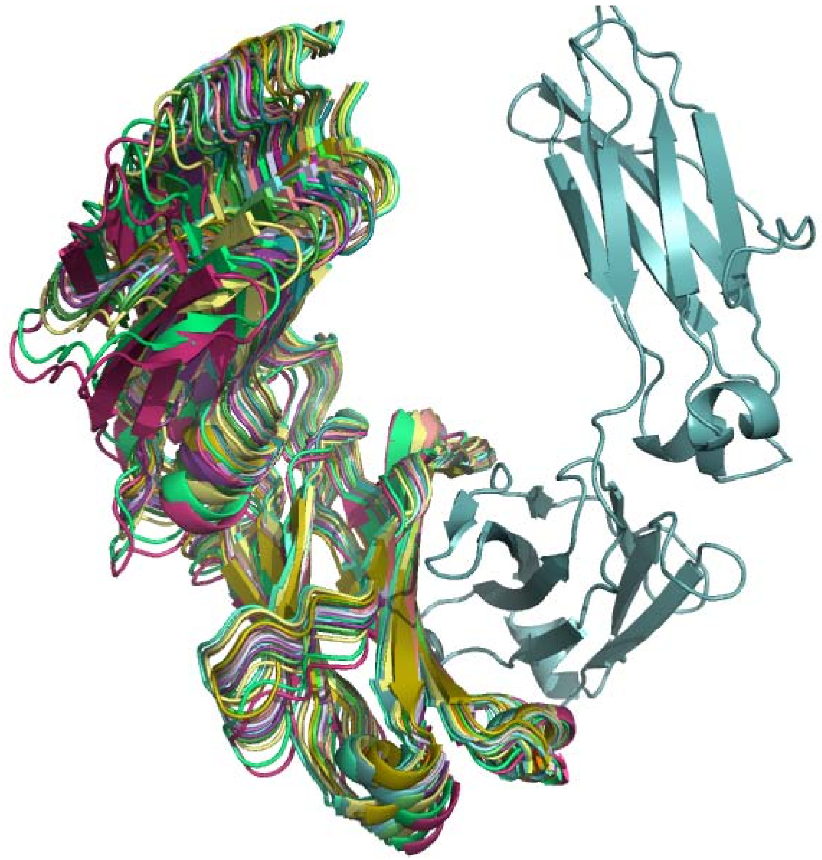
[Your Multistate Design Project Here]



Andrew Leaver-Fay

Kuhlman Lab, UNC

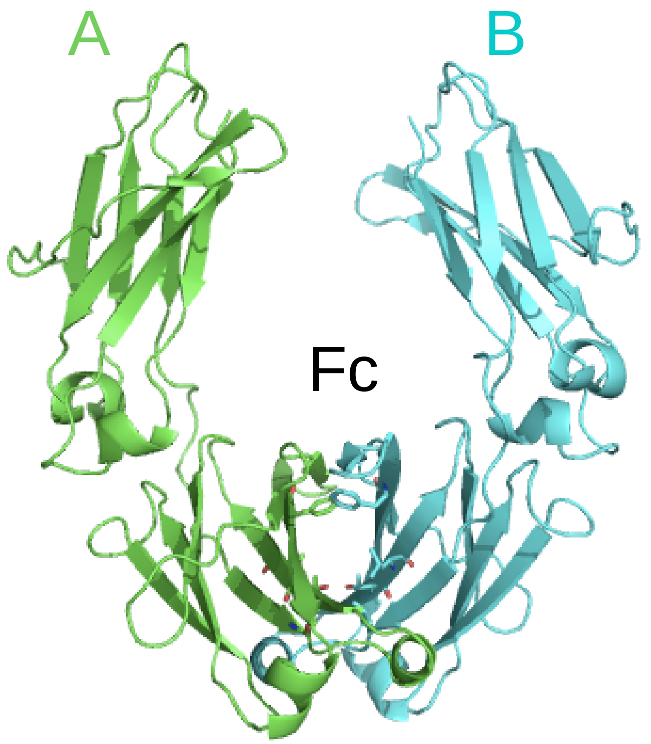
Before I begin…

- This is not the first use of multistate design in Rosetta
  - See Ambroggio & Kuhlman (2006)
  - See Thyme et al. (2009)
  - See Babor & Kortemme (2009)

Motivating Design Project

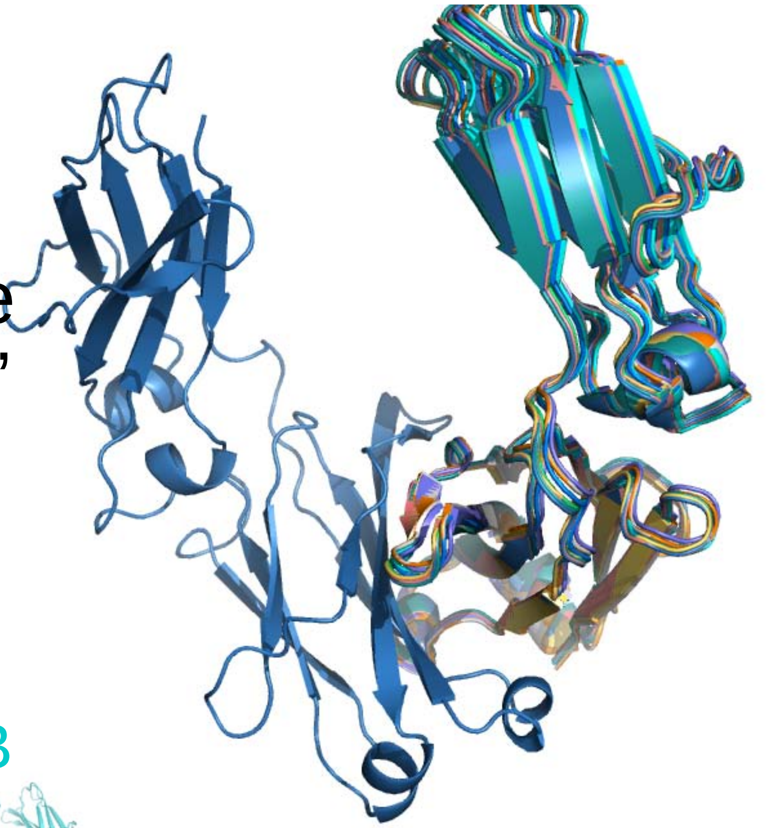
- Heterodimeric antibody design

$$\begin{tikzcd}
 & A & + & B & \\
AA & & BB & & AB
\arrow[from=1-2, to=2-1, "\times" description]
\arrow[from=1-4, to=2-3, "\times" description]
\arrow[from=1-4, to=2-5]
\end{tikzcd}$$

