA-L*. After this model, we will introduce the other way of counting the number of contacts.

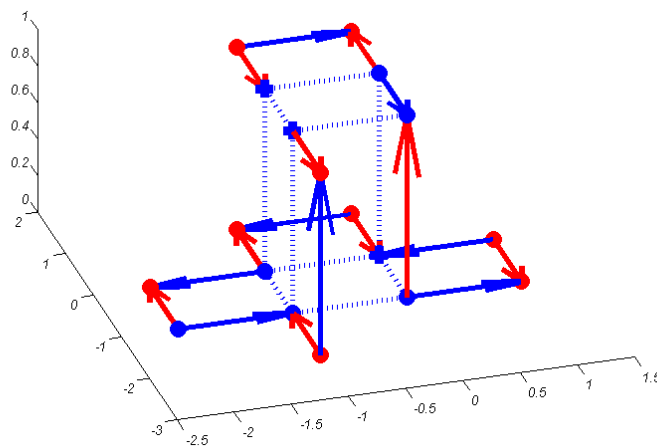


Figure 3-5 A sample solution in DLA for sequence HP2H2PHP2HP2H3P2H

Decision Variables

The first set of binary variables, which we refer to as *double link assignment variables*, determines if amino acid k of parity 0 is assigned to a feasible lattice point p , while its successor $(k + 1)$ is assigned to an adjacent point along forward direction f , and its

predecessor $(k-1)$ is assigned to an adjacent point along backward direction b , for f and b different.

$$X_{p,f,b}^k = \begin{cases} 1, & (k, k+1, k-1) \text{ occupy } (p, p+f, p+b), \\ 0, & \text{otherwise} \end{cases}, \quad (\text{DLA-1})$$

$$k \in \mathcal{R}^0, f \in \delta, b \in \delta - \{f\}, p \in F_k: p+f \in F_{k+1}, p+b \in F_{k-1}.$$

The second set of variables is the same as $(BLA-2)$ and $(SLA-2)$ variables.

$$W_p^d = \begin{cases} 1, & (p, p+d) \text{ are occupied by } H \text{ amino acids} \\ 0, & \text{otherwise} \end{cases}, \quad (\text{DLA-2})$$

$$p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1.$$

Model (DLA-L)

$$\max \sum_{p \in \mathcal{F}^0} \sum_{d \in \delta: p+d \in \mathcal{F}^1} W_p^d \quad (\text{DLA-3})$$

Subject to:

$$\sum_{p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} \sum_{b \in \delta - \{f\}: p+b \in F_{k-1}} X_{p,f,b}^k = 1, \quad \forall k \in \mathcal{R}^0, \quad (\text{DLA-4})$$

$$\sum_{k \in \mathcal{R}^0: p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} \sum_{b \in \delta - \{f\}: p+b \in F_{k-1}} X_{p,f,b}^k \leq 1, \quad \forall p \in \mathcal{F}^0, \quad (\text{DLA-5})$$

$$\sum_{k \in \mathcal{R}^0: p \in F_{k+1}} \sum_{f \in \delta: p-f \in F_k} \sum_{b \in \delta - \{f\}: p-f+b \in F_{k-1}} X_{p-f,b}^k + \sum_{b \in \delta: p-b \in F_1} \sum_{f \in \delta - \{b\}: p-b+f \in F_2} X_{p-b,f,b}^1 \leq 1, \quad \forall p \in \mathcal{F}^1, \quad (\text{DLA-6})$$

$$\sum_{f \in \delta: p-f \in F_{k-1}} \sum_{b \in \delta - \{f\}: p-f+b \in F_{k-2}} X_{p-f,b}^{k-1} = \sum_{b \in \delta: p-b \in F_{k+1}} \sum_{f \in \delta - \{b\}: p-b+f \in F_{k+2}} X_{p-b,f,b}^{k+1}, \quad \forall k \in \mathcal{R}^1 - \{0\}, p \in F_k, \quad (\text{DLA-7})$$

$$W_p^d \leq \sum_{k \in \mathcal{H}^0: p \in F_k} \sum_{f \in \delta - \{d\}: p+f \in F_{k+1}} \sum_{b \in \delta - \{d,f\}: p+b \in F_{k-1}} X_{p,f,b}^k, \quad \forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{DLA-8})$$

$$W_p^d + \sum_{k \in \mathcal{H}^1 - \{0, N-1, N-2\}: p \in F_{k+1}, p+d \in F_k} \sum_{f \in \delta - \{d\}: p+f \in F_{k+2}} X_{p,f,d}^{k+1} \quad (\text{DLA-9})$$

$$\leq \sum_{k \in \mathcal{H}^1 - \{0\}: p+d \in F_k} \sum_{f \in \delta - \{d\}: p+d-f \in F_{k-1}} \sum_{b \in \delta - \{f\}: p+d-f+b \in F_{k-2}} X_{p+d-f,b}^{k-1}$$

$$+ \rho_0 \sum_{b \in \delta - \{d\}: p+d-b \in F_1} \sum_{f \in \delta - \{b\}: p+d-b+f \in F_2} X_{p+d-b,f,b}^1,$$

$$\forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1,$$

$$X_{p,f,b}^k \in \{0,1\}, k \in \mathcal{R}^0, f \in \delta, b \in \delta - \{f\}, p \in F_k: p + f \in F_{k+1}, p + b \in F_{k-1}, \quad (\text{DLA-10})$$

$$W_p^d \in \{0,1\}, \quad p \in \mathcal{F}^0, d \in \delta: p + d \in \mathcal{F}^1. \quad (\text{DLA-11})$$

Similar to SLA model, constraint set (DLA – 4) requires that each amino acid of parity 0 is assigned to one of its feasible lattice points. Moreover, these constraints ensure that two different adjacent points are selected for the forward (successor) and the backward (predecessor) amino acids of any amino acid of parity 0. Constraint set (DLA – 5) prevents assigning more than one amino acid of parity 0 to a lattice point of parity 0. It also prevents choosing more than one pair of distinct basic directions for each amino acid of parity 0. On the other hand, constraint set (DLA – 6) guarantees that at most one amino acid of parity 1 is assigned to a lattice point of parity 1. Note that amino acids of parity 1 are in even positions in the sequence. We determine the location of these amino acids using the location of their predecessor amino acid, except amino acid 0, which does not have any predecessor, and for which we determine its position using its successor, i.e. amino acid 1.

Constraint set (DLA – 7) guarantees that there is no inconsistency in the location of amino acids of parity 1 determined by their predecessor and successor amino acids of parity 0. Constraint sets (DLA – 8) and (DLA – 9) require that two ends of the edge $(p, p + d)$ are occupied by hydrophobic amino acids. However, unlike BLA and SLA models, in DLA all of the backbone links are excluded from these contacts, by restricting the forward (f) and backward (b) directions of double link assignment variables to be different from the contact direction (d).

Note that since location of amino acid 0 is defined by its successor (amino acid 1), if amino acid 0 is hydrophobic, then it should appear in the form of

$$\sum_{\substack{b \in \delta - \{d\}: \\ p+d-b \in F_1}} \sum_{\substack{f \in \delta - \{b\}: \\ p+d-b+f \in F_2}} X_{p+d-b,f,b}^1 \text{ in the right hand side of (DLA – 9).}$$

Furthermore, note that we need to exclude the backward backbone links coming from successors of hydrophobic amino acids of parity 1 from contacts. That is why we also add the term

$$\sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_{k+1}, \\ p+d \in F_k}} \sum_{\substack{f \in \delta - \{d\}: \\ p+f \in F_{k+2}}} X_{p,f,d}^{k+1} \text{ to the left hand side of (DLA – 9).}$$

Now, we introduce the second way of counting the number of contacts. In this model, we count the number of contacts between pairs of nonconsecutive hydrophobic amino acids of different parities along given directions. We refer to this objective function as the *pairwise* objective function and denote the associated model with *DLA-P*. Let us define binary decision variables Z as:

$$Z_{p,d}^{k,l} = \begin{cases} 1, & (k,l) \text{ occupy } (p, p+d) \\ 0, & \text{otherwise} \end{cases}, \quad (DLA-12)$$

$$k \in \mathcal{H}^0, p \in F_k, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta: p+d \in F_l.$$

Model (DLA-P)

$$\max \sum_{k \in \mathcal{H}^0} \sum_{p \in F_k} \sum_{l \in \mathcal{H}^1 - \{k \pm 1\}} \sum_{d \in \delta: p+d \in F_l} Z_{p,d}^{k,l} \quad (DLA-13)$$

Subject to:

$$(DLA-4) - (DLA-7), (DLA-10),$$

$$\sum_{\substack{l \in \mathcal{H}^1 - \{k \pm 1\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} \leq \sum_{\substack{f \in \delta - \{d\}: \\ p+f \in F_{k+1}}} \sum_{\substack{b \in \delta - \{d, f\}: \\ p+b \in F_{k-1}}} X_{p,f,b}^k, \quad \forall k \in \mathcal{H}^0, p \in F_k, d \in \delta, \quad (DLA-14)$$

$$\sum_{\substack{k \in \mathcal{H}^0 - \{l \pm 1\}: \\ p \in F_k}} Z_{p,d}^{k,l} + \sum_{\substack{f \in \delta - \{d\}: \\ p+f \in F_{k+2}}} X_{p,f,d}^{l+1} \leq \sum_{\substack{f \in \delta - \{d\}: \\ p+d-f \in F_{k-1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p+d-f+b \in F_{k-2}}} X_{p+d-f,b}^{l-1}, \quad (DLA-15)$$

$$\forall l \in \mathcal{H}^1, d \in \delta, p \in \mathcal{F}^0: p+d \in F_l,$$

$$Z_{p,d}^{k,l} \in \{0,1\}, \quad k \in \mathcal{H}^0, p \in F_k, l \in \mathcal{H}^1 - \{k \pm 1\}, d \in \delta: p+d \in F_l. \quad (DLA-16)$$

Constraint set (DLA-12) requires that if any hydrophobic amino acid of parity 1 is in contact with amino acid k of parity 0 at lattice point p along direction d , k must be located at p , while $p+d$ is not occupied by either of its forward or backward amino acids. Similarly, constraint set (DLA-14) ensures that if any hydrophobic amino acid of parity 0 is going to be in contact with amino acid l of parity 1 at point p along direction d , then l must be located at $p+d$, while p is not occupied by either of its forward or backward amino acids.

Valid Inequalities

Now we present two sets of valid inequalities that further help strengthening the DLA model.

Square single contact valid inequalities in DLA

The first set of valid inequalities are based on the observation that amino acids k and $k+3$ form a contact if and only if amino acids k , $k+1$, $k+2$, and $k+3$ are placed on the four corners of a square, which implies that the direction of contact between k and $k+3$ must be opposite of the direction of link from $k+2$ to $k+1$. Note that the same applies to k , $k-1$, $k-2$, and $k-3$.

$$Z_{p,d}^{k,k+3} \leq \sum_{\substack{r \in \delta - \{d, -d\}: \\ p+d+r \in F_{k+2}}} X_{p+d+r, -r, -d}^{k+2}, \quad (\text{DLA-17})$$

$$\forall k \in \mathcal{H}^0, d \in \delta, p \in F_k: k+3 \in \mathcal{H}^1, p+d \in F_{k+3},$$

$$Z_{p,d}^{k,k-3} \leq \sum_{\substack{r \in \delta - \{d, -d\}: \\ p+d+r \in F_{k-2}}} X_{p+d+r, -d, -r}^{k-2}, \quad (\text{DLA-18})$$

$$\forall k \in \mathcal{H}^0, d \in \delta, p \in F_k: k-3 \in \mathcal{H}^1, p+d \in F_{k-3}.$$

Note that the right hand side of these inequalities implies that amino acids $k+1$, $k+2$, and $k+3$ (or $k-1$, $k-2$, and $k-3$) are located at three corners of a square. Also note that these valid inequalities are not in the equality form, because placing amino acids $k+1$, $k+2$, and $k+3$ (or $k-1$, $k-2$, and $k-3$) at three corners of a square does not necessarily imply the existence of a contact between k and $k+3$ (or $k-3$).

Rectangular single contact valid inequalities in DLA

The second set of valid inequalities, which is specific to the 2D square, is based on the observation that if there exists a contact between amino acids k and $k+5$ along direction d , then direction of forward link from amino acid $k+2$ to amino acid $k+3$ must be also d . This observation can be verified by inspecting two cases, as shown in Figure 3-6, and generalizing them to all possible cases by series of reflections and rotations. Note that the same applies to k , $k-5$, $k-2$, and $k-3$.

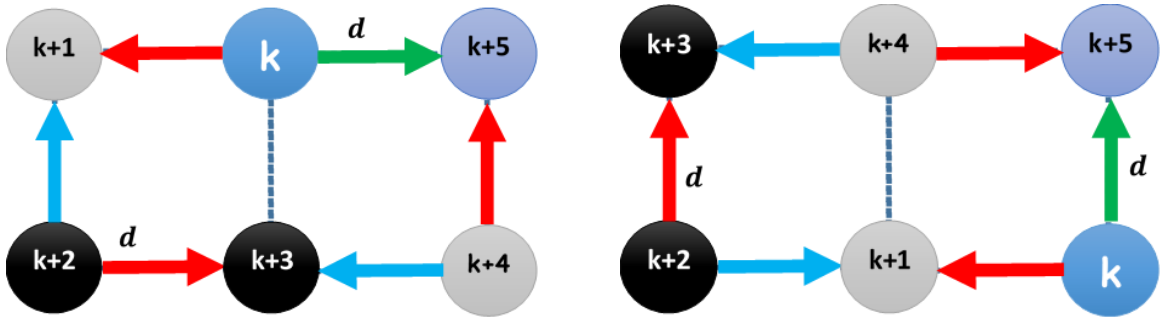


Figure 3-6 Direction of contact between k and $k+5$ is always the same as the forward link from $k+2$ to $k+3$ in square lattice

$$Z_{p,d}^{k,k+5} \leq \sum_{\substack{r_1 \in \delta - \{d\}: \\ p+r_1 \in F_{k+1}}} \sum_{\substack{r_2 \in \delta - \{-r_1, -d\}: \\ p+r_1+r_2 \in F_{k+2}}} X_{p+r_1+r_2, d, -r_2}^{k+2}, \quad (\text{DLA-19})$$

$$\forall k \in \mathcal{H}^0, d \in \delta, p \in F_k: k+5 \in \mathcal{H}^1, p+d \in F_{k+5},$$

$$Z_{p,d}^{k,k-5} \leq \sum_{\substack{r_1 \in \delta - \{d\}: \\ p+r_1 \in F_{k-1}}} \sum_{\substack{r_2 \in \delta - \{-r_1, -d\}: \\ p+r_1+r_2 \in F_{k-2}}} X_{p+r_1+r_2, -r_2, d}^{k-2}, \quad (\text{DLA-20})$$

$$\forall k \in \mathcal{H}^0, d \in \delta, p \in F_k: k-5 \in \mathcal{H}^1, p-d \in F_{k-5}.$$

Note that in (DLA – 19) and (DLA – 20), δ is limited to the basic directions of square lattice, as this set of valid inequalities is only applicable to this planar lattice.

Double Link Variables for All Amino Acids (DLA-A)

Similar to double link variables that we define for amino acids of parity 0, one can define such variables for amino acids of parity 1. With these variables, the link between consecutive amino acids k and $k + 1$ will be defined via two variables. One with forward link emitting from k to $k + 1$ and the other with backward link emitting from $k + 1$ towards k . Let us generalize the definition of $X_{p,f,b}^k$ to $k \in \mathcal{R} - \{0, N - 1\}$ and define single link assignment variables $T_{p,d}^k$ for $k \in \{0, N - 1\}$, which determines if terminal amino acid k and its predecessor (or successor) amino acid are located at edge $(p, p + d)$. With these variables we can write the link assignment consistency constraints:

$$\sum_{\substack{b \in \delta - \{d\}: \\ p+b \in F_{k-1}}} X_{p,d,b}^k = \sum_{\substack{f \in \delta - \{-d\}: \\ p+d+f \in F_{k+2}}} X_{p+d,f,-d}^{k+1}, \quad (DLA-21)$$

$$\forall k \in \mathcal{R} - \{0, N - 1, N - 2\}, p \in F_k, d \in \delta: p + d \in F_{k+1}.$$

It is not hard to prove that (DLA – 21) implies (DLA – 7). Moreover, the assignment constraints (DLA – 4) and non-overlapping constraints (DLA – 5) can be generalized to all amino acids. Finally, the relations between double link variables and decision variables related to the objective function can be written using these extra variables. It is important to note that with these variables we no longer require m to be odd.

3.6 Quadruple Neighborhood Assignment Model

Quadruple Neighborhood Assignment model (QNA) is specific to the square lattice, due to its special planar properties. Since in the square lattice each hydrophobic amino acid can form at most two contacts, it can either have no contacts (*double neighborhood assignment*), one contact (*triple neighborhood assignment*) or two contacts (*quadruple neighborhood assignment*). We define three sets of variables for each of these three cases. These variables help us to identify the combinations that cannot happen as triple or quadruple contacts, and to remove them from the problem, thus strengthening the formulation.

Similar to DLA, in this model we set $m = 2 \left(\left\lfloor \frac{N}{2} \right\rfloor - \left\lfloor \frac{N}{4} \right\rfloor \right) - 1$, to ensure that amino acid 1 is always of parity 0. In the following formulation, it is assumed that the chain consists of odd number of amino acids. In case it is even, according to definition of the middle amino acid, parity of the last amino acid will be 0, and it will be able to form three

contacts. In this case, we define variables for single backward link, as well as one-, two- or three-contacts, and update the formulations accordingly.

Sets and Parameters

We define Ψ as the set of possible *quadruples of basic directions* in square lattice.

$$\Psi = \{(d_1, d_2, d_3, d_4) \in \delta^4: d_i \neq d_j, \quad \forall 1 \leq i < j \leq 4\}. \quad (\text{QNA-1})$$

We define Π^3 and Π^4 as the set of possible combinations of triple and quadruple neighborhoods in the square lattice, respectively.

$$\Pi^3 = \left\{ \begin{array}{l} (k, \psi, l_1) \in \mathcal{H}^0 \times \Psi \times \mathcal{H}^1: \\ l_1 \text{ can be neighbor of } k \text{ along } \psi_3 \text{ while } k+1 \text{ and } k-1 \\ \text{are neighbors of } k \text{ along } \psi_1 \text{ and } \psi_2, \text{ respectively} \end{array} \right\}, \quad (\text{QNA-2})$$

$$\Pi^4 = \left\{ \begin{array}{l} (k, \psi, l_1, l_2) \in \mathcal{H}^0 \times \Psi \times \mathcal{H}^1 \times \mathcal{H}^1: \\ l_1 \text{ and } l_2 \text{ can be neighbors of } k \text{ along } \psi_3 \text{ and } \psi_4, \\ \text{while } k+1 \text{ and } k-1 \text{ are neighbors of } k \text{ along } \psi_1 \text{ and } \psi_2 \end{array} \right\} \quad (\text{QNA-3})$$

In some cases we need to change the position of the last two directions of a quadruple of directions. We define operator $\bowtie$ which operates on a quadruple and changes the order of its last two elements, i.e. if $\psi = (d_1, d_2, d_3, d_4) \in \Psi$, then $\psi^{\bowtie} = (d_1, d_2, d_4, d_3)$.

Decision Variables

The first set of decision variables, which we refer to as the *double neighborhood assignment variables*, is similar to double link assignment variables (*DLA* – 1). We define these variables for both hydrophobic and polar amino acids of parity 0.

$$XD_{p,f,b}^k = \begin{cases} 1, & (k, k+1, k-1) \text{ occupy } (p, p+f, p+b), \\ 0, & \text{otherwise} \end{cases}, \quad (\text{QNA-4})$$

$$k \in \mathcal{R}^0, f \in \delta, b \in \delta - \{f\}, p \in F_k: p+f \in F_{k+1}, p+b \in F_{k-1}.$$

The second set of decision variables, which we refer to as the *triple neighborhood assignment variables*, defined only for the hydrophobic amino acids, not only determines the location of hydrophobic amino acid of parity 0 and its successor and predecessor amino acids, but also determines if it forms a contact with a nonconsecutive amino acid of parity 1 along a direction different from its forward and backward directions.

$$HT_{p,\psi}^{k,l_1} = \begin{cases} 1, & (k, k+1, k-1, l_1) \text{ occupy } (p, p+\psi_1, p+\psi_2, p+\psi_3), \\ 0, & \text{otherwise} \end{cases}, \quad (\text{QNA-5})$$

$$k \in \mathcal{H}^0, l_1 \in \mathcal{H}^1, \psi \in \Psi, p \in F_k: p+\psi_1 \in F_{k+1} \wedge p+\psi_2 \in F_{k-1} \wedge p+\psi_3 \in F_{l_1} \wedge (k, \psi, l_1) \in \Pi^3.$$

Similar to triple neighborhood assignment variables, the third set of decision variables, which we refer to as the *quadruple neighborhood assignment variables*, defined only for hydrophobic amino acids, further determines if a hydrophobic amino acid of parity 0

forms two contacts with two nonconsecutive amino acids of parity 1 along directions different from its forward and backward directions.

$$HQ_{p,\psi}^{k,l_1,l_2} = \begin{cases} 1, & (k, k+1, k-1, l_1, l_2) \text{ occupy } (p, p+\psi_1, p+\psi_2, p+\psi_3, p+\psi_4), \\ 0, & \text{otherwise} \end{cases}, \quad (\text{QNA-6})$$

$$k \in \mathcal{H}^0, l_1 < l_2 \in \mathcal{H}^1, \psi \in \Psi, p \in F_k: p+\psi_1 \in F_{k+1} \wedge p+\psi_2 \in F_{k-1} \wedge p+\psi_3 \in F_{l_1} \wedge p+\psi_4 \in F_{l_2} \wedge (k, \psi, l_1, l_2) \in \Pi^4.$$

Before mathematically representing this model and later expressions, since expressions in this model are mostly too long, we first define certain expressions and then use these definitions in the model.

Model (QNA)

$$\max \sum_{k \in \mathcal{H}^0} \sum_{p \in F_k} \sum_{\substack{\psi \in \Psi: \\ p+\psi_1 \in F_{k+1} \\ p+\psi_2 \in F_{k-1}}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p+\psi_3 \in F_{l_1} \\ (k, \psi, l_1) \in \Pi^3}} \left(HT_{p,\psi}^{k,l_1} + 2 \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p+\psi_4 \in F_{l_2} \\ (k, \psi, l_1, l_2) \in \Pi^4}} HQ_{p,\psi}^{k,l_1,l_2} \right) \quad (\text{QNA-7})$$

Subject to:

$$\sum_{p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} \sum_{b \in \delta - \{f\}: p+b \in F_{k-1}} XD_{p,f,b}^k = 1, \quad \forall k \in \mathcal{R}^0 - \mathcal{H}^0, \quad (\text{QNA-8})$$

$$L_1 := \sum_{\substack{\psi \in \Psi: \\ p+\psi_1 \in F_{k+1} \\ p+\psi_2 \in F_{k-1}}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p+\psi_3 \in F_{l_1} \\ (k, \psi, l_1) \in \Pi^3}} \left(HT_{p,\psi}^{k,l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p+\psi_4 \in F_{l_2} \\ (k, \psi, l_1, l_2) \in \Pi^4}} HQ_{p,\psi}^{k,l_1,l_2} \right),$$

$$\sum_{p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} \sum_{b \in \delta - \{f\}: p+b \in F_{k-1}} XD_{p,f,b}^k + \sum_{p \in F_k} L_1 = 1, \quad \forall k \in \mathcal{H}^0, \quad (\text{QNA-9})$$

$$\sum_{k \in \mathcal{R}^0: p \in F_k} \sum_{f \in \delta: p+f \in F_{k+1}} \sum_{b \in \delta - \{f\}: p+b \in F_{k-1}} XD_{p,f,b}^k + \sum_{k \in \mathcal{H}^0: p \in F_k} L_1 \leq 1, \quad \forall p \in \mathcal{F}^0, \quad (\text{QNA-10})$$

$$L_2 := \sum_{k \in \mathcal{R}^0: p \in F_{k+1}} \sum_{f \in \delta: p-f \in F_k} \sum_{b \in \delta - \{f\}: p-f+b \in F_{k-1}} X_{p-f,b}^k,$$

$$L_3 := \sum_{\substack{k \in \mathcal{H}^0: \\ p \in F_{k+1}}} \sum_{\substack{\psi \in \Psi: \\ p-\psi_1 \in F_k \\ p-\psi_1+\psi_2 \in F_{k-1}}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-\psi_1+\psi_3 \in F_{l_1} \\ (k, \psi, l_1) \in \Pi^3}} \left(HT_{p-\psi_1, \psi}^{k, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-\psi_1+\psi_4 \in F_{l_2} \\ (k, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-\psi_1, \psi}^{k, l_1, l_2} \right),$$

$$L_4 := \sum_{\substack{\psi \in \Psi: \\ p-\psi_2 \in F_1 \\ p-\psi_2+\psi_1 \in F_2}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-\psi_2+\psi_3 \in F_{l_1} \\ (1, \psi, l_1) \in \Pi^3}} \left(HT_{p-\psi_2, \psi}^{1, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-\psi_2+\psi_4 \in F_{l_2} \\ (1, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-\psi_2, \psi}^{1, l_1, l_2} \right),$$

$$L_2 + L_3 + \sum_{\substack{b \in \delta: \\ p-b \in F_1}} \sum_{\substack{f \in \delta - \{b\}: \\ p-b+f \in F_2}} X_{p-b, f, b}^1 + \rho_1 L_4 \leq 1, \quad \forall p \in \mathcal{F}^1, \quad (\text{QNA-11})$$

$$L_5 := \sum_{\substack{\psi \in \Psi: \\ p-\psi_1 \in F_{k-1} \\ p-\psi_1+\psi_2 \in F_{k-2}}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-\psi_1+\psi_3 \in F_{l_1} \\ (k-1, \psi, l_1) \in \Pi^3}} \left(HT_{p-\psi_1, \psi}^{k-1, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-\psi_1+\psi_4 \in F_{l_2} \\ (k-1, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-\psi_1, \psi}^{k-1, l_1, l_2} \right),$$

$$\sum_{\substack{f \in \delta: \\ p-f \in F_{k-1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p-f+b \in F_{k-2}}} X_{p-f, f, b}^{k-1} + \rho_{k-1} L_5 = \sum_{\substack{b \in \delta: \\ p-b \in F_{k+1}}} \sum_{\substack{f \in \delta - \{b\}: \\ p-b+f \in F_{k+2}}} X_{p-b, f, b}^{k+1} + \rho_{k+1} L_6, \quad (\text{QNA-12})$$

$$\forall k \in \mathcal{R}^1, p \in F_k,$$

$$L_7 := \sum_{\substack{k \in \mathcal{H}^0: \\ p-d \in F_k}} \sum_{\substack{\psi \in \Psi: \\ \psi_1 \neq d \\ \psi_2 \neq d \\ \psi_3 = d \\ p-d+\psi_1 \in F_{k+1} \\ p-d+\psi_2 \in F_{k-1}}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p \in F_{l_1} \\ (k, \psi, l_1) \in \Pi^3}} \left(HT_{p-d, \psi}^{k, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-d+\psi_4 \in F_{l_2} \\ (k, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-d, \psi}^{k, l_1, l_2} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 > l_2 \\ p-d+\psi_4 \in F_{l_2} \\ (k, \psi^\infty, l_2, l_1) \in \Pi^4}} HQ_{p-d, \psi^\infty}^{k, l_2, l_1} \right),$$

$$L_8 := \sum_{\substack{\psi \in \Psi: \\ \psi_2 = d \\ p-\psi_2+\psi_1 \in F_{h+2}}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-\psi_2+\psi_3 \in F_{l_1} \\ (h+1, \psi, l_1) \in \Pi^3}} \left(HT_{p-\psi_2, \psi}^{h+1, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-\psi_2+\psi_4 \in F_{l_2} \\ (h+1, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-\psi_2, \psi}^{h+1, l_1, l_2} \right),$$

$$L_7 + \sum_{\substack{f \in \delta \setminus \{d\}: \\ p-d+f \in F_{h-1}}} XD_{p-d,f,d}^{h+1} + \rho_{h+1} L_8 \leq \sum_{\substack{f \in \delta \setminus \{d\}: \\ p-f \in F_{h-1}}} \sum_{\substack{b \in \delta \setminus \{f\}: \\ p-f+b \in F_{h-2}}} XD_{p-f,f,b}^{h-1} + \rho_{h-1} L_9, \quad (\text{QNA-13})$$

$$\forall h \in \mathcal{H}^1, p \in F_h, d \in \delta,$$

$$XD_{p,f,b}^k \in \{0,1\}, k \in \mathcal{R}^0, f \in \delta, b \in \delta - \{f\}, p \in F_k: p + f \in F_{k+1} \wedge p + b \in F_{k-1}, \quad (\text{QNA-14})$$

$$HT_{p,\psi}^{k,l_1} \in \{0,1\}, \forall k \in \mathcal{H}^0, l_1 \in \mathcal{H}^1, \psi \in \Psi, p \in F_k: p + \psi_1 \in F_{k+1} \wedge p + \psi_2 \quad (\text{QNA-15})$$

$$\in F_{k-1} \wedge p + \psi_3 \in F_{l_1} \wedge (k, \psi, l_1) \in \Pi^3,$$

$$HQ_{p,\psi}^{k,l_1,l_2} \in \{0,1\}, \forall k \in \mathcal{H}^0, l_1 < l_2 \in \mathcal{H}^1, \psi \in \Psi, p \in F_k: p + \psi_1 \in F_{k+1} \wedge p + \psi_2 \quad (\text{QNA-16})$$

$$\in F_{k-1} \wedge p + \psi_3 \in F_{l_1} \wedge p + \psi_3 \in F_{l_2} \wedge (k, \psi, l_1, l_2) \in \Pi^4.$$

Constraint set (QNA – 8) ensures that each polar amino acid of parity 0 is assigned to exactly one of its feasible points. Moreover, these constraints ensure that forward and backward amino acids are assigned to different lattice points adjacent to the location of this amino acid. Constraint set (QNA – 9) guarantees that each hydrophobic amino acid of polarity 0 is assigned to exactly one of its feasible points. Furthermore, the location of forward/backward amino acids, along with the location of nonconsecutive hydrophobic amino acids of parity 1 that are in triple or quadruple contact with this amino acid is uniquely determined. Constraint set (QNA – 10) ensures that at most one amino acid of parity 0 (either polar or hydrophobic) is assigned to each lattice point of parity 0. Similar to what is explained in DLA, constraint set (QNA – 11) requires that at most one amino acid of parity 1 is assigned to each lattice point of parity 1. Constraint set (QNA – 12) is again similar to the constraints given in DLA model to prevent inconsistency of the location of amino acids of parity 1.

Since we are assigning the hydrophobic amino acids of parity 1 to hydrophobic amino acids of parity 0, we only need to guarantee that they are in contact along direction d at point $p - d$ only if they are indeed located at these points.

Feasible single contact combinations (Π^3)

Let $\pi^3 = (k, \psi, l_1) \in \mathcal{H}^0 \times \Psi \times \mathcal{H}^1$. π^3 represents a feasible single contact combination, unless any of the following cases occur:

Case 1: $l_1 = k - 1 \vee l_1 = k + 1$. In this case l_1 is either forward or backward amino acid of k , hence they cannot be in contact.

Case 2: $l_1 < k - 1 \wedge \psi_3 = -\psi_2 \wedge k - 1 - l_1 < 4$. As illustrated in Figure 3-7 (left), in this case there must be a SAW from amino acid $k - 1$ to amino acid l_1 , which with the given combination of directions cannot be shorter than four steps. Consequently, the

greatest value that l_1 can get to be able to form a contact with k in this situation is $l_1 = k - 5$.

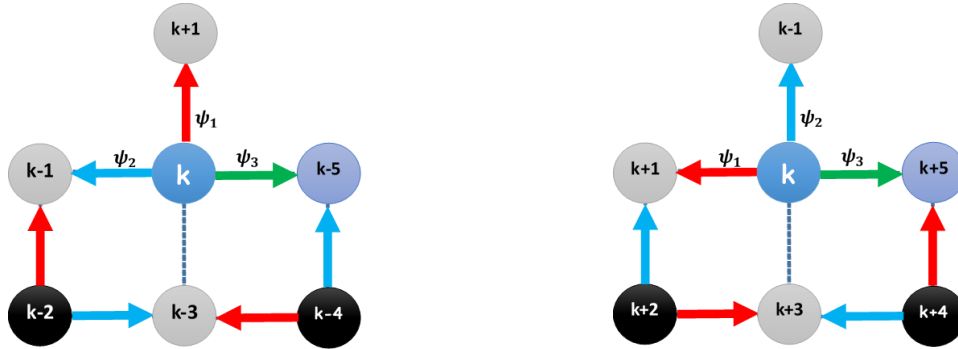


Figure 3-7 Feasible single contact combination, when $\psi_3 = -\psi_2 \wedge k - 1 - l_1 \geq 4$ (left), or $\psi_3 = -\psi_1 \wedge l_1 - k - 1 \geq 4$ (right)

Case 3: $l_1 > k + 1 \wedge \psi_3 = -\psi_1 \wedge l_1 - k - 1 < 4$. Similar to case 2, and as demonstrated in Figure 3-7 (right), again a SAW from amino acid $k + 1$ to l_1 is impossible. Similarly, the least value that l_1 can get in order for this combination to represent a feasible combination is $l_1 = k + 5$.

Feasible double contact combinations (Π^4)

Before excluding the infeasible double contact combinations, let us define function $M(c)$, which is an integer number such that $M(c) + 2$ is equal to the minimal perimeter of a rectangle that can accommodate a string of c amino acids inside its inner points. Therefore $M(c)$ is equal to $\min\{2(a + b) + 2 | (a, b) \in \mathbb{N}^2, ab \geq c\}$, which is a convex minimization problem when integrality constraints are relaxed, and the optimal integer value can be obtained as follows:

$$M(c) = \min \left\{ 4 \lceil \sqrt{c} \rceil + 2, 4 \left\lceil \frac{1 + \sqrt{4c + 1}}{2} \right\rceil \right\}. \quad (\text{QNA-17})$$

Now, let $\pi^4 = (k, \psi, l_1, l_2) \in \mathcal{H}^0 \times \Psi \times \mathcal{H}^1 \times \mathcal{H}^2$. In the following cases π^4 cannot represent a feasible double contact combination:

Case 1: $(k, \psi, l_1) \notin \Pi^3 \vee (k, \psi^\circ, l_2) \notin \Pi^3 \vee l_1 \geq l_2$. If any of the single contacts is infeasible, so is the double contact.

Case 2: $l_1 > k + 1$.

Case 2.1: $\psi_1 = -\psi_2 \wedge l_2 - l_1 < \min\{M(k), M(l_1 - k - 1)\}$. In this case forward (ψ_1) and backward (ψ_2) links are collinear, as shown in Figure 3-8 (left). Since $l_1 > k + 1$ and $l_2 > l_1$, there must be a SAW from amino acid $k + 1$ to amino acid l_1 , and a SAW from

amino acid l_1 to amino acid l_2 . If the SAW from l_1 to l_2 is taken from left side of amino acid k (which contains amino acids 0 to $k-1$) then it must be long enough to accommodate k lattice points, because none of the amino acids prior to k in the sequence must be crossed by this walk. On the other hand, if this walk is taken from the right side of k (which contains amino acids from $k+1$ to l_1), then it must be long enough to accommodate $l_1 - k - 1$ lattice points, because it should not cross any of amino acids coming after amino acid $k+1$ and prior to l_1 . Hence, the SAW from l_1 to l_2 must be at least as long as minimum of the two required number of lattice points.

Case 2.2: $\psi_1 = -\psi_3 \wedge l_1 - k - 1 < \min\{M(k), M(N - l_1)\}$. As illustrated in Figure 3-8 (right), in this combination of directions, forward (ψ_1) and backward (ψ_2) links are perpendicular. Similar to case 2.1, in this case there must be a SAW from $k+1$ to l_1 . If the walk is taken from above amino acid k , this walk must be long enough to accommodate k lattice points, and if it is taken from below k , it must be long enough to accommodate $N - l_1$ lattice points, because there must be a SAW from l_1 to l_2 , whose length is $l_2 - l_1$, and then $N - l_2$ more points must be available to accommodate the rest of amino acids.

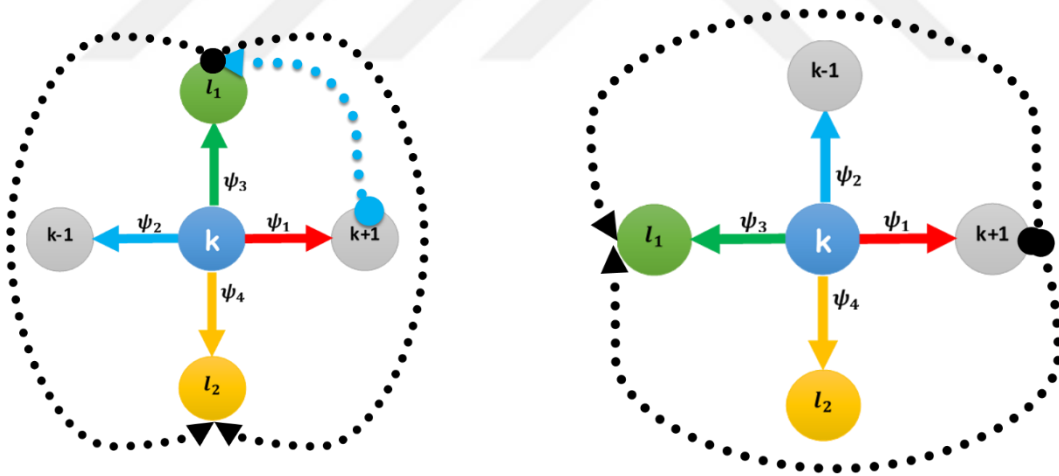


Figure 3-8 Double contact combination with collinear (left) and perpendicular (right) forward backward links, when $l_1 > k + 1$

Case 3: $l_2 < k - 1$. As illustrated in Figure 3-9, correctness of cases 3.1 (left) and 3.2 (right) can be easily verified similar to cases 2.1 and 2.2.

Case 3.1: $\psi_1 = -\psi_2 \wedge l_2 - l_1 < \min\{M(k - 1 - l_2), M(N - k - 1)\}$.

Case 3.2: $\psi_1 = -\psi_3 \wedge k - 1 - l_2 < \min\{M(l_1 + 1), M(N - k - 1)\}$.

Case 4: $l_1 < k - 1 \wedge l_2 > k + 1 \wedge \psi_2 = -\psi_3$. Figure 3-10 shows an example of combination of directions in this case. Since $l_1 < k - 1$, there must be a SAW from amino acid l_1 to amino acid $k - 1$ without passing through amino acids between $k + 1$ and l_2 , because it would make the solution self-crossing. Similarly, since $l_2 > k + 1$, there must be a path from amino acid $k + 1$ to l_2 without passing through amino acids between $k - 1$ and l_1 . It is not hard to see that this can never happen without these two walks crossing each other.

