MATH 177 - LINEAR AND NON-LINEAR OPTIMIZATION CLASS PROJECT: PROTEIN FOLDING

MILICA MISKOVIC
012446863

Abstract. Protein folding is the physical process by which chain of amino acids acquires its multidimensional structure, and by which it gets transformed into molecule that is usually biologically functional. Predicting optimal folding pattern can be modeled by integer programming model.

Introduction

The function of a protein is determined by its amino acid sequence, but understanding the effect of different multidimensional layouts on that structure would provide additional understanding of how does protein get. It embodies a celebrated problem in computational biology, known as the protein folding problem.

Forrester and Greenberg in their paper *Quadratic Binary Programming Models in Computational Biology* address the question whether this problem can be modeled by Quadratic Binary Programming and solved by the general purpose solver.

They consider two solution strategies for solving the Quadratic Binary Programming. The first strategy, called **linearization** is reformulating the Quadratic Binary Programming as a Mixed-Integer Linear Program through the introduction of auxiliary variables and constraints. Then, the reformulation can be solved using any standard mixed-integer linear solver. The second strategy, known as **convex quadratic reformulation**, is to rewrite the Quadratic Binary Programming into an equivalent problem with a convex quadratic objective function. This does not require auxiliary variables or constraints, but it does require an adjustment to the objective, called convexification, to make the quadratic form a convex function.[1]

The goal of this project is to explore protein folding modeled by simple integer program and finding optimal folding patterns for protein with various chain lengths and number (and position) of hydrophobic amino acids.

1. Importance of Protein Folding

Protein folding is process by which random coil shaped poly-peptide chain of amino acids folds into conformation that is biologically functional. It usually happens shortly after translation of nucleotide sequence (genesis of the amino acid chain). The correct tridimensional structure of the protein is very important for correct biological functioning, although some parts of proteins can be unfolded without any consequence to functionality. Misfolded proteins generally produce biologically inactive molecules, but

in some instances malformation of proteins can lead to toxic functionality. For example, some allergies are caused by incorrect folded structure of protein which prevents immune system to produce antibodies for certain protein structures.[3]

Folding happens since hydrophobic acids tend to attract each other which causes hydrophobic collapse. In the folded protein, the hydrophobic amino acids collapsed toward the center become shielded from aqueous environment.

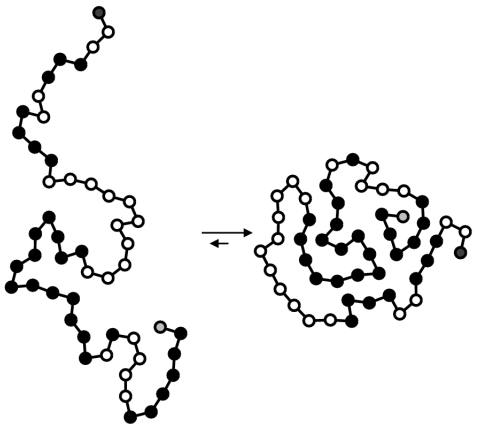

FIGURE 1. Hydrophobic collapse. In the compact fold (to the right), the hydrophobic amino acids (shown as black spheres) collapse toward the center to become shielded from aqueous environment.[3]

2. METHOD

As we previously stated, the problem at hand is predicting optimal folding pattern of one dimensional amino acid chain and it can be modeled by integer programming model. We will take an amino acid sequence which consists out of some number of amino acids with hydrophobic acids at some known positions. For each pair of hydrophobic acids, we will pair and match them if they are not consecutive (paired already) and if they have an even number of acids between them. Also, there can be only one fold between two amino acids, that we want to match.