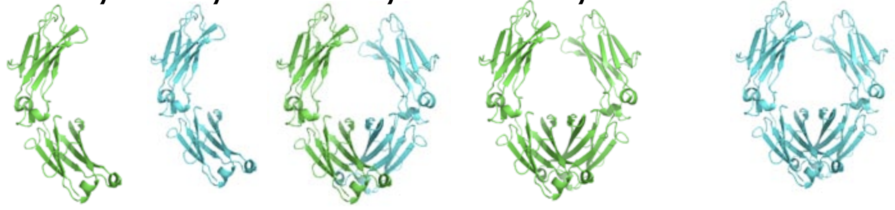


Positive and Negative Design

- (+) Design an asymmetric sequence onto "AB" state
- (-) Evaluate that sequence explicitly on "AA" and "BB" states (50 shown)



- Five species of interest:
  - A, B, AB, AA, & BB



MSD Search

1. Sequence space search
   (outer loop)
   - Pick a sequence, *q*
   - Pack each state, s_i, with sequence *q* to calculate E_i
     (inner loop)
     - E_i = pack_rotamers(s_i, q)
       (innermost loop)
   - Evaluate the *fitness, F,* for *q* given the state *energies*
2. Output the sequence with the best fitness

MSD: fitness function

- Fitness(sequence)
- There is no general fitness function for all MSD projects…
- … so the MSD code expects you to program in your fitness function

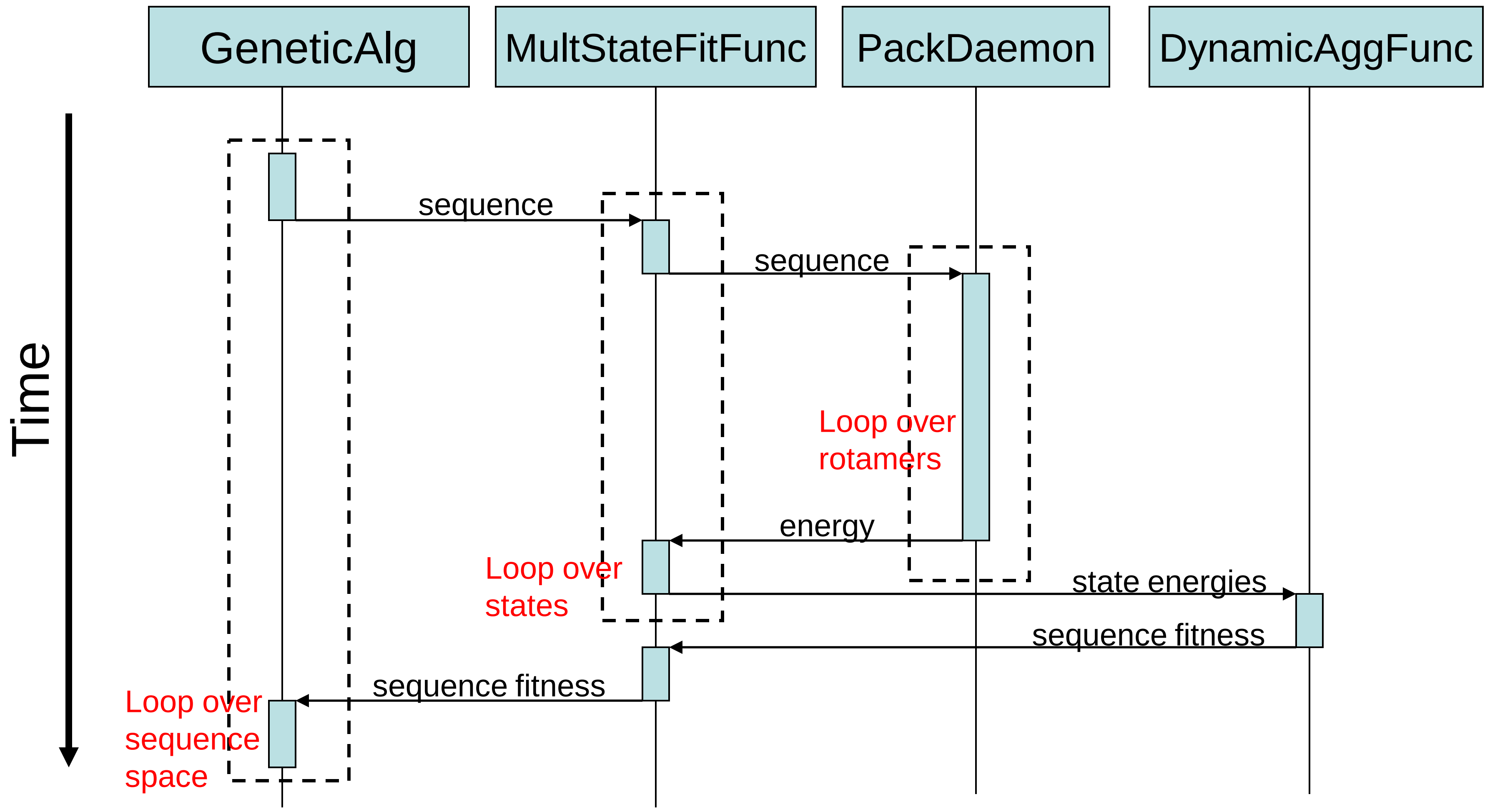
MSD: Fitness Function “programming language”

- Fitness function read from a text file

Outline

- MSD motivation
- MSD implementation
  - Control of flow / MPI
  - Fitness Function File Format
    - Expression parser (broadly useful)
  - Arbitrary Sequence Constraints (NPD)
- MSD example
  - Heterodimeric Antibody Design

