

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
use Math::Trig ;
use strict;
```

```
#{my %coor,my $chnum}=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}}( my $sqn=substr($r,0,1); if ( $sqn ne $ch) { $chnum++; $ch=$sqn; } );
my %qwa=find_quart( $coor{"O"});
```

```
if ($sqnum >0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=~ s/^\.//;
$filename=~ s/\.pdb//;
#$filename=$chnum.".".$sqnum.".".$filename.".dat";
$filename="$dir".".$filename.".dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $sqnum\n";
```

```
foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets;
my %q= find_q( $coor{$m} );
```

```
# foreach my $q { keys %qartets } { print join " ", @{$qartets{$q}}, "\n";
foreach my $q { keys %qartets } {
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res ( @{$qartets{$q}} ){
# print "$q $coor{$m}{ $res } {"R"}->x, "\n";
```

```
$nx=$nx+ $coor{$m}{ $res }{"N9"}->x;
$ny=$ny+ $coor{$m}{ $res }{"N9"}->y;
$nz=$nz+ $coor{$m}{ $res }{"N9"}->z;

$ox=$ox+ $coor{$m}{ $res }{"O6"}->x;
$oy=$oy+ $coor{$m}{ $res }{"O6"}->y;
$oz=$oz+ $coor{$m}{ $res }{"O6"}->z;
$r=$res;
}
```

Структурная Биоинформатика

Лекция 8. Моделирование самосборки белка

Головин А.В.¹

¹МГУ им М.В. Ломоносова, Факультет Биотехнологии и Биоинформатики

Москва, 2012

Содержание

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Введение

```
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $chnum++; $ch=$ggg } };
```

Методы поиска

```
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
if ( $qnum > 0 ){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename--~/\//;
# $filename=$chnum . $qnum . $filename . $r;
$filename="$dir". $filename . ".dat";
print "$filename\n";
open OUT ">$filename";
print OUT "O chain $chnum qnum $qnum\n";
```

Поиск глобального минимума

```
foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets= %qwa; #find_quart( $coor{ $m } );
my %q= find_q( $coor{ $m } );
```

REMD

```
foreach my $q ( keys %qartets ){
join " " , @{$qartets{ $q } } , "\n";
```

Самосборка белка

```
foreach my $q ( keys %qartets ){
my $r;
```

Folding@home

```
my $r;
```

```
foreach my $res ( @{$ qartets{ $q } } ){
my $or{ $m } { $res } { "R" }->x, "\n";
$nx=$nx+ $coor{ $m } { $res } { "N9" }->x;
$ny=$ny+ $coor{ $m } { $res } { "N9" }->y;
$nz=$nz+ $coor{ $m } { $res } { "N9" }->z;
```

FoldIt Game

```
$ox=$ox+ $coor{ $m } { $res } { "O6" }->x;
$oy=$oy+ $coor{ $m } { $res } { "O6" }->y;
$oz=$oz+ $coor{ $m } { $res } { "O6" }->z;
```



```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Систематические методы: Поиск по сетке

```
#!/(my $coor,my $chnum)=read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=subst($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } ;
```

• Находим торсионные углы способные вращаться.

• Генерируем ротамеры по каждому углу с небольшим шагом.

• Минимизируем энергию для каждого ротамера.

```
if ($qnum > 0){
#system("if");
my $filename="";
$filename="-- s/~/~/V/";
$filename="-- s/\,pdb//";
#$filename="$ch$dir"."$qnum"."$filename".dat;
$filename="$ch$dir"."$qnum"."$filename".dat;
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum\n";
```

```
foreach my $m (sort { $a-<=>$b } keys %coor){
my %qartets = %qwa ; #find_quart( $coor{$m} );
my %q = find_q( $coor{$m} );
```

```
# foreach my $q { keys %qartets } { print join " ", @{$qartets{$q}} , "\n";
```

```
foreach my $q { keys %qartets } {
```

```
my $ox, my $oy, my $oz;
my $nx, my $ny, my $nz;
foreach my $res ( @{$qartets{$q}} ){
```

```
# print "$q $coor{$m} {$res} {"$r"}->x,\n";
```

```
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
```

```
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
```

```
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
```

```
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
```

```
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

$$N_{conf} = \prod_{i=1}^N \frac{360}{\alpha_i}$$

Если у нас пять вращаемых связей и инкремент 30 то количество конформаций 248832

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Поиск по сетке: подходы