We will define two sets of binary variables:

$x_{ij} = 1$ – if hydrophobic acid in position **i** is matched with the one in position **j**

$y_k = 1$ – if a fold occurs between the i^{th} and $(i+1)^{th}$ acid in the chain

Since the goal of the folding process (as already stated) is clustering as many hydrophobic acids (bringing as many of them them as clse together as possible), the final solution should be optimal with respect to energy functions. In particular, our energy function includes pairwise interaction preferences and allowing variable gaps. Therefore, objective function will have following form:

$$max \sum_{\forall i,j} x_{ij} \tag{2.1}$$

where

$i, j \in \overrightarrow{v}$– vector of known positions of hydrophobic acids in chain

and previous folding and matching rules impose that our objective function has to be subject to set of constraints:

$$y_k + x_{i,j} \leq 1, \tag{2.2}$$
$$\left(\forall i,j,k | i \leq k \leq j \wedge k \neq \frac{i+j-1}{2}\right)$$

and

$$x_{i,j} = y_{\frac{i+j-1}{2}}, \tag{2.3}$$

$$\left(\forall i,j | i+j-1 = 2p\right), p \in \mathbb{N}$$

3. Solution

Problem at hand is solved using R programming language and Package 'lpSolve' since solving via LP assistant or MS Office Excel would be very hard, if not impossible since we are expecting more than 150 variables and over 1000 constraints. Example chain consists of 50 amino acids with hydrophobic acids on positions 2, 4, 5, 6, 11, 12, 17, 20, 21, 25, 27, 28, 30, 31, 33, 37, 44 and 46. Same script can be used to determine optimal folding path for arbitrarily long protein, with different position pattern of hydrophobic acids by changing those 2 input variables. We defined following objects required[1] by lp function:

- **objective.in** (Numeric vector of coefficients of objective function) – Since our objective function is 2.1, and we have 147 x[i,j] variables that appear in objective function with coefficient 1, our objective vector will take form of

$$objective.in = \underbrace{(1,1,1,...,1}_{147}, \underbrace{0,0,...,0)}_{49}$$

- **const.mat** (Matrix of numeric constraint coefficients, one row per constraint, one column per variable) – In accordance with 2.2 and 2.3, for every x[i,j] in constraint (indicated with coefficient 1 in const.mat row of that particular constraint) we will assign y[k] so that 2.2 and 2.3 in adequate constraint row, conventionally we will make a matrix with as many columns as we have variables and as many rows as we have constraints in form:

$$\mathbf{const.mat} = \begin{array}{c} \begin{array}{ccccccc} \text{x[2,4]} & \text{x[2,5]} & \dots & \text{y[1]} & \text{y[2]} & \dots & \text{y[49]} \end{array} \\ \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 1 \end{bmatrix} \end{array}$$

- **const.dir** (Vector of character strings giving the direction of the constraint: each value should be one of " < "," <= "," = "," == "," > ", *or*" >= ") – Since 2.2 first 147 elements of const.dir vector will be 'less or equal' ($\leq$) and since 2.3 the rest will be 'equal' signs (=).

[1]The list of requirements found in lpSolve package documentation [4]

- **const.rhs** (Vector of numeric values for the right-hand sides of the constraints) – First 147 elements are 1, rest is 0.

Problem at hand is solved using LpSolve library. We also had to indicate direction all our variables have to be binary, therefore:

- **direction** (Character string giving direction of optimization: "min" (default) or "max.")
- **all.bin** (Logical, if "T" or "True", all variables should be binary.)

```
library(lpSolve)
solution<-lp (direction = "max", objective.in, const.mat,
const.dir, const.rhs, all.bin = T)
```

Solved problem gave us eight (optimal objective value) hydrophobic acids that were matched (additional to those already contiguous) with folds between acids 3 and 4, 8 and 9, 22 and 23 and 35 and 36 (point in which optimum is achieved).

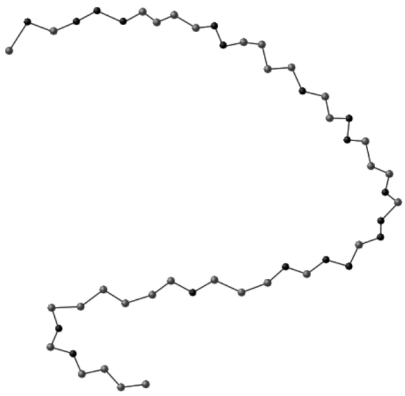

FIGURE 2. Unfolded protein chain with 50 hydrophobic acids indicated as black dots on positions 2, 4, 5, 6, 11, 12, 17, 20, 21, 25, 27, 28, 30, 31, 33, 37, 44 and 46.[2]