MSD Control Diagram



The Players

Entity

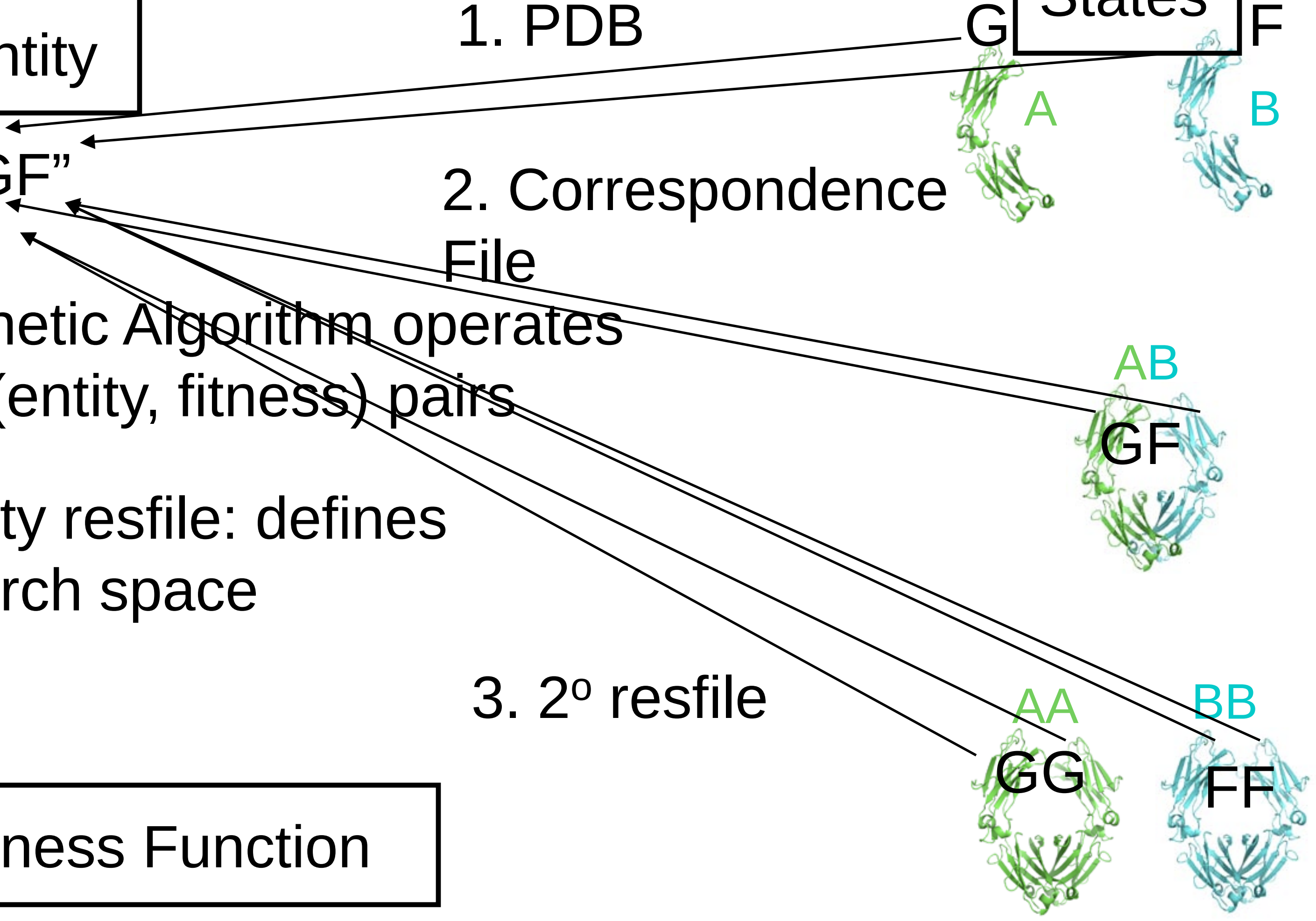
“GF”

States

1. PDB

2. Correspondence File

3. 2^o resfile



Genetic Algorithm operates on (entity, fitness) pairs

entity resfile: defines search space

Fitness Function

State energies vector -> fitness

Fitness Function File Format

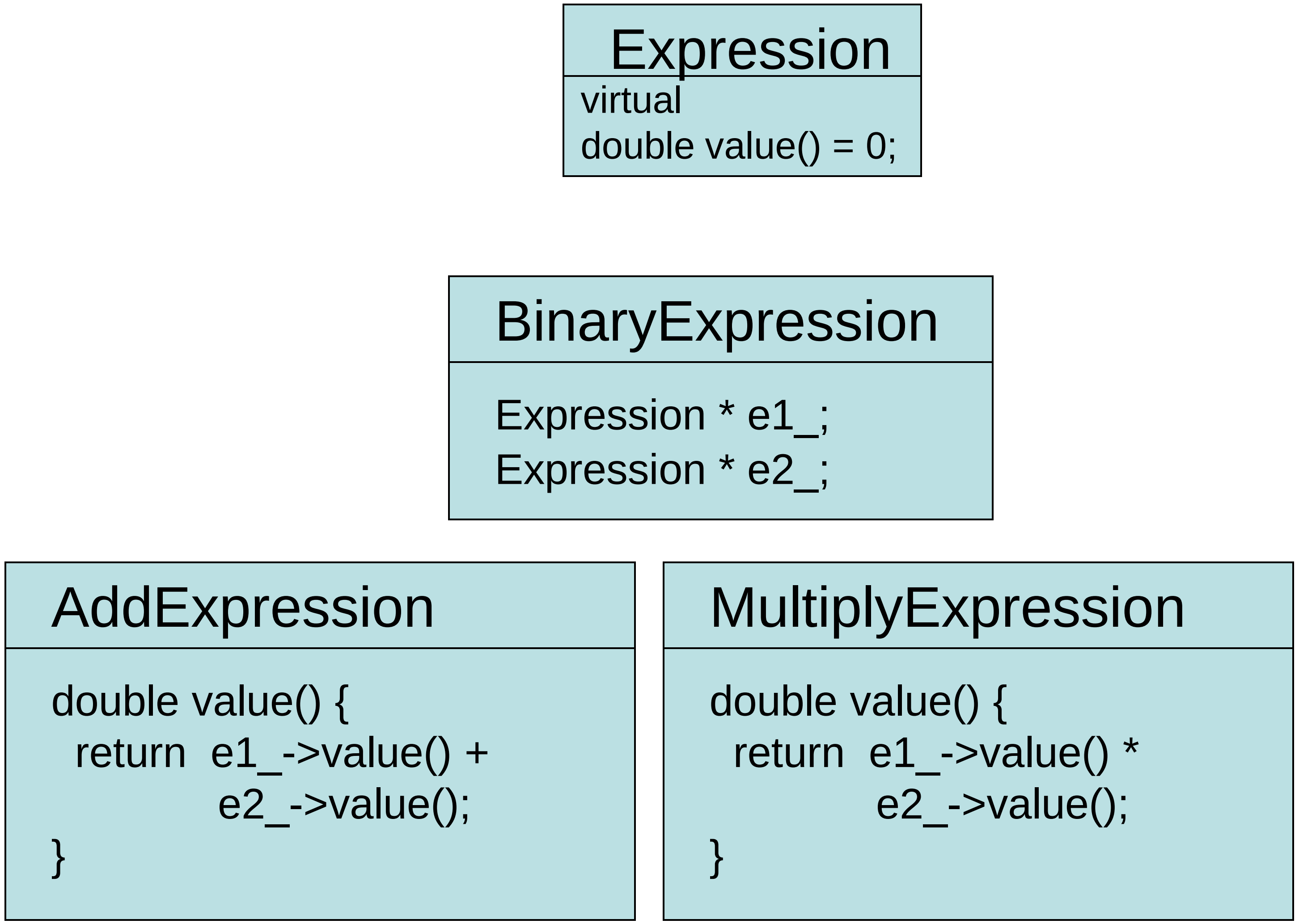
- Programming language for multistate design
- Six commands
  - STATE <name> <pdb> <corrfile> <2º resfile>
  - STATE_VECTOR <name> <statevec_file>
  - VECTOR_VARIABLE …
  - SCALAR_EXPRESSION <varname > = *Expression*
  - VECTOR_EXPRESSION …
  - ENTITY_FUNCTION <varname > <efunc_file>
  - FITNESS *Expression*
- One command per line