```
my ($my $coor, my $snum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $sdir=$ARGV[1];
my $sch, my $snum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $sggg=substr($r,0,1); if ( $sggg ne $sch ){ $snum++; $sch=$sggg } ;
```

- Не обязательно рассматривать все торсионные углы, только те, которые образуются между “строительными блоками”
- Важно иметь базу фрагментов и уметь точно вычленять фрагменты из целевой молекулы.
- Первое допущение: фрагменты в молекуле не зависимы
- Второе допущение: набор заготовок должен покрывать все конформации фрагмента в составе молекулы

```
my %qwa;
my $qnum=0;
my $qcoor={"0"};
my $qnum=keys %qwa;
```

```
if ($qnum > 0){
#system("rm -f $qcoor");
my $filename=$ARGV[0];
$filename="--s/~/V//";
$filename="--s0.pdb//";
#system("cp $filename $qcoor");
$filename="--sdir/$filename.dat";
print "filename=$filename\n";
open OUT, ">$filename";
print OUT "#INFO chain $snum qnum $qnum\n";
```

```
foreach my $r ( sort keys %{$coor{"0"}} ){
my $q;
my $q=find_q($coor{"0"});
#
foreach my $q ( keys %qartets ){ print join " ", @{$qartets{"$q"}}, "\n";
foreach my $s ( @{$qartets{"$q"}} ){
my $sx;
my $sx;
my $sy;
my $sz;
my $r;
```

```
foreach my $res ( @{$ $qartets{"$q"}} ){
#
print "$q $coor{$s}{$res} {"$r"}->x,\n";
$nx=$nx+ $coor{$s}{$res} {"$r"}->x;
$ny=$ny+ $coor{$s}{$res} {"$r"}->y;
$nz=$nz+ $coor{$s}{$res} {"$r"}->z;
$ox=$ox+ $coor{$s}{$res} {"O6"}->x;
$oy=$oy+ $coor{$s}{$res} {"O6"}->y;
$oz=$oz+ $coor{$s}{$res} {"O6"}->z;
```


Дистанционная геометрия

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
#( my %$coor, my $snum)=read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $snum;
foreach my $r ( sort keys %{$coor{"O"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch ){ $snum++; $sch=$ggg } ;
```

- Конформацию молекулы можно представить как по парные расстояния между всеми атомами.
- В молекуле с N атомами количество расстояний $N \frac{(N-1)}{2}$.
- Есть удобная особенность: верхняя граница для некоторых расстояний.
- Тут становится удобно использовать матрицы для оптимизации случайных значений.

```
foreach my $q ( keys %$qartets ){ print join " ", @{$qartets{$q}} , "\n";
foreach my $s ( keys %$coor ){ my $x=$coor{$s}{x}; my $y=$coor{$s}{y}; my $z=$coor{$s}{z};
my $sx, my $sy, my $sz;
my $r;
foreach my $res ( @{$qartets{$q}} ){
# print "$q $coor{$s} {$res} {"$r"}->x,\n";
$nx=$nx+ $coor{$s} {$res} {"$r"}->x;
$ny=$ny+ $coor{$s} {$res} {"$r"}->y;
$nz=$nz+ $coor{$s} {$res} {"$r"}->z;
$ox=$ox+ $coor{$s} {$res} {"O6"}->x;
$oy=$oy+ $coor{$s} {$res} {"O6"}->y;
$oz=$oz+ $coor{$s} {$res} {"O6"}->z;
```



```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Выбор метода для поиска конформаций

```

#(my %coor,my $cnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $cnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $cnum++; $ch=$ggg } ;

```

Поиск конформеров циклогептадекана C₁₇H₃₄

```

if ($cnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];

```

| Метод | Количество конформеров |
|---------------------------|------------------------|
| Систематический поиск | 211 |
| Случайный коор. поиск | 222 |
| Случайный торс. поиск | 249 |
| Дистанционные ограничения | 176 |
| МД | 169 |

```

my $nx, my $ny, my $nz,
my $ox, my $oy, my $oz;
my $r;

foreach my $res (@{ $partets{$q} }){
#
print "$q $coor{$m}{$res} {"$R"}->x,"n";
$nx=$nx+ $coor{$m}{$res} {"$R"}->x;
$ny=$ny+ $coor{$m}{$res} {"$R"}->y;
$nz=$nz+ $coor{$m}{$res} {"$R"}->z;