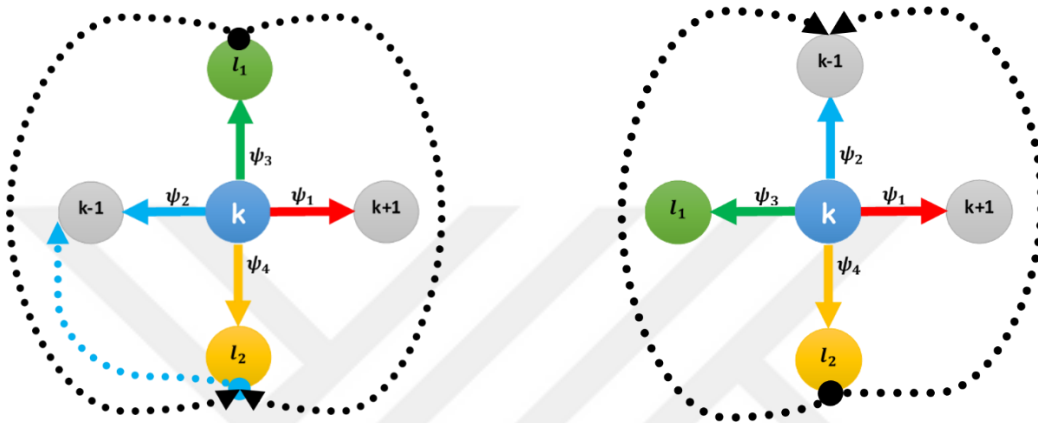


Figure 3-9 Double contact combination with collinear (left) and perpendicular (right) forward backward links, when $l_2 < k - 1$

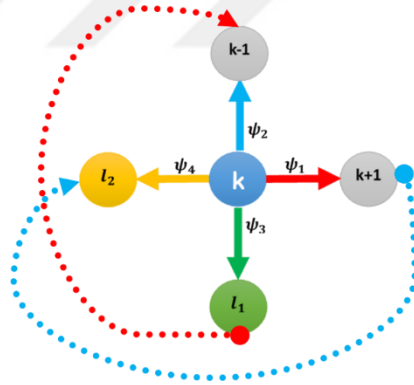


Figure 3-10 Infeasible self-intersecting walk in double contact combinations, when $l_1 < k - 1 \wedge l_2 > k + 1 \wedge \psi_2 = -\psi_3$.

Valid Inequalities

Now we introduce three sets of valid inequalities associated with QNA model.

Square double contact valid inequalities in QNA

As illustrated in Figure 3-11 (left), these valid inequalities establish that if hydrophobic amino acids h and $h + 2$ of parity 1 are connected to amino acid $h + 1$ respectively via links b (backward) and f (forward), which form a right angle (hence occupying three corners of a square), then any hydrophobic amino acid of parity 0 that forms a quadruple

contact with both h and $h+2$ must be located on the remaining corner of the square. Similar to the model, we first define certain expressions and then use these definitions in representing the valid inequalities.

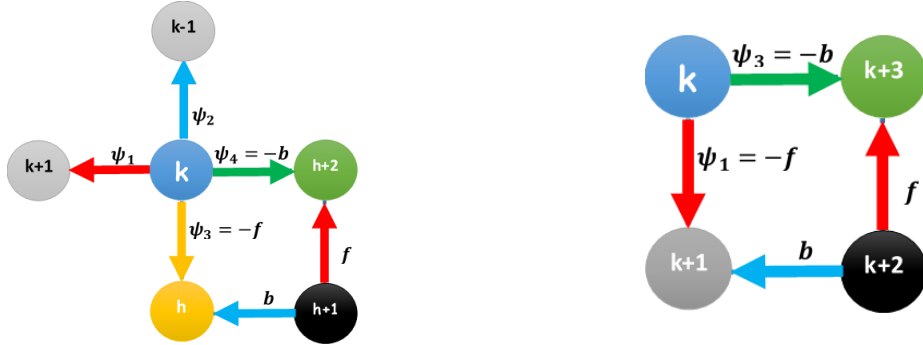


Figure 3-11 Square double (left) and single (right) contact valid inequalities in QNA

$$\begin{aligned}
 L_{10} &:= \sum_{\substack{k \in \mathcal{H}^0: \\ p+b+f \in F_k}} \sum_{\substack{\psi \in \Psi: \psi_3 = -f \\ \psi_4 = -b \\ p+b+f+\psi_1 \in F_{k+1} \\ p+b+f+\psi_2 \in F_{k-1} \\ (k, \psi, h, h+2) \in \Pi^4}} HQ_{p+b+f, \psi}^{k, h, h+2} \\
 L_{11} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_1 = -f \\ \psi_3 = -b \\ p+b+f+\psi_2 \in F_{h-2} \\ (h-1, \psi, h+2) \in \Pi^3}} \left(HT_{p+b+f, \psi}^{h-1, h+2} + \sum_{\substack{l_1 \in \mathcal{H}^1: \\ h+2 < l_1 \\ p+\psi_4 \in F_{l_1} \\ (h-1, \psi, h+2, l_1) \in \Pi^4}} HQ_{p+b+f, \psi}^{h-1, h+2, l_1} + \sum_{\substack{l_1 \in \mathcal{H}^1: \\ h-1 > l_1 \\ p+\psi_4 \in F_{l_1} \\ (h-1, \psi^\infty, l_1, h+2) \in \Pi^4}} HQ_{p+b+f, \psi^\infty}^{h-1, l_1, h+2} \right) \\
 L_{12} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_2 = -b \\ \psi_3 = -f \\ p+b+f+\psi_1 \in F_{h+4} \\ (h+3, \psi, h) \in \Pi^3}} \left(HT_{p+b+f, \psi}^{h+3, h} + \sum_{\substack{l_1 \in \mathcal{H}^1: \\ h+2 < l_1 \\ p+\psi_4 \in F_{l_1} \\ (h+3, \psi, h, l_1) \in \Pi^4}} HQ_{p+b+f, \psi}^{h+3, h, l_1} + \sum_{\substack{l_1 \in \mathcal{H}^1: \\ h-1 > l_1 \\ p+\psi_4 \in F_{l_1} \\ (h+3, \psi^\infty, l_1, h) \in \Pi^4}} HQ_{p+b+f, \psi^\infty}^{h+3, l_1, h} \right) \\
 L_{13} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_1 = f \\ \psi_2 = b}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p+\psi_3 \in F_{l_1} \\ (h+1, \psi, l_1) \in \Pi^3}} \left(HT_{p, \psi}^{h+1, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p+\psi_4 \in F_{l_2} \\ (h+1, \psi, l_1, l_2) \in \Pi^4}} HQ_{p, \psi}^{h+1, l_1, l_2} \right)
 \end{aligned}$$

And the valid inequality is:

$$\begin{aligned}
 L_{10} + \rho_{h-1} L_{11} + \rho_{h+3} L_{12} &\leq XD_{p, f, b}^{h+1} + \rho_{h+1} L_{13}, \forall h \in \mathcal{H}^1: h+2 \in \mathcal{H}^1, p \\
 &\in F_{h+1}, f \in \delta: p+f \in F_{h+2}, b \in \delta - \{f, -f\}: p+b \in F_h.
 \end{aligned} \tag{QNA-18}$$

Square single contact valid inequalities in QNA

As shown in Figure 3-11 (right), and similar to square double contact, if $k \in \mathcal{H}^0$ is going to form a contact with $k + 3 \in \mathcal{H}^1$, while amino acid $k + 1$ is polar, then amino acids k , $k + 1$, $k + 2$, and $k + 3$ must occupy corners of a square. Note that if amino acid $k + 1$ is hydrophobic, then these constraints will be covered in stronger forms as in square double contact valid inequalities. Similar to the model, we first define certain expressions and then use these definitions in representing the valid inequalities.

$$\begin{aligned}
 L_{14} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_2=b \\ \psi_3=d \\ p+\psi_1 \in F_{k+1}}} \left(HT_{p,\psi}^{k,k-3} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ k+1 < l_2 \\ p+\psi_4 \in F_{l_2} \\ (k,\psi,k-3,l_2) \in \Pi^4}} HQ_{p,\psi}^{k,k-3,l_2} \right) \\
 L_{15} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_2=b \\ \psi_4=d \\ p+\psi_1 \in F_{k+1}}} \sum_{\substack{l_2 \in \mathcal{H}^1: \\ k-3 > l_2 \\ p+\psi_3 \in F_{l_2} \\ (k,\psi,l_2,k-3) \in \Pi^4}} HQ_{p,\psi}^{k,l_2,k-3} \\
 L_{16} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_1=-d \\ \psi_2=-b}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p+b+d+\psi_3 \in F_{l_1} \\ (k-2,\psi,l_1) \in \Pi^3}} \left(HT_{p+b+d,\psi}^{k-2,l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p+b+d+\psi_4 \in F_{l_2} \\ (k-2,\psi,l_1,l_2) \in \Pi^4}} HQ_{p+b+d,\psi}^{k-2,l_1,l_2} \right)
 \end{aligned}$$

And the valid inequality reads as:

$$\begin{aligned}
 L_{14} + L_{15} &\leq XD_{p+b+d,-d,-b}^{k-2} + \rho_{k-2} L_{16}, \forall k \in \mathcal{H}^0: k-3 \in \mathcal{H}^1, k-1 \notin \mathcal{H}^1, p \\
 &\in F_k, b \in \delta: p+b \in F_{k-1}, d \in \delta - \{b, -b\}: p+d \in F_{k-3}.
 \end{aligned} \tag{QNA-19}$$

Similar to forward square single contact, we can write these valid inequalities considering amino acids $k-1$, $k-2$, and $k-3$.

$$L_{17} := \sum_{\substack{\psi \in \Psi: \\ \psi_1=f \\ \psi_3=d \\ p+\psi_2 \in F_{k-1}}} \left(HT_{p,\psi}^{k,k+3} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ k+3 < l_2 \\ p+\psi_4 \in F_{l_2} \\ (k,\psi,k+3,l_2) \in \Pi^4}} HQ_{p,\psi}^{k,k+3,l_2} \right)$$

$$L_{18} := \sum_{\substack{\psi \in \Psi: \\ \psi_1=f \\ \psi_4=d \\ p+\psi_2 \in F_{k-1}}} \sum_{\substack{l_2 \in \mathcal{H}^1: \\ k-1 > l_2 \\ p+\psi_3 \in F_{l_2}}} HQ_{p,\psi}^{k,l_2,k+3}$$

$$L_{19} := \sum_{\substack{\psi \in \Psi: \\ \psi_1=-f \\ \psi_2=-d}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p+f+d+\psi_3 \in F_{l_1} \\ (k+2,\psi,l_1) \in \Pi^3}} \left(HT_{p+f+d,\psi}^{k+2,l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p+f+d+\psi_4 \in F_{l_2} \\ (k+2,\psi,l_1,l_2) \in \Pi^4}} HQ_{p+f+d,\psi}^{k+2,l_1,l_2} \right)$$

And the valid inequality is defined as:

$$L_{17} + L_{18} \leq XD_{p+f+d,-f,-d}^{k+2} + \rho_{k+2} L_{19}, \forall k \in \mathcal{H}^0: k+3 \in \mathcal{H}^1, k+1 \notin \mathcal{H}^1, p \in F_k, f \in \delta: p+f \in F_{k+1}, d \in \delta - \{f, -f\}: p+d \in F_{k+3}. \quad (\text{QNA-20})$$

Rectangular single contact valid inequalities in QNA

These valid inequalities are in fact derived from the observations in case 2 and case 3 of feasible single contact combinations. Note that as illustrated in Figure 3-7 (left), amino acids k and $k-5$ being in contact along direction ψ_3 , while this direction is collinear with backward direction of k (ψ_2), and both perpendicular to the forward direction of k (ψ_1), implies that amino acids k to $k-5$ must form a 2 by 1 rectangle, such that amino acids $k-1$, $k-2$, $k-4$, and $k-5$ are located at the four corners of this rectangle. This in turn leads to two implications: 1) the forward and backward directions of $k-2$ are respectively ψ_3 and ψ_1 , 2) the forward and backward directions of $k-4$ are respectively ψ_2 and ψ_1 . Note that similar conclusions can be drawn for case 3 of feasible single contact.

The first valid inequality, as given in (QNA-21), addresses the necessity of $k-4$ possessing perpendicular forward and backward directions. In this inequality, f and b are respectively the forward and the backward directions of k . Therefore, if amino acid k is located at lattice point p , then $k-4$ must be located at point $p-f-b$, while its forward and backward directions are respectively b and f . The left hand side of (QNA-21) accounts for all possible ways satisfying these requirements.

$$\begin{aligned}
L_{20} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_1 = b \\ \psi_2 = f}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-f-b+\psi_3 \in F_{l_1} \\ (k-4, \psi, l_1) \in \Pi^3}} \left(HT_{p-f-b, \psi}^{k-4, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-f-b+\psi_4 \in F_{l_2} \\ (k-4, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-f-b, \psi}^{k-4, l_1, l_2} \right) \\
L_{21} &:= \sum_{\substack{l_2 \in \mathcal{H}^1: k-5 < l_2 \\ p-f \in F_{l_2} \\ (k, (f, b, -b, -f), k-5, l_2) \in \Pi^4}} HQ_{p, (f, b, -b, -f)}^{k, k-5, l_2} \\
L_{22} &:= \sum_{\substack{l_2 \in \mathcal{H}^1: k-5 > l_2 \\ p-f \in F_{l_2} \\ (k, (f, b, -f, -b), l_2, k-5) \in \Pi^4}} HQ_{p, (f, b, -f, -b)}^{k, l_2, k-5}
\end{aligned}$$

The right hand side of (QNA – 21) accounts for all possible ways of k being in contact with $k - 5$, with the given location and directions.

$$\begin{aligned}
XD_{p-f-b, b, f}^{k-4} + \rho_{k-4} L_{20} &\geq HT_{p, (f, b, -b, -f)}^{k, k-5} + L_{21} + L_{22}, \forall k \in \mathcal{H}^0: k-5 \in \mathcal{H}^1, p \in F_k, f \\
&\in \delta: p+f \in F_{k+1}, b \in \delta - \{f, -f\}: p+b \in F_{k-1} \wedge p-b \in F_{k-5}.
\end{aligned} \tag{QNA-21}$$

Similarly, second valid inequality of this type as given in (QNA – 22) addresses the requirement that if amino acid k is located at lattice point p , then $k - 2$ must be located at point $p - f + b$, while its forward and backward directions are respectively f and $-b$.

$$L_{23} := \sum_{\substack{\psi \in \Psi: \\ \psi_1 = f \\ \psi_2 = -b}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-f+b+\psi_3 \in F_{l_1} \\ (k-2, \psi, l_1) \in \Pi^3}} \left(HT_{p-f+b, \psi}^{k-2, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-f+b+\psi_4 \in F_{l_2} \\ (k-2, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-f+b, \psi}^{k-2, l_1, l_2} \right)$$

$$\begin{aligned}
XD_{p-f+b, f, -b}^{k-2} + \rho_{k-2} L_{23} &\geq HT_{p, (f, b, -b, -f)}^{k, k-5} + L_{21} + L_{22}, \forall k \in \mathcal{H}^0: k-5 \in \mathcal{H}^1, p \in F_k, f \\
&\in \delta: p+f \in F_{k+1}, b \in \delta - \{f, -f\}: p+b \in F_{k-1} \wedge p-b \in F_{k-5}.
\end{aligned} \tag{QNA-22}$$

Inequalities (QNA – 23) and (QNA – 24) are respectively analogous to (QNA – 22) and (QNA – 21) when considering the contact between k and $k + 5$.

$$\begin{aligned}
L_{24} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_1=b \\ \psi_2=-f}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-b+f+\psi_3 \in F_{l_1} \\ (k+2, \psi, l_1) \in \Pi^3}} \left(HT_{p-b+f, \psi}^{k+2, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-b+f+\psi_4 \in F_{l_2} \\ (k+2, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-b+f, \psi}^{k+2, l_1, l_2} \right) \\
L_{25} &:= \sum_{\substack{l_2 \in \mathcal{H}^1: k+5 > l_2 \\ p-b \in F_{l_2} \\ (k, (f, b, -b, -f), l_2, k+5) \in \Pi^4}} HQ_{p, (f, b, -b, -f)}^{k, l_2, k+5}, \quad L_{26} := \sum_{\substack{l_2 \in \mathcal{H}^1: k+5 < l_2 \\ p-b \in F_{l_2} \\ (k, (f, b, -f, -b), k+5, l_2) \in \Pi^4}} HQ_{p, (f, b, -f, -b)}^{k, k+5, l_2} \\
L_{27} &:= \sum_{\substack{\psi \in \Psi: \\ \psi_1=b \\ \psi_2=f}} \sum_{\substack{l_1 \in \mathcal{H}^1: \\ p-b-f+\psi_3 \in F_{l_1} \\ (k+4, \psi, l_1) \in \Pi^3}} \left(HT_{p-b-f, \psi}^{k+4, l_1} + \sum_{\substack{l_2 \in \mathcal{H}^1: \\ l_1 < l_2 \\ p-b-f+\psi_4 \in F_{l_2} \\ (k+4, \psi, l_1, l_2) \in \Pi^4}} HQ_{p-b-f, \psi}^{k+4, l_1, l_2} \right)
\end{aligned}$$

And the valid inequalities read as:

$$XD_{p-b+f, b, -f}^{k+2} + \rho_{k+2} L_{24} \geq HT_{p, (f, b, -f, -b)}^{k, k+5} + L_{25} + L_{26}, \forall k \in \mathcal{H}^0: k+5 \in \mathcal{H}^1, p \in F_k, f \quad (\text{QNA-23})$$

$$\in \delta: p+f \in F_k, b \in \delta - \{f, -f\}: p+b \in F_{k-1} \wedge p-f \in F_{k+5},$$

$$XD_{p-b-f, b, f}^{k+4} + \rho_{k+4} L_{27} \geq HT_{p, (f, b, -f, -b)}^{k, k+5} + L_{25} + L_{26}, \forall k \in \mathcal{H}^0: k+5 \in \mathcal{H}^1, p \in F_k, f \quad (\text{QNA-24})$$

$$\in \delta: p+f \in F_k, b \in \delta - \{f, -f\}: p+b \in F_{k-1} \wedge p-f \in F_{k+5}.$$

Chapter 4

METAHEURISTIC APPROACHES FOR LINEAR CHAIN HP MODEL IN CUBIC LATTICE

As we will show in the computational results in Chapter 5, solving the problem under square lattice using the presented MILP models even for sequences as short as 50 amino acids becomes intractable. This length is less than 30 amino acids in the cubic lattice. Hence, we resort to *metaheuristic* methods to find *near optimal* solutions for the longer sequences. Later we will show how we can combine these metaheuristic approaches with the models to improve the performance of the models.

Due to the difficult nature of the PSP problem, we focus on metaheuristic algorithms rather than the simple heuristics. In this chapter we first introduce a method for generating random feasible solutions (*RFSG*) because it is an important element of the metaheuristic algorithms. Then we introduce a *Tabu Search* algorithm which can be used either as a standalone method, or as a local search method implemented inside other metaheuristic algorithms. Then we define an *Iterated Local Search (ILS)* and a *Scatter Search (SS)* for PSP problem.

Solution Representation and Standardization

Solution representation is the first decision to be made in any metaheuristic algorithm and has an effect on the success of them. We represent solutions with an array of points in the 2D and 3D space. To standardize the solutions, we eliminate the symmetric solutions using either central or terminal symmetry breaking techniques explained earlier in Chapter 3.

Biased Random Solution Generation

In many steps of our metaheuristic methods, we need to generate random solutions. These solutions may or may not be inclined towards some lattice points. We call the former biased random solution and the latter unbiased or uniform random solution. In RFSG, we take advantage of the set of feasible lattice points introduced in IP models. Let w_p^k denote the relative tendency of assigning amino acid k to lattice point p . Note that if w_p^k is a constant value for all k and p , then the procedure will be an unbiased random solution generation. The idea is to fix amino acids m and $m + 1$ at their predefined feasible lattice points, and extend the chain from both sides, choosing feasible lattice points one at a time, until both terminus amino acids are assigned to feasible lattice points. During this procedure, we might end up with infeasible solutions that one of the ends gets trapped in a dead-end inside other amino acids, and we cannot extend the chain from that end. In such cases we apply a back-track mechanism, in which we unfold the

chain back to a desired position, and continue with the procedure. The detailed pseudocode of RFSG using a back-track mechanism is given in Appendix III.

Iterative Biased Random Feasible Solution Generation

In many occasions, we need to find near optimal solutions with respect to some objective functions. Since RFSG is very quick, we can iteratively update the relative tendency values (w_p^k) towards the solutions that improve the objective function, and start over with generating random solutions biased towards these new values, until some steady state or stopping condition (for example predefined number of iterations) is met. We refer to this method as *Iterative Biased Random Feasible Solution Generation (IB-RFSG)*, and use it as an alternative for finding solutions with relatively good objective values in a very short amount of time.

4.1 Tabu Search

In this section we propose a variation of Tabu Search (TS) algorithm for finding near optimal solutions for the PSP problem under cubic lattice. TS, proposed by Glover (1986), is a metaheuristic search method that employs Local Search (LS) methods for solving mathematical optimization problems. Given a solution, LS methods greedily find the best solution in the neighborhood of this solution, which may quickly result in stagnation in the local optimum solutions.

TS uses mechanisms for preventing the LS to visit the regions of the solution space that have recently been visited, thus the term *tabu*. TS may employ variety of strategies to force prohibition. A common way of forcing such prohibitions is to completely or partially store the recently visited solutions in a *tabu list* for a specific number of iterations, and exclude them from the possible solutions that LS would visit. Variety of such strategies for incorporating memory to mimic *hill climbing* have been proposed in the literature. One such strategy, which we implement in the proposed tabu search, is explained in the following.

The proposed TS is inspired by the *Human-Guidable Tabu Search (GTabu)* proposed by Klau et al. (2002), and applied to HP PSP problem under square lattice by Lesh et al. (2003). In our TS method, we adapt this method to the cubic lattice model, and slightly modify the basic concepts of GTabu. This implementation of TS is applicable to both square and cubic lattices, and can work with any arbitrary objective function, including all 20×20 contacts between amino acids.

In the concept of GTabu, each problem instance consists of finite number of *elements*. In the PSP problem, these elements can be defined as the amino acids in the sequence. Furthermore, each move in the neighborhood structure is initiated by an element, and alters that element and possibly other elements of the problem instance. In order to guide the search, instead of working with *tabu lists* as in the conventional TS algorithms, GTabu assigns three types of *mobility states* to the elements of the problem, namely *high*, *medium* and *low* mobilities. When an element is in the high mobility, it can initiate a move, or be altered by a move initiated by another element. If the element is in medium mobility state, it cannot initiate a move, but can be altered by a move initiated by another element. Finally, if an element is in low mobility state, it can neither initiate a move, nor be altered by any other move. In the basic GTabu, elements of low mobility are selected in the beginning by the user, and do not change till the end of the algorithm.

In the generic GTabu, each element is initially in the high mobility state. In each iteration, all elements of high mobility that can initiate a move are identified. Then the best move that does not violate the mobility states of other elements is applied. After each iteration, the mobilities are updated to prevent cycling and returning to the already explored regions of the search space. GTabu sets the mobility state of the altered elements to medium, and they remain in that state for a *memorySize* number of iterations, where *memorySize* is a parameter of GTabu. After this many iterations are passed, mobility of each of these elements is set back to high (unless further delayed due to being altered by other moves before reaching this number of iterations).

GTabu also encourages the less frequently altered elements to be chosen, by setting the mobility of the more frequently altered elements to medium for a number of iterations. To do so, GTabu defines *diversity* measures for the elements, moves, and the algorithm, and changes the mobility of the elements when diversity of algorithm falls below a threshold *minDiv*, where *minDiv* is another parameter of GTabu. For a detailed explanation of this measure, along with further details of GTabu, the reader is referred to Klau et al. (2002).

In the GTabu proposed by Lesh et al. (2003), the elements are defined as the amino acids in the sequence, therefore the problem representation of a sequence of N amino acids will contain N elements. In their implementation, an element is altered if the location of the amino acid is changed due to the move. In our implementation, however, we define the elements as the possible contacts between nonconsecutive amino acids, because we believe that in the search procedure, it is not the location of amino acids that determines the state of the solution and the algorithm, but the interactions between different amino acids, as contacts determine the objective function. With this definition of elements, in

the cubic or square lattices, the problem representation of a sequence of N amino acids, with $|\mathcal{R}^0|$ amino acids of parity 0, and $|\mathcal{R}^1|$ amino acids of parity 1, will contain $|\mathcal{R}^0| \times |\mathcal{R}^1| - (N - 1)$ elements.

In our implementation, we assign mobility states to the *pairs* of nonconsecutive amino acids. An element kl is altered by a move, if the contact between amino acids kl is broken or formed. If a move does not break an existing contact between amino acids k and l , or does not form a contact between amino acids k and l that are not in contact, we say that the element kl is not altered by the move.

A set of local moves for the HP PSP problem under lattice square is proposed by Lesh et al. (2003). These move sets along with other moves are used to locally move from a feasible solution X to another feasible solution X' , in the local search procedure. Therefore, the neighborhood of a solution X is the set of all feasible solutions X' that are obtained by specific changes on X . In our TS algorithm, for both square and cubic lattices, we use four types of neighborhoods: *corner move neighborhood*, *pull move neighborhood*, *crankshaft move neighborhood*, and *end move neighborhood*.

Corner Moves

Corner moves (sometimes referred to as *king move*) along with end moves are proposed by Verdier & Stockmayer (1962) and only consider single amino acid moves. A nonterminal amino acid k can be moved if its predecessor ($k - 1$) and successor ($k + 1$) amino acids in the sequence occupy diagonally adjacent lattice points. As shown in Figure 4-1 (top left), suppose that amino acids k , $k - 1$ and $k + 1$ are located at lattice points p_k , p_{k-1} , and p_{k+1} , respectively. Since p_{k-1} and p_{k+1} are diagonally adjacent, p_{k-1} , p_k and p_{k+1} form three corners of a square. Let L denote the other corner of the square. If L is not occupied by any other amino acid, then we can move k from p_k to L without losing feasibility of the current solution. Note that for a sequence of N amino acids, at most $N - 2$ corner moves are possible. Note that the same move can be defined for the cubic lattice. Furthermore, in this move amino acid k is called the initiating amino acid.

Pull Moves

This set of moves is introduced by Lesh et al. (2003) and are considered to be the most effective moves proposed for the PSP HP problem under the square lattice. Lesh et al. (2003) prove that this set of moves is *complete* in the square lattice, in the sense that any arbitrary solution can be transformed to any other arbitrary solution by performing

a series of pull moves. In the following we first introduce this move in the square lattice, and further explain how it can be extended to the cubic lattice.

In the square lattice, suppose that amino acids k and $k + 1$ are respectively located at lattice points p_k and p_{k+1} . As illustrated in Figure 4-1 (bottom left), further suppose that there exists an empty lattice point L which is adjacent to p_{k+1} and diagonally adjacent to p_k , therefore lattice points p_k , p_{k+1} , and L form three corners of a square. Let us denote by C the other corner of the square. The idea of pull move is to “pull” amino acid k from p_k to L while maintaining the connectivity of the chain. Note that if C is occupied by amino acid $k - 1$, then this move is equivalent to corner move. Therefore, we assume that in pull move this lattice point is always empty. In this case, by moving k from p_k to L , the lattice points p_k and C become empty, but the chain also becomes disconnected. If we move amino acid $k - 1$ to C , then connectivity with amino acid k is restored, but $k - 1$ may remain disconnected with its predecessor. The idea of pull move is to recursively move the predecessors of k to newly emptied lattice points, until the chain becomes fully connected again.

Since in the explained procedure we try to retain connectivity by moving backward in the sequence, we refer to this move as the *backward pull move*. We can similarly define this move by moving forward in the sequence, which we refer to as the *forward pull move*. Moreover, note that in this move the amino acid that initiates the move is k , hence we refer to it as the initiating amino acid.

This move is extended to the 3D cubic lattice by Thachuk et al. (2007), in their Replica Exchange Monte Carlo (REMC) algorithm proposed for the HP PSP problem under cubic lattice. Note that in the square lattice, points L and C are restricted to a single plane. This plane contains both p_k and p_{k+1} , therefore to generalize this move to the 3D cubic lattice, we can as well consider the plane which is perpendicular to the plane that contains p_k and p_{k+1} . Also note that in the square (cubic) lattice, a given amino acid k , may initiate four (eight) different pull moves.

Crankshaft Moves

A crankshaft can be initiated by amino acid k , if amino acids $k - 1$, and $k + 2$ occupy adjacent lattice points. Let p_l denote the position of amino acid l , for $l \in \{k - 1, \dots, k + 2\}$. Note that this implies that p_{k-1} , p_k , p_{k+1} , and p_{k+2} are four corners of a square, i.e. $p_k - p_{k-1} = p_{k+1} - p_{k+2}$. Note that $p_k - p_{k-1}$ and $p_{k+1} - p_{k+2}$ are perpendicular to the direction $p_{k+2} - p_{k-1}$, which we refer to as the *rotation direction*. Let L and C

respectively denote the lattice points adjacent to p_{k-1} and p_{k+2} that are respectively obtained by rotating p_k and p_{k+1} around the rotation direction by 180 degrees. If both lattice points L and C are empty, we can move k and $k+1$ respectively to L and C , without losing connectivity of the chain.

As illustrated in Figure 4-1 (bottom right), we can extend this move to the 3D cubic lattice, by rotating p_k and p_{k+1} around the rotation direction by ± 90 or 180 degrees. This extension also appears in Thachuk et al. (2007). However, in REMC rotating by 180 degrees is excluded in the 3D space. This can be due to the fact that rotation by 180 degrees can be obtained by two consecutive $+90$ (or -90) rotations, and including the 180 rotation might affect locality of the move. However, we must note that these consecutive rotations can only occur if both of them are possible, i.e. if the first rotation is not possible we cannot achieve the 180 degrees rotation. Therefore, in our implementation we also include the 180 degrees rotation in case neither of ± 90 degrees can be applied. In the crankshaft move, only position of amino acids k and $k+1$ is altered, and we refer to k as the initiating amino acid. Note that in crankshaft move $k \in \{1, \dots, N-3\}$.

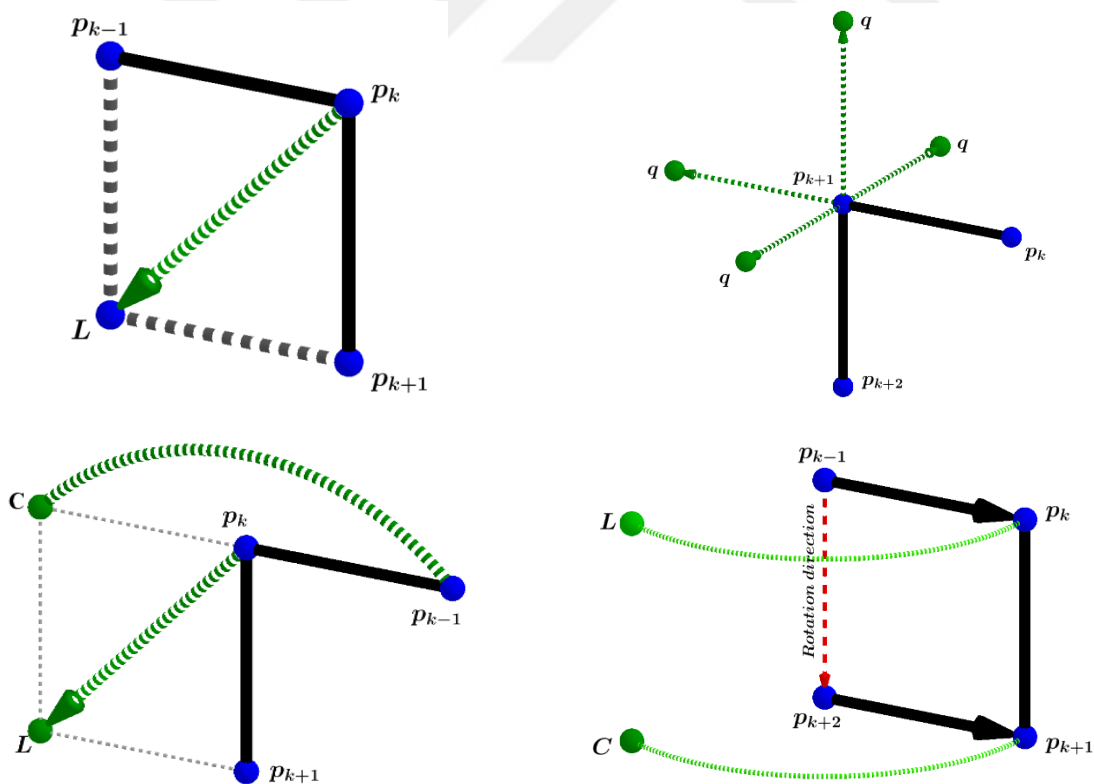


Figure 4-1 Possible moves used for linear chain lattice models in cubic lattice. Top left: corner move, top right: end move, bottom left: pull move, bottom right: crankshaft

End Moves

End move is the last and simplest type of move that can be performed on the terminal amino acids only, i.e. $k \in \{0, N - 1\}$. We explain the end move for $k = 0$, and move for $k = N - 1$ is exactly the same. Let p_k , p_{k+1} and p_{k+2} respectively denote the lattice points occupied by amino acids k , $k + 1$, and $k + 2$. As illustrated in Figure 4-1 (top right), amino acid k can be moved to any lattice points $q = p_{k+1} + d$, where $d \in \delta$, if q is not occupied by $k + 1$ and $k + 2$ (i.e. $q \notin \{p_k, p_{k+2}\}$) or other amino acids. Note that in square (cubic) lattice, at most 2 (4) such empty lattice points exist.

Proposed Tabu Search for PSP

Let M_{kl}^t denote the mobility state of pair kl at iteration t of Tabu Search. Also let M_{kl}^0 denote the initial mobility states. By D_{kl}^M we denote the iteration of TS in which mobility of pair kl must change, and initially $D_{kl}^M = \infty$ for all pairs. In our implementation of TS, change of state of mobility can occur either from medium to high (*MtH*), or from low to medium (*LtM*) in the presence of low mobility states. At the beginning of iteration t of TS, first we check the mobility states of pairs. If the iteration in which mobility of pair kl should change state is due (i.e. $D_{kl} \leq t$), we reset its mobility to high if the current state is medium, and to medium if the current state is low, and we set D_{kl}^M to ∞ in either case.

Next, in all allowable neighborhood structures (i.e. corner, pull, crankshaft, and end moves), we look for an initiating amino acid that transforms the current solution to the best neighbor solution. Let k denote the amino acid that initiates the move. Note that in the context of GTabu, this amino acid must be in the high mobility state. However, since we define mobility states for pairs, we say a move is initiated with high mobility, if there exists at least one amino acid l , with which M_{kl}^t is high and the pair kl is altered by the move. In other words, we abort a move initiated by k if all of the altered pairs kl are in the medium state. Note that in the presence of low mobility states, none of the entire altered pairs should be in the low mobility state.

In our implementation of TS, the objective function is to be maximized, therefore we evaluated the goodness of solutions with how high their objective value is. After choosing the best neighbor, we set the mobility state of altered pairs (high or medium) to medium, and set D_{kl}^M to $t + HtM_{kl}$, where HtM_{kl} is the number of iterations required for the pair kl to remain in the medium mobility state after changing from high mobility state. Finally we set the current solution to the obtained neighbor, and continue the procedure with this new solution. These steps are repeated until a stopping condition is met. This

stopping condition can be reaching a limit for non-improving iterations, and/or when the best known objective value reaches a predefined upper bound.

The procedure explained above is the most basic form of our TS algorithm. In the original implementation of GTabu, a stochastic component called noise is introduced to slightly change the mobility of elements in each iteration. With this component, at the beginning of each iteration mobility of elements of high mobility is changed to medium with probability P_{noise} and is kept in this state for one iteration, where P_{noise} is a parameter of GTabu. Moreover, in our TS we may or may not allow all four neighborhood structures. Furthermore, in this method we consider aspiration conditions, i.e. we neglect the mobility state violations if the neighbor solution globally improves the incumbent solution. This is reasonable due to the fact that if a solution is better than the best known solution, then it is the first time that we are visiting this solution. As we will explain in ILS, we use TS for different purposes.

In the HP model case, at each iteration we also record the “good” visited solutions. By good solutions we imply the solutions that the difference between their absolute energy and the globally best known absolute energy (e^*) is not less than a given threshold. Note that if TS is used inside another algorithm, such as ILS or SS, e^* is not necessarily the best objective value of the solutions visited by TS in the current run. As we will explain in ILS and SS, we record these solutions for training purposes. This training includes but is not limited to detecting the contacts to fix, and contacts to avoid. A detailed pseudocode of TS is given in Algorithm 4 in Appendix III.

Trainable Tabu Search

We can iteratively use TS inside another algorithm, which we refer to as the *main algorithm*. During the main algorithm, multiple independent or dependent TSs are performed, starting with different initial solutions. During these runs, one TS may explore regions of the solution space that may remain unexplored by other runs of TS. Therefore, we can use these “*experiences*” in TS and in the main algorithm. One way of gaining experience is to detect the frequency of occurrence of each contact in different solutions. These frequencies can be used for many training purposes. As we will describe in ILS, at some points we need to avoid the regions of the solution space that have been explored frequently. In that case, we incorporate the recorded frequencies in the objective function in different manners, as a perturbation mechanism.

We can also use these frequencies for detecting the promising partial solutions. It is believed that in real proteins, once a hydrophobic contact is formed, it cannot be broken. This hypothesis is the basis of heuristic methods such as hydrophobic zipper (Dill et al., 1993) and contact interaction algorithm (Toma & Toma, 1996). Inspired by this

hypothesis, we allow TS to fix the contacts that have been frequently observed in “good” solutions. By good solution we mean the solutions whose absolute energy falls inside a certain interval of fixed length centered at the globally best known absolute energy by the main algorithm. We also refer to the frequently observed contacts in the good solutions as the good contacts.

We can adapt our TS algorithm to this new feature by temporarily changing the mobility state of such contacts to low, once they have been observed in the current solution. Note that in our implementation, these contact can be hydrophobic or any other type of contact. However, our experiments show that only hydrophobic contacts frequently appear in the good solutions. It is worth mentioning that we choose a limited number of good contacts that only appear in the globally best known solution, and we provide these contacts at the beginning of TS.

4.2 Iterated Local Search

Iterated Local Search (ILS) is a metaheuristic that extends LS and hill climbing to broader concepts of neighborhoods. Here we briefly explain the basic concepts of ILS, and the interested reader should refer to Lorenço et al. (2010) for a detailed framework for this metaheuristic, and to Johnson & McGeoch (1997) for historical developments of ILS.

Let us denote by S the set of all feasible solutions, and by S^* set of local optima obtained by the LS algorithm. Starting with an arbitrary solution $s \in S$, LS uses “moves” to improve this solution and maps it to a solution $s^* \in S^*$. The idea of Iterated Local Search (ILS) is to find a way to “jump” from the “good” solution $s_1^* \in S^*$ to another good solution $s_2^* \in S^*$, and start LS from s_2^* . This jump should be big enough to take us out of basin of attraction of s_1^* . In our proposed ILS, we use TS as the local search method. Rather than starting over with multiple randomly generated solutions, and performing TS over these solutions, we aim to start TS with good initial solutions previously obtained by TS itself. Algorithm 1 shows the pseudocode explaining the procedure of ILS using TS. In this algorithm after performing the LS on a given solution, we perform the perturbation phase to mimic the jump, and obtain another good solution to be used as the initial solution of LS in the next iteration. These steps are repeated until a predefined number of non-improving iterations is reached.