Expression Syntax
(textbook)

- Expression :=
  Term |
  ☐Term [+,-] Expression
- Term :=
  Factor |
  Factor [*,/] Term
- Factor :=
  variable |
  literal |
  (Expression) |
  function(Expression [, Expression]*)

$$\begin{tikzcd}
\text{string} \arrow[d, "\textit{1. scan}"'] \\
\text{tokens} \arrow[d, "\textit{2. parse}"'] \\
\text{abstract syntax tree (AST)} \arrow[d, "\textit{3. traverse AST}"'] \\
\text{expression tree}
\end{tikzcd}$$

Expression Hierarchy



Expression

```
virtual
double value() = 0;
```

BinaryExpression

```
Expression * e1_;
Expression * e2_;
```

AddExpression

```
double value() {
  return  e1_->value() +
              e2_->value();
}
```

MultiplyExpression

```
double value() {
  return  e1_->value() *
              e2_->value();
}
```

Expression

- Auto-differentiable

```
ExpressionCOP
MultiplyExpression::differentiate(
   std::string const & varname
) const
{
   de1_dvar = e1_->differentiate(varname)
   de2_dvar = e2_->differentiate(varname)
   return new AddExpression(
       new MultiplyExpression( de1_dvar, e2 ),
       new MultiplyExpression( de2_dvar, e1 ) );
}
```

Expressions: functions

- Variety of already-existing functions:
  - Real valued
    - max, min
    - vmax, vmin (vector max, vector min)
    - ln, exp, sqrt, pow, abs
  - Boolean (returns 1.0 or 0.0)
    - and, or, not, gt, gte, lt, lte
  - Control-of-flow
    - ite (if, then else)

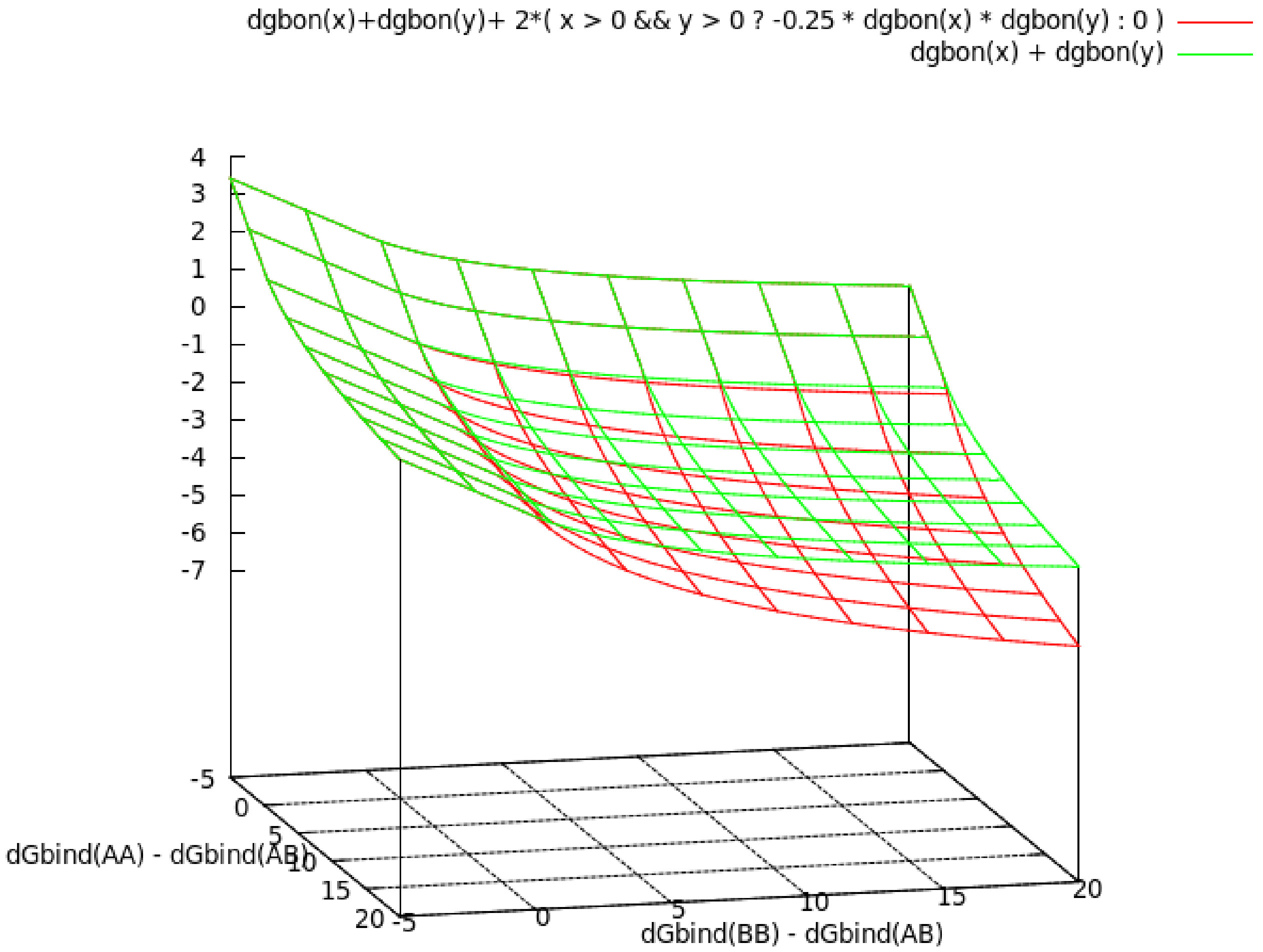
Example fitness file

```
STATE AB dimer.pdb    ab.corr    dimer.2res
STATE AA dimer.pdb    aa.corr    dimer.2res
STATE BB dimer.pdb    bb.corr    dimer.2res
STATE A    mon.pdb        a.corr     mon.2res
STATE B    mon.pdb        b.corr     mon.2res

SCALAR_EXPRESSION dGAB = AB - A - B
SCALAR_EXPRESSION dGAA = min( AA - 2 * A, 0.0 )
SCALAR_EXPRESSION dGBB = min( BB - 2 * B, 0.0 )

SCALAR_EXPRESSION ddGAA = dGAA - dGAB
SCALAR_EXPRESSION ddGBB = dGBB - dGAB

FITNESS AB + -1 * (  ddGAA + ddGBB )
```