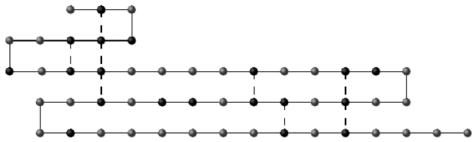

FIGURE 3. Folded protein chain with folds between acids 3 and 4, 8 and 9, 22 and 23 and 35 and 36 [2]

4. APPENDIX – CODE

```
#project MATH 177

#positions of hydrophobic acides
pos<-c(2,4,5,6,11,12,17,20,21,25,27,28,30,31,33,37,44,46)

#binar variables x[i,j]
i.j.k<-matrix(0, nrow = 0, ncol = 3)
numvar<-0

for (i in pos) {
  for(j in pos) {
    if(j > (i+1)){
      i.j.k<-rbind(i.j.k, c(i,j,0))
      numvar<-numvar+1
    }
  }
}
#binar variables y[k] indicating contiguous amino acids
i.j.k[,"k!="]<-(i.j.k[,"xi"]+i.j.k[,"xj"]-1)/2
possibleK<-matrix(0, 147, 49)

for (ij in 1:147) {
  for(k in 1:49){
    if (( i.j.k[ij,"xi"] <= k) & (k < i.j.k[ij,"xj"] )
    &( k != i.j.k[ij,"k!="])){
      possibleK[ij,k]<-k}}}

xmat1 <-matrix(0, sum(colSums(possibleK != 0)),nrow(i.j.k) )

ConstraintsForXij<-rowSums(possibleK != 0)
for (ij in 1:nrow(possibleK)){
  for (i in 1:ConstraintsForXij[ij]){
    xmat1[i, ij]<-1
  }
}
#second constraint x[i,j]=y[(i+j-1)/2]
inx<-numeric()
iny<-numeric()
for (k in 1:nrow(i.j.k)) {
  if (((i.j.k[k,"xi"]+i.j.k[k,"xj"]-1)/2)%%1==0){
    inx<-c(inx,k)
    iny<-c(iny, (as.integer(i.j.k[k,"xi"]+i.j.k[k,"xj"]-1)/2))
  }
```

```
}
#xpart const 2
xmat2<-matrix(0,length(inx),147)
for (number in 1:length(inx)){
  xmat2[number,inx[number]]<-1
}
#ypart const2
ymat2<-matrix(0, length(iny), 49)
for(number in 1:length(iny)){
  ymat2[number, iny[number]]<-(-1)
}
#objective function (coefficients)
objective.in<-c(rep(1, numvar), rep(0, 49))

#matrix of constraints
xmat<-diag(1, 147, 147) #X part of matrix of constraints
ncol(cbind(xmat1, ymat1))
ncol(cbind(xmat2, ymat2))
const.mat<-rbind(cbind(xmat1, ymat1), cbind(xmat2, ymat2))

#vector of RHS values
const.rhs<-c(rep(1, 147), rep(0, nrow(xmat2)))
#vector of directions
const.dir<-c(rep("<=", 147), rep("=", nrow(xmat2)))

#solving the problem using lpSolver pacage
library(lpSolve)
solution<-lp (direction = "max", objective.in, const.mat,
const.dir, const.rhs, all.bin = T)
```

References

[1] Richard J. Forrester, Harvey J. Greenberg, *Quadratic Binary Programming Models in Computational Biology*, Algorithmic Operations Research,(Vol.3),(2008), 110-129.

[2] Williams, H. Paul *Model Building in Mathematical Programming*, John Wiley and Sons, Incorporated, 2013.

[3] Wikipedia, *Protein Folding*, https://en.wikipedia.org/wiki/Protein_folding

[4] Michel Berkelaar and others, *Package 'lpSolve'* , https://cran.r-project.org /web/packages/lpSolve/lpSolve.pdf

Milica Miskovic, 012446863
Email address: zlotrov@gmail.com