$ox=$ox+ $coor{$m}{$res} {"O6"}->x;
$oy=$oy+ $coor{$m}{$res} {"O6"}->y;
$oz=$oz+ $coor{$m}{$res} {"O6"}->z;

```



```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Эволюционные алгоритмы

```
my ($my $coor, my $schnum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"O"}} ) { my $ggg=substr($r,0,1); if ( $ggg ne $sch ) { $schnum++; $sch=$ggg } ;
```

Три основных метода

- Генетический алгоритм
- Эволюционное программирование
- Эволюционные стратегии

Общее: создаётся набор вероятных решений проблемы.

Решения становятся хорошими или плохими и эволюционируют улучшаясь. Мера “хорошо” это внутренняя энергия молекулы определяемая из молекулярной механики.

```
foreach my $m (sort { $a<=>$b } keys %coor) {
#
    print "$q $coor{$m} {$sres} {"$R"}->x,\n";
    $nx=$nx+ $coor{$m} {$sres} {"N9"}->x;
    $ny=$ny+ $coor{$m} {$sres} {"N9"}->y;
    $nz=$nz+ $coor{$m} {$sres} {"N9"}->z;

    $ox=$ox+ $coor{$m} {$sres} {"O6"}->x;
    $oy=$oy+ $coor{$m} {$sres} {"O6"}->y;
    $oz=$oz+ $coor{$m} {$sres} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Моделирование отжига

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){$chnum++; $ch=$ggg} };

```

```
my %qwa=find_quart( %coor{"O"} ); my $qnum=keys %qwa;
```

- Отжиг — это процесс при котором расплавленное вещество медленно остужают и оно кристаллизуется в идеальный кристалл.

- Идеальный кристалл и есть искомый минимум энергии

- Моделирование отжига направлено именно на достижение глобального минимума с при моделировании охлаждения системы.

```

foreach my $m (sort {$a<=>$b} keys %coor){
  my $q;
  my $q;
# foreach my $q ( keys %qartets){
  foreach my $q ( keys %qartets){
    my $ix, my $ny, my $nz;
    my $ox, my $oy, my $oz;
    my $r;

    foreach my $res ( @{$ $qartets{$q} }){
      print "$q $coor{$m}{$res} {"$r"}->x,"$n";
      $nx=$nx+ $coor{$m}{$res} {"$r"}->x;
      $ny=$ny+ $coor{$m}{$res} {"$r"}->y;
      $nz=$nz+ $coor{$m}{$res} {"$r"}->z;

      $ox=$ox+ $coor{$m}{$res} {"O6"}->x;
      $oy=$oy+ $coor{$m}{$res} {"O6"}->y;
      $oz=$oz+ $coor{$m}{$res} {"O6"}->z;

```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Моделирование отжига

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){$chnum++; $ch=$ggg} };

```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
```

```
  #system("mkdir $ARGV[1]*");
```

```
  my $fil;
```

```
  $filename=
```

```
  $filename="-- s/\.pdb//;
```

```
  # $filename=$chnum." ".$qnum." ".$filename.".dat";
```

```
  $filename=
```

```
  print "fil: ";
```

```
  open OUT,">$filename";
```

```
  print OUT " ";
```

```
  foreach my $m (sort { $a->-$b } keys %coor{1})
```

```
    my %q;
```

```
    my %q=find_q(%coor{$m});
```

```
  #   foreach my $q ( keys %qartets ){ print join " ",@{$qartets{$q}},"n";
```

```
    foreach my $q ( keys %qartets){
```

```
      my $nx; my $ny; my $nz;
```

```
      my $ox; my $oy; my $oz;
```

```
      my $r;
```

```
      foreach my $res ( @{$qartets{$q}} ){
```

```
        #   print "$q $coor{$m}{$res}{\"N\"}->x,\"n\";
```

```
        $nx=$nx+ $coor{$m}{$res}{\"N9\"}->x;
```

```
        $ny=$ny+ $coor{$m}{$res}{\"N9\"}->y;
```

```
        $nz=$nz+ $coor{$m}{$res}{\"N9\"}->z;
```

```
        $ox=$ox+ $coor{$m}{$res}{\"O6\"}->x;
```

```
        $oy=$oy+ $coor{$m}{$res}{\"O6\"}->y;
```

```
        $oz=$oz+ $coor{$m}{$res}{\"O6\"}->z;
```

- Важно тщательно контролировать температуру
- Не факт что метод может привести к глобальному минимуму, но если несколько запусков привели к одной структуре, то можно это предположить.