Example fitness file

```
STATE_VECTOR vAB ab.states
STATE_VECTOR vAA aa.states
STATE_VECTOR vBB bb.states
STATE_VECTOR vA    a.states
STATE_VECTOR vB    b.states

SCALAR_EXPRESSION AB = vmin( vAB )
SCALAR_EXPRESSION AA = vmin( vAA )
SCALAR_EXPRESSION BB = vmin( vBB )
SCALAR_EXPRESSION A   = vmin( vA )
SCALAR_EXPRESSION B   = vmin( vB )

SCALAR_EXPRESSION dGAB = AB - A - B
SCALAR_EXPRESSION dGAA = min( AA - 2 * A, 0.0 )
SCALAR_EXPRESSION dGBB = min( BB - 2 * B, 0.0 )

SCALAR_EXPRESSION ddGAA = dGAA - dGAB
SCALAR_EXPRESSION ddGBB = dGBB - dGAB

FITNESS AB + -1 * (  ddGAA + ddGBB )
```

Fitness function search space



Sequence “constraints”

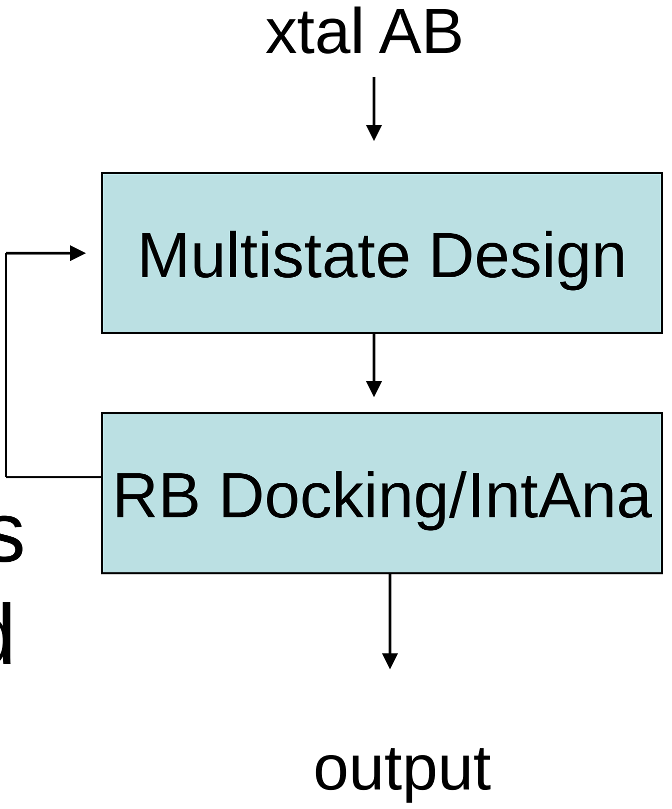
- Pairwise & Non-pairwise decomposable functions
  - “Mutate at most 5 of these 7 residues”
  - “Assign a different sequence to chain A and chain B”
  - “Put positive charge on chain A and negative charge on chain B”
  - “Favor beta-branched residues at position 116”
- “EntityFunction” from a file
  - Variables: ee_* for the entity elements
  - Commands:
    - AA_SET <aaset> = { [aa1let]* }
    - SET_CONDITON <varname> = <entity_element> in {[aa1let]*} | <aaset>
    - SUB_EXPRESSION <varname> = <Expression>
    - SCORE <Expression>