*Algorithm 1 Overview of Iterated Local Search***Input:**

- P : Initial solution
- NI_{max} : Maximum number of non-improving iterations

Output:

- P^* : Improved solution

-
1. Set $NI = 0$
 2. **While** $NI < NI_{max}$ **Do**
 - 2.1. Perturb P and obtain P' (Jump)
 - 2.2. Improve P' and obtain P^* (Local Search)
 - 2.3. **If** $E(P^*) > E(P)$
 - 2.3.1. $P = P^*$
 - 2.3.2. $NI = 0$
 - 2.4. **Else**
 - 2.4.1. $NI = NI + 1$
 - 2.5. **End If**
 3. **End While**
-

Perturbation Phase

One way to define a jump from a good solution to another good solution, is to slightly perturb the objective function, and apply LS on s_1^* with the new objective function. In the context of PSP problem, perturbing the objective function is implied by changing the coefficients of different contacts in the objective function. To define the new objective function, first we define three contact matrices H_{kl} , F_{kl} , and G_{kl} . H_{kl} is the conventional HP contact matrix, in which a cell is 1 if k and l are both H, and 0 otherwise. F and G are defined based on the experience we have gained in the previous LS iterations. F_{kl} is simply the relative frequency of contact kl as calculated in $(ILS - 1)$, and shows the rate of observing a contact between k and l during the searching procedure. Note that we want to investigate new regions of the solution space, hence contacts with higher frequencies are less desirable, and contribute negatively to the objective function. These values help increase the length of the jump by reducing compactness of the current solution, hence guiding it out of basin of attraction of s_1^* . On the other hand, G_{kl} is the relative “goodness” of contact kl , and is calculated as in $(ILS - 2)$:

$$F_{kl} = \frac{\sum_{e=0}^{e^*} f_{kl}^e}{\sum_{e=0}^{e^*} n^e}, \quad (\text{ILS-1})$$

$$G_{kl} = \frac{Q_{kl}}{\max_{ij}\{Q_{ij}\}}, \quad Q_{kl} = \sum_{e=e_m}^{e^*} \frac{f_{kl}^e}{n^e} (e - e_m + 1). \quad (\text{ILS-2})$$

In (ILS-1) and (ILS-2), f_{kl}^e is frequency of contact kl at absolute energy level e , n^e is the total number of recorded solutions of absolute energy level e , e^* is the highest absolute energy level ever recorded, and e_m is the least acceptable absolute energy level (usually $e_m = e^* - 2$). Finally, contact coefficients in the perturbed objective function are obtained by linear combinations of these three matrices:

$$C_{lk} = \alpha_H H_{kl} + \alpha_G G_{kl} - \alpha_F F_{kl}, \quad \alpha_H + \alpha_F + \alpha_G = 1, \quad \alpha_H, \alpha_F, \alpha_G \geq 0. \quad (\text{ILS-3})$$

Parameters α_H , α_F , and α_G , which we refer to as perturbation vector, control the length of the jump, as well as quality of obtained solutions. Since in ILS, s_1^* is updated only upon improvements in the energy level, it is possible that at some non-improving iterations, due to lack of “fresh” information in the recent attempts of LS, F_{kl} and G_{kl} might not change significantly from one iteration to another non-improving iteration. Hence, we might end up with the same s_2^* starting with s_1^* in consecutive iterations. To tackle this issue, we introduce some stochastic elements in the objective function, by multiplying α_H , α_F and α_G with uniformly generated random numbers r_H , r_F and r_G , respectively:

$$C_{lk} = r_H \alpha_H H_{kl} + r_G \alpha_G G_{kl} - r_F \alpha_F F_{kl}. \quad (\text{ILS-4})$$

Figure 4-2 shows a sample solution for the benchmark instance S1-7 of standard benchmark instances before and after performing the perturbation phase. In this example, the absolute energy of the initial solution (left) before perturbation is 35, and 33 in the resulting solution, while they are totally different solutions. It is worth mentioning that in most cases, as in this example, the perturbation phase transforms the solution into a solution with lower absolute energy value. However, in some cases it might even help improving the current solution. We must also note that in the perturbation phase, we allow LS to incorporate all the four neighborhoods, to increase the irreversibility of obtained solutions.

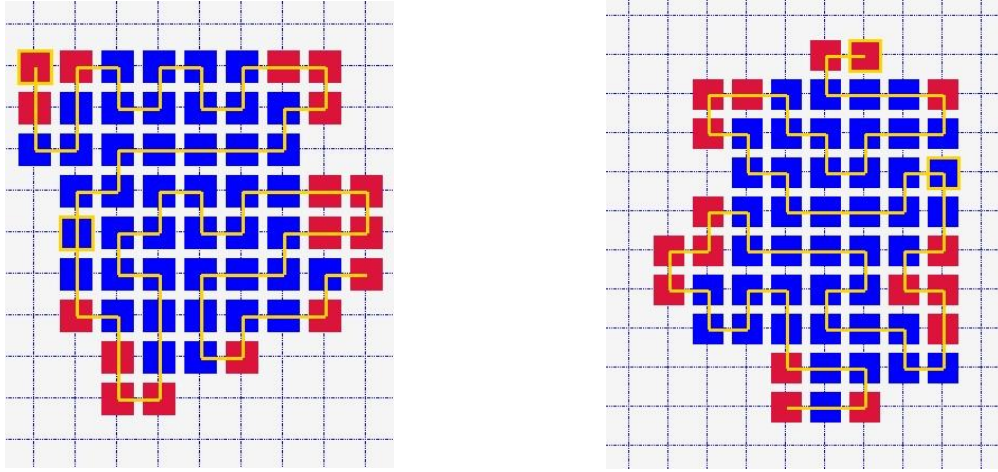


Figure 4-2 A sample solution before and after perturbation in ILS

4.3 Scatter Search

Scatter Search (SS) is a population based metaheuristic proposed by Glover (1977), in an attempt to solve combinatorial optimization problems using linear combinations of solutions. SS starts with generating a pool of solutions. It then chooses a subset of good solutions in the initial pool as the reference set, and uses strategies to select subsets of reference set, and to combine these solution. It then updates the current reference set based on the quality of the obtained offspring solutions. Our SS method for PSP problem is based on the framework of Scatter Search provided by Glover (1998). Our SS consists of the following routines:

- Diverse Solution Generation Routine
- Subset Generation Routine
- Solution Combination Routine
- Improvement (Local Search) Routine

In Scatter Search, we first generate a pool of initial diverse solutions, denoted by P_0 , with size $PSize$, where $PSize$ is a parameter of SS. To generate P_0 , first we generate a feasible random solution using RFSG method, and improve it using the Improvement Routine, which is the Tabu Search algorithm in our implementation. Then we generate next solution using the Diverse Solution Generation Routine. After improving this solution, we add it to P_0 , and repeat this step until size of P_0 reaches $PSize$.

In our implementation of SS, we divide the reference set, denoted by RS , into two mutually exclusive subsets. The first subset, which we denote by RS_H and call it high quality reference set, consists of B_1 solutions with the best objective values. The other

subset, which we denote by RS_D and refer to as the diverse reference set, consists of B_2 solutions with highest diversity values with respect to RS_H .

To generate the reference set, we first choose B_1 best solutions in P_0 , remove them from P_0 , and add them to RS_H . Then one by one, we choose B_2 most diverse solutions of P_0 with respect to RS_H , and add them to RS_D . Next steps consist of generating subsets of reference set, combining them, and generating new offspring solutions. In each iteration of SS, we choose subsets of RS using Subset Generation Routine. Then using the Combination Routine, we generate offspring solutions as combination of the solutions in each subset. Finally, we improve the offspring using Improvement Routine, and consider adding it to the reference set. The newly improved solution is first compared with the solutions in RS_H . If it is better than the worse solution in RS_H , we add it to RS_H . Otherwise, we compare it with the solutions in RS_D . If it is more diverse than the least diverse solution in RS_D with respect to RS_H , then we add this solution to RS_D . We discard the solution if it cannot be added to any of these sets. We repeat these steps until no new solution is added to RS_H and a number of non-improving iterations have been observed.

A detailed pseudocode explaining our implementation of Scatter Search is given in Algorithm 5 in the Appendix III. In the following subsections, we further explain each of the elements of SS in more details.

Subset Generation Routine

One simple method for generating subsets of RS is to choose all possible pairs of solutions from RS (which we refer to as all pairs subsets), and further append more solutions to these subsets, if required. For instance, after choosing a pair of solutions, one can add the first, second, etc. other best (or most diverse) solutions to this pair to generate subsets with more elements. An example of this method can be found in Campos et al. (2001). There are two issues regarding this method. The first problem is that it generates too many subsets. For instance, if one aims to generate subsets of up to four elements, total number of generated subsets using this method will approximately be $4 \frac{|RS| \times (|RS| + 1)}{2}$. Since after each subset generation a combination and an improvement method are performed, high number of subsets increases running time of the algorithm. Moreover, this method “blindly” selects all possible subsets of this type without considering the similarity of solutions. This could result in generating similar offspring solutions for different combinations, without increasing the chance of global improvement.

Rather than blindly combining any possible solutions, in order to generate subsets of size r , first we partition the reference set into r categories, based on the similarity of its solutions. Then, to generate the subsets, we draw exactly one solution from each partition. In this way, we reduce the chance of combining the solutions that may result in offspring solutions similar to their parents existing in the reference set, hence increasing the chance of improvement. Moreover, we reduce the number of combinations, hence increasing the speed of the algorithm. For instance, if the reference set contains 12 solutions and we aim to generate subsets of up to four elements, then according to the former method, we will have to perform (almost) $2 \times 12 \times 11 = 264$ combinations and improvements, whereas according to the latter method, the number of subsets will be $\sum_{p=2}^4 \left(\frac{12}{p}\right)^p = 181$, hence as many combination and improvements. A rigorous mathematical treatment of partitioning a set of solutions into subsets is given in Appendix IV.

Solution Combination Routine

During SS we need to generate offspring solution(s) by combining a set of solutions. Inspired by the original ideas of SS for combining solutions of a combinatorial optimization problem, we use the mathematical representation of solutions as a basis for treating this problem. Our mathematical representation of the solutions is based on the MILP model proposed in BLA. Based on constraints introduced in BLA, let us define the set of all feasible solutions χ as:

$$\chi = \{X_p^k \in \{0,1\} \mid (BLA - 4) - (BLA - 6)\}. \quad (\text{SS-1})$$

Let $R \subset \chi$ denote a subset of feasible solutions that we aim to combine. Let $X^i \in \chi$, for $i = 1, \dots, |R|$ denote the i -th solution in R , and let $\bar{X}$ be an arbitrary affine combination of these solutions, i.e. $\bar{X} = \sum_{i=1}^{|R|} \lambda_i X^i$ for $\lambda_i \in \mathbb{R}$. For instance, for the case in which $\lambda_i = \frac{1}{|R|}$, $\bar{X}$ is simply the center of gravity of R . Note that $\bar{X}$, which we refer to as the pivot point, is not necessarily a feasible solution, i.e. $\bar{X} \notin \chi$, therefore it may not be an offspring solution of the combination. One way for finding a feasible offspring solution based on this pivot point, is to find the closest feasible solution in χ to the pivot point. We measure the distance of two (feasible or infeasible) points by their Euclidean distance. Therefore, we should find a feasible solution in χ that minimizes this value, i.e.:

$$\min_{X \in \chi} \|X - \bar{X}\|^2. \quad (\text{SS-2})$$

Proposition 3: (SS – 2) can be solved by integer linear programming model.

Proof: Note that $\|X - \bar{X}\|^2$ can be linearized in the following way:

$$\begin{aligned}
\|X - \bar{X}\|^2 &= \sum_{k \in \mathcal{R}} \sum_{p \in F_k} (X_p^k - \bar{X}_p^k)^2 = \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k + (\bar{X}_p^k)^2 - 2X_p^k \bar{X}_p^k \quad (\text{SS-3}) \\
&= \left(N + \sum_{k \in \mathcal{R}} \sum_{p \in F_k} (\bar{X}_p^k)^2 \right) - 2 \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{X}_p^k
\end{aligned}$$

Since $N + \sum_{k \in \mathcal{R}} \sum_{p \in F_k} (\bar{X}_p^k)^2$ is constant, (SS-2) can be reduced to:

$$\max_{X \in \mathcal{X}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{X}_p^k \quad (\text{SS-4})$$

which is a linear integer programming model ■

Although solving (SS-4) to obtain exact global optimal solution is computationally less expensive than solving BLA due to the relaxed constraints, it can be still intractable for the large instances. However, we do not need to find the optimal solution to (SS-4), and a good enough solution suffices. We can solve this problem by using IB-RFSG with the objective of maximizing (SS-4). Note that $\bar{X}_p^k$ values might be negative, but we can replace them with nonnegative values w_p^k defined as in Corollary 4.

Corollary 4: Let $\bar{x}_{min} = \min\{\bar{X}_p^k\}$ and $\bar{x}_{max} = \max\{\bar{X}_p^k\}$. Replacing $\bar{X}_p^k$ with $w_p^k = \frac{\bar{X}_p^k - \bar{x}_{min}}{\bar{x}_{max} - \bar{x}_{min}}$ in (SS-4) does not change the optimal solution.

Proof:

$$\begin{aligned}
\sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k w_p^k &= \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \frac{\bar{X}_p^k - \bar{x}_{min}}{\bar{x}_{max} - \bar{x}_{min}} = \frac{1}{\bar{x}_{max} - \bar{x}_{min}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k (\bar{X}_p^k - \bar{x}_{min}) \quad (\text{SS-5}) \\
&= \frac{1}{\bar{x}_{max} - \bar{x}_{min}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{X}_p^k - \frac{1}{\bar{x}_{max} - \bar{x}_{min}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{x}_{min} \\
&= \frac{1}{\bar{x}_{max} - \bar{x}_{min}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{X}_p^k - \frac{\bar{x}_{min}}{\bar{x}_{max} - \bar{x}_{min}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \\
&= \frac{1}{\bar{x}_{max} - \bar{x}_{min}} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{X}_p^k - \frac{N \bar{x}_{min}}{\bar{x}_{max} - \bar{x}_{min}}.
\end{aligned}$$

Since $\frac{1}{\bar{x}_{max} - \bar{x}_{min}} > 0$ and $-\frac{N \bar{x}_{min}}{\bar{x}_{max} - \bar{x}_{min}}$ is constant, maximizing $\sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k w_p^k$ also maximizes $\sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{X}_p^k$. ■

Diversity and Diverse Solution Generation

Before representing the Diverse Solution Generation Routine, we first introduce measures for assigning *distance* and *similarity* values to a pair of solutions.

Distance and similarity of two solutions

We denote the distance of two solutions X and Y by $D(X, Y)$, which we define it as half of their *Euclidean distance*, i.e.:

$$D(X, Y) = \frac{1}{2} \|X - Y\|^2. \quad (\text{SS-6})$$

Similar to (SS – 3), we can write (SS – 6) in the expanded form as:

$$\begin{aligned} D(X, Y) &= \frac{1}{2} \|X - Y\|^2 = \frac{1}{2} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} (X_p^k - Y_p^k)^2 \\ &= \frac{1}{2} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} ((X_p^k)^2 + (Y_p^k)^2 - 2X_p^k Y_p^k) \\ &= \frac{1}{2} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k + Y_p^k - 2X_p^k Y_p^k \\ &= \frac{1}{2} \left(\sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k + \sum_{k \in \mathcal{R}} \sum_{p \in F_k} Y_p^k - 2 \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_p^k \right). \end{aligned} \quad (\text{SS-7})$$

Note that since both X and Y belong to χ , $\sum_{p \in F_k} X_p^k = \sum_{p \in F_k} Y_p^k = 1$ for each $k \in \mathcal{R}$, therefore $\sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k = \sum_{k \in \mathcal{R}} \sum_{p \in F_k} Y_p^k = |\mathcal{R}| = N$. Hence, we can simply write $D(X, Y)$ as:

$$D(X, Y) = N - \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_p^k. \quad (\text{SS-8})$$

We call $\sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_p^k$ *similarity* of solutions X and Y , and denote it by $S(X, Y)$. Hence, we have:

$$D(X, Y) = N - S(X, Y). \quad (\text{SS-9})$$

Now that we have defined the distance of two solutions, we can define values for measuring the diversity of a solution with respect to a set of solutions, e.g. $R = \{X^1, \dots, X^{|R|}\}$. The first measure of diversity that we denote by $Div_1(X, R)$, is the minimum distance of X to the solutions in R , i.e.:

$$Div_1(X, R) = \min_{i=1, \dots, |R|} \{D(X, X^i)\}. \quad (\text{SS-10})$$

Now we introduce a method for generating the most diverse solution with respect to a given set of solutions. With this definition of diversity, generating the most diverse solution is equivalent with maximizing the minimum distance to members of R (d), i.e.:

$$\max_{X \in \chi} d \quad (\text{SS-11})$$

$$\text{s.t. } d \leq D(X, Y), \quad \forall Y \in R. \quad (\text{SS-12})$$

Now, let us define $s = N - d$, and let Y_i denote the i -th solution in R . After replacing $D(X, Y)$ with its equivalent value $N - S(X, Y)$, (SS – 11)- (SS – 12) can be rewritten as a mixed integer linear programming model by (SS – 13)-(SS – 14):

$$\min_{X \in \chi} s \quad (\text{SS-13})$$

$$s. t. \quad s \geq S(X, Y_i), \quad \forall i = 1, \dots, |R|. \quad (SS-14)$$

Note that $S(X, Y) = \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_{k,p}^i$ is linear. In other words, maximizing the minimum distance (d) is equivalent to minimizing the maximum similarity (s). Therefore we can define the first measure of similarity of solution X to the set R as:

$$Sim_1(X, R) = N - Div_1(X, R). \quad (SS-15)$$

Now we show how the model (SS-13)-(SS-14) results in another measure for representing the diversity of a solution with respect to a set. By moving constraint set (SS-14) with *Lagrangian multipliers* $\pi_i \geq 0$ to the objective function, (SS-13)-(SS-14) can be relaxed to:

$$\min_{X \in \mathcal{X}} s \left(1 - \sum_{i=1}^{|R|} \pi_i \right) + \sum_{i=1}^{|R|} \pi_i \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_{k,p}^i. \quad (SS-16)$$

Note that (SS-16) being a relaxation to (SS-13)-(SS-14) implies that for any choice of vector of multiplier π , the optimal solution obtained by (SS-16) provides a lower bound to (SS-13)-(SS-14), i.e. a more similar or less diverse solution. If we choose π_i such that $\sum_{i=1}^{|R|} \pi_i = 1$, the term $s(1 - \sum_{i=1}^{|R|} \pi_i)$ disappears from (SS-16), and (SS-16) will be nothing more than finding the solution with least weighted average similarity to all of the solutions in R , with weights equal to π_i , i.e.

$$s \left(1 - \sum_{i=1}^{|R|} \pi_i \right) + \sum_{i=1}^{|R|} \pi_i \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_{k,p}^i = \sum_{i=1}^{|R|} \pi_i S(X, Y_i). \quad (SS-17)$$

Also note that if we choose $\pi_i = \frac{1}{|R|}$, the average similarity with all of the solutions of R will be equivalent to the similarity with the center of gravity of R denoted by $\bar{Y}$:

$$\sum_{i=1}^{|R|} \frac{1}{|R|} \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k Y_{k,p}^i = \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \left(\frac{1}{|R|} \sum_{i=1}^{|R|} Y_{k,p}^i \right) = \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \bar{Y}_p^k \quad (SS-18)$$

Inspired by this observation, we can think of another measure for similarity of solution X to the set R , denoted by $Sim_2(X, R)$, as the number of lattice points that X shares with any of solutions in R , i.e.:

$$Sim_2(X, R) = \sum_{k \in \mathcal{R}} \sum_{p \in F_k} X_p^k \vartheta_p^k, \quad (SS-19)$$

where ϑ_p^k is 1 if $Y_p^k = 1$ for at least one solution $Y \in R$, and 0 otherwise, i.e.:

$$\vartheta_p^k = 1 - \prod_{Y \in R} (1 - Y_p^k), \quad \forall k \in \mathcal{R}, p \in F_k. \quad (\text{SS-20})$$

In other words, we are thinking of a (probably infeasible) point ϑ whose values are determined by (SS – 20), and try to find the most distant feasible solution to this point. Therefore, the second definition of diversity can be given as follows:

$$Div_2(X, R) = \frac{1}{2} \|X - \vartheta\|^2 \quad (\text{SS-21})$$

Although $Sim_2(X, R)$ is no longer equal to $N - Div_2(X, R)$, it is not hard to see that minimizing $Sim_2(X, R)$ is still equivalent to maximizing $Div_2(X, R)$. This definition can be more desirable, as it does not impose any further constraints, and can be solved to optimality in a much shorter time. It is also of importance to note that in (SS – 13)-(SS – 14), as we proceed to generate more solutions, more and more constraints are added to the model, hence the time needed to generate solutions grows by number of solutions in the set. On the other hand, however, the time required to generate diverse solutions using (SS – 21) does not change significantly by number of existing solutions.

Although this measure of diversity still yields diverse solutions, the diversity as measured by Div_2 is different from the value measured by Div_1 , and the solution is less diverse. Furthermore, it is important to note that after a specific number of solution generations based on this measure, inevitably all the ϑ_p^k values will eventually become 1, and from that point on, the generated solutions will be the same. Therefore, this method is applicable only if a few number of solutions exist in the set. Nevertheless, similar to the Solution Combination method, solving any of these models to the exact optimal solution is computationally expensive. Therefore, this time with the direction of optimization as maximizing the diversity, we can generate diverse enough solutions in a very short amount of time using IB-RFSG method.

4.4 Metaheuristic Approaches Combined with Mathematical Models

In this section we propose methods for combining the mathematical models with the metaheuristic algorithms. Our aim is to benefit from the information provided by the metaheuristic algorithms for finding the optimal solution via mathematical models in a shorter time.

Warm Starts

When solving an MILP model, we can provide initial solutions to the solver, called advanced starts or *warm starts*. With these solutions, we provide a good lower bound for the solver before starting the *Branch and Cut* ($B\&C$) process. This lower bound, or

incumbent solution, allows the solver to truncate portions of the search space, and thus may result in relatively smaller branch and cut trees. Using the metaheuristic approaches provided in the previous sections, we can provide the solver with good warm starts.

Reducing Solution Space

In the globular proteins the optimal solution tends to be very compact. In fact hydrophobic amino acids tend to be located inside the conformation, covered by polar amino acids. Therefore, if a given solution is forming a compact hydrophobic core, it has a high chance of determining how compact the optimal solution should be. Apart from the lower bound and the warm starts that the initial solutions provide, if the solutions are guaranteed to be close to optimality, we can also reduce the solution space by choosing middle amino acid (m) and the maximum distance allowable (L) according to the starting solution.

Without having the knowledge obtained by the starting solution, we usually choose m as the amino acid in the middle of the sequence. As illustrated in Figure 4-3, choosing m as the amino acid that is located at the center of the fold, and defining parameter L as the maximum distance of other amino acids to m , would greatly reduce the size of the search space. If the solutions obtained by the metaheuristic algorithms are guaranteed to be near optimal, then this restriction maintains the feasibility of the optimal solution.

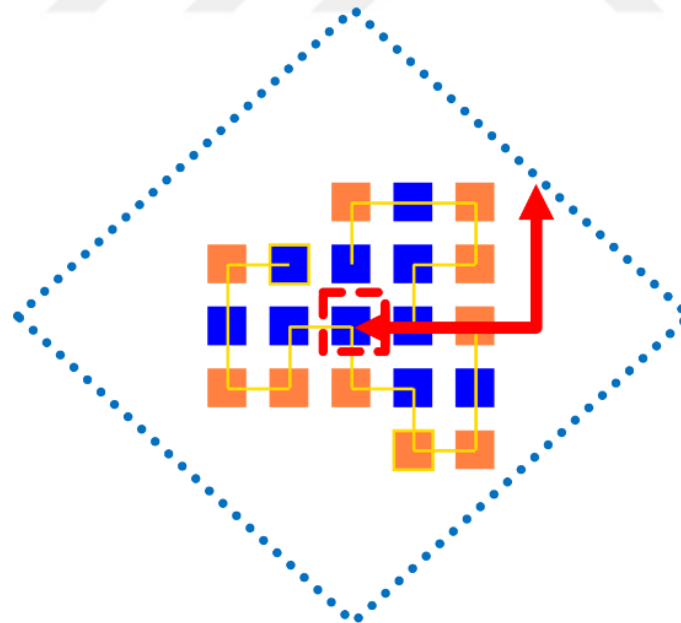


Figure 4-3 Limiting the size of problem by choosing m and L based on information provided by the initial solutions

Chapter 5

COMPUTATIONAL RESULTS FOR LINEAR CHAIN HP MODEL IN CUBIC LATTICE

In this chapter we present the computational results for the models and metaheuristic algorithms proposed for the HP PSP problem under cubic and square lattices in Chapter 3 and Chapter 4, respectively. First we introduce some classes of benchmark instances widely used in the literature. Then we present and analyze the performance of different models proposed in Chapter 3 under square and cubic lattices, with different settings regarding the valid inequalities and symmetry breaking techniques. Finally we represent the computational results of the metaheuristics algorithms that we presented in Chapter 4, and combination of these two approaches under several benchmark instances.

5.1 Benchmark Instances

Several classes of benchmark instances are available in the literature. In the following we introduce these instances, along with their optimal or best known objective values.

Standard benchmarks for square lattice

This set of benchmark instances, labeled $S1 - 0$ to $S1 - 11$ and shown in Table 5-1, is the most common instances used to compare the performance of different methods proposed for HP PSP under square lattice. To the best of our knowledge, instances $S1 - 1$ to $S1 - 7$ are first introduced by Uger & Moulton (1993) in their Genetic Algorithm proposed for this problem. Instance $S1 - 0$ has been first used by Toma & Toma (1996) and later used by Carr et al. (2002) for evaluating the performance of their LP models proposed for this problem. This instance does not appear in the conventional standard instances, hence labeled with index 0. Instances $S1 - 1$ to $S1 - 11$ fully or in part appear in numerous works such as Rego et al. (2011) and Thachuk et al. (2007).

In this table, under $|E^*|$ the best known absolute energy of each sequence is given. These values are obtained via total enumeration for short sequences or have been obtained by researchers using variety of exact or heuristic methods. Each sequence is comprised of terms $(XnYm)l$, which means a subsequence of n amino acids of type X followed by m amino acids of type Y is repeated l times. For instance sequence $S1-0$ can be expanded as $HHHPPHPHPHPHPHPHPH$.

Standard benchmarks for cubic lattice

This set of instances, also known as *Harvard* instances, is introduced by Yue et al. (1995). This set consists of 10 sequences of 48 amino acids, and mostly appears along with the

standard square lattice instances, such as in the works of Thachuk et al. (2007), Hsu et al. (2003), and Shmygelska & Hoos (2005).

Table 5-1 Standard benchmark instances for square lattice

ID	Length	$ E^* $	Sequence
S1 – 0	20	10	H3P2(HP)3(PH)3P2H
S1 – 1	20	9	(HP)2PH2PHP2HPH2P2HPH
S1 – 2	24	9	H2(P2H)7H
S1 – 3	25	8	P2HP2(H2P4)3H2
S1 – 4	36	14	P3H2P2H2P5H7P2H2P4H2P2HP2
S1 – 5	48	23	P2H(P2H2)2P5H10P6(H2P2)2HP2H5
S1 – 6	50	21	H2(PH)3PH4PH(P3H)2P4H(P3H)2PH4(PH)4H
S1 – 7	60	36	P2H3PH8P3H10PHP3H12P4H6PH2PHP
S1 – 8	64	42	H12(PH)2(P2H2)2P2HP2H2PPH2P2HP2(H2P2)2(HP)2H12
S1 – 9	85	53	H4P4H12P6(H12P3)3HP2(H2P2)2HPH
S1 – 10	100	50	P3H2P2H4P2H3(PH2)2PH4P8H6P2H6P9HPH2PH11P2H3 PH2PHP2HPH3P6H3
S1 – 11	100	48	P6HPH2P5H3PH5PH2P4H2P2H2PH5PH10PH2PH7P11H7 P2HPH3P6HPH2

Table 5-2 Standard benchmark instances for cubic lattice

ID	Length	$ E^* $	Sequence
S2 – 1	48	32	HPH2P2H4PH3P2H2P2HPH3PHPH2P2H2P3HP8H2
S2 – 2	48	34	H4PH2PH5P2HP2H2P2HP6HP2HP3HP2H2P2H3PH
S2 – 3	48	34	PHPH2PH6P2HPHP2HPH2(PH)2P3H(P2H2)2P2HPHP2HP
S2 – 4	48	33	PHPH2P2HPH3P2H2PH2P3H5P2HPH2(PH)2P4HP2(HP)2
S2 – 5	48	32	P2HP3HPH4P2H4PH2PH3P2(HP)2HP2HP6H2PH2PH
S2 – 6	48	32	H3P3H2PH(PH2)3PHP7HPHP2HP3HP2H6PH
S2 – 7	48	32	PHP4HPH3PHPH4PH2PH2P3HPHP3H3(P2H2)2P3H
S2 – 8	48	31	PH2PH3PH4P2H3P6HPH2P2H2PHP3H2(PH)2PH2P3
S2 – 9	48	34	(PH)2P4(HP)2HP2HPH6P2H3PHP2HPH2P2HPH3P4H
S2 – 10	48	33	PH2P6H2P3H3PHP2HPH2(P2H)2P2H2P2H7P2H2

Z-Structure Benchmarks

This class of artificial sequences, proposed by Aichholzer et al. (2003), possesses unique optimal conformations in the HP model under the square lattice. Sequences of this class are defined by the general term $Z_n = (HP)_u(PH)_d$, where $d = \left\lfloor \frac{n}{2} \right\rfloor$ and $u = \left\lceil \frac{n}{2} \right\rceil$. Examples of these instances are shown in Table 5-3. Aichholzer et al. (2003) prove that optimal absolute energy for sequence Z_n is $2n$. As shown in Figure 5-1, instances of this class all possess diagonal optimal solutions. These solutions are special, in the sense that if amino

acid $\frac{N}{2}$ is chosen as the middle amino acid, then the whole solution space must be available in order for the optimal solution to remain feasible. This can be problematic for the algorithms that work with the reduced square solution space.

Table 5-3 Samples of Z-Structure benchmarks

ID	Length	$ E^* $	Sequence
Z_4	8	3	HPHPHPHPH
Z_8	16	7	HPHPHPHPHPHPHPH
Z_9	18	8	HPHPHPHPHPHPHPHPH
Z_{12}	24	11	HPHPHPHPHPHPHPHPHPH
Z_{16}	32	15	HPHPHPHPHPHPHPHPHPHPHPHPHPHPH

Other Instances

Five instances, shown in Table 5-4, have been introduced by Yue & Dill (1993) in their study of finding relationships between HP sequences and their optimal solutions in the cubic lattice. We refer to these instances as *Yue-Dill instances*, and label them with $YD - 1$ to $YD - 5$.

Table 5-4 Yue-Dill instances for cubic lattice

ID	Length	$ E^* $	Sequence
$YD - 1$	27	-	HP4H4P2(HP)2H3PHP2H2P2H
$YD - 2$	27	-	HP3H4(PH)2HP2(PH)2(HP)2P2HP
$YD - 3$	27	-	H(PH2P)2PH4P3HP3H3P2H
$YD - 4$	31	-	(H2P)3H6(P2H5)2H2
$YD - 5$	36	-	(PHP)12

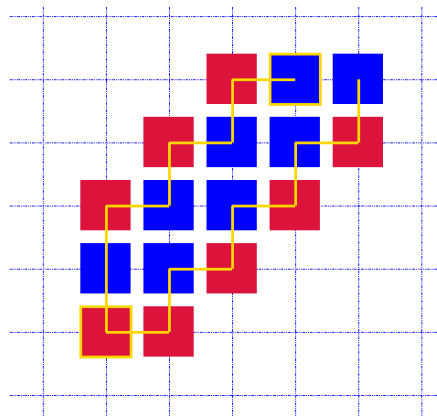


Figure 5-1 Optimal solution for an instance of Z-structure class (Z_8)

5.2 Computational Results for Mathematical Models

In this section we present the computational results regarding the mathematical models proposed for the linear chain HP model under square and cubic lattices. In addition to the abbreviations specific to the name of each model, we use the style given in Table 5-5 for referring to different models under different settings. Note that if an abbreviation appears without additional specifications, it implies that the model has been used in the very basic specifications, i.e. without additional valid inequalities.

Table 5-5 Specifications of mathematical models in linear chain HP model under cubic and square lattices

Abbreviation	Meaning
$BGM + V$	BGM with valid inequalities
$AAM + V$	AAM with valid inequalities
$BLA + V(l)$	BLA with distance based extended connectivity with parameter l
$BLA + CS$	BLA with central symmetry breaking technique with additional variables
$SLA + A$	SLA with aggregate connectivity constraints
$SLA + S$	SLA with single connectivity constraints
$SLA + A, S$	SLA with both aggregate and single connectivity constraints
$DLA_p + V_1$	DLA with pairwise objective function, and the first set of valid inequalities
$DLA_p + V_2$	DLA with pairwise objective function, and the second set of valid inequalities
$DLA_p + V_{1,2}$	DLA with pairwise objective function, and both sets of valid inequalities
$QNA + V_1$	QNA with the first set of valid inequalities
$QNA + V_2$	QNA with the second set of valid inequalities
$QNA + V_3$	QNA with the third set of valid inequalities
$QNA + V_{1,2}$	QNA with the first and second sets of valid inequalities
$QNA + V_{1,3}$	QNA with the first and third sets of valid inequalities
$QNA + V_{2,3}$	QNA with the second and third sets of valid inequalities
$QNA + V_{1,2,3}$	QNA with all sets of valid inequalities

We opt three strategies for choosing m and L . The first strategy, which we refer to as “half-length” strategy, is to select m as $\frac{N}{2}$ and $L = \min\{m, N - 1 - m\}$, which is the default setting of all of our models. The second option, which we refer to as “quarter-length”, is to select m as $\frac{N}{4}$ and L as $\min\{m, \frac{L}{2} - m\}$. The last option, which is subject to the availability of initial solution and we refer to as “learning”, is to obtain the best m and L based on the initial solution as explained in Section 4.4. Note that in some of the models we may choose slightly different values for m based on the requirement of the model. For instance in SLA we need m to be even, but in DLA and QNA we need it to be odd.

We have conducted extensive computational experiments under variety of settings for the models on several benchmark instances. In running the mathematical models, we have used solver CPLEX 12.50 integrated with our software package using the Concert technology, in a computer with Intel Xeon E5620 2.40GHz CPU and 12GB RAM. For different instances we impose different limits on the running time, varying from 300 to 2000 seconds and more.

We present three metrics for analyzing the models. In all of the tables in the following under *MILP*, we present the optimal (or obtained) integer solution along with the running time of solving the model within the specified time limit and/or optimality gap. Similar to MILP, under *Root Node*, we present the obtained integer solution along with the optimality gap and running time of solving the first node of the B&C tree. Finally, under *LP* we present the optimal objective value of the LP relaxation of the models. Each table is captioned in the format of lattice model (square or cubic), strategy of choosing m and L (half-length, quarter-length, or learning), symmetry breaking technique (central or terminal) and the ID of the benchmark instance. In all of the tables, the bold faced entries indicate the best value of each metric among other models and settings. In some of the rows, we fill the entries with dash (-), which means that the solver has not been able to solve the corresponding models in the given time limit.

We classify the results based on the benchmark instances on which the models have been tested. For each instance, we evaluate the performance of models under variety of settings. These settings vary based on the lattice type (square or cubic), symmetry breaking techniques (central or terminal), choice of m and L (half-length, quarter-length, or learning), as well as different choices of valid inequalities for each model as presented in Table 5-5.

Computational results for square lattice without initial solutions

From Table 5-6 to Table 5-11 we present the computational results of the mathematical models for instances S1-0 to S1-5, respectively. These models are solved under square lattice with m and L chosen based on the half-length strategy. Equivalently, from Table 5-12 to Table 5-17, m and L are chosen based on the quarter-length strategy.

Comparing the half-length and quarter-length strategies, we observe that the performance of the models highly depends on the choice of m and L . As we could expect, mathematical models generally show superior performance in quarter-length strategy, compared to the half-length strategy. This is due to the fact that the feasible region is pruned in the quarter-length strategy. Moreover, as the results indicate, this strategy does not exclude the optimal solution. An unexpected outcome of opting quarter-length strategy appears

in benchmark instance S1-4. As shown in Table 5-18, average fractional optimal values are significantly increased in the quarter-length strategy for all of the models. The outcome is even more severe for QNA, in which the average integer optimal value is dramatically inferior to the results in half-length strategy.

A closer inspection of S1-4 ($P3H2P2H2P5\text{HHHHHHHHHP}2H2P4H2P2HP2$) reveals that hydrophobic amino acids are mostly located towards the center of this sequence. Therefore, choosing m as the central amino acid of S1-4 results in limiting the lattice points of hydrophobic amino acids thus gives a tighter upper bound. Since QNA heavily relies on the location of hydrophobic amino acids, limiting the feasible region via quarter-length strategy does not improve running time of QNA. A similar property can be observed in S1-5 ($P2H)2HP2H2P5\text{HHHHHHHHHHHP}6H2P2H(HP2)2H4$) in which a long run of hydrophobic amino acids is cumulated in the center of the protein. As shown in Table 5-19 and illustrated in Figure 5-2, choosing m as the center of benchmark S1-4 yields the minimum values of the fractional objective function for both QNA and DLA.

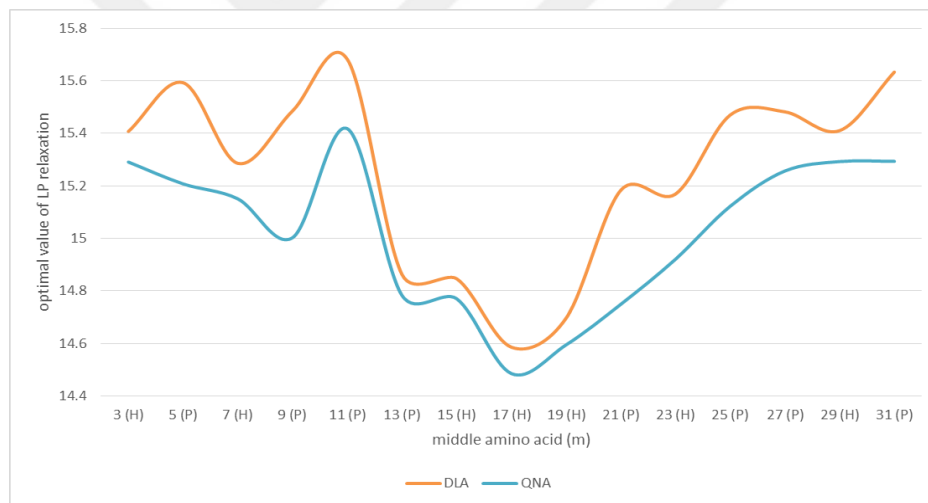


Figure 5-2 Sensitivity of LP relaxation of DLA and QNA to m in square lattice

Comparing the performance of different models, we observe that models BGM and AAM are extremely slow in finding the integer solutions, and very quick in finding the fractional upper bound. As we will later discuss this in comparing the upper bounds provided by each of these models with previously proposed models, these two models only obtain the upper bound provided by model $(BGM - 1)$ - $(BGM - 4)$ which is included in these models in the form of valid inequality constraints. We also note that while BLA does not generally provide good upper bounds, it is still usable for finding the optimal solutions of short sequences.