Конформации белка из ограниченной МД

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $key ( keys %coor ) {
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
if ($qnum > 0) {
#system("mv $filename-$ARGV[0]");
$filename="--s/~/V//";
$filename="--s/~/pdb//";
#system("mv $filename-$ARGV[0]");
$filename="--s/~/V//";
$filename="--s/~/pdb//";
print "File name: $filename\n";
open OUT, ">$filename";
print OUT "#INFO Chain $chnum $chnum $chnum\n";
foreach my $key ( keys %qartets ) {
my %qartets= %qwa; #find_quart( $coor{$m} );
my %q= find_q( $coor{$m} );
# foreach my $q { keys %qartets } { print join " ", @{$qartets{$q}}, "\n";
foreach my $q { keys %qartets } {
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
foreach my $res ( @{$qartets{$q}} ) {
print "$q $coor{$m} {$res} {" "N9"}->x,\n";
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
}
```

- Суть идеи состоит в том, что в молекулярную динамику вводятся потенциалы, которые увеличивают энергию если конформация не соответствует экспериментальным данным.

- Например: проверяем в ходе оптимизации геометрии R-фактор структуры и если он отличается от экспериментального, то увеличиваем энергию.

$$E_{tot} = U_{pot} + E_{sf}$$

$$E_{sf} = S \sum ||F_{obs}| - |F_{calc}||^2$$

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Метод обмена репликами (REMD)

```

#(my %$coor,$my $schnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch,$my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $sggg=substr($r,0,1); if ( $sggg ne $sch ){ $schnum++; $sch=$sggg } ;

```

- Основная идея: запустить параллельно несколько счётов с разными температурами.
- Мы можем выбрать правило когда производить обмен конформациями.
- Если мы проводим обмен когда потенциальная энергия одной из реплик ниже чем других, то это похоже на моделирование отжига.
- Такой подход часто используется для моделирования самосборки.

```

if ($schnum) {
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=$dir.$filename;
# $filename=$dir.$filename;
$filename=$dir.$filename;
print "$filename\n";
open OUT,">$filename";
print OUT;
foreach my $r ( sort keys %{$coor{"0"}} ){
my %$q;
my %$q=find_of($coor{$r});
# foreach my $i ( sort keys %{$q}{keys %{$q}} ){ print join ("@",$q{$i},"$i\n");
my $nx=$coor{$r}{$i}{"x"};
my $ny=$coor{$r}{$i}{"y"};
my $nz=$coor{$r}{$i}{"z"};
my $sx=$coor{$r}{$i}{"O6"};
my $sy=$coor{$r}{$i}{"O6"};
my $sz=$coor{$r}{$i}{"O6"};

```

REMD

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

- Цель метода это ускорить сканирование (sampling) конформационного пространства.
- Применимо к переходам через значимые энергетические барьеры.
- В Gromacs обмен между репликами происходит случайно по условию:

$$P(1 \leftrightarrow 2) = \min \left(1, \exp \left[\left(\frac{1}{k_B T_1} - \frac{1}{k_B T_2} \right) (U_1 - U_2) \right] \right)$$

А скорости масштабируются: $(T_1/T_2)^{\pm 0.5}$

```
#
print "$q $coor{$m} {$res} {"$R"}->x,"$n";
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;

$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

REMD: количество реплик

Разница между температурами (репликами):

$$U_1 - U_2 = N_{df} \frac{c}{2} k_B (T_1 - T_2)$$

где N_{df} это количество степеней свободы и "с" это величина от 1 до 2 для системы белок вода.

Если $T_2 = (1 + \epsilon)T_1$ тогда вероятность обмена:

$$P(1 \leftrightarrow 2) = \exp\left(-\frac{\epsilon^2 c N_{df}}{2(1 + \epsilon)}\right) \approx \exp\left(-\epsilon^2 \frac{c}{2} N_{df}\right)$$

Таким образом для вероятности обмена $e^{-2} \approx 0.135$ получаем $\epsilon \approx 2/\sqrt{c N_{df}}$.