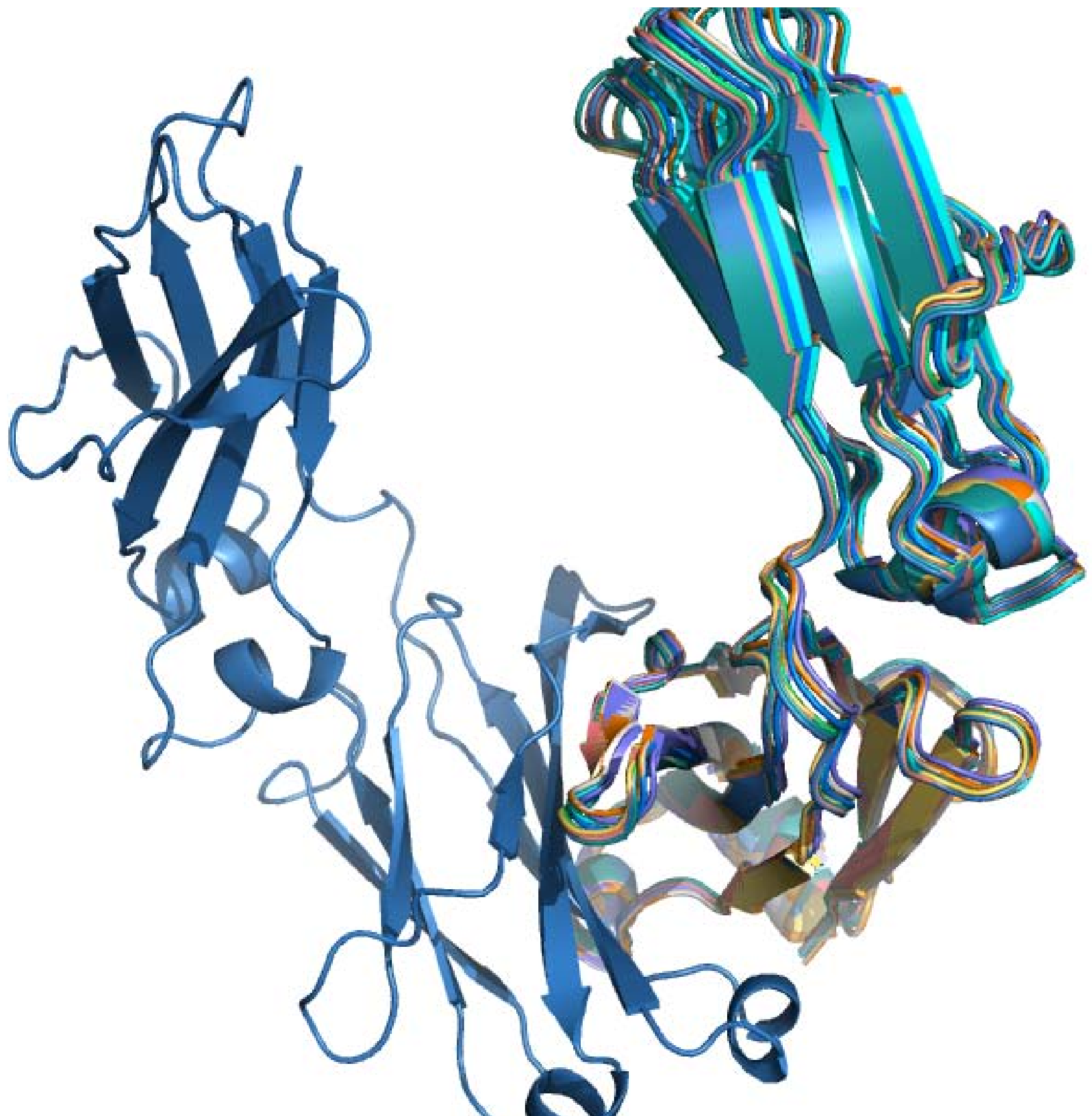
Outline

- MSD motivation
- MSD implementation
  - Control of flow / MPI
  - Fitness Function File Format
    - Expression parser (broadly useful)
  - Arbitrary Sequence Constraints (NPD)
- MSD example
  - Heterodimeric Antibody Design

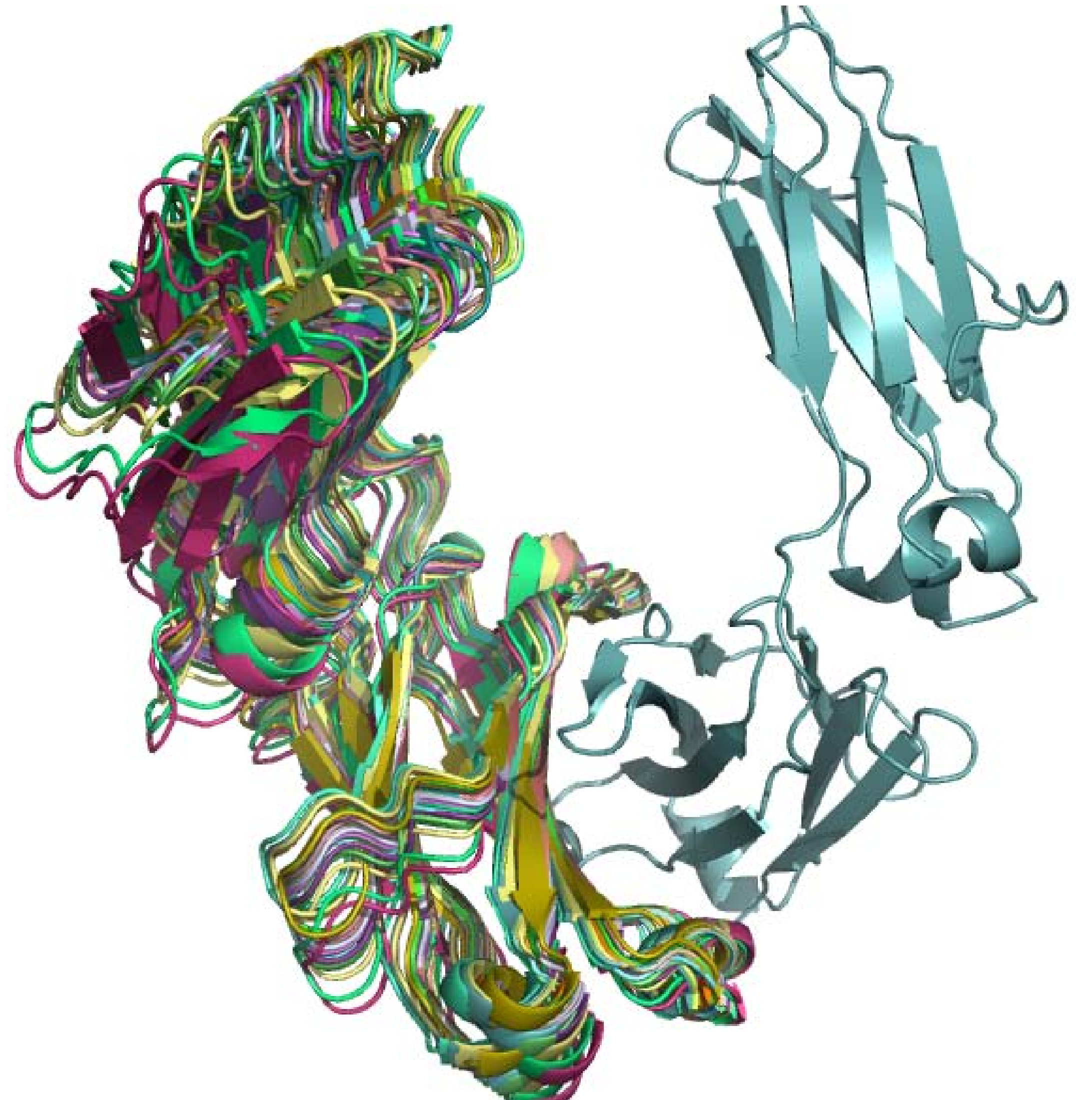
Heterodimeric Antibody Design

- Start with 1L6X xtal
- 1st iteration, take low energy docked homodimer conformations as negative states for 2nd iteration