When used with aggregate connectivity constraints, SLA generally shows fair performance in solving instances up to S1-4. DLA turns out to be the best choice for

finding the integer optimal solutions. On the one hand, this model provides tight upper bound on the absolute energy, and on the other hand it outperforms almost all other models in terms of the time required for solving the integer models. Finally, QNA in all cases provides the tightest upper bound on the absolute energy, except in S1-4 and S1-5 when opting the quarter-length strategy for the explained reasons. In spite of the extremely high number of variables in QNA, this model still finds the best results even in instances as long as S1-5. Due to memory issues, QNA cannot be used for longer sequences.

Table 5-6 Performance of MILP models (square, half-length, central, S1-0)

Model	MILP (time limit=300s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM</i>	7 (200.00%)	300.08	0 (inf)	1.19	23.00
<i>BGM + V</i>	2 (450.00%)	300.08	2 (450.00%)	8.47	11.00
<i>AAM</i>	8 (165.83%)	300.04	1 (inf)	10.65	23.00
<i>AAM + V</i>	9 (22.22%)	300.13	0 (inf)	10.00	11.00
<i>BLA + CS</i>	10	10.33	8 (51.05%)	5.40	13.48
<i>BLA + V(2)</i>	10	13.93	6 (106.21%)	5.23	13.48
<i>BLA + V(3)</i>	10	12.11	7 (73.06%)	7.22	13.48
<i>SLA + A</i>	10	6.07	9 (33.73%)	2.46	12.43
<i>SLA + S</i>	10	5.29	8 (49.65%)	4.17	12.53
<i>SLA + A,S</i>	10	7.19	6 (101.61%)	2.11	12.41
<i>DLA_L</i>	10	0.76	10	0.76	10.14
<i>DLA_P</i>	10	1.61	10	1.61	10.09
<i>DLA_P + V₂</i>	10	2.07	10	2.07	10.08
<i>QNA</i>	10	5.97	10	5.97	10.03
<i>QNA + V₁</i>	10	5.49	10	5.49	10.01
<i>QNA + V₂</i>	10	4.93	10	4.93	10.02
<i>QNA + V₃</i>	10	5.23	10	5.23	10.00
<i>QNA + V_{1,2}</i>	10	5.34	10	5.34	10.01
<i>QNA + V_{1,3}</i>	10	5.44	10	5.44	10.00
<i>QNA + V_{2,3}</i>	10	5.15	10	5.15	10.00
<i>QNA + V_{1,2,3}</i>	10	5.52	10	5.52	10.00

Table 5-7 Performance of MILP models (square, half-length, central, S1-1)

Model	MILP (time limit=300s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM</i>	6 (283.33%)	300.02	0 (inf)	1.29	23.00
<i>BGM + V</i>	0 (inf)	300.02	0 (inf)	8.05	11.00
<i>AAM</i>	8 (145.83%)	300.15	5 (340.00%)	23.60	23.00
<i>AAM + V</i>	8 (37.50%)	300.16	0 (inf)	3.81	11.00
<i>BLA + CS</i>	9	11.01	8 (54.74%)	4.46	13.20
<i>BLA + V(2)</i>	9	19.87	8 (53.14%)	7.02	13.19
<i>BLA + V(3)</i>	9	11.62	9 (36.82%)	4.17	13.20
<i>SLA + A</i>	9	3.37	8 (35.70%)	1.40	11.03
<i>SLA + S</i>	9	4.35	5 (119.03%)	1.47	11.08
<i>SLA + A, S</i>	9	6.13	8 (35.71%)	1.34	10.97
<i>DLA_L</i>	9	2.78	8 (28.10%)	2.14	10.40
<i>DLA_P</i>	9	4.24	6 (72.31%)	1.95	10.35
<i>DLA_P + V₁</i>	9	5.99	8 (28.21%)	2.53	10.32
<i>DLA_P + V₂</i>	9	6.19	5 (105.23%)	2.46	10.27
<i>DLA_P + V_{1,2}</i>	9	3.99	6 (68.38%)	3.37	10.13
<i>QNA</i>	9	11.04	6 (64.85%)	6.77	10.01
<i>QNA + V₁</i>	9	5.48	9	5.48	9.96
<i>QNA + V₂</i>	9	9.44	9	9.44	9.98
<i>QNA + V₃</i>	9	5.82	9	5.82	9.92
<i>QNA + V_{1,2}</i>	9	8.11	9	8.11	9.94
<i>QNA + V_{1,3}</i>	9	7.07	9	7.07	9.86
<i>QNA + V_{2,3}</i>	9	6.58	9	6.58	9.91
<i>QNA + V_{1,2,3}</i>	9	6.69	9	6.69	9.85

Table 5-8 Performance of MILP models (square, half-length, central, S1-2)

Model	MILP (time limit=300s)		Root Note		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM</i>	5 (360.00%)	300.11	1 (2200.00%)	2.29	23.00
<i>BGM + V</i>	0 (inf)	300.04	0 (inf)	14.63	11.00
<i>AAM</i>	6 (246.30%)	301.63	5 (360.00%)	6.55	23.00
<i>AAM + V</i>	7 (57.14%)	300.13	3 (266.67%)	22.57	11.00
<i>BLA + CS</i>	9	130.74	8 (62.33%)	14.76	13.64
<i>BLA + V(2)</i>	9	276.06	7 (85.46%)	12.62	13.64
<i>BLA + V(3)</i>	9	191.44	7 (82.07%)	7.99	13.64
<i>SLA + A</i>	9	87.69	5 (157.47%)	6.16	12.83
<i>SLA + S</i>	9	129.25	6 (113.26%)	10.56	12.88
<i>SLA + A, S</i>	9	167.86	6 (83.33%)	4.15	12.82
<i>DLA_L</i>	9	51.43	9 (21.54%)	5.69	11.00
<i>DLA_P</i>	9	43.23	9 (21.15%)	6.75	10.95
<i>DLA_P + V₁</i>	9	27.36	9 (19.80%)	14.91	10.92
<i>QNA</i>	9	42.96	9 (19.43%)	17.04	10.84
<i>QNA + V₁</i>	9	39.87	9 (19.43%)	17.04	10.75

Second valid inequalities in DLA, and second and third valid inequalities in QNA are not applicable to S1-2.

Table 5-9 Performance of MILP models (square, half-length, central, S1-3)

Model	MILP (time limit=300s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM</i>	5 (220.00%)	300.02	0 (inf)	2.78	16.00
<i>BGM + V</i>	0 (inf)	300.10	0 (inf)	18.74	8.00
<i>AAM</i>	6 (141.67%)	301.22	0 (inf)	7.18	16.00
<i>AAM + V</i>	7 (14.29%)	300.13	0 (inf)	23.09	8.00
<i>BLA + CS</i>	8	16.86	5 (81.14%)	6.40	9.81
<i>BLA + V(2)</i>	8	39.08	1 (798.39%)	18.55	9.80
<i>BLA + V(3)</i>	8	14.07	2 (344.30%)	10.95	9.77
<i>SLA + A</i>	8	10.36	2 (300.00%)	7.96	8.00
<i>SLA + S</i>	8	7.86	8	7.86	8.00
<i>SLA + A, S</i>	8	13.93	8	13.93	8.00
<i>DLA_L</i>	8	1.78	8	1.78	8.00
<i>DLA_P</i>	8	2.50	8	2.50	8.00
<i>DLA_P + V₁</i>	8	2.46	8	2.46	8.00
<i>DLA_P + V₂</i>	8	2.36	8	2.36	8.00
<i>DLA_P + V_{1,2}</i>	8	3.67	8	3.67	8.00
<i>QNA</i>	8	3.85	8	3.85	8.00
<i>QNA + V₁</i>	8	5.16	8	5.16	8.00
<i>QNA + V₂</i>	8	3.99	8	3.99	8.00
<i>QNA + V₃</i>	8	3.88	8	3.88	8.00
<i>QNA + V_{1,2}</i>	8	6.22	8	6.22	8.00
<i>QNA + V_{1,3}</i>	8	5.05	8	5.05	8.00
<i>QNA + V_{2,3}</i>	8	4.06	8	4.06	8.00
<i>QNA + V_{1,2,3}</i>	8	6.05	8	6.05	8.00

Table 5-10 Performance of MILP models (square, half-length, central, S1-4)

Model	MILP (time limit=500s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM + V</i>	3 (433.33%)	500.09	3 (433.33%)	192.32	16.00
<i>AAM + V</i>	0 (inf)	500.30	0 (inf)	462.68	16.00
<i>BLA + CS</i>	12 (30.85%)	500.11	8 (98.46%)	90.98	16.05
<i>BLA + V(2)</i>	5 (209.37%)	500.05	7 (123.17%)	500.03	16.05
<i>BLA + V(3)</i>	5 (211.54%)	500.03	6 (157.30%)	500.11	16.05
<i>SLA + A</i>	13 (12.62%)	500.06	11 (33.92%)	151.10	15.08
<i>SLA + S</i>	14	413.71	11 (34.56%)	225.48	15.09
<i>SLA + A, S</i>	11 (34.15%)	500.06	11 (34.48%)	277.62	15.08
<i>DLA_L</i>	14	204.05	10 (44.87%)	35.38	14.69
<i>DLA_P</i>	14	297.96	6 (143.10%)	25.29	14.59
<i>DLA_P + V₁</i>	14	473.62	8 (82.32%)	27.86	14.59
<i>DLA_P + V₂</i>	12 (21.30%)	500.08	7 (108.37%)	28.92	14.59
<i>DLA_P + V_{1,2}</i>	13 (12.17%)	500.06	7 (108.37%)	39.13	14.59
<i>QNA</i>	12 (20.70%)	501.22	5 (189.67%)	102.45	14.49
<i>QNA + V₁</i>	12 (20.66%)	500.45	7 (106.95%)	101.93	14.49
<i>QNA + V₂</i>	14	484.80	3 (382.89%)	82.60	14.49
<i>QNA + V₃</i>	14	401.66	8 (80.89%)	258.79	14.49
<i>QNA + V_{1,2}</i>	14	342.14	9 (60.95%)	77.27	14.49
<i>QNA + V_{1,3}</i>	14	189.42	9 (60.95%)	101.70	14.49
<i>QNA + V_{2,3}</i>	13 (11.33%)	500.11	9 (60.95%)	133.37	14.49
<i>QNA + V_{1,2,3}</i>	14	453.88	6 (141.41%)	76.91	14.49

Table 5-11 Performance of MILP models (square, half-length, central, S1-5)

Model	MILP (time limit=2000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	5 (425.97%)	2000.10	5 (425.97%)	2000.07	26.54
<i>SLA + A</i>	13 (95.74%)	2000.09	14 (82.77%)	2000.10	25.83
<i>SLA + S</i>	15 (70.57%)	2001.06	14 (82.93%)	1053.66	25.86
<i>SLA + A, S</i>	8 (222.84%)	2000.24	8 (222.84%)	2000.09	-
<i>DLA_L</i>	21 (18.76%)	2000.20	7 (256.36%)	81.85	24.95
<i>DLA_P</i>	10 (143.75%)	2000.93	11 (121.59%)	244.86	24.38
<i>DLA_P + V₁</i>	5 (387.38%)	2000.39	5 (387.38%)	362.09	24.37
<i>QNA</i>	2 (inf)	2116.67	2 (inf)	2116.67	24.34
<i>QNA + V₃</i>	13 (95.74%)	2000.09	13 (95.74%)	2000.09	24.32

Table 5-12 Performance of MILP models (square, quarter-length, central, S1-0)

Model	MILP (time limit=300s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM + V</i>	5 (120.00%)	300.13	5 (120.00%)	300.13	11.00
<i>AAM + V</i>	10 (10.00%)	300.15	10 (10.00%)	300.15	11.00
<i>BLA + CS</i>	10	3.88	9 (30.97%)	3.41	12.27
<i>BLA + V(2)</i>	10	5.35	10 (10.00%)	4.86	12.27
<i>BLA + V(3)</i>	10	4.04	10	4.04	12.27
<i>SLA + A</i>	10	0.62	10	0.62	11.06
<i>SLA + S</i>	10	1.22	7 (56.30%)	0.54	11.11
<i>SLA + A, S</i>	10	1.17	9 (18.05%)	0.47	11.05
<i>DLA_L</i>	10	0.17	10	0.17	10.03
<i>DLA_P</i>	10	0.47	10	0.47	10.00
<i>DLA_P + V₁</i>	10	0.45	10	0.45	10.00
<i>DLA_P + V₂</i>	10	0.47	10	0.47	10.00
<i>DLA_P + V_{1,2}</i>	10	0.58	10	0.58	10.00
<i>QNA</i>	10	1.67	10	1.67	10.00
<i>QNA + V₁</i>	10	1.92	10	1.92	10.00
<i>QNA + V₂</i>	10	1.67	10	1.67	10.00
<i>QNA + V₃</i>	10	1.81	10	1.81	10.00
<i>QNA + V_{1,2}</i>	10	1.83	10	1.83	10.00
<i>QNA + V_{1,3}</i>	10	2.04	10	2.04	10.00
<i>QNA + V_{2,3}</i>	10	1.90	10	1.90	10.00
<i>QNA + V_{1,2,3}</i>	10	2.11	10	2.11	10.00

Table 5-13 Performance of MILP models (square, quarter-length, central, S1-1)

Model	MILP (time limit=300s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM + V</i>	7 (57.14%)	300.13	7 (57.14%)	300.13	11.00
<i>AAM + V</i>	8 (37.50%)	300.10	8 (37.50%)	300.10	11.00
<i>BLA + CS</i>	9	7.75	9 (32.51%)	4.34	12.92
<i>BLA + V(2)</i>	9	7.39	8 (45.76%)	6.70	12.92
<i>BLA + V(3)</i>	9	6.61	8 (54.13%)	3.68	12.92
<i>SLA + A</i>	9	2.39	8 (37.70%)	1.53	11.23
<i>SLA + S</i>	9	2.11	8 (37.42%)	1.78	11.33
<i>SLA + A, S</i>	9	2.54	8 (38.31%)	2.15	11.21
<i>DLA_L</i>	9	0.33	9	0.33	9.90
<i>DLA_P</i>	9	0.73	9	0.73	9.89
<i>DLA_P + V₁</i>	9	0.84	9	0.84	9.87
<i>DLA_P + V₂</i>	9	0.72	9	0.72	9.80
<i>DLA_P + V_{1,2}</i>	9	0.94	9	0.94	9.68
<i>QNA</i>	9	2.29	9	2.29	9.75
<i>QNA + V₁</i>	9	2.12	9	2.12	9.72
<i>QNA + V₂</i>	9	2.42	9	2.42	9.75
<i>QNA + V₃</i>	9	2.34	9	2.34	9.74
<i>QNA + V_{1,2}</i>	9	2.04	9	2.04	9.71
<i>QNA + V_{1,3}</i>	9	1.87	9	1.87	9.69
<i>QNA + V_{2,3}</i>	9	2.36	9	2.36	9.74
<i>QNA + V_{1,2,3}</i>	9	2.26	9	2.26	9.68

Table 5-14 Performance of MILP models (square, quarter-length, central, S1-2)

Model	MILP (time limit=300s)		LP		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM + V</i>	-	-	-	-	11.00
<i>AAM + V</i>	8 (37.50%)	300.16	8 (37.50%)	300.16	11.00
<i>BLA + CS</i>	9	26.13	5 (148.61%)	3.15	13.08
<i>BLA + V(2)</i>	9	59.72	7 (79.05%)	10.81	13.08
<i>BLA + V(3)</i>	9	59.47	4 (211.39%)	5.63	13.08
<i>SLA + A</i>	9	23.43	7 (78.22%)	2.95	12.51
<i>SLA + S</i>	9	14.59	6 (108.67%)	2.89	12.53
<i>SLA + A, S</i>	9	19.13	7 (79.59%)	2.87	12.51
<i>DLA_L</i>	9	20.56	7 (57.14%)	2.89	11.00
<i>DLA_P</i>	9	13.15	8 (36.93%)	4.76	10.92
<i>DLA_P + V₁</i>	9	13.35	8 (36.59%)	5.96	10.88
<i>QNA</i>	9	16.91	7 (53.11%)	14.62	10.78
<i>QNA + V₁</i>	9	16.77	9 (18.51%)	15.97	10.73

Table 5-15 Performance of MILP models (square, quarter-length, central, S1-3)

Model	MILP (time limit=300s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM + V</i>	6 (33.33%)	300.10	-	-	8.00
<i>AAM + V</i>	8	159.43	-	-	8.00
<i>BLA + CS</i>	8	4.78	6 (39.91%)	4.21	9.13
<i>BLA + V(2)</i>	8	12.73	7 (23.01%)	11.53	9.13
<i>BLA + V(3)</i>	8	16.07	5 (69.54%)	7.71	9.13
<i>SLA + A</i>	8	3.23	8	3.46	8.00
<i>SLA + S</i>	8	4.93	3 (166.67%)	1.56	8.00
<i>SLA + A, S</i>	8	5.69	8	1.42	8.00
<i>DLA_L</i>	8	9.13	5 (60.00%)	3.03	8.00
<i>DLA_P</i>	8	4.13	8	4.13	8.00
<i>DLA_P + V₁</i>	8	5.26	5 (60.00%)	3.85	8.00
<i>DLA_P + V₂</i>	8	15.04	6 (33.33%)	5.15	8.00
<i>DLA_P + V_{1,2}</i>	8	4.84	8	4.84	8.00
<i>QNA</i>	8	3.45	8	3.45	8.00
<i>QNA + V₁</i>	8	3.35	8	3.35	8.00
<i>QNA + V₂</i>	8	4.37	8	4.37	8.00
<i>QNA + V₃</i>	8	3.48	8	3.48	8.00
<i>QNA + V_{1,2}</i>	8	3.68	8	3.68	8.00
<i>QNA + V_{1,3}</i>	8	3.46	8	3.46	8.00
<i>QNA + V_{2,3}</i>	8	4.34	8	4.34	8.00
<i>QNA + V_{1,2,3}</i>	8	3.78	8	3.78	8.00

Table 5-16 Performance of MILP models (square, quarter-length, central, S1-4)

Model	MILP (time limit=500s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BGM + V</i>	-	-	-	-	16.00
<i>AAM + V</i>	5 (220.00%)	500.09	2 (700.00%)	83.74	16.00
<i>BLA + CS</i>	13 (25.61%)	500.12	6 (172.60%)	110.70	17.23
<i>BLA + V(2)</i>	12 (31.20%)	500.17	9 (84.66%)	112.13	17.23
<i>BLA + V(3)</i>	12 (32.34%)	500.06	11 (49.36%)	322.41	17.23
<i>SLA + A</i>	14	79.33	9 (73.18%)	44.15	15.56
<i>SLA + S</i>	13 (17.20%)	500.08	10 (56.47%)	50.34	15.56
<i>SLA + A, S</i>	14	232.85	10 (56.51%)	71.34	15.55
<i>DLA_L</i>	14	190.37	9 (68.76%)	22.53	15.44
<i>DLA_P</i>	14	74.26	8 (91.50%)	48.44	15.31
<i>DLA_P + V₁</i>	14	111.46	8 (90.55%)	61.48	15.28
<i>DLA_P + V₂</i>	14	69.86	6 (157.32%)	30.33	15.28
<i>DLA_P + V_{1,2}</i>	14	78.13	9 (69.18%)	72.10	15.22
<i>QNA</i>	13 (13.84%)	500.25	6 (149.40%)	375.93	14.91
<i>QNA + V₁</i>	13 (13.13%)	500.12	8 (86.17%)	429.80	14.85
<i>QNA + V₂</i>	7 (111.41%)	500.11	6 (149.22%)	500.12	14.89
<i>QNA + V₃</i>	14	454.54	14 (6.77%)	335.53	14.82
<i>QNA + V_{1,2}</i>	10 (47.36%)	500.15	7 (112.63%)	460.81	14.84
<i>QNA + V_{1,3}</i>	12 (22.50%)	500.09	9 (65.87%)	319.13	14.82
<i>QNA + V_{2,3}</i>	7 (110.39%)	500.09	6 (147.97%)	500.05	14.82
<i>QNA + V_{1,2,3}</i>	3 (392.96%)	500.15	10 (48.51%)	490.86	14.81

Table 5-17 Performance of MILP models (square, quarter-length, central, S1-5)

Model	MILP (time limit=2000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	17 (55.30%)	2000.68	8 (243.62%)	253.02	28.29
<i>SLA + A</i>	14 (93.05%)	2000.09	14 (93.05%)	1436.04	27.14
<i>DLA_L</i>	20 (25.00%)	2000.31	12 (126.82%)	2000.14	25.00
<i>DLA_P</i>	7 (286.68%)	2000.12	7 (286.68%)	2000.12	25.00
<i>DLA_P + V₁</i>	8 (212.50%)	2000.96	8 (212.50%)	186.51	25.00
<i>DLA_P + V₂</i>	4 (525.00%)	2000.15	4 (525.00%)	1283.14	25.00
<i>DLA_P + V_{1,2}</i>	4 (525.00%)	2000.18	4 (525.00%)	1472.84	25.00
<i>QNA</i>	3 (733.33%)	-	3 (733.33%)	1264.79	24.96
<i>QNA + V₃</i>	7 (256.65%)	2080.74	3 (733.33%)	1732.27	24.96

Table 5-18 Comparative analysis of half- and quarter-length strategies in instance S1-4

	QNA	DLA	SLA	BLA
	(m, L)			
Quarter-length	(9,10)	(9,10)	(8,11)	(9,10)
Half-length	(17,18)	(17,18)	(18,18)	(18,18)
Average integer optimal value				
Quarter-length	9.875	14	13.7	12.25
Half-length	13.4	13.6	12.7	8.5
Average fractional optimal value				
Quarter-length	14.84	15.30	15.56	17.29
Half-length	14.49	14.61	15.08	16.05

Table 5-19 Sensitivity of LP relaxation, time, and number of variables of DLA and QNA models with the choice of m in square lattice

	Obj. Value		Time (s)		Number of Variables	
m	DLA	QNA	DLA	QNA	DLA	QNA
3	15.41	15.29	45.80	90.01	35,519	194,255
5	15.59	15.21	36.08	84.69	32,098	167,429
7	15.29	15.15	30.36	54.58	28,550	140,070
9	15.49	15.00	20.20	34.80	25,733	123,124
11	15.68	15.42	15.30	32.34	23,736	110,928
13	14.86	14.78	18.97	16.01	22,331	94,154
15	14.85	14.77	11.65	14.84	21,087	79,551
17	14.59	14.49	13.82	11.76	20,403	69,848
19	14.70	14.59	11.15	15.55	20,602	69,302
21	15.18	14.75	9.31	12.70	20,798	75,812
23	15.17	14.92	14.55	19.55	21,649	89,154
25	15.47	15.12	14.52	32.65	23,260	104,630
27	15.48	15.26	23.34	40.05	25,621	117,829
29	15.41	15.29	28.25	65.89	28,542	137,286
31	15.63	15.29	41.28	88.37	32,126	165,182

Comparative analysis of DLA and DLA-A

Now we compare the performance of DLA and DLA with double link assignment variables for all amino acids (DLA-A). Since both models are able to solve all of the problems to optimality within the given time limit of 500 seconds (except for some instances of S1-4, in which we report optimality gap instead), here we only present the running times for integer solutions, and objective values for fractional solutions. From Table 5-20 we observe that although DLA provides tighter bounds, due to memory issues, DLA is more favorable for solving the integer models.

Table 5-20 Comparative analysis of performance of DLA and DLA-A under square lattice for five standard benchmark instances

Model	Instance	MILP (time)		LP (Obj.)		Integrality Gap	
		DLA	DLA-A	DLA	DLA-A	DLA	DLA-A
Plain	S1-0	1.61	2.15	10.09	10	0.09	0
Valid 1		1.84	2.23	10.09	10	0.09	0
Valid 2		2.07	2.68	10.08	10	0.08	0
Valid 1,2		2.12	2.65	10.08	10	0.08	0
Plain	S1-1	4.24	13.63	10.35	10.24	1.35	1.24
Valid 1		5.99	10.59	10.32	10.20	1.32	1.20
Valid 2		6.19	9.72	10.27	10.17	1.27	1.17
Valid 1,2		3.99	10.14	10.13	10.01	1.13	1.01
Plain	S1-2	43.23	62.81	10.95	10.88	1.95	1.88
Valid 1		27.36	82.88	10.92	10.85	1.92	1.85
Plain	S1-3	2.50	6.04	8.00	8.00	0.00	0.00
Valid 1		2.46	4.52	8.00	8.00	0.00	0.00
Valid 2		2.36	5.91	8.00	8.00	0.00	0.00
Valid 1,2		3.67	5.97	8.00	8.00	0.00	0.00
Plain	S1-4	297.96s	61.47%	14.59	14.59	0.5862	0.5862
Valid 1		473.62s	21.36%	14.59	14.58	0.5862	0.5862
Valid 2		21.30%	12.17%	14.59	14.59	0.5862	0.5862
Valid 1,2		12.17%	498.18s	14.59	14.58	0.5858	0.5858

Comparison of proposed MILP models with previous models

Before presenting the computational results for the cubic lattice, we first compare the performance of our models with the previously proposed models. Because of the differences between the computers on which the models have been run, in Table 5-21 we only present the optimal values of the LP relaxation of the models. To the best of our knowledge, the models have never been tested on the cubic lattice, therefore we only present the results for square lattice. In Table 5-21, under column C we present the upper bound reported by Carr et al. (2002). Columns N and Y, respectively, denote the upper bounds obtained by MILP models proposed by Namsu & Park (2010) and Yanev et al. (2011). Under other columns we present the upper bounds obtained by our models when all of the valid inequalities are added to models. In these models we select the central symmetry breaking technique for eliminating symmetries, as well as the half-length strategy for choosing m and L . In this table we present the results for the first six standard benchmark instances for square lattice, because only these sequences appear in the aforementioned papers.

The analysis reveals that DLA and QNA significantly outperform the previously proposed models in terms of the upper bound provided by LP relaxation of the models. An interesting result of this comparison appears in the case of benchmarks S1-1 and S1-2. Note that if we consider the integral part of these values, the optimal values for S1-1 and

S1-2 will be 9 and 10 respectively, which indicate that the optimal integer value can be at most equal to these values. These values, however, are 10 and 11 for Carr et al. (2002), which implies that our models provide tighter bounds on the optimal integer value. Another interesting result of these models is that QNA is able to find the complete integer solution for instances S1-0 and S1-3 even when integrality constraints are relaxed. Comparing BGM and AAM with Y and N, we observe that all of these models result in the same upper bounds. This is due to the fact that these four models report nothing more than the upper bound provided by the simple model $(BGM - 1)$ -($BGM - 4$) presented in Section 3.1 in Chapter 3.

Table 5-21 Comparison of MILP models with previous models in square lattice

Instance	C	N	Y	BGM	AAM	BLA	SLA	DLA	QNA
S1-0	10.76	NA	NA	11	11	13.48	12.41	10.08	10
S1-1	10.68	11	11	11	11	13.19	10.97	10.13	9.85
S1-2	11	11	11	11	11	13.64	12.83	10.92	10.75
S1-3	8	8	8	8	8	9.77	8	8	8
S1-4	14.90	16	16	16	16	16.05	15.08	14.59	14.49
S1-5	24.89	25	25	25	25	26.54	25.83	24.37	24.32

Computational results for cubic lattice without initial solutions

For evaluating the performance of the proposed MILP models under the cubic lattice, we only consider the most promising specifications of BLA and SLA. In order to avoid unnecessary large feasible region, we limit the range of z to the interval $[-3, 4]$, since the instances tend to form hydrophobic cores. We must mention that since the benchmark instances proposed for cubic lattice are very long, and therefore too difficult for MILP models, to capture the essence of performance of the models under cubic lattice, we run them on benchmark instances S1-0 to S1-3 and YD-1 to YD-3. In the following tables, we present the results of BLA without additional constraints, SLA with aggregate connectivity constraints, and all possible variations of DLA (note that rectangular single contact valid inequalities are specific to square lattice and are not included here.)

From Table 5-22 to Table 5-37, we present the performance of each of these models under four combinations of symmetry breaking techniques (terminal and central) and choices of middle amino acid (half- and quarter-length) for each of instances. From these results we conclude that in general quarter-length strategy helps models to find the optimal solution in a shorter time. In contrast with the square lattice, the integrality gap in cubic lattice is significantly higher. This implies that the models need more effort to close this gap even if they start with the optimal solution. Moreover, the results indicate that although the central symmetry breaking results in tighter upper bound compared to the terminal strategy, models generally require shorter time to find the optimal solution when the terminal strategy is used. We further investigate the performance of the models on

benchmark instances YD-1 (Table 5-38), YD-2 (Table 5-39), and YD-3 (Table 5-40) within the time limit of 1000 seconds. These results also support our conclusions based on former tests that in general DLA-L coupled with terminal symmetry breaking outperforms other combinations.

Table 5-22 Performance of MILP models (cubic, half-length, central, S1-0)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i> + <i>CS</i>	11 (68.81%)	1000.08	6 (191.06%)	56.64	18.82
<i>SLA</i> + <i>A</i>	10 (75.12%)	1000.08	8 (128.54%)	42.06	18.32
<i>DLA_L</i>	12	576.55	6 (157.44%)	18.88	15.46
<i>DLA_P</i>	12 (25.85%)	1000.29	10 (52.36%)	62.57	15.28
<i>DLA_P</i> + <i>V₁</i>	12 (24.01%)	1000.43	9 (68.98%)	70.34	15.25

Table 5-23 Performance of MILP models (cubic, quarter-length, central, S1-0)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i> + <i>CS</i>	12	759.82	8 (91.23%)	43.57	17.06
<i>SLA</i> + <i>A</i>	12 (31.13%)	1000.09	8 (97.12%)	46.58	15.83
<i>DLA_L</i>	12	508.98	4 (291.24%)	9.24	15.68
<i>DLA_P</i>	12	389.63	9 (71.47%)	22.68	15.44
<i>DLA_P</i> + <i>V₁</i>	12	388.89	6 (155.85%)	20.95	15.35

Table 5-24 Performance of MILP models (cubic, half-length, terminal, S1-0)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i>	11 (83.42%)	1000.45	9 (117.01%)	53.52	20.47
<i>SLA</i> + <i>A</i>	11 (81.69%)	1000.15	5 (299.73%)	53.32	19.99
<i>DLA_L</i>	11 (47.83%)	1000.26	9 (80.69%)	25.04	16.27
<i>DLA_P</i>	12	866.49	10 (59.71%)	28.28	15.98
<i>DLA_P</i> + <i>V₁</i>	12 (19.03%)	1000.82	9 (75.65%)	31.53	15.82

Table 5-25 Performance of MILP models (cubic, quarter-length, terminal, S1-0)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i>	12	357.12	9 (85.95%)	52.67	18.41
<i>SLA</i> + <i>A</i>	12 (42.88%)	1000.34	10 (72.43%)	20.06	17.39
<i>DLA_L</i>	12	404.12	10 (57.98%)	4.90	15.80
<i>DLA_P</i>	12 (17.79%)	1000.04	7 (129.37%)	16.85	16.07
<i>DLA_P</i> + <i>V₁</i>	12 (17.79%)	1000.06	10 (59.85%)	18.03	16.00

Table 5-26 Performance of MILP models (cubic, half-length, central, S1-1)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	11 (54.43%)	1000.08	7 (147.38%)	50.64	18.64
<i>SLA + A</i>	11 (41.79%)	1000.04	6 (172.28%)	17.69	16.34
<i>DLA_L</i>	11 (32.66%)	1000.18	10 (59.38%)	21.04	15.94
<i>DLA_P</i>	11 (44.09%)	1000.09	8 (99.08%)	44.73	15.93
<i>DLA_P + V₁</i>	11 (37.97%)	1000.23	10 (58.73%)	49.14	15.88

Table 5-27 Performance of MILP models (cubic, quarter-length, central, S1-1)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	11	673.28	10 (60.71%)	34.09	17.88
<i>SLA + A</i>	11 (40.38%)	1000.11	10 (59.29%)	22.99	15.96
<i>DLA_L</i>	11	667.03	9 (65.36%)	6.61	14.88
<i>DLA_P</i>	11 (32.25%)	1000.15	9 (65.20%)	11.37	14.87
<i>DLA_P + V₁</i>	11 (21.55%)	1000.09	11 (33.88%)	18.22	14.73

Table 5-28 Performance of MILP models (cubic, half-length, terminal, S1-1)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i>	11 (72.95%)	1000.43	6 (218.49%)	29.11	20.30
<i>SLA + A</i>	11 (60.30%)	1000.14	4 (341.77%)	15.97	17.67
<i>DLA_L</i>	11 (47.90%)	1000.14	8 (114.09%)	12.07	17.13
<i>DLA_P</i>	11 (57.10%)	1000.14	8 (116.52%)	25.74	17.32
<i>DLA_P + V₁</i>	11 (50.91%)	1000.15	9 (90.80%)	32.50	17.17

Table 5-29 Performance of MILP models (cubic, quarter-length, terminal, S1-1)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i>	11 (31.00%)	1000.12	9 (85.65%)	24.76	18.05
<i>SLA + A</i>	11 (42.62%)	1000.06	6 (199.38%)	12.59	17.98
<i>DLA_L</i>	11	310.99	6 (157.95%)	7.39	15.48
<i>DLA_P</i>	11	863.51	5 (217.07%)	16.04	15.85
<i>DLA_P + V₁</i>	11	752.99	4 (287.32%)	17.91	15.49

Table 5-30 Performance of MILP models (cubic, half-length, central, S1-2)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	12 (60.65%)	1000.11	5 (291.11%)	340.18	19.74
<i>SLA + A</i>	12 (56.45%)	1000.04	4 (379.78%)	114.29	19.20
<i>DLA_L</i>	13 (40.06%)	1000.29	13 (40.10%)	81.31	18.26
<i>DLA_P</i>	11 (57.76%)	1000.12	11 (59.88%)	154.88	17.63
<i>DLA_P + V₁</i>	13 (33.29%)	1000.29	10 (73.76%)	219.37	17.39

Table 5-31 Performance of MILP models (cubic, quarter-length, central, S1-2)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	12 (53.73%)	1000.06	9 (99.11%)	118.06	19.12
<i>SLA + A</i>	11 (70.73%)	1000.09	3 (526.02%)	35.60	18.78
<i>DLA_L</i>	11 (62.42%)	1000.17	9 (101.59%)	32.51	18.16
<i>DLA_P</i>	11 (57.16%)	1000.09	9 (92.14%)	74.99	17.34
<i>DLA_P + V₁</i>	13 (31.33%)	1003.59	10 (70.90%)	97.16	17.13

Table 5-32 Performance of MILP models (cubic, half-length, terminal, S1-2)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i>	9 (125.12%)	1000.25	3 (563.19%)	220.54	20.84
<i>SLA + A</i>	9 (116.63%)	1000.67	7 (178.53%)	403.54	19.50
<i>DLA_L</i>	11 (75.97%)	1000.26	10 (93.65%)	75.72	19.38
<i>DLA_P</i>	13 (40.75%)	1000.09	11 (66.62%)	69.31	18.33
<i>DLA_P + V₁</i>	13 (38.63%)	1000.23	9 (101.10%)	305.33	18.10

Table 5-33 Performance of MILP models (cubic, quarter-length, terminal, S1-2)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA</i>	13 (43.64%)	1000.08	8 (115.12%)	160.62	19.87
<i>SLA + A</i>	13 (42.14%)	1000.08	8 (134.12%)	45.41	18.91
<i>DLA_L</i>	13 (38.47%)	1000.37	11 (63.65%)	22.31	18.00
<i>DLA_P</i>	11 (56.54%)	1000.23	8 (115.50%)	40.39	17.24
<i>DLA_P + V₁</i>	13 (31.47%)	1000.11	11 (55.68%)	52.48	17.12

Table 5-34 Performance of MILP models (cubic, half-length, central, S1-3)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	6 (142.19%)	1000.68	2 (624.60%)	270.37	14.73
<i>SLA + A</i>	9 (47.67%)	1000.08	7 (89.14%)	174.39	13.29
<i>DLA_L</i>	7 (83.52%)	1000.86	5 (156.93%)	87.92	12.85
<i>DLA_P</i>	9 (42.43%)	1000.22	7 (83.12%)	191.71	12.84
<i>DLA_P + V₁</i>	9 (42.37%)	1001.82	7 (83.07%)	114.68	12.83

Table 5-35 Performance of MILP models (cubic, quarter-length, central, S1-3)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
<i>BLA + CS</i>	9 (45.31%)	1000.14	5 (167.36%)	128.54	14.08
<i>SLA + A</i>	9 (46.40%)	1000.12	7 (81.26%)	112.87	13.18
<i>DLA_L</i>	7 (91.63%)	1000.86	7 (91.63%)	149.42	13.90
<i>DLA_P</i>	8 (65.53%)	1000.14	6 (121.30%)	192.75	13.35
<i>DLA_P + V₁</i>	9 (43.93%)	1000.15	6 (117.58%)	81.20	13.12

Table 5-36 Performance of MILP models (cubic, half-length, terminal, S1-3)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
BLA	9 (59.36%)	1000.15	5 (172.29%)	275.98	14.51
SLA + A	9 (55.58%)	1000.22	5 (173.68%)	121.49	14.00
DLA _L	9 (40.59%)	1000.50	5 (153.47%)	38.24	12.69
DLA _P	8 (55.51%)	1000.11	6 (108.69%)	81.12	12.54
DLA _P + V ₁	8 (54.53%)	1000.25	6 (107.36%)	93.93	12.45

Table 5-37 Performance of MILP models (cubic, quarter-length, terminal, S1-3)

Model	MILP (time limit=1000s)		Root Node		LP (Obj.)
	Obj. (Gap)	Time (s)	Obj. (Gap)	Time (s)	
BLA	9 (44.75%)	1000.17	7 (74.65%)	229.21	13.88
SLA + A	9 (52.15%)	1000.20	2 (584.67%)	28.91	13.69
DLA _L	9 (65.09%)	1000.12	7 (112.26%)	17.64	14.86
DLA _P	9 (53.49%)	1000.17	5 (182.19%)	31.67	14.11
DLA _P + V ₁	9 (56.33%)	1000.31	4 (252.28%)	30.25	14.09

Table 5-38 Performance of MILP models (cubic, quarter-length, YD-1)

Model	Central			Terminal		
	MILP	LP (Obj.)	LP (Time)	MILP	LP (Obj.)	LP (Time)
BLA	11 (100.00%)	22.52	22.67	15 (47.35%)	22.77	24.52
DLA _L	14 (46.46%)	20.70	16.33	16 (35.60%)	21.70	24.26
DLA _P	11 (87.78%)	20.67	65.97	16 (36.39%)	21.99	46.16
DLA _P + V ₁	9 (129.33%)	20.64	148.20	4 (inf)	21.93	108.51

Table 5-39 Performance of MILP models (cubic, quarter-length, YD-2)

Model	Central			Terminal		
	MILP	LP (Obj.)	LP (Time)	MILP	LP (Obj.)	LP (Time)
BLA	14 (43.33%)	21.49	22.35	14 (54.34%)	22.38	21.89
DLA _L	15 (29.94%)	20.04	19.91	15 (40.61%)	21.10	14.94
DLA _P	15 (29.22%)	19.41	52.68	12 (76.77%)	21.23	55.60
DLA _P + V ₁	13 (48.69%)	19.34	101.43	14 (50.48%)	21.09	96.86

Table 5-40 Performance of MILP models (cubic, quarter-length, YD-3)

Model	Central			Terminal		
	MILP	LP (Obj.)	LP (Time)	MILP	LP (Obj.)	LP (Time)
BLA	7 (239.51%)	24.13	24.37	13 (77.56%)	24.45	19.66
DLA _L	16 (38.15%)	22.17	17.24	14 (59.68%)	22.39	14.26
DLA _P	13 (70.44%)	22.22	73.27	15 (52.31%)	22.85	57.03
DLA _P + V ₁	12 (83.31%)	22.04	111.01	13 (74.58%)	22.71	72.23

5.3 Computational Results for Metaheuristic Algorithms

In this section we present the computational results of the metaheuristic algorithms proposed for PSP problem. All results were obtained after tuning the algorithms. While the details of the tuning of the algorithms can be found in Appendix IV, here we briefly explain the results of these tuning experiments before presenting the results. We must note that we have only tuned the parameters of TS rigorously, and for ILS and SS we have obtained the parameters in some trial and error procedures. For tuning parameters of ILS, we have run the algorithm on benchmark instance S1-6 and S1-8 in a similar procedure for tuning parameters of TS. We consider several combinations of parameters, and for each combination we run ILS with five independent streams of random numbers.