И если мы контролируем длину связей, то: $N_{df} \approx 2 N_{atoms}$
и при $c = 2$ надо использовать: $\epsilon = 1/\sqrt{N_{atoms}}$.

```
$sx=$sx+ $coor{$m} {$res} {"O6"}->x;
$sy=$sy+ $coor{$m} {$res} {"O6"}->y;
$sz=$sz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Вызовы молекулярного моделирования

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){$chnum++; $ch=$ggg } ;

```

```
my %qwa=fin
```

Sampling
(достоверность)

Анализ
(понимание)

```

if ($qnum > 0){
#system("mkdir
my $filename
$filename="-- s/~.~/V//;
$filename="-- s/\.pdb//;
#$filename=$chnum."."$qnum."."$filename."$dir
$filename="$dir"."$filename"."dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum \n";

```

```

foreach my $m (sort {$a<=>$b} keys %coor){
my %qartets = %qwa ; #find_quart($coor{$m}) ;
my %q= find_q($coor{$m}) ;

```

```
# foreach my $q { keys %qartets } { print join " ",
```

```
foreach my $q { keys %qartets } {
```

```

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

```

```
foreach my $res (@{ $qartets{$q} }){
```

```
# print "$q $coor{$m} {$res} {"$r"}->x,"n";
```

```
$nx=$nx+ $coor{$m} {$res} {"$r"}->x;
```

```
$ny=$ny+ $coor{$m} {$res} {"$r"}->y;
```

```
$nz=$nz+ $coor{$m} {$res} {"$r"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
```

```
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
```

```
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

Модели
(силовые поля)

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Моделирование самосборки белка

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}} ){ my $ggg=subst($r,0.1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } };
my %qwa

```

- Как иметь дело с большими временами?

Самые "быстрые" белки собираются за 10-100 μ s, моделирование должно быть на порядок длинее.

- На сколько хороши наши силовые поля?

Сможем ли мы получать нативное состояние белка без знания структуры.

Будут ли предсказанные структуры иметь правильные параметры: ΔG , K_f .

- Сможем ли мы изучать сборку моделированием?

Механизмы, Теория ...

```

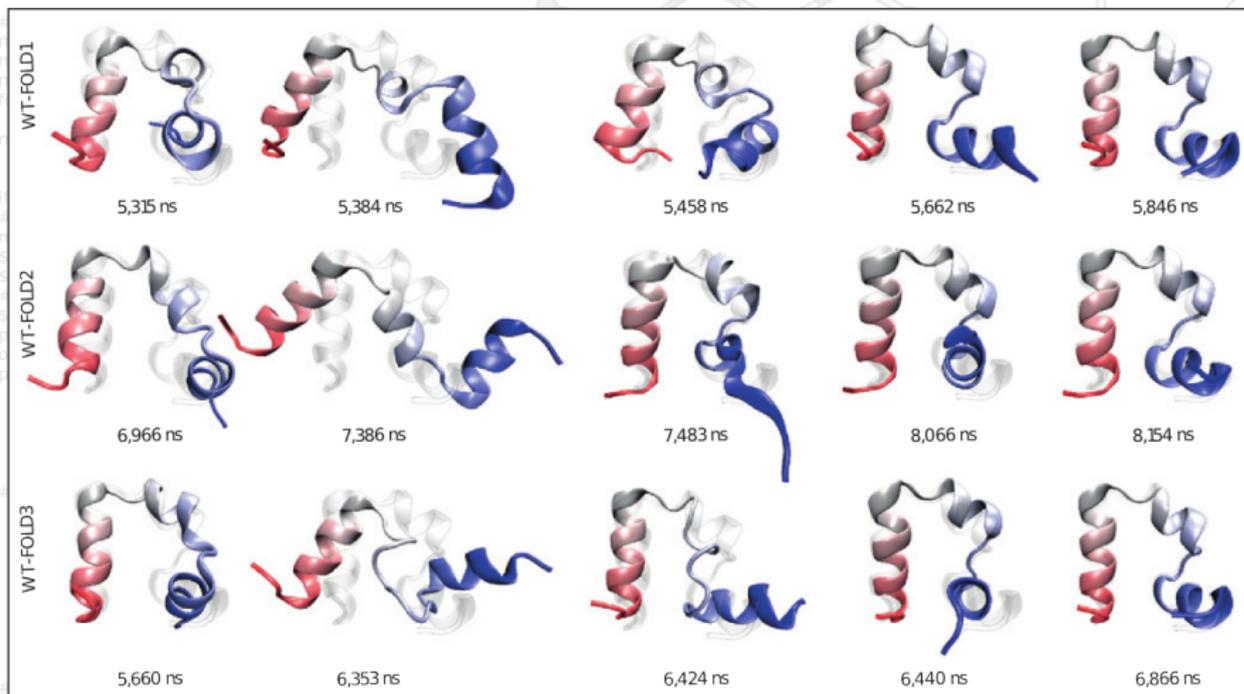
# foreach my $m ( sort { $a-<=>$b } keys %{$coor{"O"}} ){
my %qwa;
my %q;
# foreach my $q ( keys %qartets ){
foreach my $q ( keys %qartets ){
my $so; my $oz;
foreach my $res ( @{$qartets{$q}} ){
print "$q $coor{$m}{$res} {"$res"}->x,"n";
$nx=$nx+ $coor{$m}{$res} {"$res"}->x;
$ny=$ny+ $coor{$m}{$res} {"$res"}->y;
$nz=$nz+ $coor{$m}{$res} {"$res"}->z;

$ox=$ox+ $coor{$m}{$res} {"O6"}->x;
$oy=$oy+ $coor{$m}{$res} {"O6"}->y;
$oz=$oz+ $coor{$m}{$res} {"O6"}->z;