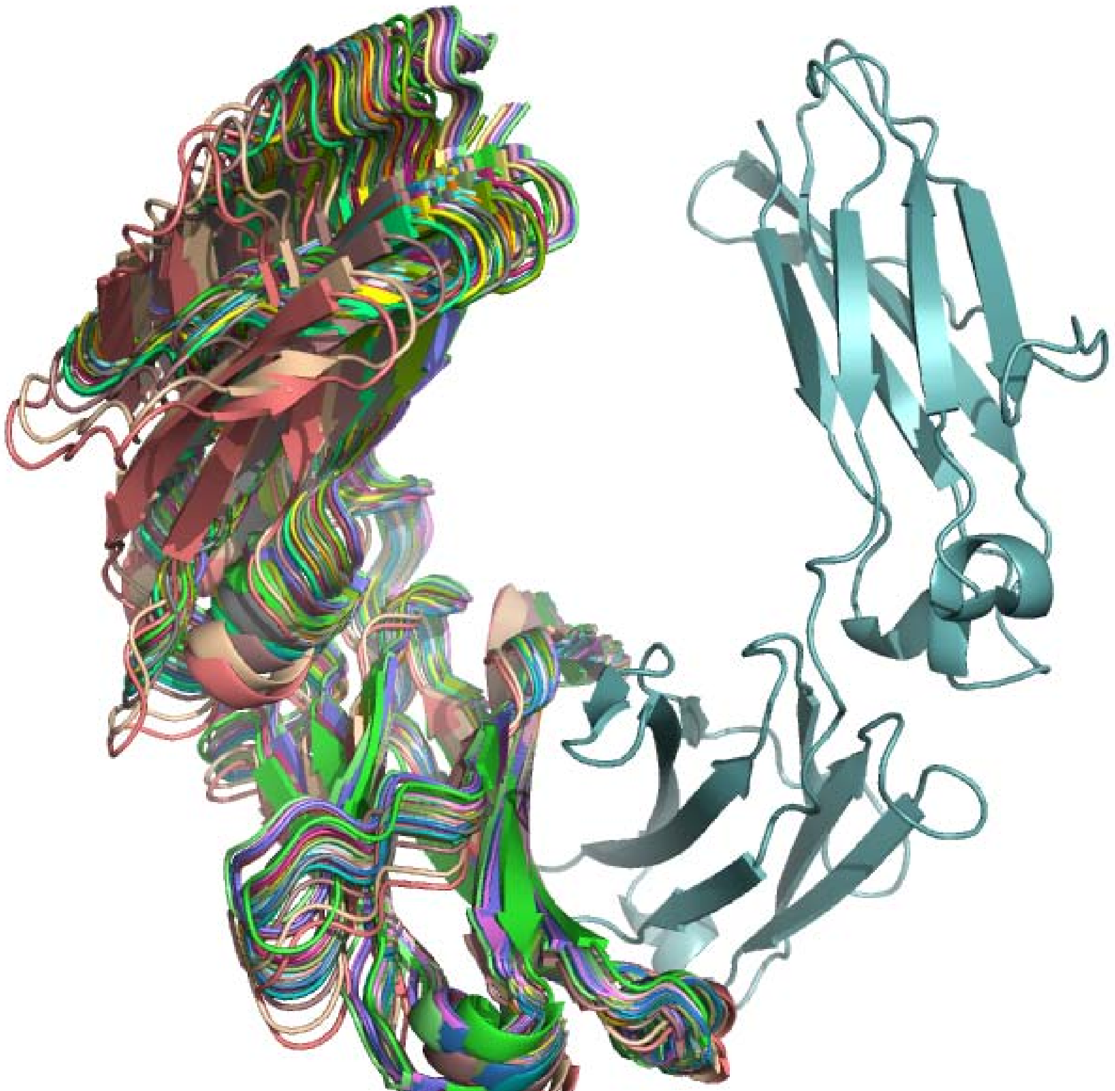




v4



v5



v6

Example Designs

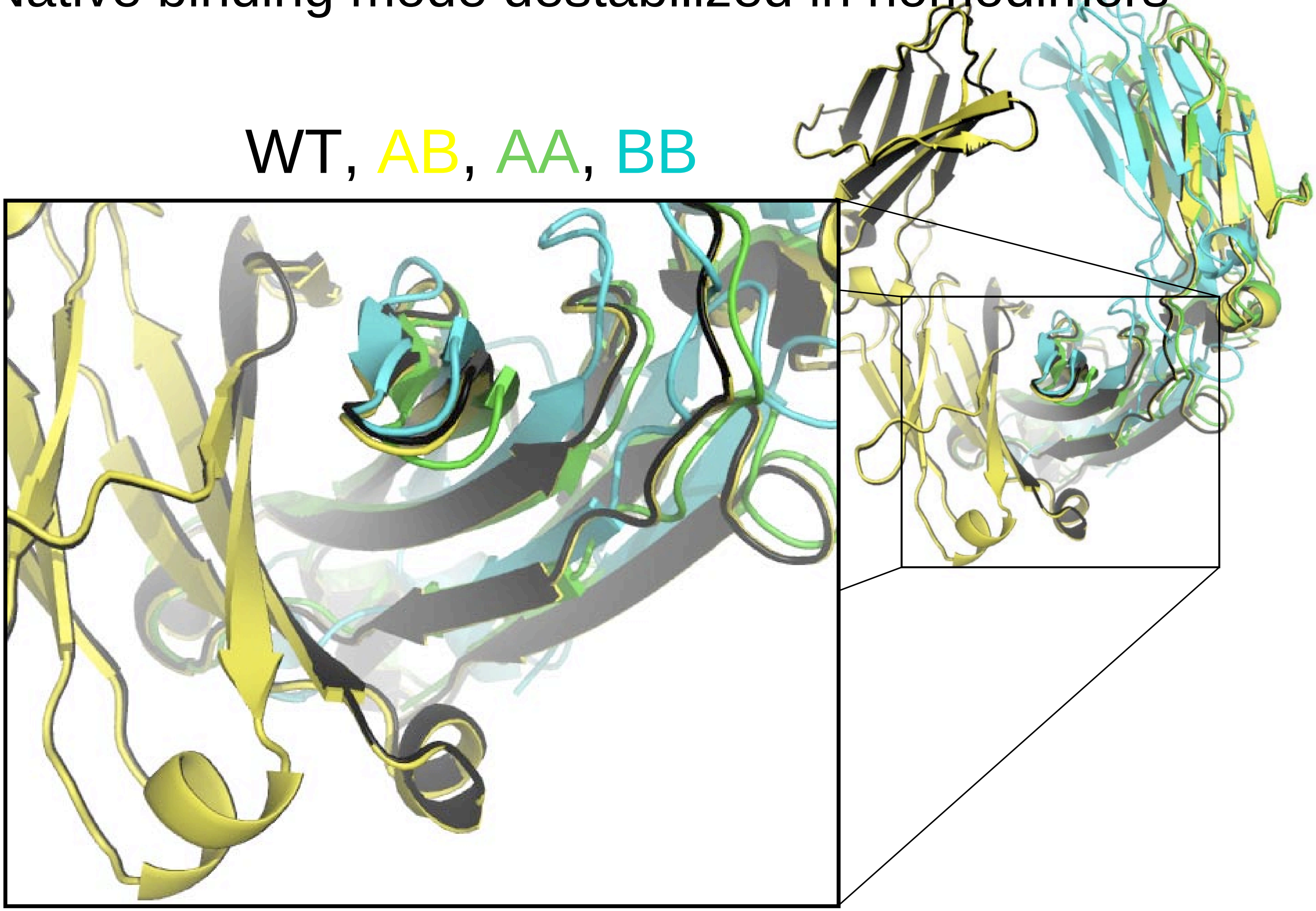
	Interaction Energies Following Rigid Body Docking (REU*)		
	AB	AA	BB
hetAB1	-35.5	-23.6	-18.4
hetAB2	-33.9	-20.2	-20.1
hetAB3	-34.1	-21.9	-21.8
hetAB4	-34.8	-17.5	-21.3