Apart from parameters of TS, for perturbation vector (α_H , α_F , and α_G) we consider four vectors (0.2,0.6,0.2), (0.3,0.1,0.6), (0.6,0.1,0.3) and (0.3,0.4,0.3). Another controlling parameter of ILS is whether or not including stochastic coefficients in the perturbed objective function. Finally, the maximum number of good contacts that we temporarily fix during the local search is chosen from 0, 3, and 6. Our computational results suggest that setting perturbation vector to (0.6,0.1,0.3), multiplying perturbation vector with uniform random numbers, and fixing at most 3 contacts result in the best solutions.

In a similar procedure, we concluded that including six solutions (three high quality, three diverse) in the reference set and combining three and two solutions based on the partitioning of reference set results in the best solutions. We must note that all these parameters are chosen based on instances in square lattice, and may be different for cubic lattice instances. Finally, it is worth mentioning that due to the swarm nature of SS, the long running time of algorithm is inevitable. In fact, even if the algorithm finds the optimal solution in the early stages, it should keep looking until a satisfactory condition is met. On the other hand, since TS and ILS are single solution based algorithms, their running times are significantly less than SS. Therefore, we believe that these two algorithms have the potential to be further improved and become competitive with the state of the art heuristics.

In Table 5-41 the absolute energy values obtained by each of the proposed metaheuristic algorithms in solving the standard benchmark instances of square lattice model are given. Under columns TS, ILS, and SS we present the best absolute energy found by each of these algorithms as well as their running times. The results indicate that the algorithms are able to find the best known solutions for all of these instances, except S1-10 and S1-11, which are still very close to the optimal values. To evaluate the robustness of the proposed algorithms in finding the optimal solution of instances with unique optimal solutions, we further test our models on the so called Z-Structure instances, as given in Table 5-42. We have not included TS in this table as finding the optimal solution of Z-

Structure highly depends on the initial solution, and TS mostly fails to find the optimal solution in a single trial.

Similar to parameter tuning of TS for square lattice, we tune its parameters for cubic lattice based on its performance on three benchmark instances S2-1, S2-2 and S2-9. In this procedure, three choices of HtM (10, 20, 35), three choices of P_{noise} (0.1, 0.3, and 0.5) and four choices of moves as explained in parameter tuning of TS for square lattice are considered. Our computations suggest that the best choice of parameters is to set P_{noise} to 0.2 and HtM to 20, while excluding crankshaft move from allowable moves. We use the former parameters tuned for ILS and SS in the cubic lattice as well.

Table 5-41 Performance of metaheuristic algorithms on standard square benchmarks

Instance	$ E^* $	TS	ILS	SS
S1-0	10	10 (< 1 s)	10 (< 1 s)	10 (3 s)
S1-1	9	9 (< 1 s)	9 (< 1 s)	9 (3 s)
S1-2	9	9 (< 1 s)	8 (< 1 s)	9 (3 s)
S1-3	8	8 (< 1 s)	8 (< 1 s)	8 (3 s)
S1-4	14	14 (1 s)	14 (4 s)	14 (8 s)
S1-5	23	23 (4 s)	23 (17 s)	23 (31 s)
S1-6	21	20 (9 s)	21 (27 s)	21 (32 s)
S1-7	36	35 (11 s)	36 (51 s)	36 (89 s)
S1-8	42	41 (13 s)	42 (1 min)	42 (3 min)
S1-9	53	51 (3 min)	53 (11 min)	53 (14 min)
S1-10	50	47 (9 min)	48 (53 min)	49 (96 min)
S1-11	48	46 (7 min)	47 (13 min)	47 (84 min)

Table 5-42 Performance of metaheuristic algorithms on Z-structure instances

Instance	$ E^* $	ILS	SS
Z_4	3	3 (< 1s)	3 (1s)
Z_8	7	7 (< 1s)	7 (3s)
Z_{12}	11	11 (6s)	11 (12s)
Z_{16}	15	15 (9s)	15 (17s)
Z_{20}	19	19 (9s)	19 (12s)
Z_{24}	23	23 (8s)	23 (33s)
Z_{28}	27	27 (12s)	27 (36s)
Z_{32}	31	31 (14s)	31 (41s)
Z_{36}	35	35 (23s)	35 (50s)
Z_{40}	39	39 (29s)	39 (73s)

Table 5-43 Performance of metaheuristic algorithms on standard cubic benchmarks

Instance	$ E^* $	TS	ILS	SS
S2-1	32	32 (21 s)	32 (1 min)	32 (3 min)
S2-2	34	32 (26 s)	33 (1 min)	34 (4 min)
S2-3	34	32 (37 s)	32 (1 min)	33 (3 min)
S2-4	33	31 (20 s)	32 (1.5 min)	33 (4 min)
S2-5	32	31 (18 s)	32 (2 min)	32 (5 min)
S2-6	32	30 (29 s)	32 (1.5 min)	32 (4 min)
S2-7	32	30 (43 s)	31 (2 min)	31 (7 min)
S2-8	31	29 (13 s)	30 (2 min)	31 (4 min)
S2-9	34	32 (37 s)	32 (1.5 min)	33 (4 min)
S2-10	33	31 (23 s)	33 (2 min)	33 (3 min)

5.1 Computational Results for Combination of Methods

Now we present the computational results for different MILP models when we use an initial solution as a warm start. These solutions (both under square and cubic lattice) are provided by ILS algorithm, and in most of the cases are proved to be the global optimal solutions. Based on the performance of MILP models run without initial solutions, we limit the models and their specifications to the most promising ones, which we will identify in the corresponding subsections. To see the effect of constraining the feasible region with and without the information provided by the initial solution, we choose m and L based on both quarter-length and learning strategies.

Computational Results for Square Lattice with Initial Solutions

Table 5-44 shows the performance of models when using quarter-length strategy. In this table the column under S1-4 corresponds to the running time of models solving S1-4, as all of the instances are solved to optimality within the given time limit of 2000 seconds. On the other hand, since all of the models are terminated due to the same time limit, we only report the corresponding optimality gaps for other instances. In Table 5-45 we present the results obtained when letting the models to choose m and L based on the initial solution (learning strategy). In addition to S1-4, all of the models are solved to optimality for S1-5, therefore we report the running times for these two instances. For other instances, we report the optimality gaps. For benchmark instance S1-8 three of the models are solved to the optimality, hence we report the running times of these models as well as their optimality gap (which is zero).

Even though theoretically QNA provides the tightest bounds for all of the instances, it is worth mentioning that even in the most limited feasible region and a good initial solution at hand, QNA practically fails to start solving the model within the allowed time limit. This is due to the extremely high number of variables in this model. As given in

Table 5-46, number of variables in QNA is an order of magnitude more than other models, even in the most limited feasible region.

Table 5-44 Performance of MILP models with initial solution (square, quarter-length)

Model	S1-4	S1-5	S1-6	S1-7	S1-8
	Time (s)	Gap	Gap	Gap	Gap
<i>BLA + CS</i>	920.56s	17.4%	54.81%	18.45%	14.22%
<i>SLA + A</i>	283.63s	17.6%	45.50%	13.80%	inf
<i>DLA_L</i>	47.39s	8.7%	19.05%	11.11%	2.38%
<i>DLA_P</i>	141.20s	8.7%	19.05%	inf	inf
<i>DLA_P + V₁</i>	249.18s	8.7%	19.05%	11.11%	inf
<i>DLA_P + V₂</i>	233.38s	8.7%	19.05%	inf	inf
<i>DLA_P + V_{1,2}</i>	261.21s	8.7%	19.05%	11.11%	inf
<i>QNA</i>	243.17s	inf	inf	-	-
<i>QNA + V₁</i>	128.98s	inf	inf	-	-
<i>QNA + V₂</i>	245.36s	8.6%	inf	-	-
<i>QNA + V₃</i>	93.27s	8.6%	inf	-	-
<i>QNA + V_{1,2}</i>	115.24s	inf	inf	-	-
<i>QNA + V_{1,3}</i>	94.27s	8.5%	inf	-	-
<i>QNA + V_{2,3}</i>	113.29s	inf	inf	-	-
<i>QNA + V_{1,2,3}</i>	117.23s	inf	19.05%	-	-

Table 5-45 Performance of MILP models with initial solution (square, learning)

Model	S1-4	S1-5	S1-6	S1-7	S1-8
	Time (s)	Time (s)	Gap	Gap	Gap/Time
<i>BLA + CS</i>	20.79s	153.83s	39.29%	10.01%	7.51%
<i>SLA + A</i>	3.35s	1109.26s	36.04%	10.28%	8.38%
<i>DLA_L</i>	1.23s	273.10s	19.05%	7.59%	2.38%
<i>DLA_P</i>	5.69s	100.57s	19.05%	8.91%	0 (202.66 s)
<i>DLA_P + V₁</i>	4.38s	172.26s	19.05%	8.88%	0 (108.19 s)
<i>DLA_P + V₂</i>	4.63s	71.51s	19.05%	8.92%	inf
<i>DLA_P + V_{1,2}</i>	3.53s	162.65s	19.05%	8.84%	0 (76.38 s)
<i>QNA</i>	18.08s	601.54s	19.05%	-	-
<i>QNA + V₁</i>	18.33s	532.92s	19.05%	-	-
<i>QNA + V₂</i>	17.30s	578.20s	19.05%	-	-
<i>QNA + V₃</i>	18.24s	522.74s	19.05%	-	-
<i>QNA + V_{1,2}</i>	17.25s	524.23s	19.05%	-	-
<i>QNA + V_{1,3}</i>	16.58s	510.70s	19.05%	-	-
<i>QNA + V_{2,3}</i>	15.18s	509.95s	19.05%	-	-
<i>QNA + V_{1,2,3}</i>	14.31s	452.57s	19.05%	-	-

Table 5-46 Number of variables for different models under square lattice

	QNA	DLA-P	DLA-L	SLA	BLA
S1-4					
Half-length	68,356	23,612	18,296	8,072	4,819
Quarter-length	83,606	16,148	8,748	4,537	2,332
Learning	39,874	10,909	7,354	3,206	2,004
S1-5					
Half-length	788,519	89,789	47,148	19,583	11,438
Quarter-length	740,820	66,928	29,909	10,329	5,992
Learning	269,334	25,468	11,292	5,693	2,813
S1-6					
Quarter-length	737,764	73,821	32,098	12,354	7,111
Learning	501,797	38,177	15,678	4,925	2,971
S1-7					
Quarter-length	4,442,036	213,431	52,967	21,842	11,156
Learning	2,332,458	97,366	18,309	7,399	4,373
S1-8					
Quarter-length	-	298,950	70,120	23,897	13,374
Learning	2,209,369	90,605	20,493	8,622	4,815

Computational Results for Cubic Lattice with Initial Solutions

Similar to square lattice, we present the results of MILP models with initial solutions provided by the ILS algorithm in two ways. First, to observe the performance of the models without constraining the feasible region based on the conformation of the initial solution, we choose m and L with quarter-length strategy as shown in Table 5-47 within the time limit of 1000 seconds. Then we use the initial solution to reform the feasible region. As shown in Table 5-48, this new information extensively decreases the computational times and optimality gaps (within 1000 seconds). The running times required to find the optimal solution in the absence of initial solution is also given in the last two rows of Table 5-48, which shows the effectiveness of the integrated method.

Table 5-47 Performance of MILP models with initial solution (cubic, quarter-length, central)

Model	S1-0	S1-1	S1-2	S1-3
	Time(s)	Time(s)/Gap(%)	Gap(%)	Gap(%)
$BLA + CS$	306.62s	605.13s	38.85%	48.36%
DLA_L	282.21s	734.27s	37.43%	48.99%
DLA_P	282.19s	26.25%	32.98%	65.52%
$DLA_P + V_1$	457.41s	24.69%	31.24%	43.92%
$DLA_P + V_2$	283.20s	26.25%	32.98%	65.52%
$DLA_P + V_{1,2}$	458.69s	24.69%	31.24%	43.92%

Table 5-48 Performance of MILP models with initial solution (cubic, learning, central)

Model	S1-0	S1-1	S1-2	S1-3
	Time(s)	Time(s)	Time(s)	Time(s)/Gap(%)
$BLA + CS$	14.13s	25.51s	33.99s	104.47s
DLA_L	8.53s	16.49s	819.79s	34.69%
DLA_P	11.39s	27.72s	756.53s	33.97%
$DLA_P + V_1$	11.48s	28.00s	755.20s	33.97%
$DLA_P + V_2$	10.81s	28.10s	515.68s	33.95%
$DLA_P + V_{1,2}$	10.98s	28.14s	516.19s	33.95%
CPU times for unlimited time in half-length strategy without initial solution				
$BLA + CS$	1503s	13,010s	311,750s	14,084s
DLA_L	576s	2,689s	215,768s	12,462s

Chapter 6

DRAWBACKS OF CUBIC HP MODEL AND SOLUTIONS

Despite the benefits of HP model under cubic and square lattices due to its simplicity, this model is criticized for lacking the ability to reflect many features of real proteins. In this chapter we introduce the extensions of HP PSP under cubic lattice model to other modes of the problem from different perspectives to address the drawbacks of linear chain HP model under square and cubic lattice. One of the drawbacks of the conventional HP model is that it considers each amino acid as a single bead, therefore another extension of the HP model has been proposed in which each amino acid occupies two lattice points, one for the side-chain and the other for the backbone of the amino acid.

Another drawback of the conventional HP model is that cubic lattice limits the possible contacts to the amino acids of opposite parities. This issue is addressed in the so called triangular (or hexagonal) lattice in 2D and more complicated lattice structures such as FCC and KW lattices in 3D. Due to the high number of basic vectors in the FCC and KW lattices (respectively 12 and 24), in the second section of this chapter we only consider extending our mathematical models to the triangular lattice in 2D.

Lastly, conventional HP model only considers the interactions between hydrophobic amino acids, while it is believed that these interactions are not enough for determining the native conformation of proteins. One solution to this issue is to consider the interactions between all 20 types of amino acids. Empirical contact energies between the 20×20 amino acids have been proposed to tackle this problem. In the last section of this chapter we present the extension of our methods to one of such energy functions, and show how these energy functions affect the running time of the methods.

6.1 Side-Chain HP Models under Cubic/Square Lattices

In this section we address the first drawback of conventional HP model, i.e. considering amino acids as single points. An alternative to the linear chain model is the side-chain lattice model, in which each amino acid occupies two adjacent lattice points; one for the *backbone* and the other for the *side-chain* of the amino acid. In the side-chain HP model, hydrophobicity and polarities are associated with the side-chains, therefore we denote the backbones by B, and reserve hydrophobic (H) and polar (P) for the side-chains only.

Six types of interactions are possible in this model: *backbone-backbone* ($B-B$), *backbone-hydrophobic side-chain* ($B-H$), *backbone-polar side-chain* ($B-P$), *hydrophobic-polar side-chains* ($H-P$), *polar-polar side-chains* ($P-P$), and finally *hydrophobic-hydrophobic side-chains* ($H-H$). Each of these interactions may contribute differently to the energy function; however, analogous to the linear chain HP model, mostly only H-H contacts

contribute -1 to the energy function, and all other contacts are neutral. Therefore, similar to linear chain HP model, in side-chain HP model, finding the minimum energy is equivalent to finding the maximum number of H-H contacts.

Figure 6-1 shows a sample solution of this problem for the sequence S1-0. In this figure, black and green solid lines respectively represent the backbone and side-chain bonds, while the dashed blue lines represent the H-H contacts. Black, blue, and red dots respectively denote the backbones, hydrophobic and polar side-chains.

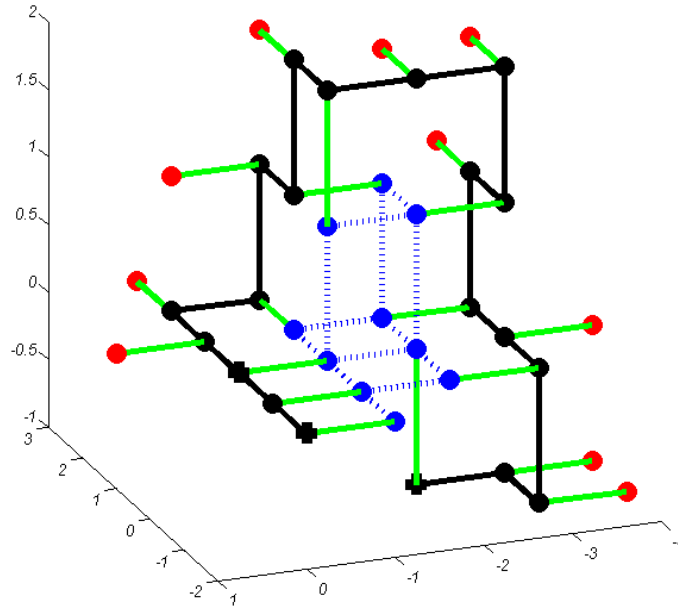


Figure 6-1 A solution of side-chain HP PSP under cubic lattice for sequence S1-0

Before going through the mathematical models, it is important to note that in the side-chain cubic lattice, side-chains occupy lattice points of opposite parity of the index of their amino acid, contrasted with backbones that occupy lattice points of the same parity as the index.

Notation

In addition to the sets and parameters defined for linear chain cubic lattice models, we define further parameters specific to the side-chain models. All of the parameters in these two models are the same, except explicitly defined otherwise. By $\mathcal{B}_k$ and $\mathcal{S}_k$, we respectively denote the backbone and the side-chain of the k -th amino acid, for $k \in \mathcal{R}$. By $F_k^{\mathcal{B}}$ and $F_k^{\mathcal{S}}$ we denote the set of feasible lattice points for $\mathcal{B}_k$ and $\mathcal{S}_k$, respectively. $F_k^{\mathcal{B}}$ is equivalent to F_k in the linear chain models, however, we need to define $F_k^{\mathcal{S}}$ according to side-chain model. Note that in the side-chain model, the feasible region for a sequence of N amino acids is equivalent to the feasible region of a sequence of $N + 2$ amino acids

in the linear chain model. Hence, if amino acid m is chosen as the middle amino acid, and $\mathcal{B}_m$ is fixed at the origin, then $\mathcal{S}_k$ can be as apart from the origin as $|k - m| + 1$.

$$F_k^{\mathcal{S}} = \{p \in \mathcal{F}^{1-\mu_k} \setminus F_m^{\mathcal{B}} : \Delta_p \leq \min\{|k - m| + 1, L\}\}. \quad (\text{SC-1})$$

Furthermore, note that similar symmetries associated with SAWs exist for SAMs. We can eliminate these symmetries using the terminal symmetry breaking technique in the exact same way as explained for SAWs. However, we must note that the lattice points that become infeasible due to elimination of symmetries (Φ_k and Υ_k) should be excluded from the set of feasible lattice points of backbones ($F_k^{\mathcal{B}}$) only. Similar to QNA, we may need tuples of basic directions in the cubic lattice, which are defined as:

$$\Psi = \{(d_1, d_2, d_3) \in \delta^3 : d_i \neq d_j, \quad \forall 1 \leq i < j \leq 3\}. \quad (\text{SC-2})$$

In the following, we present extensions of the linear chain mathematical models to the side-chain model. Due to the poor performance of BGM and AAM, in this section we only consider extending location assignment type models. Moreover, since the idea of eliminating the impossible contacts as proposed in QNA no longer exist in the side-chain model (or are very limited), we do not extend QNA to side-chain either.

Recently, an extension of BLA to the side-chain model has been proposed by Nunes et al. (2016). In their implementation, they restrict the feasible region to cubes of small side lengths (e.g. $5 \times 5 \times 5$). Moreover, they do not attempt to eliminate any of the symmetries associated with the cubic lattice. We do not present their extension here, as it is analogous to BLA, but will refer to it as *Basic Location Assignment Side-chain model (BLASC)* throughout the rest of this thesis. In our implementation, however, we use the feasible sets implied by symmetry breaking techniques. It is worth mentioning that in their model, they use the weak connectivity constraints. Analogous to constraint (BLA – 6), we can write the strengthened connectivity constraints as in (SC – 3), in which Y_p^k is a binary variable indicating whether $\mathcal{S}_k$ is assigned to lattice point $p \in F_k^{\mathcal{S}}$.

$$X_p^{k+1} + X_p^{k-1} + Y_p^k \leq \sum_{d \in \delta : p+d \in F_k^{\mathcal{B}}} X_{p+d}^k, \quad k \in \mathcal{R}, p \in F_{k+1}^{\mathcal{B}} \cup F_{k-1}^{\mathcal{B}} \cup F_k^{\mathcal{S}}. \quad (\text{SC-3})$$

Backbone Side-chain Link Assignment Model

In *Backbone Side-chain Link Assignment* model (BSLA), similar to SLA, we define variables for selecting links (edges) rather than points. However, in BSLA we define these variables for the links between backbone and side-chain of each amino acid only. In this model we refer to the direction emitted from $\mathcal{B}_k$ to $\mathcal{S}_k$, connecting the two, as the *side-chain direction*, and denote it by s . The first set of decision variables includes binary

variables concerning assignment of backbone and side-chain of amino acids at the same time.

$$BS_{p,s}^k = \begin{cases} 1, & (\mathcal{B}_k, \mathcal{S}_k) \text{ occupy } (p, p+s), \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{R}, p \in F_k^{\mathcal{B}}, s \in \delta: p+b \in F_k^{\mathcal{S}}. \quad (\text{SC-4})$$

Note that since these variables are defined for all amino acids, unlike SLA, we do not need to restrict the middle amino acid to even indices. Similar to SLA and BLA, we define decision variables for counting the number of edges with hydrophobic side-chains located at both ends.

$$W_p^d = \begin{cases} 1, & p \text{ and } p+d \text{ are occupied by H side-chains} \\ 0, & \text{otherwise} \end{cases}, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1. \quad (\text{SC-5})$$

Model (BSLA-L)

$$\max \sum_{p \in \mathcal{F}^0} \sum_{d \in \delta: p+d \in \mathcal{F}^1} W_p^d \quad (\text{SC-6})$$

Subject to:

$$\sum_{p \in F_k^{\mathcal{B}}} \sum_{s \in \delta: p+s \in F_k^{\mathcal{S}}} BS_{p,s}^k = 1, \quad \forall k \in \mathcal{R}, \quad (\text{SC-7})$$

$$\sum_{\substack{k \in \mathcal{R}^\mu: \\ p \in F_k^{\mathcal{B}}}} \sum_{\substack{s \in \delta: \\ p+s \in F_k^{\mathcal{S}}}} BS_{p,s}^k + \sum_{\substack{k \in \mathcal{R}^{1-\mu}: \\ p \in F_k^{\mathcal{S}}}} \sum_{\substack{s \in \delta: \\ p-s \in F_k^{\mathcal{B}}}} BS_{p-s,s}^k \leq 1, \quad \forall \mu \in \{0,1\}, p \in \mathcal{F}^\mu, \quad (\text{SC-8})$$

$$BS_{p,s_1}^{k-1} + BS_{p,s_1}^{k+1} \leq \sum_{b \in \delta - \{s_1\}: p+b \in F_k^{\mathcal{B}}} \sum_{s_2 \in \delta - \{-b\}: p+b+s_2 \in F_k^{\mathcal{S}}} BS_{p+b,s_2}^k, \quad (\text{SC-9})$$

$$\forall k \in \mathcal{R}, p \in F_{k \pm 1}^{\mathcal{B}}, s_1 \in \delta: p+s_1 \in F_{k \pm 1}^{\mathcal{S}},$$

$$W_p^d \leq \sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_k^{\mathcal{S}}}} \sum_{\substack{s \in \delta - \{-d\}: \\ p-s \in F_k^{\mathcal{B}}}} BS_{p-s,s}^k, \quad \forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{SC-10})$$

$$W_p^d \leq \sum_{\substack{k \in \mathcal{H}^0: \\ p+d \in F_k^{\mathcal{S}}}} \sum_{\substack{s \in \delta - \{d\}: \\ p+d-s \in F_k^{\mathcal{B}}}} BS_{p-s,s}^k, \quad \forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1, \quad (\text{SC-11})$$

$$BS_{p,s}^k \in \{0,1\}, \quad \forall k \in \mathcal{R}, p \in F_k^{\mathcal{B}}, s \in \delta: p+b \in F_k^{\mathcal{S}}, \quad (\text{SC-12})$$

$$W_p^d \in \{0,1\}, \quad \forall p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1. \quad (\text{SC-13})$$

Constraint sets (SC – 7) to (SC – 9) are respectively equivalent to the assignment, non-overlapping, and connectivity constraints in BLASC. Note that the connectivity between side-chains and their respective backbone is guaranteed by the definition in (SC – 4). Hence, we only need to ensure that the backbones are connected. Constraint set (SC – 9), which we refer to as the *single connectivity* constraint, maintains connectivity of $\mathcal{B}_k$

with its successor ($\mathcal{B}_{k+1}$), and predecessor ($\mathcal{B}_{k-1}$) at the same time. This constraint set implies that if any of $\mathcal{B}_{k-1}$ or $\mathcal{B}_{k+1}$ is assigned to lattice point p , while their side-chain is assigned to $p + s_1$, then $\mathcal{B}_k$ must be assigned to one of the lattice points adjacent to p , except $p + s_1$ as it is already occupied by either of $\mathcal{S}_{k+1}$ or $\mathcal{S}_{k-1}$. Note that in this constraint set, if k is one of the terminal amino acids, or $(p, p + s_1)$ is not a feasible edge for $(\mathcal{B}_{k-1}, \mathcal{S}_{k-1})$ or $(\mathcal{B}_{k+1}, \mathcal{S}_{k+1})$, only one of these terms will appear in the left hand side of the inequality.

Unlike BLA and SLA, note that in BSLA we no longer count the sequential bonds while maximizing summation of W_p^d , as the contact only occurs between side-chains of amino acids. This implies that the location-wise objective function yields exactly the total number of contacts between hydrophobic side-chains. Similar to DLA-P, we can measure the absolute energy equivalently using a pairwise objective function, i.e. by counting the number of contacts between pairs of hydrophobic side-chains of different parities along given directions. We define binary decision variables Z as:

$$Z_{p,d}^{k,l} = \begin{cases} 1, & (\mathcal{S}_k, \mathcal{S}_l) \text{ occupy } (p, p + d) \\ 0, & \text{otherwise} \end{cases}, \quad (SC-14)$$

$$k \in \mathcal{H}^1, p \in F_k^S, l \in \mathcal{H}^0, d \in \delta: p + d \in F_l^S.$$

Model (BSLA-P)

$$\max \sum_{k \in \mathcal{H}^1} \sum_{p \in F_k^S} \sum_{l \in \mathcal{H}^0} \sum_{d \in \delta: p+d \in F_l^S} Z_{p,d}^{k,l} \quad (SC-15)$$

Subject to:

$$\sum_{l \in \mathcal{H}^0: p+d \in F_l^S} Z_{p,d}^{k,l} \leq \sum_{s \in \delta - \{-d\}: p-s \in F_k^B} (SC-7) - (SC-9), (SC-12), \quad BS_{p-s,s}^k, \quad \forall k \in \mathcal{H}^1, p \in F_k^S, d \in \delta, \quad (SC-16)$$

$$\sum_{k \in \mathcal{H}^1: p \in F_k^S} Z_{p,d}^{k,l} \leq \sum_{s \in \delta - \{d\}: p+d-s \in F_l^B} BS_{p+d-s,s}^l, \quad \forall l \in \mathcal{H}^0, p \in F_l^S, d \in \delta, \quad (SC-17)$$

$$Z_{p,d}^{k,l} \in \{0,1\}, \quad \forall k \in \mathcal{H}^1, p \in F_k^S, l \in \mathcal{H}^0, d \in \delta: p + d \in F_l^S. \quad (SC-18)$$

Aggregate connectivity constraints

Another way of maintaining the connectivity of the backbones is to write the strengthened connectivity constraints (SC-3) in terms of link assignment variables (i.e. BS .) Note that X_p^k (assignment of $\mathcal{B}_k$) and Y_p^k (assignment of $\mathcal{S}_k$) can be written in terms of BS variables as:

$$X_p^k = \sum_{s \in \delta: p+s \in F_k^S} BS_{p,s}^k, \quad Y_p^k = \sum_{s \in \delta: p-s \in F_k^B} BS_{p-s,s}^k. \quad (SC-19)$$

Therefore, constraint (SC – 3) reads as:

$$\sum_{\substack{s \in \delta: \\ p+s \in F_k^S}} BS_{p,s}^{k-1} + \sum_{\substack{s \in \delta: \\ p+s \in F_k^S}} BS_{p,s}^{k+1} + \sum_{\substack{s \in \delta: \\ p-s \in F_k^B}} BS_{p-s,s}^k \leq \sum_{\substack{d \in \delta: \\ p+d \in F_k^B}} \sum_{\substack{s \in \delta: \\ p+d+s \in F_k^S}} BS_{p+d,s}^k. \quad (\text{SC-20})$$

Note that the right hand side of (SC – 20) can be broken into two terms, based on whether $s = -d$ or $s \neq -d$:

$$\sum_{\substack{d \in \delta: \\ p+d \in F_k^B}} \sum_{\substack{s \in \delta: \\ p+d+s \in F_k^S}} BS_{p+d,s}^k = \sum_{\substack{d \in \delta: \\ p+d \in F_k^B}} \sum_{\substack{s \in \delta - \{-d\}: \\ p+d+s \in F_k^S}} BS_{p+d,s}^k + \sum_{\substack{s \in \delta: \\ p \in F_k^S}} BS_{p-s,s}^k. \quad (\text{SC-21})$$

Therefore, (SC – 20) can be reduced to (SC – 22) which we refer to as the *aggregate connectivity* constraint.

$$\sum_{\substack{s \in \delta: \\ p+s \in F_k^S}} BS_{p,s}^{k-1} + \sum_{\substack{s \in \delta: \\ p+s \in F_k^S}} BS_{p,s}^{k+1} \leq \sum_{\substack{d \in \delta: \\ p+d \in F_k^B}} \sum_{\substack{s \in \delta - \{-d\}: \\ p+d+s \in F_k^S}} BS_{p+d,s}^k. \quad (\text{SC-22})$$

Square single contact valid inequality in BSLA

Note that side-chains of two consecutive amino acid can form a contact in cubic lattice, only if these side-chains along with the backbones are located at four corners of a square. As illustrated in Figure 6-2, one implication of this observation is that hydrophobic $\mathcal{S}_k$ forms a contact with either of its successor ($\mathcal{S}_{k+1}$) or predecessor ($\mathcal{S}_{k-1}$) in the sequence along a given direction, only if the side-chain directions are perpendicular to the contact direction. (SC – 20) and (SC – 21) show this property as valid inequalities. Note that in these constraints if either of the terms on the left hand side of the inequality are not defined we replace them with zero.

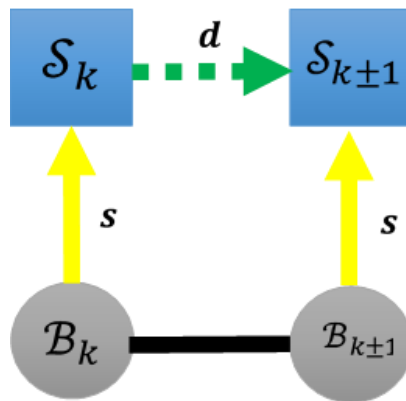


Figure 6-2 Square single contact valid inequalities in BSLA

$$Z_{p,d}^{k,k+1} + Z_{p,d}^{k,k-1} \leq \sum_{\substack{s \in \delta - \{d, -d\}: \\ p-s \in F_k^{\mathcal{B}}}} BS_{p-s,s}^k, \quad (\text{SC-23})$$

$$\begin{aligned} \forall k \in \mathcal{H}^1, p \in F_k^{\mathcal{S}}, d \in \delta: k \pm 1 \in \mathcal{H}^0, p+d \in F_{k \pm 1}^{\mathcal{S}}, \\ Z_{p-d,d}^{k+1,k} + Z_{p-d,d}^{k-1,k} \leq \sum_{\substack{s \in \delta - \{d, -d\}: \\ p-s \in F_k^{\mathcal{B}}}} BS_{p-s,s}^k, \quad (\text{SC-24}) \\ \forall k \in \mathcal{H}^0, p \in F_k^{\mathcal{S}}, d \in \delta: k \pm 1 \in \mathcal{H}^1, p-d \in F_{k \pm 1}^{\mathcal{S}}. \end{aligned}$$

Single-Triple Link Assignment Model

In *Single-Triple Link Assignment (STLA)* model, which is the extension of DLA to the side-chain model, we take advantage of parity property of cubic lattice for assigning backbones and side-chains to lattice points. Here, we only assign backbones of parity 0 to lattice points of the same parity, and side-chains of parity 1 to the points of opposite parity. Note that since two backbones and a side-chain are connected to each backbone, we assign a triple of distinct basic directions to determine the adjacent points in which each of these three are assigned to (variables XB). Moreover, since side-chains of parity 1 are actually equivalent to backbones of parity 0, we treat them in a similar manner, i.e. we assign side-chains of parity 1 to points of parity 0, along with a direction that indicates to which adjacent node its backbone is assigned (variables XS).

In this model we use the notation of Ψ as the set of all triples of distinct basic directions. Note that in the square lattice, cardinality of Ψ is 24, while this number is as large as 120 triples in the cubic lattice.

The link assignment variables are divided into two sets. The first, denoted by XB , accounts for assignment of $\mathcal{B}_k$ of parity 0 to one of its feasible lattice points, while successor and predecessor backbones, as well as the side-chain connected to $\mathcal{B}_k$ are assigned to its adjacent points, along a *triple* of basic directions.

$$XB_{p,\psi}^k = \begin{cases} 1, & (\mathcal{B}_k, \mathcal{B}_{k+1}, \mathcal{B}_{k-1}, \mathcal{S}_k) \text{ occupy } (p, p + \psi_1, p + \psi_2, p + \psi_3), \\ 0, & \text{otherwise} \end{cases}, \quad (\text{SC-25})$$

$$k \in \mathcal{R}^0, p \in F_k^{\mathcal{B}}, \psi \in \Psi: p + \psi_1 \in F_{k+1}^{\mathcal{B}}, p + \psi_2 \in F_{k-1}^{\mathcal{B}}, p + \psi_3 \in F_k^{\mathcal{S}}.$$

The second set of variables, denoted by XS , contains binary variables determining the location of side-chains of parity 1 to one of their feasible lattice points, and assignment of their associated backbones to one of the neighboring lattice points.

$$XS_{p,s}^k = \begin{cases} 1, & (\mathcal{S}_k, \mathcal{B}_k) \text{ occupy } (p, p + s), \\ 0, & \text{otherwise} \end{cases}, \quad k \in \mathcal{R}^1, p \in F_k^{\mathcal{S}}, s \in \delta: p + s \in F_k^{\mathcal{B}}. \quad (\text{SC-26})$$

Finally, we define the binary variables determining whether a contact exists between two hydrophobic side-chains on a given edge.

$$Z_{p,d}^{k,l} = \begin{cases} 1, & (\mathcal{S}_k, \mathcal{S}_l) \text{ occupy } (p, p+d) \\ 0, & \text{otherwise} \end{cases}, \quad (SC-27)$$

$$k \in \mathcal{H}^1, p \in F_k^\mathcal{S}, l \in \mathcal{H}^0, d \in \delta: p+d \in F_l^\mathcal{S}.$$

Model (STLA)

$$\max \sum_{k \in \mathcal{H}^1} \sum_{p \in F_k^\mathcal{S}} \sum_{l \in \mathcal{H}^0} \sum_{d \in \delta: p+d \in F_l^\mathcal{S}} Z_{p,d}^{k,l} \quad (SC-28)$$

Subject to:

$$\sum_{p \in F_k^\mathcal{B}} \sum_{\psi \in \Psi: \substack{p+\psi_1 \in F_{k+1}^\mathcal{B} \\ p+\psi_2 \in F_{k-1}^\mathcal{B} \\ p+\psi_3 \in F_k^\mathcal{S}}} XB_{p,\psi}^k = 1, \quad \forall k \in \mathcal{R}^0 \quad (SC-29)$$

$$\sum_{p \in F_k^\mathcal{S}} \sum_{b \in \delta: p+b \in F_k^\mathcal{B}} XS_{p,b}^k = 1, \quad \forall k \in \mathcal{R}^1 \quad (SC-30)$$

$$\sum_{\substack{k \in \mathcal{R}^0: \\ p \in F_k^\mathcal{B}}} \sum_{\psi \in \Psi: \substack{p+\psi_1 \in F_{k+1}^\mathcal{B} \\ p+\psi_2 \in F_{k-1}^\mathcal{B} \\ p+\psi_3 \in F_k^\mathcal{S}}} XB_{p,\psi}^k + \sum_{\substack{k \in \mathcal{R}^1: \\ p \in F_k^\mathcal{S}}} \sum_{b \in \delta: p+b \in F_k^\mathcal{B}} XS_{p,b}^k \leq 1, \quad \forall p \in \mathcal{F}^0 \quad (SC-31)$$

$$\sum_{\psi \in \Psi: \substack{p-\psi_1 \in F_k^\mathcal{B} \\ p-\psi_1+\psi_2 \in F_{k-1}^\mathcal{B} \\ p-\psi_1+\psi_3 \in F_k^\mathcal{S}}} XB_{p-\psi_1,\psi}^k = \sum_{\psi \in \Psi: \substack{p-\psi_2 \in F_{k+1}^\mathcal{B} \\ p-\psi_2+\psi_1 \in F_{k+2}^\mathcal{B} \\ p-\psi_2+\psi_3 \in F_{k+1}^\mathcal{S}}} XB_{p-\psi_2,\psi}^{k+1}, \quad \forall k \in \mathcal{R}^1, p \in F_k^\mathcal{B} \quad (SC-32)$$

$$\sum_{\psi \in \Psi: \substack{p-\psi_1 \in F_k^\mathcal{B} \\ p-\psi_1+\psi_2 \in F_{k-1}^\mathcal{B} \\ p-\psi_1+\psi_3 \in F_k^\mathcal{S}}} XB_{p-\psi_1,\psi}^k = \sum_{\substack{b \in \delta: \\ p-d \in F_k^\mathcal{S}}} XS_{p-b,b}^k, \quad \forall k \in \mathcal{R}^1, p \in F_k^\mathcal{B} \quad (SC-33)$$

$$\sum_{\substack{l \in \mathcal{H}^0: \\ p+d \in F_l^\mathcal{S}}} Z_{p,d}^{k,l} \leq \sum_{\substack{b \in \delta - \{d\}: \\ p+b \in F_k^\mathcal{B}}} XS_{p,b}^k, \quad \forall k \in \mathcal{H}^1, p \in F_k^\mathcal{S}, d \in \delta \quad (SC-34)$$

$$\sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_k^\mathcal{S}}} Z_{p,d}^{k,l} \leq \sum_{\substack{\psi \in \Psi: \\ \psi_3 \neq d \\ p+d-\psi_3 \in F_l^\mathcal{B} \\ p+d-\psi_3+\psi_1 \in F_{l+1}^\mathcal{B} \\ p+d-\psi_3+\psi_2 \in F_{l-1}^\mathcal{B}}} XB_{p+d-\psi_3,\psi}^l, \quad \forall l \in \mathcal{H}^0, d \in \delta, p \in \mathcal{F}^0: p+d \in F_l^\mathcal{S} \quad (SC-35)$$

$$XB_{p,\psi}^k \in \{0,1\}, \quad k \in \mathcal{R}^0, p \in F_k^\mathcal{B}, \psi \in \Psi: p+\psi_1 \in F_{k+1}^\mathcal{B} \wedge p+\psi_2 \in F_{k-1}^\mathcal{B} \wedge p+\psi_3 \in F_k^\mathcal{S} \quad (SC-36)$$

$$XS_{p,s}^k \in \{0,1\}, \quad k \in \mathcal{R}^1, p \in F_k^\delta, s \in \delta: p + s \in F_k^\mathcal{B} \quad (\text{SC-37})$$

$$Z_{p,d}^{k,l} \in \{0,1\}, \quad k \in \mathcal{H}^1, p \in F_k^\delta, l \in \mathcal{H}^0, d \in \delta: p + d \in F_l^\delta. \quad (\text{SC-38})$$

Constraint set (SC – 29) requires that each backbone of parity 0 is assigned to one of its feasible points. Furthermore, it assigns distinct basic directions to forward and backward backbones as well as to the side-chain connected to this backbone. Constraint set (SC – 30) guarantees that each side-chain of parity 1 is assigned to one of its feasible points of parity 0. Constraint set (SC – 31) ensures that at most one backbone of parity 0 or side-chain of parity 1 is assigned to each point of parity 0. Similarly, constraint set (SC – 32) guarantees that at most one backbone of parity 1 or side-chain of parity 0 is assigned to each lattice point of parity 1.