```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

МД в моделировании самосборки белка



DOI: 10.1038/NPHYS1713

МД в моделировании самосборки белка

- Традиционный подход: несколько длинных траекторий
- Альтернатива: методы стохастического кинетического сканирования.

Фолдинг это стохастический процесс с экспоненциальной кинетикой, т.е. количество молекул, которые собрались:

$$f(t) = M[1 - \exp(-kt)]$$

для малых времён:

$$f(t) \sim Mkt \quad M \sim 10,000 \text{procs}, k \sim 1/10,000 \text{ns}, t \sim 20 \text{ns}/\text{proc}$$

ожидается, что можно увидеть 20 раз сборку белка.

- Это эффективно.
- Эргодично

Folding@home

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}} ) {
    my %qwa=find_quart( %coor{"0"} ); m
```

```
if ($qnum > 0) {
    #system("mkdir $ARGV[1]");
    my $filename=$ARGV[0];
    $filename =~ s/^\.//;
    $filename =~ s/\./_/;
    # $filename=$chnum."_"$qnum."."$f;
    $filename="$dir"/$filename.".dat";
    print "$filename\n";
    open OUT,">$filename";
    print OUT "#INFO chain $chnum qnum
```

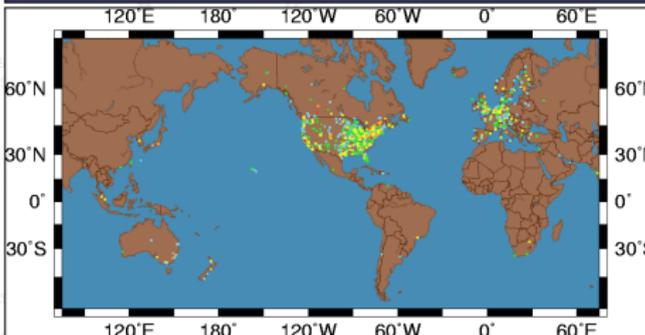
```
foreach my $m (sort { $a-<=>$b } keys %qwa) {
    my %qartets = %qwa; #find_quart(
    my %q = find_q( %coor{$m} );

    # foreach my $q { keys %qartets }
    foreach my $q { keys %qartets } {
        my $nx; my $ny; my $nz;
        my $ox; my $oy; my $oz;
        my $r;

        foreach my $res ( @{$ $qartets
```

```
# print "$q $coor{$m} {"$
    $nx=$nx+ $coor{$m} {$res} {"N9"}->x;
    $ny=$ny+ $coor{$m} {$res} {"N9"}->y;
    $nz=$nz+ $coor{$m} {$res} {"N9"}->z;

    $nx=$nx+ $coor{$m} {$res} {"O6"}->x;
    $oy=$oy+ $coor{$m} {$res} {"O6"}->y;
    $oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```




```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Результаты и ответы