WT homodimer interface energy: -32.2

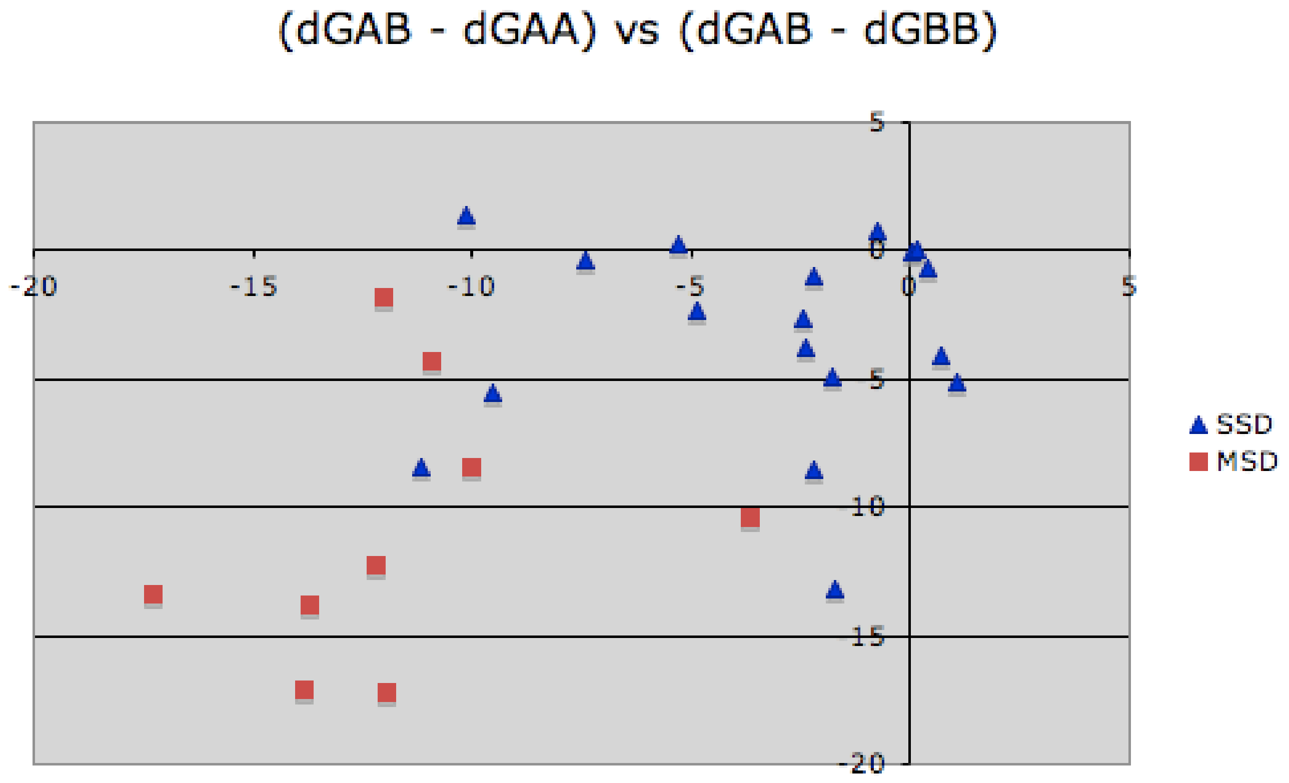
*score12 w/ -no_his_his_pairE

Native binding mode destabilized in homodimers

WT, AB, AA, BB



MSD Improves Specificity



Acknowledgements

- Bryan Der
- Brian Kuhlman

- Jim Havranek, Justin Ashworth & Colin Smith (GA code)

- Funding from NIH