Constraint sets (SC – 33) and (SC – 34) prevent inconsistencies in locations assigned to backbones of parity 1. These constraints require that the lattice point, assigned to $\mathcal{B}_k$ of parity 1 by its successor ($\mathcal{B}_{k+1}$) through backward links, is the same determined by its predecessor ($\mathcal{B}_{k-1}$) through a forward link to $\mathcal{B}_k$. Moreover, these constraints guarantee the connectivity of backbones. Constraint sets (SC – 34) and (SC – 35) define the relation between contacts and locations, excluding side-chain links, in a similar manner as in DLA-P.

Square single contact valid inequality in BSLA

Analogous to the square single contact valid inequalities in BSLA, consecutive hydrophobic side-chain can form contacts only if they and their associated backbones occupy four corners of a square. As shown in Figure 6-2, one implication of this observation is that the direction of backbone bonds of these amino acids must be the same as the direction of contact, as presented in (SC – 39) to (SC – 40). Note that unlike BSLA, we cannot integrate (SC – 39) and (SC – 40) into a single constraint.

$$\rho_{(k+1)} Z_{p,d}^{k,k+1} + \rho_{(k-1)} Z_{p,d}^{k,k-1} \leq \sum_{b \in \delta - \{d, -d\}: p+b \in F_k^\mathcal{B}} XS_{p,b}^k, \quad (\text{SC-39})$$

$$Z_{p-d,d}^{l+1,l} \leq \sum_{\substack{\psi \in \Psi: \\ \psi_1 = -d \\ \psi_3 \notin \{d, -d\} \\ p-\psi_3 \in F_l^\mathcal{B} \\ p-\psi_3+\psi_1 \in F_{l+1}^\mathcal{B} \\ p-\psi_3+\psi_2 \in F_{l-1}^\mathcal{B}}} XB_{p-\psi_3,\psi}^l, \quad (\text{SC-40})$$

$$\forall l \in \mathcal{H}^0, p \in F_l^\delta, d \in \delta: l+1 \in \mathcal{H}^1, p-d \in F_{l+1}^\delta$$

$$Z_{p-d,d}^{l-1,l} \leq \sum_{\substack{\psi \in \Psi: \\ \psi_2 = -d \\ \psi_3 \notin \{d, -d\} \\ p-\psi_3 \in F_l^B \\ p-\psi_3+\psi_1 \in F_{l+1}^B \\ p-\psi_3+\psi_2 \in F_{l-1}^B}} XB_{p-\psi_3,\psi}^l, \quad (\text{SC-41})$$

$$\forall l \in \mathcal{H}^0, p \in F_l^\delta, d \in \delta: l-1 \in \mathcal{H}^1, p-d \in F_{l-1}^\delta.$$

Computational Result

Now we present the computational results of the mathematical models proposed for side-chain HP model plus the existing model (BLASC) adjusted to our symmetry breaking constraints. Our aim is to show how considering the side-chain affects the computational results of linear-chain models. In the following tables, we use similar abbreviations as given in the computational results for linear chain models in Section 5.2 in Chapter 5. For instance, $BSLA_p + A, S, V$ reads as BSLA with pairwise objective function, while single and aggregate connectivity constraints, as well as the square single contact valid inequalities are added to the model. First we present the computational results of the models under square lattice. As shown from Table 6-1 to Table 6-8, for benchmark instances S1-0 to S1-3, we solve overall thirteen models under four combinations of symmetry breaking strategy (central, terminal) and choice of m (half- and quarter-length) within time limit of 500 seconds.

In each table, the columns under half-length and quarter-length represent the results obtained by opting these strategies for choosing m . Under each of these columns, we give the integer and fractional results obtained by each model. Note that under column MILP we present the objective function in addition to CPU time, or optimality gap, based on whether the model has been terminated due to time limit (optimality gap), or has found the optimal solution before the time limit (CPU time). The results suggest that STLA and BSLA outperform BLASC, which in turn implies that under equal conditions, these two models must result in better solutions compared to results obtained by Nunes et al. (2016). Quarter-length strategy generally results in better solutions. Moreover, compared to central symmetry breaking, we note that in most of the cases, terminal symmetry breaking in spite of looser upper bound helps the models to find the optimal solution in a shorter time.

Based on these observations, we conclude that STLA, STLA+V, BSLA-L+A, BSLA-P+A, and BSLA-P+A,V show the best performances among other models. Therefore, in order to evaluate the models under cubic lattice, we limit the models to these five cases. Additional to these five models, we also keep running BLASC-L and BLASC-P, to observe the differences between our models and Nunes et al. (2016). For the same

benchmark instances and the time limit of 500 seconds, we run these models under quarter-length strategy and both choices of central and terminal symmetry breaking strategies.

Results obtained by these runs support our former claim that STLA and BSLA are superior to BLASC. Here we do not compare our results with Nunes et al. (2016) for three reasons. First, in their models they reduce the feasible region to a $5 \times 5 \times 5$ cube, which significantly reduces the size of the problem, hence have shorter running times. Second, they use an extremely fast workstation which is not comparable with the machine on which we have performed the runs. Finally, in their runs, they have set the time limit to 604,800 seconds (seven days), which is not affordable for us due to the lack of time.

Table 6-1 Performance of MILP models for side-chain model (square,central,S1-0)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC</i> – <i>L</i>	11 (263.25)	17.03	11 (175.94s)	15.82
<i>BLASC</i> – <i>P</i>	9 (69.9%)	17.03	11 (183.33s)	15.82
<i>STLA</i>	11 (48.11s)	14.14	11 (53.76s)	13.44
<i>STLA</i> + <i>V</i>	11 (26.93s)	14.20	11 (27.44s)	13.57
<i>BSLA_L</i> + <i>A</i>	11 (54.15s)	14.30	11 (36.79s)	13.54
<i>BSLA_L</i> + <i>S</i>	11 (238.56s)	14.23	11 (83.21s)	13.51
<i>BSLA_L</i> + <i>A,S</i>	11 (117.81s)	14.14	11 (123.05s)	13.41
<i>BSLA_P</i> + <i>A</i>	11 (126.42s)	14.30	11 (121.85s)	13.54
<i>BSLA_P</i> + <i>S</i>	11 (217.79s)	14.23	11 (455.63s)	13.51
<i>BSLA_P</i> + <i>A,S</i>	11 (159.67s)	14.14	11 (326.34s)	13.41
<i>BSLA_P</i> + <i>A,V</i>	11 (124.94s)	14.23	11 (42.45s)	13.45
<i>BSLA_P</i> + <i>S,V</i>	11 (127.02s)	14.12	11 (226.42s)	13.42
<i>BSLA_P</i> + <i>A,S,V</i>	11 (273.97s)	14.05	11 (297.92s)	13.34

Table 6-2 Performance of MILP models for side-chain model (square,terminal,S1-0)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC</i> – <i>L</i>	8 (105.6%)	17.64	11 (368.90s)	16.46
<i>BLASC</i> – <i>P</i>	7 (135.1%)	17.64	11 (42.2%)	16.46
<i>STLA</i>	10 (43.4%)	14.47	11 (74.29s)	13.94
<i>STLA</i> + <i>V</i>	9 (48.7%)	14.55	11 (41.71s)	13.95
<i>BSLA_L</i> + <i>A</i>	11 (310.63s)	14.66	11 (41.64s)	13.98
<i>BSLA_L</i> + <i>S</i>	10 (42.6%)	14.63	11 (90.64s)	13.96
<i>BSLA_L</i> + <i>A,S</i>	9 (57.6%)	14.52	11 (87.05s)	13.85
<i>BSLA_P</i> + <i>A</i>	10 (31.5%)	14.66	11 (81.14s)	13.98
<i>BSLA_P</i> + <i>S</i>	9 (58.9%)	14.63	11 (247.95s)	13.96
<i>BSLA_P</i> + <i>A,S</i>	9 (57.8%)	14.52	11 (200.73s)	13.85
<i>BSLA_P</i> + <i>A,V</i>	11 (222.05s)	14.50	11 (93.74s)	13.98
<i>BSLA_P</i> + <i>S,V</i>	10 (41.1%)	14.44	11 (217.37s)	13.95
<i>BSLA_P</i> + <i>A,S,V</i>	9 (56.7%)	14.37	11 (156.45s)	13.84

Table 6-3 Performance of MILP models for side-chain model (square,central,S1-1)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	11 (230.40s)	16.47	11 (193.57s)	15.55
<i>BLASC – P</i>	11 (344.51s)	16.47	11 (252.83s)	15.55
<i>STLA</i>	11 (35.02s)	13.96	11 (27.07s)	14.12
<i>STLA + V</i>	11 (32.11s)	13.99	11 (24.32s)	14.13
<i>BSLA_L + A</i>	11 (28.88s)	14.11	11 (34.63s)	13.99
<i>BSLA_L + S</i>	11 (69.69s)	13.94	11 (91.37s)	13.95
<i>BSLA_L + A, S</i>	11 (57.47s)	13.87	11 (73.23s)	13.92
<i>BSLA_P + A</i>	11 (42.51s)	14.11	11 (69.73s)	13.99
<i>BSLA_P + S</i>	11 (127.69s)	13.94	11 (178.70s)	13.95
<i>BSLA_P + A, S</i>	11 (114.46s)	13.87	11 (97.24s)	13.92
<i>BSLA_P + A, V</i>	11 (44.10s)	14.07	11 (61.54s)	13.98
<i>BSLA_P + S, V</i>	11 (89.79s)	13.89	11 (185.30s)	13.95
<i>BSLA_P + A, S, V</i>	11 (99.81s)	13.83	11 (188.22s)	13.89

Table 6-4 Performance of MILP models for side-chain model (square,terminal,S1-1)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	10 (53.3%)	17.48	11 (432.76s)	16.50
<i>BLASC – P</i>	10 (59.4%)	17.48	11 (341.80s)	16.50
<i>STLA</i>	11 (56.69s)	14.13	11 (37.32s)	14.47
<i>STLA + V</i>	11 (88.12s)	14.13	11 (40.83s)	14.50
<i>BSLA_L + A</i>	11 (44.49s)	14.36	11 (27.77s)	14.51
<i>BSLA_L + S</i>	11 (151.79s)	14.21	11 (88.06s)	14.26
<i>BSLA_L + A, S</i>	11 (214.98s)	14.15	11 (54.80s)	14.20
<i>BSLA_P + A</i>	11 (117.91s)	14.36	11 (57.63s)	14.51
<i>BSLA_P + S</i>	11 (201.57s)	14.21	11 (156.16s)	14.26
<i>BSLA_P + A, S</i>	11 (232.35s)	14.15	11 (75.22s)	14.20
<i>BSLA_P + A, V</i>	11 (140.37s)	14.33	11 (60.40s)	14.51
<i>BSLA_P + S, V</i>	11 (238.07s)	14.19	11 (133.35s)	14.26
<i>BSLA_P + A, S, V</i>	11 (342.98s)	14.13	11 (77.20s)	14.19

Table 6-5 Performance of MILP models for side-chain model (square, central, S1-2)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	11 (41.5%)	16.65	11 (37.7%)	16.32
<i>BLASC – P</i>	12 (31.9%)	16.65	11 (35.0%)	16.32
<i>STLA</i>	12 (16.7%)	14.70	12 (255.58s)	14.67
<i>STLA + V</i>	12 (21.1%)	14.70	12 (281.13s)	14.67
<i>BSLA_L + A</i>	12 (449.25s)	14.68	12 (481.00s)	14.48
<i>BSLA_L + S</i>	12 (20.6%)	14.65	11 (26.2%)	14.47
<i>BSLA_L + A, S</i>	12 (21.1%)	14.65	11 (27.5%)	14.46
<i>BSLA_P + A</i>	12 (21.1%)	14.68	11 (23.4%)	14.48
<i>BSLA_P + S</i>	12 (21.4%)	14.65	11 (28.3%)	14.47
<i>BSLA_P + A, S</i>	11 (32.8%)	14.65	11 (27.5%)	14.46
<i>BSLA_P + A, V</i>	12 (20.2%)	14.67	11 (29.0%)	14.48
<i>BSLA_P + S, V</i>	12 (21.0%)	14.65	11 (28.5%)	14.47
<i>BSLA_P + A, S, V</i>	11 (33.0%)	14.65	11 (28.0%)	14.46

Table 6-6 Performance of MILP models for side-chain model (square, terminal, S1-2)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	11 (49.6%)	17.72	10 (61.4%)	17.49
<i>BLASC – P</i>	11 (51.7%)	17.72	11 (47.2%)	17.49
<i>STLA</i>	12 (20.6%)	14.77	12 (337.37s)	14.86
<i>STLA + V</i>	12 (21.2%)	14.88	12 (296.51s)	14.86
<i>BSLA_L + A</i>	12 (23.7%)	14.92	11 (352.31s)	14.63
<i>BSLA_L + S</i>	12 (23.7%)	14.92	11 (31.0%)	14.63
<i>BSLA_L + A, S</i>	12 (24.1%)	14.91	11 (31.4%)	14.60
<i>BSLA_P + A</i>	12 (23.0%)	14.92	11 (23.4%)	14.63
<i>BSLA_P + S</i>	12 (23.8%)	14.92	11 (31.9%)	14.63
<i>BSLA_P + A, S</i>	12 (24.1%)	14.91	11 (32.4%)	14.60
<i>BSLA_P + A, V</i>	12 (23.7%)	14.87	11 (32.1%)	14.62
<i>BSLA_P + S, V</i>	12 (23.7%)	14.86	11 (31.6%)	14.62
<i>BSLA_P + A, S, V</i>	11 (34.8%)	14.86	11 (31.2%)	14.60

Table 6-7 Performance of MILP models for side-chain model (square,central,S1-3)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	11 (16.7%)	14.73	11 (219.56s)	14.38
<i>BLASC – P</i>	11 (20.3%)	14.73	11 (311.58s)	14.38
<i>STLA</i>	11 (209.42s)	12.00	11 (488.38s)	12.00
<i>STLA + V</i>	11 (208.06s)	12.00	11 (311.39s)	12.00
<i>BSLA_L + A</i>	11 (111.59s)	12.00	11 (71.39s)	12.00
<i>BSLA_L + S</i>	11 (9.1%)	12.00	11 (417.44s)	12.00
<i>BSLA_L + A, S</i>	11 (9.1%)	12.00	11 (343.58s)	12.00
<i>BSLA_P + A</i>	11 (293.48s)	12.00	11 (189.51s)	12.00
<i>BSLA_P + S</i>	11 (9.1%)	12.00	11 (468.41s)	12.00
<i>BSLA_P + A, S</i>	10 (20.0%)	12.00	11 (9.1%)	12.00
<i>BSLA_P + A, V</i>	11 (250.33s)	12.00	11 (159.12s)	12.00
<i>BSLA_P + S, V</i>	6 (100.0%)	12.00	11 (491.31s)	12.00
<i>BSLA_P + A, S, V</i>	11 (9.1%)	12.00	9 (33.3%)	12.00

Table 6-8 Performance of MILP models for side-chain model (square,terminal,S1-3)

Model	Half-length		Quarter-length	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	10 (36.4%)	14.82	11 (197.42s)	14.67
<i>BLASC – P</i>	10 (35.7%)	14.82	11 (467.99s)	14.67
<i>STLA</i>	11 (266.26s)	12.00	11 (279.40s)	12.00
<i>STLA + V</i>	11 (268.37s)	12.00	11 (185.05s)	12.00
<i>BSLA_L + A</i>	11 (198.90s)	12.00	11 (121.95s)	12.00
<i>BSLA_L + S</i>	10 (20.0%)	12.00	11 (210.52s)	12.00
<i>BSLA_L + A, S</i>	11 (9.1%)	12.00	11 (338.66s)	12.00
<i>BSLA_P + A</i>	11 (234.52s)	12.00	11 (150.93s)	12.00
<i>BSLA_P + S</i>	11 (9.1%)	12.00	11 (467.24s)	12.00
<i>BSLA_P + A, S</i>	10 (20.0%)	12.00	11 (474.06s)	12.00
<i>BSLA_P + A, V</i>	11 (357.35s)	12.00	11 (158.93s)	12.00
<i>BSLA_P + S, V</i>	10 (20.0%)	12.00	11 (296.65s)	12.00
<i>BSLA_P + A, S, V</i>	10 (20.0%)	12.00	11 (486.52s)	12.00

Table 6-9 Performance of MILP models for side-chain model (cubic,quarter,S1-0)

Model	Central		Terminal	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	7 (186.0%)	21.07	10 (106.88%)	21.93
<i>BLASC – P</i>	5 (289.3%)	20.92	6 (241.19%)	21.77
<i>STLA</i>	10 (88.4%)	19.35	15 (36.02%)	20.41
<i>STLA + V</i>	6 (223.8%)	19.67	9 (122.28%)	20.41
<i>BSLA_L + A</i>	13 (49.8%)	19.62	13 (52.26%)	20.42
<i>BSLA_P + A</i>	8 (151.9%)	20.38	11 (90.23%)	20.94
<i>BSLA_P + A, V</i>	5 (299.2%)	20.22	10 (109.13%)	20.93

Table 6-10 Performance of MILP models for side-chain model (cubic,quarter,S1-1)

Model	Central		Terminal	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	10 (102.2%)	20.84	7 (195.5%)	21.93
<i>BLASC – P</i>	3 (557.0%)	20.66	7 (193.5%)	21.83
<i>STLA</i>	9 (118.2%)	19.70	14 (44.6%)	20.26
<i>STLA + V</i>	10 (96.8%)	19.78	15 (34.7%)	20.36
<i>BSLA_L + A</i>	13 (51.1%)	19.77	15 (36.0%)	20.45
<i>BSLA_P + A</i>	10 (99.0%)	20.30	10 (105.5%)	21.20
<i>BSLA_P + A, V</i>	10 (101.1%)	20.29	12 (71.4%)	21.08

Table 6-11 Performance of MILP models for side-chain model (cubic,quarter,S1-2)

Model	Central		Terminal	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	3 (644.7%)	22.34	4 (492.7%)	24.21
<i>BLASC – P</i>	4 (438.1%)	22.20	2 (1061.7%)	24.01
<i>STLA</i>	4 (441.0%)	21.70	4 (439.1%)	21.57
<i>STLA + V</i>	2 (982.0%)	21.70	2 (986.2%)	21.75
<i>BSLA_L + A</i>	11 (93.4%)	21.60	15 (49.1%)	22.50
<i>BSLA_P + A</i>	1 (2042.9%)	21.48	12 (88.6%)	22.78
<i>BSLA_P + A, V</i>	1 (2042.9%)	21.48	6 (274.9%)	22.69

Table 6-12 Performance of MILP models for side-chain model (cubic,quarter,S1-3)

Model	Central		Terminal	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
<i>BLASC – L</i>	4 (380.8%)	19.23	3 (544.9%)	19.67
<i>BLASC – P</i>	2 (824.2%)	19.15	2 (847.1%)	19.51
<i>STLA</i>	-	18.23	12 (54.6%)	-
<i>STLA + V</i>	2 (825.9%)	-	9 (112.2%)	19.12
<i>BSLA_L + A</i>	8 (127.2%)	18.21	9 (103.2%)	18.33
<i>BSLA_P + A</i>	1 (1703.6%)	18.09	7 (166.4%)	18.68
<i>BSLA_P + A, V</i>	1 (1703.6%)	18.09	2 (821.5%)	18.46

6.2 Linear Chain HP Models under Triangular Lattice

Due to the parity restriction of cubic and square lattices, in these lattices only contacts between amino acids of opposite parities are possible. To tackle this problem, many sophisticated lattice models that do not obey this restriction have been proposed, among which are Face-Centered Cubic (FCC), Knight Walk (KW), and triangular (or hexagonal) lattices. Interested reader is referred to Godzik et al. (1993) for detailed comparison of different lattices.

Due to high cardinality of set of basic directions of FCC (12) and KW (24) in 3D, and considering the performance of mathematical models under cubic lattice, it is unlikely that presented mathematical models can show satisfactory performance for these complicated parity free lattices. To capture the essence of parity free lattices, in the following we extend DLA, which is the best performing model in most of the cases in the square lattice, to the triangular lattice.

Before presenting the model, let us investigate the triangular lattice in more details. Note that in this lattice, each lattice point is adjacent with six lattice points each along one of the basic directions of triangular lattice. The basic directions of triangular lattice are $(\pm 1, 0)$ and $(\pm 1/2, \pm \sqrt{3}/2)$. Note that multiplying these vectors by 2 yield vectors $(\pm 2, 0)$ and $(\pm 1, \pm \sqrt{3})$. Therefore, for the sake of simplicity in implementation, instead of $(\pm 1, \pm \sqrt{3})$ we consider vectors $(\pm 1, \pm 2)$. Note that every lattice point in the original triangular lattice can be transformed into a point in the modified triangular lattice by

transformation matrix $T = \begin{bmatrix} 2 & 0 \\ 0 & \frac{4\sqrt{3}}{3} \end{bmatrix}$ and transformed back to the original lattice by T^{-1} .

Figure 6-3 shows a sample solution in the triangular lattice, for the sequence S1-0, with this modified set of basic vectors.

Moreover, note that the same symmetries as explained in the case of cubic lattice appear in the triangular lattice, and we can eliminate them in a similar manner. For instance, analogous to central symmetry breaking in the square lattice, we can fix the so called middle amino acid (m) at the origin, and $m + 1$ at $(2,0,0)$. Moreover, we can exclude points $(-2(m - k - 1) \pm 1, -2)$ from feasible lattice points of amino acids $k < m$.

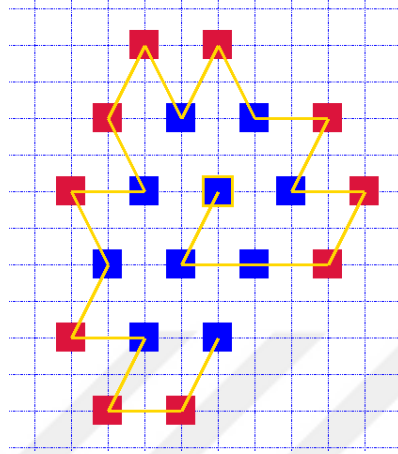


Figure 6-3 Sample optimal solution in triangular lattice for sequence S1-0

Double Link Assignment Model under Triangular Lattice

In extending DLA to triangular lattice (DLAT), we use the same sets of variables as in DLA. To point out the main differences of DLA and DLAT, we must note that in triangular lattice, amino acids are no longer limited to lattice points of the same parity. Consequently, in the assignment of amino acids of parity 0 (which emit the forward and the backward directions) to their feasible points, we should also include the assignment of amino acids of parity 1.

In DLAT, the location variables are defined in the exact way as in DLA. The location-wise objective function variables, however, are defined in a slightly different way. In order to avoid redundancies in counting the number of edges, we constrain the set of basic directions that define the edges to $\delta^+ = \{(2,0), (1,2), (-1,2)\}$. The variables counting the number of edges occupied by hydrophobic amino acids (W) are therefore defined as:

$$W_p^d = \begin{cases} 1, & (p, p + d) \text{ are occupied by } H \text{ amino acids} \\ 0, & \text{otherwise} \end{cases}, \quad (\text{DLAT-1})$$

$p \in \mathcal{F}, d \in \delta^+; p + d \in \mathcal{F}.$

Model (DLAT-L)

$$\max \sum_{p \in \mathcal{F}} \sum_{d \in \delta^+; p+d \in \mathcal{F}} W_p^d \quad (\text{DLAT-2})$$

Subject to:

$$\sum_{p \in F_k} \sum_{f \in \delta:} \sum_{\substack{b \in \delta - \{f\}: \\ p+f \in F_{k+1} \ p+b \in F_{k-1}}} X_{p,f,b}^k = 1, \quad \forall k \in \mathcal{R}^0, \quad (\text{DLAT-3})$$

$$\sum_{\substack{k \in \mathcal{R}^0: \\ p \in F_k}} \sum_{f \in \delta:} \sum_{\substack{b \in \delta - \{f\}: \\ p+f \in F_{k+1} \ p+b \in F_{k-1}}} X_{p,f,b}^k + \sum_{\substack{k \in \mathcal{R}^1: \\ p \in F_k}} \sum_{f \in \delta:} \sum_{\substack{b \in \delta - \{f\}: \\ p-f \in F_{k-1} \ p-f+b \in F_{k-2}}} X_{p-f,f,b}^k \leq 1, \quad (\text{DLAT-4})$$

$$\sum_{\substack{f \in \delta: \\ p-f \in F_{k-1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p-f+b \in F_{k-2}}} X_{p-f,f,b}^{k-1} = \sum_{\substack{b \in \delta: \\ p-b \in F_{k+1}}} \sum_{\substack{f \in \delta - \{b\}: \\ p-b+f \in F_{k+2}}} X_{p-b,f,b}^{k+1}, \quad (\text{DLAT-5})$$

$$W_p^d + \sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_k, \\ p+d \in F_{k-1}}} \sum_{\substack{b \in \delta - \{-d\}: \\ p+d+b \in F_{k-2}}} X_{p+d,-d,b}^{k-1} \quad (\text{DLAT-6})$$

$$\begin{aligned} &\leq \sum_{\substack{k \in \mathcal{H}^0: \\ p \in F_k}} \sum_{\substack{f \in \delta - \{d\}: \\ p+f \in F_{k+1}}} \sum_{\substack{b \in \delta - \{d,f\}: \\ p+b \in F_{k-1}}} X_{p,f,b}^k \\ &+ \sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_k}} \sum_{\substack{b \in \delta - \{-d\}: \\ p-b \in F_{k+1}}} \sum_{\substack{f \in \delta - \{b\}: \\ p-b+f \in F_{k+2}}} X_{p-b,f,b}^{k+1}, \\ &\forall p \in \mathcal{F}, d \in \delta^+: p+d \in \mathcal{F}, \end{aligned} \quad (\text{DLAT-7})$$

$$\begin{aligned} &W_p^d + \sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_{k+1}, \\ p+d \in F_k}} \sum_{\substack{f \in \delta - \{d\}: \\ p+f \in F_{k+2}}} X_{p,f,d}^{k+1} \\ &\leq \sum_{\substack{k \in \mathcal{H}^1: \\ p+d \in F_k}} \sum_{\substack{f \in \delta - \{d\}: \\ p+d-f \in F_{k-1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p+d-f+b \in F_{k-2}}} X_{p+d-f,f,b}^{k-1} \\ &+ \sum_{\substack{k \in \mathcal{H}^0: \\ p+d \in F_k}} \sum_{\substack{f \in \delta - \{-d\}: \\ p+d+f \in F_{k+1}}} \sum_{\substack{b \in \delta - \{-d,f\}: \\ p+d+b \in F_{k-1}}} X_{p+d,f,b}^k, \\ &\forall p \in \mathcal{F}, d \in \delta^+: p+d \in \mathcal{F}, \end{aligned}$$

$$X_{p,f,b}^k \in \{0,1\} \quad k \in \mathcal{R}^0, f \in \delta, b \in \delta - \{f\}, p \in F_k: p+f \in F_{k+1} \wedge p+b \in F_{k-1}, \quad (\text{DLAT-8})$$

$$W_p^d \in \{0,1\}, \quad p \in \mathcal{F}^0, d \in \delta^+: p+d \in \mathcal{F}^1. \quad (\text{DLAT-9})$$

The pairwise objective function (DLAT-P) is also defined in a similar manner as in DLA-P, with the only difference that contacts are extended to all hydrophobic amino acids and direction of contacts are limited to δ^+ .

Computational Result

Table 6-13 shows the results obtained by DLAT-L and DLAT-P for five benchmark instances. The time limit has been set to 2000 seconds, and the optimality gaps are reported if the models are terminated due to this limit. We note that converting the lattice to triangular lattice dramatically increases the running time of the models. This is mainly due to the parity restriction of square lattice that our models heavily benefit

from. In triangular lattice, on one hand all amino acids can form contacts, which in turn increases the number of pair-wise decision variables, and on the other hand increases the number of variables on the right side of location-wise decision variables, which results in weaker constraints.

Table 6-13 Performance of MILP models under triangular lattice

Instance	DLAT-L		DLAT-P	
	MILP (Time/Gap)	LP	MILP (Time/Gap)	LP
S1-0	17 (20.5s)	18.88	17 (245.2s)	18.40
S1-1	12 (42.3%)	20.56	15 (20.8%)	18.86
S1-2	12 (69.8%)	21.18	17 (14.0%)	19.44
S1-3	11 (44.6%)	15.98	12 (26.3%)	15.15
S1-4	9 (216.8%)	29.95	-	-

6.3 Extension to Other Energy Functions

The main abstraction regarding the HP model is limiting the contacts contributing to the energy function to the hydrophobic contacts. This abstraction results in the problem of finding unique optimal solutions in many instances, except probably some artificial sequences. Several corrections have been proposed to address this problem. For example, in some models, the $\{H, P\}$ alphabets in the HP model are extended to more alphabets, such as $\{H, P, N, X\}$ by Backofen et al. (1999), in which H-H, N-P, N-N, and P-P contacts respectively contribute -4, -1, 1, 1 to energy, while other contacts are neutral.

Berrera's Energy Function

In the most general case, all 20×20 pairs of amino acids contribute to the energy function. Such energies are first proposed by Miyazawa & Jernigan (1985), and later improved by Berrera, Molinari & Fogolari (2003). We will refer to former as *MJ matrix*, and the latter as *Berrera's matrix* throughout the rest of this section. Berrera's empirical contact energy matrix is given in Table 6-14, in which bold faced entries indicate positive contact energy between pairs.

We define two new parameters regarding the type of amino acids and contribution of each contact to the objective function.

$$\alpha_k = \text{Type of amino acid } k, \quad k \in \mathcal{R}, \quad (\text{DLAB-1})$$

$$\beta_{kl} = \text{Value of contact } (\alpha_k, \alpha_l) \text{ in Berrea's matrix}, \quad k, l \in \mathcal{R}. \quad (\text{DLAB-2})$$

In the following, we extend the DLA model to the general model with all possible interactions, which we denote by *DLAB*. To adapt DLA to this new energy function, we

must note that we no longer can use the location-wise decision variables for measuring the energy function, therefore in the following we extend DLA-P to the general problem. Note that if we replace $\mathcal{H}^\theta$ with $\mathcal{R}^\theta$ in the domain of variables and constraints, all of the constraints and variables will remain the same as defined in DLA-P. Note also that in HP model, since we are maximizing the total number of contacts, the objective function forces the contact variables to acquire value 1, if the constraints allow them to be 1. In the general model, however, we should force the variables to become 1 if they are not encouraged by the objective function to become 1 (e.g. bold faced entries in Table 6-14.)

Model (DLAB)

$$\max \sum_{k \in \mathcal{R}^0} \sum_{p \in F_k} \sum_{l \in \mathcal{R}^1 - \{k \pm 1\}} \sum_{d \in \delta: p+d \in F_l} -\beta_{kl} Z_{p,d}^{k,l} \quad (\text{DLAB-3})$$

Subject to:

$$(DLA - 4) - (DLA - 7), (DLA - 10), (DLA - 14) - (DLA - 16)$$

$$Z_{p,d}^{k,l} \geq \sum_{\substack{f \in \delta: \\ p+f \in F_{k+1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p+b \in F_{k-1}}} X_{p,f,b}^k + \sum_{\substack{f \in \delta: \\ p+d-f \in F_{l-1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p+d-f+b \in F_{l-2}}} X_{p+d-f,b}^{l-1} - 1, \quad (\text{DLAB-4})$$

$$\forall k \in \mathcal{R}^0, p \in F_k, l \in \mathcal{R}^1 - \{k \pm 1, 0\}, d \in \delta: p+d \in F_l$$

$$Z_{p,d}^{k,0} \geq \sum_{\substack{f \in \delta: \\ p+f \in F_{k+1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p+b \in F_{k-1}}} X_{p,f,b}^k + \sum_{\substack{b \in \delta: \\ p+d-b \in F_1}} \sum_{\substack{f \in \delta - \{b\}: \\ p+d-b+f \in F_2}} X_{p+d-b,b}^1 - 1, \quad (\text{DLAB-5})$$

$$\forall k \in \mathcal{R}^0 - \{0, 2\}, p \in F_k, d \in \delta: p+d \in F_0.$$

Constraint sets (DLAB - 4) and (DLAB - 5) require that if amino acids k and l are respectively located at lattice points p and $p+d$, then there must be a contact between these two over the edge $(p, p+d)$. Note that these constraints should only be written for the pairs with positive entry in Table 6-14 (bold faced). Let us define terms X_p^k analogous to the location assignment variables in BLA, therefore we have:

$$X_p^k = \sum_{\substack{f \in \delta: \\ p+f \in F_{k+1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p+b \in F_{k-1}}} X_{p,f,b}^k, \forall k \in \mathcal{R}^0, p \in F_k, \quad (\text{DLAB-6})$$

$$X_p^l = \sum_{\substack{f \in \delta: \\ p-f \in F_{l-1}}} \sum_{\substack{b \in \delta - \{f\}: \\ p-b+f \in F_{l-2}}} X_{p-f,b}^{l-1}, \forall l \in \mathcal{R}^1 - \{0\}, p \in F_l, \quad (\text{DLAB-7})$$

$$X_p^0 = \sum_{\substack{b \in \delta: \\ p-b \in F_1}} \sum_{\substack{f \in \delta - \{b\}: \\ p-b+f \in F_2}} X_{p-b,b}^1, \forall p \in F_0. \quad (\text{DLAB-8})$$

Consequently, we can rewrite (DLAB - 4) and (DLAB - 5) in terms of the location variables:

$$Z_{p,d}^{k,l} \geq X_p^k + X_{p+d}^l - 1, \quad \forall k \in \mathcal{R}^0, p \in F_k, l \in \mathcal{R}^1 - \{k \pm 1\}, d \in \delta: p + d \in F_l. \quad (\text{DLAB-9})$$

Moreover, note that constraints (DLA – 14) and (DLA – 15) can be written (in weakened form) in terms of location assignment variables as:

$$\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} \leq X_p^k, \quad \forall k \in \mathcal{R}^0, p \in F_k, d \in \delta, \quad (\text{DLAB-10})$$

$$\sum_{\substack{k \in \mathcal{R}^0 - \{l \pm 1\}: \\ p \in F_k}} Z_{p,d}^{k,l} \leq X_p^l, \quad \forall l \in \mathcal{R}^1, d \in \delta, p \in \mathcal{F}^0: p + d \in F_l. \quad (\text{DLAB-11})$$

Now we show that constraints (DLAB – 4) and (DLAB – 5) can be replaced with stronger constraints (DLAB – 12) and (DLAB – 13) as follows:

$$\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} \geq X_p^k + \sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1\}: \\ p+d \in F_l}} X_{p+d}^l - 1, \quad (\text{DLAB-12})$$

$$\sum_{\substack{k \in \mathcal{R}^0 - \{l \pm 1\}: \\ p \in F_k}} Z_{p,d}^{k,l} \geq X_{p+d}^l + \sum_{\substack{k \in \mathcal{R}^0 - \{l \pm 1\}: \\ p \in F_k}} X_p^k - 1. \quad (\text{DLAB-13})$$

Proposition 4: (DLAB – 12) and (DLAB – 13) imply (DLAB – 9).