```
my ($my $coor, my $snum)=read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
```

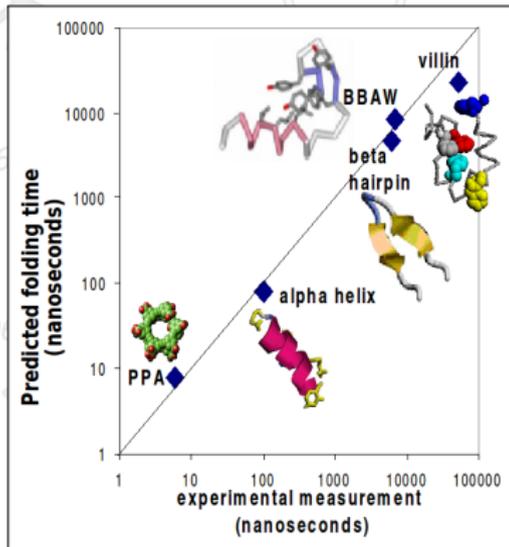
Ответы на наши вопросы:

- Выборка достаточна.
- Силовые поля работают
Сборка происходит правильно
и с нужной скоростью
- Механизм
Вероятно, что самосборка
это не универсальный
механизм и индивидуален
для каждого белка

Illustration from Stefan Larson

```
# print "$q $coor($m) ($res) {"$N"}->x,"n";
$nx=$nx+ $coor($m) ($res) {"N9"}->x;
$ny=$ny+ $coor($m) ($res) {"N9"}->y;
$nz=$nz+ $coor($m) ($res) {"N9"}->z;

$ox=$ox+ $coor($m) ($res) {"O6"}->x;
$oy=$oy+ $coor($m) ($res) {"O6"}->y;
$oz=$oz+ $coor($m) ($res) {"O6"}->z;
```



FoldIt Game

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %$coor,$my $schnum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $my $dir=$ARGV[1];
my $my $sch,$my $schnum;
foreach my $r ( sort keys %{$$coor{"0"} }){ my $my $ggg=substr($r,0,1); if ( $my $ggg ne $my $sch){$my $schnum++; $my $sch=$my $ggg } };
my $my $qnum=$my $schnum; my $my $qnum=keys %$qwa;
```

Основная идея:

- Использование **crowdsourcing** для поиска оптимальной геометрии.
- Пользователь может изменять торсионные углы и добиваться оптимальной энергии.
- В ходе игры для прохождения уровня пользователь должен достичь нужного уровня энергии для структуры
- Создатели программы собираются утилизировать способы придуманные людьми для разработки новых алгоритмов.

```
if ($my $qnum)
#system("mkdir $ARGV[1]");
my $my $filename;
$my $filename;
$my $filename=$my $dir.pdb//;
# $my $filename;
$my $filename;
print "$my $filename";
open OUT "$my $filename";
print OUT "MOL0 chain schnum $my $qnum\n";

foreach my $q ( keys %$qwa ){ print join " ", $my $q, "\n"; }
my $my $q= find of $coor{$my $q};
# foreach my $q ( keys %$qwa ){ print join " ", $my $q, "\n"; }

my $my $x,$my $y,$my $z;
my $my $x;
my $my $y;
my $my $z;

foreach my $res ( @{$my $qartets{$my $q}} ){
# print "$my $q $my $coor{$my $res} {"$my $res"}->x,\n";
$my $x=$my $x+ $my $coor{$my $res} {"$my $res"}->x;
$my $y=$my $y+ $my $coor{$my $res} {"$my $res"}->y;
$my $z=$my $z+ $my $coor{$my $res} {"$my $res"}->z;

$my $x=$my $x+ $my $coor{$my $res} {"$my $res"}->x;
$my $y=$my $y+ $my $coor{$my $res} {"$my $res"}->y;
$my $z=$my $z+ $my $coor{$my $res} {"$my $res"}->z;
```



FoldIt

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
my $c = 0;
my $score = 0;
my $chain = 0;
my $qwa = find_quart( $chain );
```

```
if ($qnum > 0) {
#system("mkdir $ARGV[0]");
my $filename = $ARGV[0];
$filename =~ s/\.//;
$filename =~ s/\.pdb//;
# $filename = $chain;
$filename = "dir". $filename;
open OUT, ">$filename";
print OUT "#INFO chain";
```

```
foreach my $m (sort {
my %qartets = %qwa;
my %q = find_q( $c );
```

```
# foreach my $q ( keys %q ) {
foreach my $q ( keys %q ) {
```

```
my $nx; my $ny;
my $ox; my $oy;
my $r;
```

```
foreach my $res ( keys %q ) {
```

```
# print "S";
$nx = $nx + $q{ $res };
$ny = $ny + $q{ $res };
$nz = $nz + $q{ $res };
```

```
$ox = $ox + $q{ $res } * $q{ "O6" }->x;
$res = "O6"->y;
$res = "O6"->z;
```

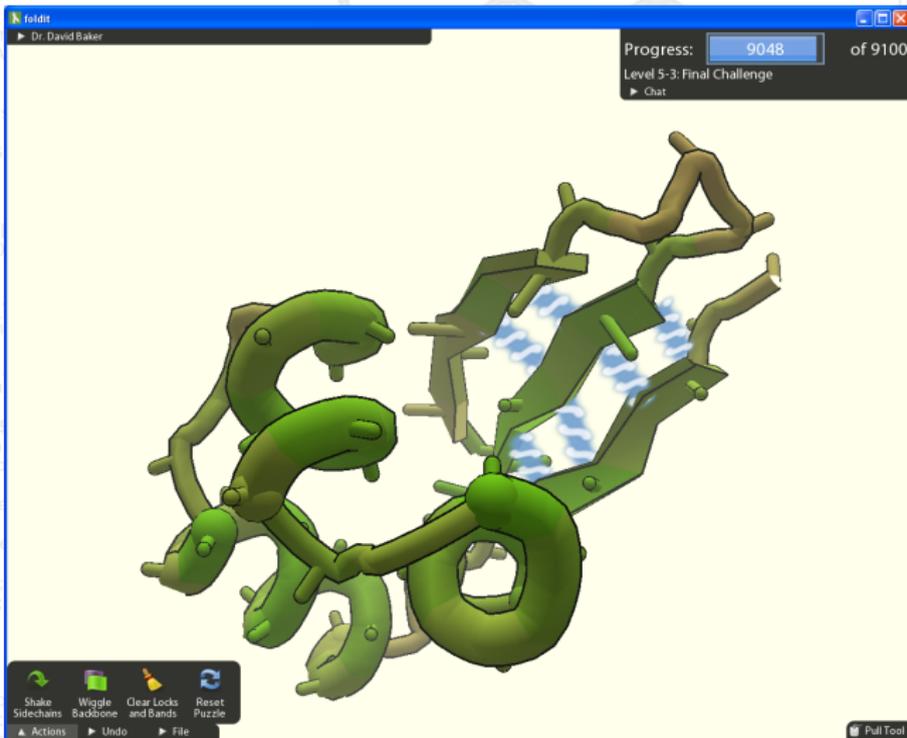


Illustration from Wikipedia

Вопросы?

```

#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %$coor,my $schnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $sch){ $schnum++; $sch=$ggg } };

my %$qwa=find_quart( $coor{"0"} ); my %$qnum=keys %$qwa;

if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=~ s/^\./V//;
$filename=~ s/\.pdb//;
#$filename=$schnum."_"$qnum."_"$filename.".dat";
$filename="dir".$filename.".dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $schnum qnum $qnum \n";

foreach my $m (sort {$a<=>$b} keys %coor){
my %$qartets = %qwa; #find_quart( $coor{$m} );
my %$q = find_q( $coor{$m} );

# foreach my $q ( keys %qartets){ print join " ",@{$qartets{$q}}," \n";

foreach my $q ( keys %qartets){

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

foreach my $res (@{ $qartets{$q} }){

# print "$q $coor{$m}{$res}{\"N\"}->x,\"n\";
$nx=$nx+ $coor{$m}{$res}{\"N9\"}->x;
$ny=$ny+ $coor{$m}{$res}{\"N9\"}->y;
$nz=$nz+ $coor{$m}{$res}{\"N9\"}->z;

$ox=$ox+ $coor{$m}{$res}{\"O6\"}->x;
$oy=$oy+ $coor{$m}{$res}{\"O6\"}->y;
$oz=$oz+ $coor{$m}{$res}{\"O6\"}->z;

```