Proof: First, we show that (DLAB – 12) implies (DLAB – 9). Note that $\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1\}: \\ p+d \in F_l}} Z_{p,d}^{k,l}$

and $\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1\}: \\ p+d \in F_l}} X_{p+d}^l$ can be respectively written as $\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} + Z_{p,d}^{k,j}$ and

$\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} X_{p+d}^l + X_{p+d}^j$ for some $j \in \mathcal{R}^1 - \{k \pm 1\}$. Therefore, (DLAB – 10) can be

rewritten as:

$$-1 + \sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} X_{p+d}^l + X_{p+d}^j + X_p^k \leq \sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} + Z_{p,d}^{k,j}, \quad (\text{DLAB-14})$$

$$\Rightarrow -1 + X_{p+d}^j + X_p^k \leq Z_{p,d}^{k,j} + \sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} - \sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} X_{p+d}^l. \quad (\text{DLAB-15})$$

Moreover, (DLAB – 11) implies that $\sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} Z_{p,d}^{k,l} \leq \sum_{\substack{l \in \mathcal{R}^1 - \{k \pm 1, j\}: \\ p+d \in F_l}} X_{p+d}^l$, therefore

the right hand side of (DLAB – 14) is less than or equal to $Z_{p,d}^{k,j}$, which is constraint (DLAB – 9). In a similar way we can show the other implication. ■

Corollary 5: Either of (DLAB – 12) or (DLAB – 13) can replace (DLAB – 9), while the other one can be used as valid inequality.

Table 6-14 Empirical contact energies for all pairs of standard amino acids as suggested by Berrera, Molinari & Fogolari (2003)

	CYS	MET	PHE	ILE	LEU	VAL	TRP	TYR	ALA	GLY
CYS	-3.477	-2.24	-2.424	-2.41	-2.343	-2.258	-2.08	-1.892	-1.7	-1.101
MET	-2.24	-1.901	-2.304	-2.286	-2.208	-2.079	-2.09	-1.834	-1.517	-0.897
PHE	-2.424	-2.304	-2.467	-2.53	-2.491	-2.391	-2.286	-1.963	-1.75	-1.03
ILE	-2.41	-2.286	-2.53	-2.691	-2.647	-2.568	-2.303	-1.998	-1.872	-0.885
LEU	-2.343	-2.208	-2.491	-2.647	-2.501	-2.447	-2.222	-1.919	-1.728	-0.767
VAL	-2.258	-2.079	-2.391	-2.568	-2.447	-2.385	-2.097	-1.79	-1.731	-0.756
TRP	-2.08	-2.09	-2.286	-2.303	-2.222	-2.097	-1.867	-1.834	-1.565	-1.142
TYR	-1.892	-1.834	-1.963	-1.998	-1.919	-1.79	-1.834	-1.335	-1.318	-0.818
ALA	-1.7	-1.517	-1.75	-1.872	-1.728	-1.731	-1.565	-1.318	-1.119	-0.29
GLY	-1.101	-0.897	-1.034	-0.885	-0.767	-0.756	-1.142	-0.818	-0.29	0.219
THR	-1.243	-0.999	-1.237	-1.36	-1.202	-1.24	-1.077	-0.892	-0.717	-0.311
SER	-1.306	-0.893	-1.178	-1.037	-0.959	-0.933	-1.145	-0.859	-0.607	-0.261
ASN	-0.788	-0.658	-0.79	-0.669	-0.524	-0.673	-0.884	-0.67	-0.371	-0.23
GLN	-0.835	-0.72	-0.807	-0.778	-0.729	-0.642	-0.997	-0.687	-0.323	0.033
ASP	-0.616	-0.409	-0.482	-0.402	-0.291	-0.298	-0.613	-0.631	-0.235	-0.097
GLU	-0.179	-0.209	-0.419	-0.439	-0.366	-0.335	-0.624	-0.453	-0.039	0.443
HIS	-1.499	-1.252	-1.33	-1.234	-1.176	-1.118	-1.383	-1.222	-0.646	-0.325
ARG	-0.771	-0.611	-0.805	-0.854	-0.758	-0.664	-0.912	-0.745	-0.327	-0.05
LYS	-0.112	-0.146	-0.27	-0.253	-0.222	-0.2	-0.391	-0.349	0.196	0.589
PRO	-1.196	-0.788	-1.076	-0.991	-0.771	-0.886	-1.278	-1.067	-0.374	-0.042
	THR	SER	ASN	GLN	ASP	GLU	HIS	ARG	LYS	PRO
CYS	-1.243	-1.306	-0.788	-0.835	-0.616	-0.179	-1.499	-0.771	-0.112	-1.196
MET	-0.999	-0.893	-0.658	-0.72	-0.409	-0.209	-1.252	-0.611	-0.146	-0.788
PHE	-1.237	-1.178	-0.79	-0.807	-0.482	-0.419	-1.33	-0.805	-0.27	-1.076
ILE	-1.36	-1.037	-0.669	-0.778	-0.402	-0.439	-1.234	-0.854	-0.253	-0.991
LEU	-1.202	-0.959	-0.524	-0.729	-0.291	-0.366	-1.176	-0.758	-0.222	-0.771
VAL	-1.24	-0.933	-0.673	-0.642	-0.298	-0.335	-1.118	-0.664	-0.2	-0.886
TRP	-1.077	-1.145	-0.884	-0.997	-0.613	-0.624	-1.383	-0.912	-0.391	-1.278
TYR	-0.892	-0.859	-0.67	-0.687	-0.631	-0.453	-1.222	-0.745	-0.349	-1.067
ALA	-0.717	-0.607	-0.371	-0.323	-0.235	-0.039	-0.646	-0.327	0.196	-0.374
GLY	-0.311	-0.261	-0.23	0.033	-0.097	0.443	-0.325	-0.05	0.589	-0.042
THR	-0.617	-0.548	-0.463	-0.342	-0.382	-0.192	-0.72	-0.247	0.155	-0.222
SER	-0.548	-0.519	-0.423	-0.26	-0.521	-0.161	-0.639	-0.264	0.223	-0.199
ASN	-0.463	-0.423	-0.367	-0.253	-0.344	0.16	-0.455	-0.114	0.271	-0.018
GLN	-0.342	-0.26	-0.253	0.054	0.022	0.179	-0.29	-0.042	0.334	-0.035
ASP	-0.382	-0.521	-0.344	0.022	0.179	0.634	-0.664	-0.584	-0.176	0.189
GLU	-0.192	-0.161	0.16	0.179	0.634	0.933	-0.324	-0.374	-0.057	0.257
HIS	-0.72	-0.639	-0.455	-0.29	-0.664	-0.324	-1.078	-0.307	0.388	-0.346
ARG	-0.247	-0.264	-0.114	-0.042	-0.584	-0.374	-0.307	0.2	0.815	-0.023
LYS	0.155	0.223	0.271	0.334	-0.176	-0.057	0.388	0.815	1.339	0.661
PRO	-0.222	-0.199	-0.018	-0.035	0.189	0.257	-0.346	-0.023	0.661	0.129

The metaheuristic algorithms can be easily adapted to this general objective function. Since these algorithms are written for maximizing the objective function, we need to pass the contact energies multiplied by -1. Due to superior performance of ILS over TS and SS, here we only adapt ILS to this new objective function.

Computational Result

To analyze the effect of considering all possible contacts on DLAB, as well as robustness of ILS, we run the DLAB and ILS on some real proteins obtained from *Protein Data Bank* (*PDB*²) both in the square and cubic lattices. The length of these proteins varies from 12 to 40 amino acids and are given in Table 6-15. In Table 6-16 we present the results of running DLAB on these proteins under square lattice within the time limit of 1000 seconds. Results for longer proteins are given in Table 6-17 in which the time limit is set to 2000 seconds. In Table 6-19 results of DLAB for some proteins under cubic lattice are given.

In these tables, we present both integer and fractional solutions. Note that the objective values are given with positive signs. As the results indicate, considering the 20×20 contacts significantly increases the running time of the model. This can be explained by two reasons: first note that the integrality gaps are higher than the instances in the HP model. This implies that the model has to close a larger gap for determining the optimality of the solution at hand. Second, considering all contacts, on one hand increases the number of variables, and on the other hand requires additional constraints (*DLAB* – 12). These additional variables/constraints impose further computational expenses on the solver, hence the longer running time.

Table 6-15 Instances of real proteins obtained from PDB

ID	Length	Sequence
1LE0	12	SWTWEGNKWTWK
1KVG	12	SCHFGPLGWVCK
1LE3	16	GEWTWDDATKTWTWTE
1EDP	17	CSCSSLMDKECVYFCHL
1PG1	18	RGRLCYCRRRFCVCVGR
1PSV	28	KPYTARIKGRTFSNEKELRDFLETFTGR
1FSD	28	QQYTAKIKGRTFRNEKELRDFIEKFKGR
1NMJ	28	DAEFGHDSGFVVRHQKLVFFAEDVGSNK
1WN4	28	ALETQKPNHLLLEALVAFKKGNLGGLP
2AJJ	28	SGNYVLDLIYSLHKQINRGLKKIVLGWA
1ZDD	34	FNMQCQRRFYALHDPNLNEEQRNAKIKSIRDDC
1VII	36	MLSDEDFKAVFGMTRSAFANLPLWKQQNLKKEKGLF
1E0M	37	SMGLPPGWDEYKTHNGKTYYYNHNTKTSTWTDPRMSS
2GP8	40	ITGDVSAANKDAIRKQMDAAASKGDVETRYRKLKAKLKGIR

² <http://www.rcsb.org/pdb>

To show the robustness of ILS, we further test this algorithm on some of the real proteins. As shown in Table 6-18, ILS is able to find the global optimum for almost all of the instances in square lattice, and four instances in Table 6-19 for which DLAB is able to find the optimal solution in the cubic lattice. This observation makes us believe that the other solutions are most probably optimal or very close to optimal. Figure 6-4 illustrates solutions found by ILS for proteins *1PSV* in 3D (left) and *1LE0* in 2D (right).

Table 6-16 Performance of DLAB on real proteins in square lattice (part 1)

ID	Length	MILP (1000s)		LP		Integrality Gap
		Obj. (Gap)	Time	Obj.	Time	
1LE0	12	6.743	0.11	6.74	0.02	0.00
1KVG	12	10.839	0.16	11.54	0.02	0.70
1LE3	16	9.217	1.37	9.43	1.81	0.21
1EDP	17	15.974	8.92	17.71	2.04	1.74
1PG1	18	16.788	3.71	17.70	2.42	0.91
1PSV	28	17.025 (20.13%)	1000.11	20.59	19.64	-
1FSD	28	16.089 (22.03%)	1000.08	19.86	18.53	-
1NMJ	28	18.872	838.61	21.40	21.14	2.53
1WN4	28	22.078	442.47	24.88	22.21	2.81
2AJJ	28	26.175 (15.43%)	1000.15	30.47	22.17	-

Table 6-17 Performance of DLAB on real proteins in square lattice (part 2)

ID	Length	MILP (1000s)		LP	
		Obj. (Gap)	Time	Obj.	Time
1ZDD	34	20.962 (54.68%)	2000.42	32.62	93.05
1VII	36	26.846 (30.54%)	2000.34	35.12	109.22
1E0M	37	17.09 (51.37%)	2000.31	26.01	116.52
2GP8	40	16.86 (103.74%)	2000.34	34.54	262.18

Table 6-18 Performance of ILS on real proteins in square and cubic lattices

ID	Length	Square Lattice		Cubic Lattice	
		Obj.	Time (s)	Obj.	Time (s)
1LE0	12	6.743	0.49	7.811	2.73
1KVG	12	10.839	0.88	13.244	1.23
1LE3	16	9.217	2.47	11.959	4.85
1EDP	17	15.974	1.32	21.687	1.26
1PG1	18	16.788	1.10	23.214	5.94
1PSV	28	17.029	4.01	28.131	10.01
1FSD	28	16.438	5.56	27.466	12.45
1NMJ	28	18.704	6.85	27.704	15.41
1WN4	28	22.078	7.10	31.062	15.56
2AJJ	28	26.078	4.27	40.076	18.57
1ZDD	34	26.297	13.51	41.253	14.64
1VII	36	29.867	39.40	46.574	49.72
1E0M	37	22.15	16.14	35.476	55.65
2GP8	40	28.5	11.64	46.289	69.43

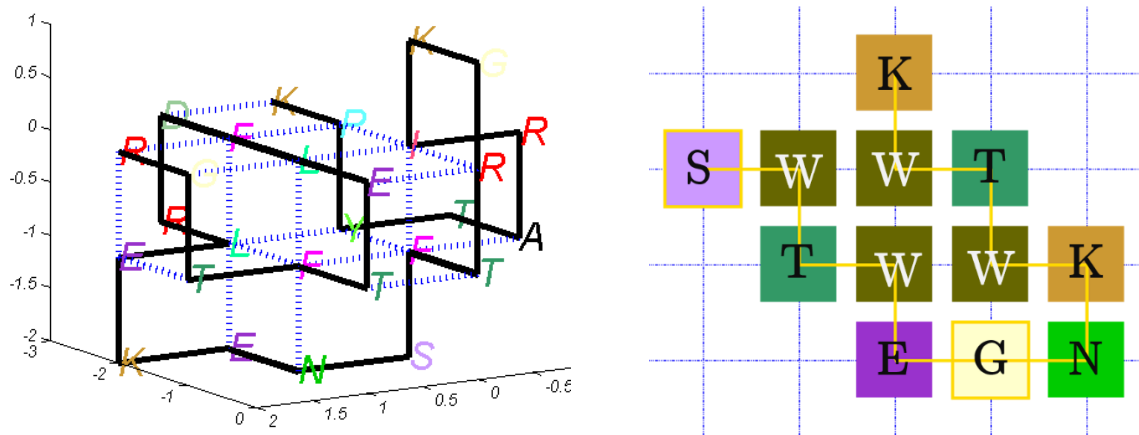


Figure 6-4 Optimal solution obtained by ILS for proteins 1PSV (left) and 1LE0 (right) based on Berrera's energy matrix

Table 6-19 Performance of DLAB on real proteins in cubic lattice

ID	Length	MILP (2000s)		LP		Integrality Gap
		Obj. (Gap)	Time	Obj.	Time	
1LE0	12	7.811	3.29	9.17	3.17	1.363
1KVG	12	13.244	4.24	16.17	2.85	2.922
1LE3	16	11.959	460.87	14.97	55.07	3.012
1EDP	17	21.572 (31.47%)	2000.14	28.85	70.73	-
1PG1	18	23.214	1179.82	27.18	105.35	3.969

Table 6-20 Distinguishable colors for amino acids obtained by maximizing the minimum similarity

k	Associated Amino Acid	Color	R	G	B
1	A		0	0	0
2	R		255	0	0
3	N		0	203	0
4	D		153	204	153
5	C		51	0	255
6	E		153	50	204
7	Q		0	153	255
8	G		255	255	204
9	H		102	0	102
10	I		255	76	127
11	L		0	255	153
12	K		204	153	51
13	M		0	51	153
14	F		255	0	255
15	P		102	255	255
16	S		204	153	255
17	T		51	153	102
18	W		102	102	0
19	Y		101	255	50
20	V		255	255	0

An interesting problem of finding the most different colors arises while visualizing these solutions. To find such colors we need the closest colors to be as distinguishable as possible. In other words, we need the colors to be as distant as possible in terms RGB. The problem is equivalent to finding the position of 20 (+2) points in a cube side length 255 (the range of RGB elements of colors) such that the minimum distance between each of these points is maximized. In Appendix VI, we also introduce mathematical models for solving this problem. The colors used in Figure 6-4 are obtained by solving this problem, and can be found in Table 6-20.



Chapter 7

CONCLUSION AND FUTURE WORK

In this thesis we propose optimization approaches for the protein structure prediction problem under lattice structures. We first introduce several integer linear programming models, sets of valid inequalities, as well as strategies for eliminating the symmetric solutions. We show that these symmetric solutions can be excluded from the search space of the models without imposing additional constraints to the models. Our results demonstrate that the models integrated with these strategies and with the valid inequalities significantly outperform previously proposed models in the literature both in terms of running times required for finding the optimal integer solutions, and the tightness of the bounds provided by the LP relaxations of the models, especially for small sized instances.

To solve moderate sized problems in an affordable time, we then propose metaheuristic algorithms that are able to find near optimal solutions in short time. We next show how we can reduce the size of the problem for the mathematical models using the information provided by these solutions. We also use these solutions in the mathematical models as initial solutions to further help improve their performance.

Our computational experiments indicate that this integrated method can significantly improve the performance of mathematical models, and it is able to find the optimal solution of instances above 64 amino acids in the square lattice, while this length can hardly exceed 40 amino acids using the plain mathematical models. Similarly in the cubic lattice, while finding the optimal solution for sequences up to 20 amino acids using the mathematical models may take thousands of seconds, the integrated method is able to find the optimal solutions in matter of seconds. Therefore, the integrated method can be used as an alternative for finding the optimality guaranteed solutions. This method can be helpful in devising new benchmark instances and can help other researchers with evaluating their heuristic algorithms.

Due to its over-simplified nature, HP model may fail to predict and represent the native structure of real proteins. Although the focus of this research is on the conventional HP model under cubic and square lattices, we also study some of the drawbacks of this model. We address some of these drawbacks by extending our methods to the side-chain model, triangular lattice, and the general energy functions. Our computational results indicate the robustness of our methods, especially when considering the general energy functions and the side-chain models. To the best of our knowledge, it is the first time that an exact method has been developed for PSP under square and cubic lattices considering the general energy functions. Moreover, our computational results show the superior

performance of our mathematical models proposed for the side-chain model over previously proposed mathematical models.

There are several directions to continue this research. To the best of our knowledge, few studies have considered the hydrophobic polar model with side-chains. Although we have extended our mathematical models to cover this variation of the problem, due to lack of good heuristic methods, we have not tested the performance of these extended models when initial solutions are available. For linear chain models under cubic and square lattices, variety of neighborhood structures and moves have been proposed in the literature; however, the literature lacks good sets of neighborhood structures for more complicated lattice structures as well as the side-chain model. Furthermore, one can consider more problem specific properties of the lattices to introduce more sophisticated methodologies for solving the PSP problem under lattices.

One can also consider grouping the amino acids thus converting the problem into an easier low resolution problem. For instance, one can group every two, three, or four consecutive amino acids, and treat them as single motifs with accumulated hydrophobicity of their members. Similarly, one can identify and fix the structure of local structures, and find the optimal solution within the reduced feasible region. Incorporating the statistical data in the form of dominance properties can be another possibility. Other methodologies, such as dynamic programming inspired by the chain growth algorithms, can be promising.

Other strategies for integrating the mathematical models with the metaheuristic algorithms can be also applied. One such strategy is to use the TS algorithm as a local search method for improving the integer solutions that the solver finds during the branch and cut procedure.

Appendix I

PROOF OF LEMMA 1

In this appendix we present the proof of Lemma 1 introduced in Chapter 1.

Proof of Lemma 1: Let p_0 denote the starting point of the walk, and $p_k = (x_k, y_k, z_k)$, denote the location of the node after taking k steps starting from p_0 . Let $\delta_1 = (1,0,0)$, $\delta_2 = (0,1,0)$, $\delta_3 = (0,0,1)$, $\delta_4 = (-1,0,0)$, $\delta_5 = (0,-1,0)$ and $\delta_6 = (0,0,-1)$, denote the basic vectors. Let s_l^d be 1, if $p_l - p_{l-1} = \delta_d$, and 0 otherwise, for $d = 1, \dots, 6$ and $l = 1, \dots, k$. Since exactly one of the basic vectors must be taken in each step, we have:

$$\sum_{d=1}^6 s_l^d = 1, \quad \forall l = 1, \dots, k \quad (\text{App-1})$$

$$p_k = p_0 + \sum_{l=1}^k \sum_{d=1}^6 s_l^d \delta_d \quad (\text{App-2})$$

As we come back to the starting position after k steps, we must have $p_k = p_0$, hence from (App – 2) we have:

$$\sum_{l=1}^k \sum_{d=1}^6 s_l^d \delta_d = 0 \Rightarrow \begin{cases} \sum_{l=1}^k s_l^1 - s_l^4 = 0 \\ \sum_{l=1}^k s_l^2 - s_l^5 = 0 \\ \sum_{l=1}^k s_l^3 - s_l^6 = 0 \end{cases} \Rightarrow \begin{cases} \sum_{l=1}^k s_l^1 = \sum_{l=1}^k s_l^4 \\ \sum_{l=1}^k s_l^2 = \sum_{l=1}^k s_l^5 \\ \sum_{l=1}^k s_l^3 = \sum_{l=1}^k s_l^6 \end{cases} \quad (\text{App-3})$$

Finally, from (App – 1)- (App – 3) we have:

$$\sum_{d=1}^6 s_l^d = 1 \Rightarrow k = \sum_{l=1}^k \sum_{d=1}^6 s_l^d = \sum_{d=1}^6 \sum_{l=1}^k s_l^d = 2 \left(\sum_{l=1}^k s_l^1 + \sum_{l=1}^k s_l^2 + \sum_{l=1}^k s_l^3 \right) \Rightarrow k = 2q. \blacksquare$$

Appendix II

COLUMN GENERATION FOR BLA

Here, we propose a Column Generation (CG) method for solving the LP relaxation of BLA model in Chapter 3. Let us call BLA in the most basic form the original model. Let us define polytope χ as follows:

$$\chi = \{0 \leq X_p^k \leq 1, \quad k \in \mathcal{R}, p \in F_k \mid (BLA - 4), (BLA - 5)\} \quad (\text{App-4})$$

In BLA, the connectivity constraints $(BLA - 5)$ as well as $(BLA - 7)$ and $(BLA - 8)$ are coupling constraints that bundle location assignment variables of opposite parities. With absence of these constraints in χ , we can decompose χ into two independent polytopes, χ^θ for $\theta \in \{0,1\}$.

$$\chi^\theta = \{0 \leq X_p^k \leq 1, k \in \mathcal{R}^\theta, p \in F_k \mid \sum_{p \in F_k} X_p^k = 1, \forall k \in \mathcal{R}^\theta, \sum_{k \in \mathcal{R}^\theta, p \in F_k} X_p^k \leq 1, \forall p \in \mathcal{F}^\theta\}, \quad \forall \theta \in \{0,1\} \quad (\text{App-5})$$

Let J_θ denote the set of all extreme points of χ^θ , and $Y^{j,\theta}$ denote the j -th extreme point of J_θ , for $j = 1, \dots, |J_\theta|$. Hence, any point $X^\theta \in \chi^\theta$ can be expressed as a convex combination of these points, i.e.:

$$\forall X^\theta \in \chi^\theta, \quad \exists \lambda_j^\theta \geq 0 \text{ s.t. } X^\theta = \sum_{j=1}^{|J_\theta|} \lambda_j^\theta Y^{j,\theta}, \quad \sum_{j=1}^{|J_\theta|} \lambda_j^\theta = 1 \quad (\text{App-6})$$

Or equivalently,

$$X_{k,p}^\theta = \sum_{j=1}^{|J_\theta|} \lambda_j^\theta Y_{k,p}^{j,\theta}, \quad \forall \theta \in \{0,1\}, k \in \mathcal{R}^\theta, p \in F_k \quad (\text{App-7})$$

Therefore, the original model can be rewritten (in canonical minimization form) using these extreme points:

$$\text{minimize } \sum_{p \in \mathcal{F}^0} \sum_{d \in \delta: p+d \in \mathcal{F}^1} -W_p^d \quad (\text{App-8})$$

Subject to:

$$\sum_{j=1}^{|J_0|} \lambda_j^0 \left(\sum_{k \in \mathcal{H}^0: p \in F_k} Y_{k,p}^{j,0} \right) - W_p^d \geq 0, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1 \quad (\text{App-9})$$

$$\sum_{j=1}^{|J_1|} \lambda_j^1 \left(\sum_{k \in \mathcal{H}^1: p+d \in F_k} Y_{k,p+d}^{j,1} \right) - W_p^d \geq 0, \quad p \in \mathcal{F}^0, d \in \delta: p+d \in \mathcal{F}^1 \quad (\text{App-10})$$

$$\sum_{j=1}^{|J_\theta|} \lambda_j^\theta \left(\sum_{d \in \delta: p+d \in F_k} Y_{k,p+d}^{j,\theta} \right) - \sum_{j=1}^{|J_{1-\theta}|} \lambda_j^{1-\theta} (Y_{k+1,p}^{j,1-\theta} + Y_{k-1,p}^{j,1-\theta}) \geq 0, \quad (\text{App-11})$$

$$\theta \in \{0,1\}, k \in \mathcal{R}^\theta, p \in F_{k+1} \cup F_{k-1}$$

$$\sum_{j=1}^{|J_\theta|} \lambda_j^\theta = 1, \quad \theta \in \{0,1\} \quad (\text{App-12})$$

$$\lambda_j^\theta \geq 0, \quad \theta \in \{0,1\}, j = 1, \dots, |J_\theta| \quad (\text{App-13})$$

$$X_{k,p}^\theta = \sum_{j=1}^{|J_\theta|} \lambda_j^\theta Y_{k,p}^{j,\theta}, \quad \forall \theta \in \{0,1\}, k \in \mathcal{R}^\theta, p \in F_k \quad (\text{App-14})$$

$$X_{k,p}^\theta \in \{0,1\}, \quad \forall \theta \in \{0,1\}, k \in \mathcal{R}^\theta, p \in F_k \quad (\text{App-15})$$

We call this model the *Master Problem (MP)*. Note that in this model the variables are contact variables (W_p^d) and coefficients of extreme points of each polytope (λ_j^θ). However, we do not need to use all of these extreme points, and in the optimal solution only a large enough subset of these points will appear. Let J'_θ denote the large enough restricted set of extreme points. Replacing J_θ in MP with J'_θ , the model is called *Reduced Master Problem (RMP)*. CG is therefore, the procedure of generating the extreme point with least reduced cost using a *Pricing Subproblem (PS)*, and adding it to RMP. CG by itself can only be used as a method for solving LP models. In order to solve the LP relaxation of MP, we can relax constraints (App – 14) and (App – 15).

Let $\pi_{p,d}^1$, $\pi_{p,d}^2$, $\pi_{\theta,k,p}^3$, and π_θ^0 respectively denote the *dual variables* associated with (App – 9) to (App – 12), defined on the domain of each constraint set. Let PS^θ denote the PS associated with χ^θ , which is finding an extreme point in χ^θ with minimum reduced cost:

PS^0 :

$$\min_{X \in \chi^0} \left\{ - \sum_{p \in \mathcal{F}^0} \sum_{d \in \delta: p+d \in \mathcal{F}^1} \sum_{k \in \mathcal{H}^0: p \in F_k} \pi_{p,d}^1 X_p^k - \sum_{k \in \mathcal{R}^0} \sum_{p \in F_{k+1} \cup F_{k-1}} \sum_{d \in \delta: p+d \in F_k} \pi_{0,k,p}^3 X_{p+d}^k \right. \\ \left. + \sum_{k \in \mathcal{R}^1} \left(\sum_{p \in F_{k+1}} \pi_{1,k+1,p}^3 X_p^{k+1} + \sum_{p \in F_{k-1}} \pi_{1,k-1,p}^3 X_p^{k-1} \right) - \pi_0^0 \right\} \quad (\text{App-16})$$

PS^1 :

$$\min_{X \in \chi^1} \left\{ - \sum_{p \in \mathcal{F}^0} \sum_{\substack{d \in \delta: \\ p+d \in \mathcal{F}^1}} \sum_{\substack{k \in \mathcal{H}^1: \\ p \in F_k}} \pi_{p,d}^2 X_{p+d}^k - \sum_{k \in \mathcal{R}^1} \sum_{p \in F_{k+1} \cup F_{k-1}} \sum_{\substack{d \in \delta: \\ p+d \in F_k}} \pi_{1,k,p}^3 X_{p+d}^k \right. \\ \left. + \sum_{k \in \mathcal{R}^0} \left(\sum_{p \in F_{k+1}} \pi_{0,k+1,p}^3 X_p^{k+1} + \sum_{p \in F_{k-1}} \pi_{0,k-1,p}^3 X_p^{k-1} \right) - \pi_1^0 \right\} \quad (\text{App-17})$$

We keep solving PSs until the reduced costs associated with extreme points yielded by both PSs become nonnegative.

Strengthening the polytopes

We can add distance valid inequalities (*BLA* – 17) with parameter $l = 2$ to each polytope, without bundling variables of opposite parities.

$$X_p^{k+2} + X_p^{k-2} \leq \sum_{\substack{q \in F_k: \\ \Delta_{p,q}=2}} X_q^k, \quad \forall \theta \in \{0,1\}, k \in \mathcal{R}^\theta, p \in F_{k+2} \cup F_{k-2} \quad (\text{App-18})$$

Adding these constraints to PS^θ on one hand makes them computationally more difficult, but on the other hand it may make the extreme points obtained from each subproblem “*more likely*” to be feasible, hence reducing the MP time, and improving the runtime of the algorithm.

Appendix III

DETAILED PSEUDOCODES OF METAHEURISTIC ALGORITHMS

In this appendix we present the detailed pseudocodes of the metaheuristic algorithms introduced in Chapter 4.

Algorithm 2 Detailed pseudocode of Random Feasible Solution Generation

Input:

- w_p^k (Relative tendency of assigning amino acid k to lattice point p)
-

Output:

- P_k (Array of selected points of amino acid)
-

1. Set $k_R = m + 2, k_L = m - 1$
 2. Set $P_k = NULL$ for all k
 3. Set P_m and P_{m+1} to their fixed locations
 4. Set $D = 0$ (Direction of extension, 0 for leftward, and 1 for rightward)
 5. **While** $k_R < N$ **OR** $k_L \geq 0$ **Do**
 - 5.1. $D = 1 - D$
 - 5.2. **If** ($D = 0$ and $k_L = -1$) **OR** ($D = 1$ and $k_R = N$)
 - 5.2.1. Continue
 - 5.3. **End If**
 - 5.4. Set $k_C = D \times k_R + (1 - D) \times k_L$
 - 5.5. Let FL denote the set of all feasible locations for k_C
 - 5.6. Let UFL be all locations in FL that are not occupied by any amino acids
 - 5.7. **If** $UFL = \{\}$ (trapped in a dead-end)
 - 5.7.1. Let BC denote the set of amino acids that have occupied any of locations in FL
 - 5.7.2. **If** all amino acids in BC are greater than $m + 1$
 - 5.7.2.1. $BC = BC \cup \left\{ \frac{m+2+\min\{BC\}}{2} \right\}$
 - 5.7.3. **Else If** all amino acids in BC are less than m
 - 5.7.3.1. $BC = BC \cup \left\{ \frac{m+\max\{BC\}}{2} \right\}$
 - 5.7.4. **End If**
 - 5.7.5. Choose k_U randomly from BC
 - 5.7.6. **If** $k_U \geq m + 2$
 - 5.7.6.1. **For** l from $\frac{m+2+k_U}{2}$ to $N - 1$
 - 5.7.6.1.1. Set $P_l = NULL$
 - 5.7.6.2. **End For**
 - 5.7.6.3. Set $k_R = \frac{m+2+k_U}{2}$
 - 5.7.7. **Else**
 - 5.7.7.1. **For** l from 0 to $\frac{m+k_U}{2}$
 - 5.7.7.1.1. Set $P_l = NULL$
 - 5.7.7.2. **End For**
 - 5.7.7.3. Set $k_L = \frac{m+k_U}{2}$
 - 5.7.8. **End If**
 - 5.8. **Else**
 - 5.8.1. Choose lattice point p from UFL randomly, with probabilities proportional to $w_p^{k_C}$
-

```

5.8.2. Set  $P_{k_C} = p$ 
5.8.3.  $k_R = k_R + D$ 
5.8.4.  $k_L = k_L - 1 + D$ 
5.9. End If
6. End While

```

Algorithm 3 Detailed pseudocode of Iterative Biased Random Solution Generation

Inputs:

- w_p^k : initial tendency values
 - **UnchagnedLimit**: iterations we wait before updating the w_p^k values
 - ω : change in w_p^k
 - f : objective function
-

Output:

- P_k^* (Array of selected location of amino acids)
-
- ```

1. Generate P^* using RFSG, with w_p^k
2. Set $NonImproving = 0$ (consecutive non-improving iterations)
3. Set $Unchagned = 0$ (consecutive unchanged tendency values)
4. While stopping condition is not met Do
 4.1. Generate P using RFSG, with w_p^k
 4.2. If $f(P)$ is better than $f(P^*)$
 4.2.1. $P^* = P$
 4.2.2. $NonImproving = 0$
 4.2.3. $Unchagned = 0$
 4.3. Else
 4.3.1. $NonImproving = NonImproving + 1$
 4.3.2. $Unchagned = Unchagned + 1$
 4.3.3. If $Unchagned \geq UnchagnedLimit$
 4.3.3.1. For k from 0 to $N - 1$
 4.3.3.1.1. $w_{P_k^*}^k = w_{P_k^*}^k + \omega$
 4.3.3.2. End For
 4.3.3.3. $Unchagned = 0$
 4.3.4. End If
 4.4. End If
5. End While

```
- 

*Algorithm 4 Detailed pseudocode of Tabu Search*

---

**Inputs:**

- $P^*$ : Initial solution
  - $NS$ : Set of allowed neighborhood structures
  - $M_{kl}^0$ : Initial mobility value of pair  $kl$
  - $P_{noise}$ : Probability of temporarily change mobility to medium for one iteration
  - $E(P)$ : Objective function to be maximized
  - $LtM_{kl}$ : Number of iterations that pair  $kl$  must remain in medium mobility, when changed from low mobility.
  - $HtM_{kl}$ : Number of iterations that pair  $kl$  must remain in medium mobility, when changed from high mobility.
-

- 
- **Rec(P)**: A function that determines whether solution **P** should be recorded, based on the goodness of **P**.
- 

**Outputs:**

- $P^*$ : Improved solution
  - $f_{kl}^e$ : Recorded frequency of contact  $kl$  in objective function level  $e$
  - $S^e$ : Recorded solutions in objective function level  $e$
- 

1. Let  $M_{kl}^t$  denote the mobility of pair  $kl$  at iteration  $t$ , and set them to  $M_{kl}^0$  for of all pairs  $kl$ .
  2. Let  $D_{kl}$  denote the iteration in which mobility of pair  $kl$  must change, and set them to  $\infty$  for all pairs  $kl$ .
  3. Set  $f_{kl}^e = 0$  for all pairs  $kl$  and objective levels  $e$ .
  4. Set  $S^e = \{\}$  for all objective levels  $e$ .
  5. Set  $t = 0$  (iteration counter)
  6. Set  $P = P^*$  (the solution which the move will be performed on)
  7. Set  $NI = 0$  (number of non-improving iterations)
  8. **While** Stopping condition is not met **Do**
    - 8.1. **For** each pair  $kl$  where  $D_{kl} \leq t$ 
      - 8.1.1. **If**  $M_{kl}^t = \text{Medium}$ 
        - 8.1.1.1. Set  $M_{kl}^t = \text{High}$
        - 8.1.1.2. Set  $D_{kl} = \infty$
      - 8.1.2. **Else If**  $M_{kl}^t = \text{Low}$ 
        - 8.1.2.1. Set  $M_{kl}^t = \text{Medium}$
        - 8.1.2.2. Set  $D_{kl} = t + LtM_{kl}$
      - 8.1.3. **Else If**  $\text{rand} \leq P_{\text{noise}}$ 
        - 8.1.3.1. Set  $M_{kl}^t = \text{Medium}$
        - 8.1.3.2. Set  $D_{kl} = t + 1$
      - 8.1.4. **End If**
    - 8.2. **End For**
    - 8.3. Set  $P_t = \text{NULL}$  (best solution in the neighborhoods of  $P$ )
    - 8.4. **For** each *Neighborhood* in  $NS$ 
      - 8.4.1. Let  $V$  denote the set of amino acids that can initiate a move in *Neighborhood* over  $P$ .
      - 8.4.2. **For** each amino acid  $v$  in  $V$ 
        - 8.4.2.1. Let  $S$  denote the set of feasible solutions obtained by initiating the move by  $v$ .
        - 8.4.2.2. **For** each  $P'$  in  $S$ 
          - 8.4.2.2.1. **If**  $E(P') > E(P^*)$  ( $P'$  globally improves  $P^*$ )
            - 8.4.2.2.1.1.  $P_t = P'$
            - 8.4.2.2.1.2. Go To 8.6
          - 8.4.2.2.2. **Else If**  $P_t = \text{NULL}$  OR  $E(P') > E(P_t)$ 
            - 8.4.2.2.2.1. Let  $W$  denote the set of all amino acids whose contact with  $v$  has been altered (either formed a contact, or contact has been eliminated)
            - 8.4.2.2.2.2. **If**  $M_{vw}^t = \text{High}$  for some  $w$  in  $W$ 
              - 8.4.2.2.2.2.1. Let  $C$  denote the set of all pairs of amino acids whose contact has been altered
              - 8.4.2.2.2.2.2. **If**  $M_{kl}^t \neq \text{Low}$  for all contacts  $kl$  in  $C$ 
                - 8.4.2.2.2.2.2.1.  $P_t = P'$
        - 8.4.2.2.3. **End If**
-

---

```

 8.4.2.2.2.3. End If
 8.4.2.2.3. End If
 8.4.2.3. End For
8.4.3. End For
8.5. End For
8.6. If $P_t \neq NULL$
 8.6.1. If $Rec(P_t)$
 8.6.1.1. $S^{E(P_t)} = S^{E(P_t)} \cup \{P_t\}$
 8.6.1.2. For each pair kl forming a contact in P_t
 8.6.1.2.1. $f_{kl}^{E(P_t)} = f_{kl}^{E(P_t)} + 1$
 8.6.1.3. End For
 8.6.2. End If
 8.6.3. If $E(P_t) > E(P^*)$ (global improvement)
 8.6.3.1. $P^* = P = P_t$
 8.6.3.2. $NI = 0$
 8.6.3.3. Set M_{kl}^t to M_{kl}^0 (initial mobility state of kl) or Low for selected
 good contacts in Trainable Tabu Search.
 8.6.4. Else
 8.6.4.1. $NI = NI + 1$
 8.6.4.2. Let C denote the set of all pairs of amino acids with different
 contact values in P and P_t
 8.6.4.3. For each pair kl in C where $M_{kl}^t = High$
 8.6.4.3.1. $M_{kl}^t = Medium$
 8.6.4.3.2. Set $D_{kl} = t + HtM_{kl}$
 8.6.4.4. End For
 8.6.4.5. $P = P_t$
 8.6.5. End If
8.7. Else (No valid move was detected)
 8.7.1. Let $j = \min\{D_{kl} | M_{kl}^t \neq High\} - t$
 8.7.2. Set $t = t + j$ (skip the next j iterations)
 8.7.3. Set $NI = NI + j$
8.8. End If
9. End While

```

---

*Algorithm 5 Detailed pseudocode of Scatter Search*

---

**Inputs:**

- **$PSize$** : Initial pool size
  - **$B_1$** : Number of high quality solutions in  **$RS$**
  - **$B_2$** : Number of diverse solutions in  **$RS$**
  - **$f$** : Objective function to be maximized
- 

**Output:**

- Best solution found by Scatter Search method
- 

1. Generate solution  $P$  using RFSG method.
  2. Improve  $P$  using Improvement routine, and obtain  $P^*$
  3. Set  $P_0 = \{P^*\}$  (initial pool of solutions)
  4. **While**  $|P_0| < PSize$  **Do**:
    - 4.1. Generate solution  $P$  using IB-RFSG method, biased towards maximizing  $Div(P, P_0)$ .
-

- 
- 4.2. Improve  $P$  using Improvement routine, and obtain  $P^*$
  - 4.3. Set  $P_0 = P_0 \cup \{P^*\}$
  5. **End While**
  6. Let  $RS$  (reference set) be the best  $B_1$  solutions in  $P_0$
  7. Set  $P_0 = P_0 - RS$
  8. **While**  $|RS| < B_1 + B_2$  **Do**:
    - 8.1. Choose the solution  $P$  in  $P_0$  with highest  $Div(P, RS)$
    - 8.2. Set  $RS = RS \cup \{P\}$
    - 8.3. Set  $P_0 = P_0 - \{P\}$
  9. **End While**
  10. Set  $RefSetUpdated = TRUE$
  11. **While**  $RefSetUpdated$  **Do**:
    - 11.1. Set  $RefSetUpdated = FALSE$ 
      - 11.1.1. Generate *Subsets* using Subset Generation routine
      - 11.1.2. **For** each set  $S$  in *Subsets* **Do**:
        - 11.1.2.1. Combine solutions in  $S$  using Solution Combination routine and obtain  $P$ .
        - 11.1.2.2. Improve  $P$  using Improvement routine, and obtain  $P^*$
        - 11.1.2.3. **If**  $P^* \notin RS$ 
          - 11.1.2.3.1. Set  $Q^* = \underset{Q \in RS}{\operatorname{argmin}} E(Q)$
          - 11.1.2.3.2. **If**  $E(P^*) > E(Q^*)$ 
            - 11.1.2.3.2.1. Set  $RS = (RS - \{Q^*\}) \cup \{P^*\}$
            - 11.1.2.3.2.2. Set  $RefSetUpdated = TRUE$
          - 11.1.2.3.3. **End If**
        - 11.1.2.4. **End If**
      - 11.1.3. **End For**
      - 11.2. **End For**
    12. **End While**

---

### Appendix IV PARTITIONING A SET OF SOLUTIONS

Here we present the detailed and mathematical explanation of partitioning a set of solutions into mutually exclusive subsets as explained in Section 4.3 of Chapter 4, such that the most similar solutions are assigned to the same partitions. Let  $R = \{X^1, \dots, X^{|R|}\}$  and  $P$  respectively denote the set of solutions and set of partitions. Let  $p_k$  denote the capacity of partition  $k$  for  $k \in P$ . Having defined distance ( $D$ ) and similarity ( $S$ ) measures as in Scatter Search, we can partition  $R$  into subsets  $P$  of least distant and most similar solutions. We will show that partitioning based on similarity measure is equivalent to partitioning with respect to distance measure. Let us define binary variables  $Y_{ik}$  which are 1, if  $X^i$  is assigned to partition  $k$ , and 0 otherwise. Let us define binary variables  $Z_{ij}$ , which are 1, if  $X^i$  and  $X^j$  belong to the same partition, and 0 otherwise. We want to partition  $R$ , such that total inter-distance in partitions is minimized, i.e.:

$$\text{minimize} \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} Z_{ij} D(X^i, X^j) \quad (\text{App-19})$$

Subject to:

$$Z_{ij} \geq Y_{ik} + Y_{jk} - 1, \quad \forall i < j, k \in P \quad (\text{App-20})$$

$$\sum_{k \in P} Y_{ik} = 1, \quad \forall i = 1, \dots, |R| \quad (\text{App-21})$$

$$\sum_{i=1}^{|R|} Y_{ik} = p_k, \quad \forall k \in P \quad (\text{App-22})$$

$$Y_{ik} \in \{0,1\}, \quad \forall k \in P, i = 1, \dots, |R| \quad (\text{App-23})$$

Now, let us define  $W_{ij} = 1 - Z_{ij}$ , which is 1, if  $X^i$  and  $X^j$  belong to different partitions, and 0 otherwise. Replacing  $Z$  and  $D$  with their equivalent values, (App – 19) can be rewritten as:

$$\begin{aligned} \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} Z_{ij} D(X^i, X^j) &= \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} (1 - W_{ij}) (N - S(X^i, X^j)) \quad (\text{App-24}) \\ &= \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} (N - S(X^i, X^j)) - N \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} W_{ij} \\ &\quad + \sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} W_{ij} S(X^i, X^j) \end{aligned}$$

Note that  $\sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} W_{ij}$  is constant for complete graphs, and is equal to  $\frac{1}{2} \sum_{k \in P} p_k (|R| - p_k) = \frac{1}{2} (|R|^2 - \sum_{k \in P} p_k^2)$ . It can be shown that replacing (App – 20) with  $W_{ij} \geq X_i^k - X_j^k$  and  $W_{ij} \geq X_j^k - X_i^k$  is enough to guarantee feasibility of partitions. Hence, model (App – 19)-(App – 23) can be rewritten in terms of similarities as (App – 25)-(App – 30):

$$\text{minimize } \sum_{i < j} W_{ij} S(V_i, V_j) \quad (\text{App-25})$$

Subject to:

$$W_{ij} \geq Y_{ik} - Y_{jk}, \quad \forall i < j, k \in P \quad (\text{App-26})$$

$$W_{ij} \geq Y_{jk} - Y_{ik}, \quad \forall i < j, k \in P \quad (\text{App-27})$$

$$\sum_{k \in P} Y_{ik} = 1, \quad \forall i = 1, \dots, |R| \quad (\text{App-28})$$

$$\sum_{i=1}^{|R|} Y_{ik} = p_k, \quad \forall k \in P \quad (\text{App-29})$$

$$Y_{ik} \in \{0,1\}, \quad \forall k \in P, i = 1, \dots, |R| \quad (\text{App-30})$$

## Appendix V

### TUNING METAHEURISTIC ALGORITHMS

In this appendix we present our computational experiments used to tune the parameters of metaheuristic algorithms introduced in Chapter 4. In order to tune the parameters of Tabu Search, we run TS under variety of combinations of its parameters. The parameters that we aim to optimize are as follows:

- $P_{noise}$ : Probability of changing mobility pairs with high mobility to medium for one iteration.
- $minDiv$ : If the diversity of the algorithm (as defined in Lesh et al. (2003)) falls below this threshold, then we change the mobility of elements with diversity less than this threshold to medium and keep them at this state for one iteration.
- Choice of moves: Combinations of pull, corner, end, and crankshaft moves.
- $HtM_{kl}$ : The number of iterations that the pair  $kl$  stays at mobility state medium, when changed from high mobility state.

We optimize these parameters using four benchmark instances S1-4 (short), S1-6 (medium), and S1-8 (long). For optimizing  $P_{noise}$  we choose values 0 (off), 0.1 (low probability), and 0.3 (moderate probability). Similarly, we choose values 0, 0.1, and 0.3 as possible choices of  $minDiv$ . We choose possible moves from combinations pull, pull+corner, pull+corner+ened, and pull+corner+end+crankshaft. We abbreviate these sets of moves as P, PC, PCE, and PCECr, respectively. Finally, we use similar values of  $HtM_{kl}$  for all of pairs, and choose it from one the values 5, 10, 20, 35, 50, 100.

For each choice of these parameters, i.e.  $3 \times 3 \times 4 \times 6(7)$  different combinations, we run the algorithm over five randomly generated solutions of each benchmark instance, and report the maximum and average value obtained from each of these sets. We should note that for each benchmark instance, and for each choice of parameters we generate and use the streams of random numbers to reduce the effect of randomness in assessing the performance of the algorithm under different settings. Before presenting the results, we should mention that our algorithm shows negligible sensitivity with respect to the different choices of  $minDiv$ , hence in the following we assume that it is equal to zero.

#### Tuning parameters with respect to S1-4

As shown in Table 7-1 to Table 7-3, there seems to exist a positive correlation between  $P_{noise}$  and average quality of solutions. As we increase  $P_{noise}$  from 0 to 0.3, the average objective value improves in almost all of the combinations of HtM and choices of moves. Therefore, we further conduct additional runs with  $P_{noise}$  equal to 0.5 to see if this value still helps improving the overall performance of the algorithm. As results shown in

Table 7-4 imply, increasing  $P_{noise}$  above **0.3** starts to reduce the overall quality. Moreover, we observe a jump in the average running times from 1.26 seconds to 2.59 seconds. Therefore we stop at this point, and assume that 0.3 is the optimal value for this parameter.

As bolded in the tables, the choice of HtM equal to **10** seems to be the best option for this instance. Finally, any of choices PCE and PCECr for the neighborhood structures seem to be equally preferable. To choose one of these two choice, we consider the average running time of each one. PCE on average takes 1.36 seconds to terminate, whereas this period is 2.76 for PCECr, therefore we choose not to include crankshaft in the moves.

### Tuning parameters with respect to S1-6

Table 7-5 to Table 7-8 correspond to the tests performed on benchmark instance S1-6. Similar to S1-4, a positive correlation between  $P_{noise}$  and average quality of solutions is observable in these instances of the problem. We follow the same procedure we applied for optimizing the parameters. As some of the solutions tend to improve while increasing  $P_{noise}$ , we further conduct additional runs with  $P_{noise}$  equal to 0.5 to see if this value still helps improving the overall performance of the algorithm. Table 7-8 however indicates that, similar to S1-4, increasing  $P_{noise}$  above 0.3 starts to reduce the overall quality. Therefore we stop at this point, and assume that either **0.1** the optimal value for this parameter. As bolded in the tables, the choice of HtM equal to **10** seems to be the best option for this parameter. Finally, PCECr seems to be the best choice for the neighborhood structures.

Table 7-1 Performance of TS on S1-4, with  $P_{noise} = 0$

| $ E^*  = 14$ | HtM   |     |           |             |             |           |           |
|--------------|-------|-----|-----------|-------------|-------------|-----------|-----------|
|              | Moves | 5   | 10        | 20          | 35          | 50        | 100       |
| Average      | P     | 10  | 12        | 11.8        | 11.2        | 11.2      | 11.2      |
|              | PC    | 9.8 | 11.6      | 12.4        | 11.6        | 11.8      | 11.8      |
|              | PCE   | 9   | 12.2      | 11.8        | <b>12.8</b> | 12.2      | 11.6      |
|              | PCECr | 8.6 | 12.2      | <b>12.8</b> | 11.8        | 11.4      | 11.4      |
| Max          | P     | 12  | 12        | <b>14</b>   | 12          | 12        | 12        |
|              | PC    | 13  | 12        | 13          | 13          | 13        | 13        |
|              | PCE   | 10  | <b>14</b> | 13          | <b>14</b>   | <b>14</b> | <b>14</b> |
|              | PCECr | 10  | <b>14</b> | 13          | 13          | 13        | 13        |

Table 7-2 Performance of TS on S1-4, with  $P_{noise} = 0.1$ 

| $ E^*  = 14$ | HtM   |           |             |      |           |      |      |
|--------------|-------|-----------|-------------|------|-----------|------|------|
|              | Moves | 5         | 10          | 20   | 35        | 50   | 100  |
| Average      | P     | 12.6      | 12.2        | 12.6 | 12.6      | 11.2 | 10.6 |
|              | PC    | 11.6      | 12.8        | 12   | 11.8      | 11.4 | 11   |
|              | PCE   | 10.6      | <b>13.4</b> | 12.8 | 12.2      | 12.4 | 11.6 |
|              | PCECr | 10.8      | <b>13.4</b> | 12   | 11.2      | 12   | 11.4 |
| Max          | P     | <b>14</b> | 13          | 13   | <b>14</b> | 13   | 12   |
|              | PC    | 13        | 13          | 13   | 13        | 12   | 11   |
|              | PCE   | 12        | <b>14</b>   | 13   | <b>14</b> | 13   | 13   |
|              | PCECr | 13        | <b>14</b>   | 13   | 12        | 13   | 12   |

Table 7-3 Performance of TS on S1-4, with  $P_{noise} = 0.3$ 

| $ E^*  = 14$ | HtM   |           |             |           |           |      |      |
|--------------|-------|-----------|-------------|-----------|-----------|------|------|
|              | Moves | 5         | 10          | 20        | 35        | 50   | 100  |
| Average      | P     | 13.2      | 13.2        | 12.6      | 11.8      | 12   | 11   |
|              | PC    | 11.4      | <b>13.4</b> | 12.6      | 12.4      | 11   | 11.  |
|              | PCE   | 11.8      | <b>13.4</b> | 12.6      | 12.2      | 12.8 | 11.8 |
|              | PCECr | 11.8      | <b>13.4</b> | 12.4      | 12        | 11   | 12.2 |
| Max          | P     | <b>14</b> | <b>14</b>   | <b>14</b> | 12        | 13   | 12   |
|              | PC    | 13        | <b>14</b>   | 13        | <b>14</b> | 12   | 13   |
|              | PCE   | 12        | <b>14</b>   | 13        | 13        | 12   | 13   |
|              | PCECr | <b>14</b> | <b>14</b>   | 13        | <b>14</b> | 13   | 13   |

Table 7-4 Performance of TS on S1-4, with  $P_{noise} = 0.5$ 

| $ E^*  = 14$ | HtM   |      |      |      |      |
|--------------|-------|------|------|------|------|
|              | Moves | 5    | 10   | 20   | 35   |
| Average      | P     | 13   | 12   | 11   | 11.8 |
|              | PC    | 13   | 12.4 | 11.4 | 10.6 |
|              | PCE   | 13.4 | 11.8 | 11.4 | 10.6 |
|              | PCECr | 12.4 | 11.4 | 11   | 10.4 |
| Max          | P     | 14   | 13   | 13   | 13   |
|              | PC    | 14   | 13   | 12   | 11   |
|              | PCE   | 14   | 12   | 12   | 12   |
|              | PCECr | 13   | 12   | 12   | 11   |

Table 7-5 Performance of TS on S1-6, with  $P_{noise} = 0$ 

| $ E^*  = 21$ | HtM   |      |      |             |      |      |      |
|--------------|-------|------|------|-------------|------|------|------|
|              | Moves | 5    | 10   | 20          | 35   | 50   | 100  |
| Average      | P     | 12.2 | 16.2 | 17.4        | 16.4 | 16   | 16   |
|              | PC    | 13.2 | 17.2 | 17          | 16.2 | 16.6 | 16.2 |
|              | PCE   | 13.4 | 15.2 | <b>18.2</b> | 17.6 | 17.4 | 16.8 |
|              | PCECr | 13.8 | 15   | 17.6        | 17.2 | 16.8 | 16.8 |
| Max          | P     | 14   | 18   | 18          | 17   | 17   | 17   |
|              | PC    | 15   | 19   | 18          | 17   | 18   | 18   |
|              | PCE   | 15   | 18   | 19          | 18   | 18   | 18   |
|              | PCECr | 15   | 18   | 19          | 18   | 18   | 18   |

Table 7-6 Performance of TS on S1-6, with  $P_{noise} = 0.1$ 

| $ E^*  = 21$ | HtM   |      |             |      |      |    |      |
|--------------|-------|------|-------------|------|------|----|------|
|              | Moves | 5    | 10          | 20   | 35   | 50 | 100  |
| Average      | P     | 16.6 | 18          | 18   | 17   | 17 | 15.8 |
|              | PC    | 15.6 | 17.2        | 17.8 | 17.2 | 17 | 15.4 |
|              | PCE   | 15.2 | 18.8        | 17.8 | 17.6 | 17 | 16.6 |
|              | PCECr | 15.6 | <b>19.2</b> | 18.8 | 17.6 | 17 | 16.8 |
| Max          | P     | 17   | 19          | 19   | 17   | 19 | 17   |
|              | PC    | 17   | 18          | 19   | 19   | 18 | 17   |
|              | PCE   | 18   | 19          | 19   | 18   | 19 | 18   |
|              | PCECr | 19   | 20          | 19   | 18   | 18 | 18   |

Table 7-7 Performance of TS on S1-6, with  $P_{noise} = 0.3$ 

| $ E^*  = 21$ | HtM   |      |      |           |      |      |      |
|--------------|-------|------|------|-----------|------|------|------|
|              | Moves | 5    | 10   | 20        | 35   | 50   | 100  |
| Average      | P     | 17.2 | 17.8 | 16.8      | 16.8 | 16.8 | 15.8 |
|              | PC    | 15.6 | 17.8 | 18.6      | 17.8 | 17.2 | 15.8 |
|              | PCE   | 17.2 | 17.6 | 18.4      | 17.6 | 17.4 | 17   |
|              | PCECr | 17.4 | 18.4 | 18.6      | 17.6 | 17.6 | 16.8 |
| Max          | P     | 18   | 19   | 18        | 17   | 18   | 17   |
|              | PC    | 17   | 19   | <b>21</b> | 19   | 18   | 16   |
|              | PCE   | 19   | 18   | 19        | 18   | 19   | 18   |
|              | PCECr | 20   | 20   | 20        | 18   | 19   | 18   |

Table 7-8 Performance of TS on S1-6, with  $P_{noise} = 0.5$ 

| $ E^*  = 21$ | HtM   |      |      |      |      |
|--------------|-------|------|------|------|------|
|              | Moves | 5    | 10   | 20   | 35   |
| Average      | P     | 17.8 | 17.4 | 16.6 | 15.4 |
|              | PC    | 17.8 | 17.6 | 16.4 | 15.6 |
|              | PCE   | 17.8 | 17.6 | 16.2 | 16.2 |
|              | PCECr | 18.8 | 18.2 | 16.6 | 15.8 |
| Max          | P     | 18   | 19   | 17   | 17   |
|              | PC    | 18   | 19   | 17   | 17   |
|              | PCE   | 19   | 18   | 17   | 17   |
|              | PCECr | 20   | 19   | 18   | 17   |

Table 7-9 Performance of TS on S1-8, with  $P_{noise} = 0$ 

| $ E^*  = 42$ | HtM   |      |      |      |      |      |      |
|--------------|-------|------|------|------|------|------|------|
|              | Moves | 5    | 10   | 20   | 35   | 50   | 100  |
| Average      | P     | 24.4 | 30.2 | 32.6 | 32.4 | 32.2 | 32.2 |
|              | PC    | 23.8 | 34   | 33.2 | 32.8 | 32.2 | 32.8 |
|              | PCE   | 23.8 | 32.4 | 34.4 | 33.4 | 31.8 | 31.6 |
|              | PCECr | 23.6 | 32.2 | 33.4 | 34   | 32   | 30   |
| Max          | P     | 27   | 34   | 35   | 35   | 35   | 38   |
|              | PC    | 27   | 37   | 36   | 36   | 35   | 35   |
|              | PCE   | 27   | 38   | 36   | 35   | 36   | 32   |
|              | PCECr | 26   | 36   | 36   | 36   | 35   | 32   |

Table 7-10 Performance of TS on S1-8, with  $P_{noise} = 0.1$ 

| $ E^* $<br>= 42 | HtM   |      |           |      |      |      |      |
|-----------------|-------|------|-----------|------|------|------|------|
|                 | Moves | 5    | 10        | 20   | 35   | 50   | 100  |
| Average         | P     | 36   | <b>37</b> | 35.4 | 34.6 | 34.2 | 32   |
|                 | PC    | 32.8 | 35.8      | 34.6 | 33.8 | 33.6 | 30.8 |
|                 | PCE   | 32.6 | 33.4      | 33   | 32.2 | 32.6 | 30.4 |
|                 | PCECr | 28.8 | 32.4      | 32.8 | 31.8 | 30.4 | 30.4 |
| Max             | P     | 40   | 40        | 38   | 38   | 38   | 36   |
|                 | PC    | 36   | 41        | 37   | 37   | 38   | 33   |
|                 | PCE   | 39   | 35        | 37   | 34   | 37   | 33   |
|                 | PCECr | 31   | 34        | 35   | 36   | 32   | 33   |

Table 7-11 Performance of TS on S1-8, with  $P_{noise} = 0.3$ 

| $ E^* $ = 42 | HtM   |      |      |             |      |      |      |
|--------------|-------|------|------|-------------|------|------|------|
|              | Moves | 5    | 10   | 20          | 35   | 50   | 100  |
| Average      | P     | 34.8 | 36   | <b>36.8</b> | 34.8 | 33.6 | 31.4 |
|              | PC    | 33.6 | 35.4 | 36.2        | 33.6 | 33.4 | 31.4 |
|              | PCE   | 30.2 | 34   | 34.8        | 33.6 | 33   | 31.2 |
|              | PCECr | 29.6 | 33.4 | 34.4        | 32.6 | 32.2 | 29.8 |
| Max          | P     | 37   | 40   | 40          | 38   | 38   | 34   |
|              | PC    | 35   | 40   | 40          | 35   | 36   | 37   |
|              | PCE   | 35   | 40   | 38          | 36   | 37   | 33   |
|              | PCECr | 35   | 35   | 38          | 34   | 35   | 33   |

## Appendix VI

### MOST DISTANT OBJECTS IN A CUBE

As discussed in Section 6.3 of Chapter 6, when considering all 20 different amino acids, we need distinguishable colors for illustrating the solutions. On the other hand we have the background and link colors which are fixed. For the sake of ease of distinguishability, we need the 20 colors to be as different from each other and from the fixed colors as possible. Since every color is characterized by three elements- Red, Green, and Blue, we can think of this problem as a problem of locating  $20 + 2$  points in a  $255 \times 255 \times 255$  cube, such that the minimum distance between every pair of freely-moving points or between freely moving points and the fixed points is maximized. Despite this simple description, solving this problem to optimality is computationally expensive.

Let  $N_C$  and  $N_F$  denote the number of free and fixed colors, respectively. Let  $C_k^d$  and  $F_f^d$  denote the  $d$ -th dimension of  $k^{th}$  free color, and  $f^{th}$  fixed color, respectively, where  $d \in \{R, G, B\}, k = 1, \dots, N_C, f = 1, \dots, N_F$ . Let  $md$  denote the minimum distance between all colors, we want colors to be chosen such that  $md$  is maximized.

$$\text{maximize } md \quad (\text{App-31})$$

Subject to:

$$md \leq \sum_{d \in \{R, G, B\}} |C_k^d - C_l^d|, \quad \forall k, l = 1, \dots, N_C, k < l \quad (\text{App-32})$$

$$md \leq \sum_{d \in \{R, G, B\}} |C_k^d - F_f^d|, \quad \forall k = 1, \dots, N_C, f = 1, \dots, N_F \quad (\text{App-33})$$

$$0 \leq C_k^d \leq 255, \quad \forall k = 1, \dots, N_C, d \in \{R, G, B\} \quad (\text{App-34})$$

To linearize this model, we define the following variables:

$$DC_{kl}^d = |C_k^d - C_l^d|, \quad k, l = 1, \dots, N_C, k < l, d \in \{R, G, B\} \quad (\text{App-35})$$

$$SDC_{kl}^d = \begin{cases} 1, & C_k^d \geq C_l^d \\ 0, & C_k^d \leq C_l^d \end{cases}, \quad k, l = 1, \dots, N_C, k < l \quad (\text{App-36})$$

$$DF_{kl}^d = |C_k^d - F_f^d|, \quad k = 1, \dots, N_C, f = 1, \dots, N_F, d \in \{R, G, B\} \quad (\text{App-37})$$

$$SDF_{kf}^d = \begin{cases} 1, & C_k^d \geq F_f^d \\ 0, & C_k^d \leq F_f^d \end{cases}, \quad k = 1, \dots, N_C, f = 1, \dots, N_F, d \in \{R, G, B\} \quad (\text{App-38})$$

The problem is hence reformulated as:

$$\text{maximize } md \quad (\text{App-39})$$

Subject to:

$$md \leq \sum_{d \in \{R, G, B\}} DC_{kl}^d, \quad \forall k, l = 1, \dots, N_C, k < l \quad (\text{App-40})$$

$$md \leq \sum_{d \in \{R, G, B\}} DF_{kl}^d, \quad \forall k = 1, \dots, N_C, f = 1, \dots, N_F \quad (\text{App-41})$$

$$DC_{kl}^d \leq C_k^d - C_l^d + 2 \times 255 \times SDC_{kl}^d, \quad \forall k, l = 1, \dots, N_C, k < l, d \in \{R, G, B\} \quad (\text{App-42})$$

$$DC_{kl}^d \leq C_l^d - C_k^d + 2 \times 255 \times (1 - SDC_{kl}^d), \quad \forall k, l = 1, \dots, N_C, k < l, d \in \{R, G, B\} \quad (\text{App-43})$$

$$DF_{kl}^d \leq C_k^d - F_f^d + 2 \times 255 \times SDF_{kf}^d, \quad \forall k = 1, \dots, N_C, f = 1, \dots, N_F, d \in \{R, G, B\} \quad (\text{App-44})$$

$$DF_{kl}^d \leq F_f^d - C_k^d + 2 \times 255 \times (1 - SDF_{kf}^d), \quad \forall k = 1, \dots, N_C, f = 1, \dots, N_F, d \in \{R, G, B\} \quad (\text{App-45})$$

$$SDC_{kl}^d \in \{0, 1\}, \quad \forall k, l = 1, \dots, N_C, k < l, d \in \{R, G, B\} \quad (\text{App-46})$$

$$0 \leq C_k^d \leq 255, \quad \forall k = 1, \dots, N_C, d \in \{R, G, B\} \quad (\text{App-47})$$

With this formulation even finding the optimal solution for 8 free colors and 0 fixed colors takes more than 3hrs, let alone solving for 20 free and 2 fixed colors. One way to tackle this problem is to equally split the  $d$ -th dimension into  $S_d$  sections, hence partitioning the cube into  $\prod_{d \in \{R, G, B\}} S_d$  identical sub-cubes, such that  $\prod_{d \in \{R, G, B\}} S_d \leq N_C - 1$ . According to the *pigeonhole principle*, in this way there must be a sub-cube which contains at least two points, and the maximum distance in that cube can be  $\sum_{d \in \{R, G, B\}} \frac{255}{S_d}$ . Note that since this value is always an upper bound on  $md$ , we want to find  $S_d$  values that minimize  $\sum_{d \in \{R, G, B\}} \frac{255}{S_d}$ . Hence, first we should solve the following problem:

$$\min \sum_{d \in \{R, G, B\}} \frac{255}{S_d} \quad (\text{App-48})$$

$$s. t. \quad \prod_{d \in \{R, G, B\}} S_d \leq N_C - 1 \quad (\text{App-49})$$

$$S_d \in \mathbb{N}, \quad \forall d \in \{R, G, B\} \quad (\text{App-50})$$

Note that if we relax the integrality constraints, setting  $y_d = \frac{1}{S_d}$ , this problem can be transformed to the model (App – 51) – (App – 52), which is a concave minimization problem. It is not hard to show that the optimal solution when the integrality constraint is relaxed is  $S_d = \frac{1}{y_d} = \sqrt[3]{N_C - 1}$ . For instance, if  $N_C = 20$ , then  $\frac{1}{y_d} = 2.668$ , and the integer solution is  $S_1 = S_2 = 3$  and  $S_3 = 2$ , and the upper bound on  $md$  is therefore  $255 \left( \frac{1}{3} + \frac{1}{3} + \frac{1}{2} \right) = 297.5$ .

$$\min 255 \sum_{d \in \{R, G, B\}} y_d \quad (\text{App-51})$$

$$s. t. \quad \prod_{d \in \{R, G, B\}} y_d \geq \frac{1}{N_C - 1} \quad (\text{App-52})$$

Moreover, in this way we can restrict points to sub-cubes and easily determine the values of distance identifier binaries. Moreover, in this way we can neglect the distances between

the sub-cubes that are not directly or diagonally adjacent. Figure 7-1 shows the solution found when we set  $S_d = 2$ , for  $N_C = 20$  and  $N_F = 0$ .

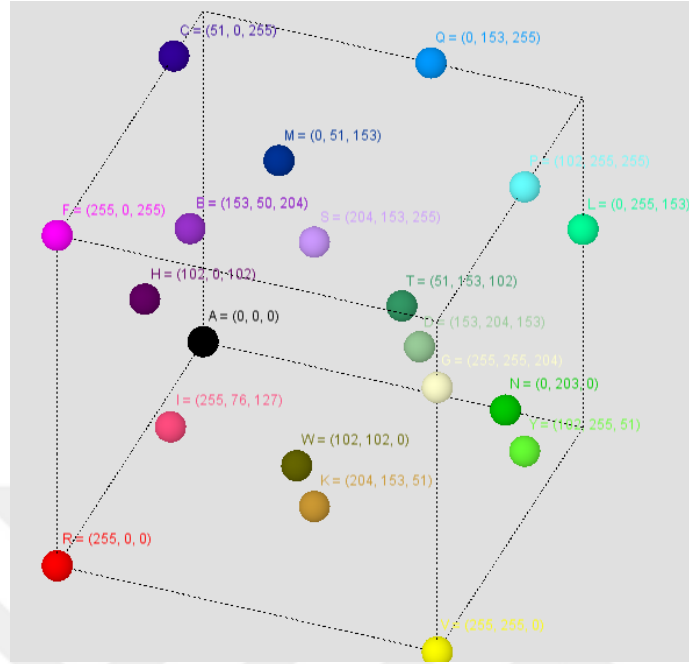


Figure 7-1 Most distant objects in a cube of side length 255, analogous to most distinguishable colors

This method yields the optimal solution in a short time, if we have free points only. However, we must note that in case of having fixed points, this method might give a local optimum. To tackle this problem, noting that the fixed points are also meant to be as different, we can allow the fixed points to freely move, find the optimal solution accordingly, and finally assign the closest points to the real fixed points. Let  $\bar{C}_k^d$  be an optimal solution to the relaxed problem, for  $k = 1, \dots, (N_C + N_F)$  and  $d \in \{R, G, B\}$ . The following simple assignment model solves the second stage problem:

$$X_f^k = \begin{cases} 1, & \text{the } k^{\text{th}} \text{ color is fixed as } f^{\text{th}} \text{ original fixed color,} \\ 0, & \text{otherwise} \end{cases}, \quad (\text{App-53})$$

$$\min \sum_{k=1}^{N_C+N_F} \sum_{f=1}^{N_F} X_f^k \sum_{d \in \{R,G,B\}} |\bar{C}_k^d - F_f^d| \quad (\text{App-54})$$

Subject to:

$$\sum_{k=1}^{N_C+N_F} X_f^k = 1, \quad \forall f = 1, \dots, N_F \quad (\text{App-55})$$

$$\sum_{f=1}^{N_F} X_f^k \leq 1, \quad \forall k = 1, \dots, N_C + N_F \quad (\text{App-56})$$

$$X_f^k \in \{0,1\}, \quad \forall k = 1, \dots, N_C + N_F, f = 1, \dots, N_F \quad (\text{App-57})$$

## BIBLIOGRAPHY

- Aichholzer, O., Bremner, D., Demaine, E. D. & Meijer, H., 2003. Long proteins with unique optimal foldings in the H-P model. *Computational Geometry*, 25(1), pp. 139-159.
- Anfinsen, C. B., 1973. Principles that govern the folding of protein chains. *Science*, 181(1), pp. 223-230.
- Backofen, R. & Will, S., 2006. A Constraint-Based approach to fast and exact structure prediction in three-dimensional protein models. *Constraints*, 11(1), pp. 5-30.
- Backofen, R., Will, S. & Bornberg-Bauer, E., 1999. Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics*, 15(3), pp. 234-242.
- Bauer, E. B., 1997. Chain growth algorithms for HP-type lattice proteins. *Proceedings of the First Annual International Conference on Computational Molecular Biology*, pp. 47-55.
- Berger, B. & Leighton, T., 1998. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1), pp. 27-40.
- Berrera, M., Molinari, H. & Fogolari, F., 2003. Amino acid empirical contact energy definitions for fold recognition in the space of contact maps. *BMC Bioinformatics*, 4(1), pp. 8-26.
- Beutler, T. C. & Dill, K. A., 1996. A fast conformational search strategy for finding low energy structures of model proteins. *Protein Science*, 5(10), pp. 2037-2043.
- Blazewicz, J., Dill, K. A., Lukasiak, P. & Milostan, M., 2004. A tabu search strategy for finding low energy structures of proteins in HP-model. *Computational Methods in Science and Technology*, 10(1), pp. 7-19.
- Campos, V., Glover, F., Laguna, M. & Martí, R., 2001. An experimental evaluation of a scatter search for the linear ordering problem. *Journal of Global Optimization*, 21(4), pp. 397-414.
- Carr, R., Hart, W. E. & Newman, A., 2002. Discrete optimization models for protein folding, Technical report, Sandia National Laboratories.
- Dal Palu, A., Dovier, A. & Fogolari, F., 2004. Constraint Logic Programming approach to protein structure prediction. *BMC Bioinformatics*, 5(1), pp. 186-198.
- Dill, K. A., 1985. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6), pp. 1501-1509.

- Dill, K. A., 1990. Dominant forces in protein folding. *Biochemistry*, 29(31), pp. 7133-7155.
- Dill, K.A., Bromberg, S., Yue, K., Chan, H.S., Ftebig, K.M., Yee, D.P. and Thomas, P.D., 1995. Principles of protein folding: A perspective from simple exact models. *Protein Science*, 4(4), pp. 561-602.
- Dill, K. A., Fiebig, K. M. & Chan, H. S., 1993. Cooperativity in protein-folding kinetics. *Proceedings of the National Academy of Sciences*, 90(5), pp. 1942-1946.
- Dittel, A., Fügenschuh, A. & Martin, A., 2011. Polyhedral aspects of self-avoiding walks. Technical Report ZR 11-11, Berlin: Zuse Institute.
- Glover, F., 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1), pp. 156-166.
- Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5), pp. 533-549.
- Glover, F., 1998. A template for scatter search and path relinking. *Lecture Notes in Computer Science*, Volume 1363, pp. 13-54.
- Glover, F., Løkketangen, A. & Woodruff, D. L., 2000. Scatter search to generate diverse MIP solutions. *Computing Tools for Modeling, Optimization and Simulation*, 12(1), pp. 299-317.
- Godzik, A., Kolinski, A. & Skolnick, J., 1993. Lattice representations of globular proteins: how good are they? *Journal of Computational Chemistry*, 14(10), pp. 1194-1202.
- Grassberger, P., 1997. Pruned-enriched Rosenbluth method: Simulations of  $\theta$  polymers of chain length up to 1,000,000. *Physical Review E*, 56(3), pp. 3682-3693.
- Greenberg, H. J., Hart, W. E. & Lancia, G., 2004. Opportunities for combinatorial optimization in computational biology. *INFORMS Journal on Computing*, 16(3), pp. 211-231.
- Gupta, A., Manuch, J. & Stacho, L., 2005. Structure-approximating inverse protein folding problem in the 2D HP model. *Journal of Computational Biology*, 12(10), pp. 1328-1345.
- Hsu, H.-P., Mehra, V., Nadler, W. & Grassberger, P., 2003. Growth-based optimization algorithm for lattice heteropolymers. *Physical Review E*, 68(2), pp. 211-217.

- Istrail, S. & Lam, F., 2009. Combinatorial algorithms for protein folding in lattice models: a survey of mathematical results. *Communications in Information & Systems*, 9(4), pp. 303-346.
- Jensen, I., 2008. Enumeration of self-avoiding walks on the square lattice. *Journal of Physics A: Mathematical and General*, 37(21), pp. 5503-5528.
- Johnson, D. S. & McGeoch, L. A., 1997. The traveling salesman problem: A case study in local optimization. *Local Search in Combinatorial Optimization*, 1(1), pp. 215-310.
- Klau, G. W., Lesh, N., Marks, J. & Mitzenmacher, M., 2002. Human-Guided Tabu Search. *AAAI Proceedings*, 16(3), pp. 41-47.
- Kloczkowski, A., Sen, T. Z. & Jernigan, R. L., 2003. The transfer matrix method for lattice proteins—an application with cooperative interactions. *Polymer*, 45(2), pp. 707-716.
- Lau, K. F. & Dill, K. A., 1989. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10), pp. 3986-3997.
- Lesh, N., Mitzenmacher, M. & Whitesides, S., 2003. A complete and effective move set for simplified protein folding. *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pp. 188-195.
- Lorenço, H. R., Martin, O. C. & Stützle, T., 2010. Iterated local search: Framework and applications. In: *Handbook of Metaheuristics*. US: Springer, pp. 363-397.
- Miyazawa, S. & Jernigan, R. L., 1985. Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, 18(3), pp. 534-552.
- Namsu, A. & Park, S., 2010. Finding an upper bound for the number of contacts in hydrophobic-hydrophilic protein structure prediction model. *Journal of Computational Biology*, 17(4), pp. 647-656.
- Ngo, J. T., Marks, J. & Karplus, M., 1992. Computational complexity of a problem in molecular structure prediction. *Protein Engineering*, 5(4), pp. 313-321.
- Nunes, L.F., Galvão, L.C., Lopes, H.S., Moscato, P. and Berretta, R., 2016. An integer programming model for protein structure prediction using the 3D-HP side chain model. *Discrete Applied Mathematics*, 198(1), pp. 206-214.
- Rego, C., Li, H. & Glover, F., 2011. A Filter-and-Fan approach to the 2D lattice model of the protein folding problem. *Annals of Operations Research*, 188(1), pp. 389-414.

- Shmygelska, A. & Hoos, H. H., 2005. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6(1), pp. 30-52.
- Thachuk, C., Shmygelska, A. & Hoos, H. H., 2007. A replica exchange Monte Carlo algorithm for protein folding in the HP model. *BMC Bioinformatics*, 8(1), pp. 342-362.
- Toma, L. & Toma, S., 1996. Contact interactions method: A new algorithm for protein folding simulations. *Protein Science*, 5(1), pp. 147-153.
- Uger, R. & Moult, J., 1993. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231(1), pp. 75-81.
- Vendruscolo, M., Najmanovich, R. & Domany, E., 2000. Can a pairwise contact potential stabilize native protein folds against decoys obtained by threading. *Proteins: Structure, Function, and Bioinformatics*, 38(2), pp. 134-148.
- Verdier, P. & Stockmayer, W., 1962. Monte Carlo calculations on the dynamics of polymers in dilute solution. *The Journal of Chemical Physics*, 36(1), pp. 227-235.
- Yanev, N., Milanov, P. & Mirchev, I., 2011. Integer programming approach to HP folding. *Serdica Journal of Computing*, 5(4), pp. 359-366.
- Yoon, H.-s., 2006. Optimization approaches to protein folding, Doctoral dissertation, Georgia Institute of Technology.
- Yue, K. & Dill, K. A., 1993. Sequence-structure relationships in proteins and copolymers. *Physical Review E*, 48(3), pp. 2267-2277.
- Yue, K., Fiebig, K. M., Thomas, P. D. & Chan, H. S., 1995. A test of lattice protein folding algorithms. *Proceedings of the National Academy of Sciences*, 92(1), pp. 325-329.

## VITA

Seyed Mojtaba Hosseini was born in 1990, in Birjand, the east Iranian provincial capital of South Khorasan. He received his diploma in Mathematics and Physics from National Organization for Development of Talents (NODET) high school in the same city in 2008, and his B.Sc. degree in Industrial Engineering from Sharif University of Technology, Tehran, in 2013. In his B.Sc. thesis he designed and developed software tools for computer aided algorithms in facility layout problems. Next year he joined the M.Sc. program in Industrial Engineering at Koç University, Istanbul, and worked as research and teaching assistant under supervision of Prof. Dr. Ceyda Oğuz. The current thesis is the result of his research under supervision of Prof. Oğuz from 2014 to 2016. Next year, he will join the Ph.D. program in Operations and Decision Technologies at the Paul Merage School of Business, UC Irvine, USA.

