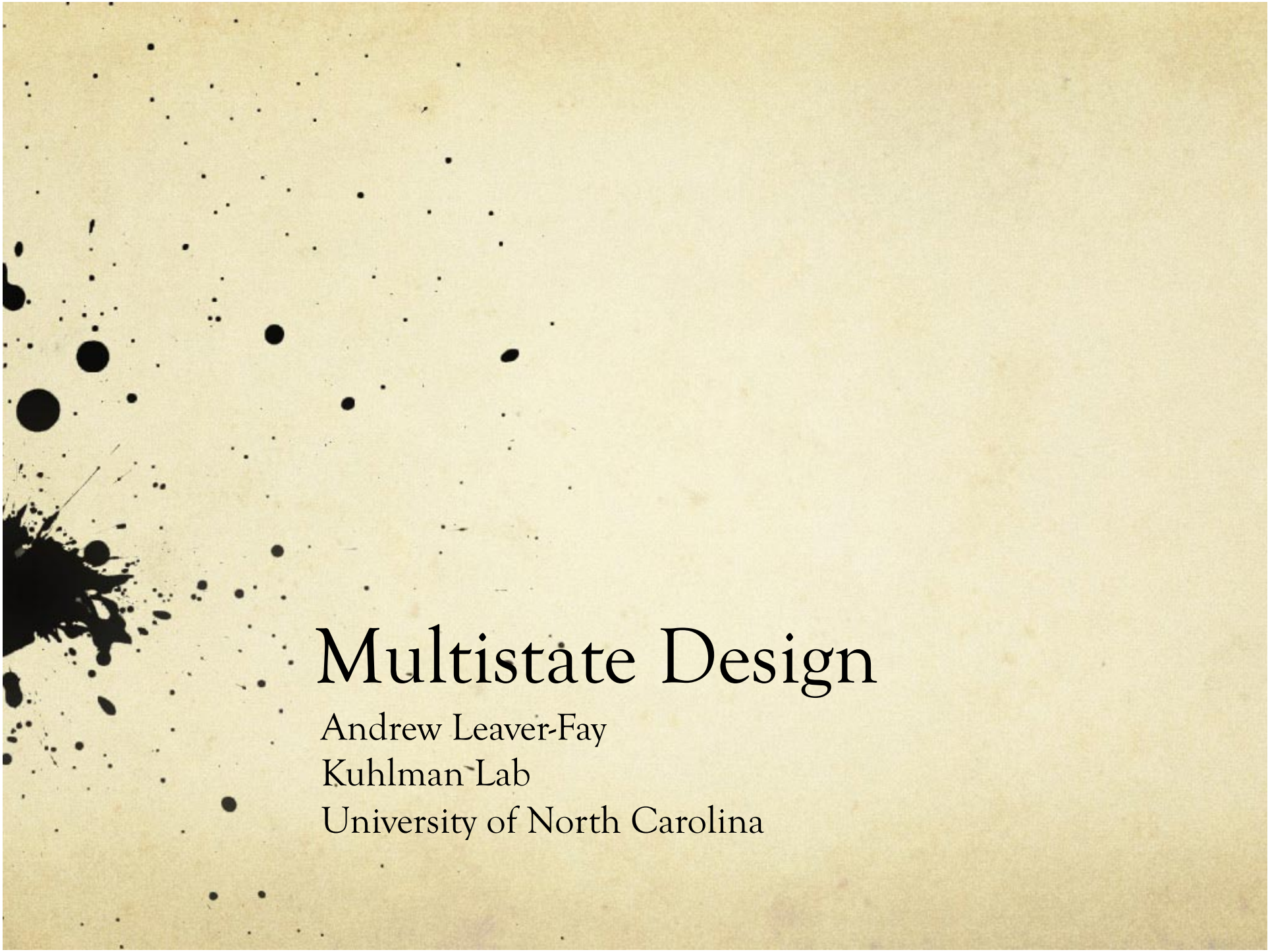


Multistate Design

Andrew Leaver-Fay
Kuhlman Lab
University of North Carolina

mpi_msd

- Very general framework for multistate design (MSD)
  - Can handle arbitrary many states
  - Can handle arbitrarily complicated MSD problems
- Necessarily complicated
  - “correspondence files”
  - “entity resfile”
- This talk:
  - Motivate the complexity
  - Explain the input files
  - Outline job-management

Single State Design?

- Optimize rotamers on a fixed backbone
  - pack_rotamers
  - Inner loop:
    - Pick random rotamer and try to substitute it in
    - Compute ΔE of rotamer substitution
    - Reject or accept rotamer substitution
- Iteratively optimize sequence and backbone structure
  - flxbb
  - Remodel
- Key: Only one conformation is considered at a time when the sequence is changed

Multistate Design

- Main purpose: design for more than one goal
- Design a(n):
  - Heterodimer from a homodimer
    - Homodimer → AB heterodimer (no AA or BB)
  - Sequence compatible w/ 2 (or more) loop conformations
  - Sequence that favors one loop conformation over another
  - Protein that binds two others
  - Orthogonal interface
    - Redesign promiscuous protein to bind only one partner
- Design for anything other than total energy: e.g.,
  - Binding energy
  - Δ buried unsatisfied hbond groups
  - Catalytically active rotamer should be lowest in energy

Multistate Design

- Implementation:
  - Fixed-backbone design
  - Search through sequence space
    - For each sequence, optimize its rotamers on each “state”
    - Evaluate each sequence based on the state energies after rotamer optimization
- Alternative (Bad) Implementation:
  - Build the “same” rotamers on all states
  - Pick a random rotamer; assign that rotamer to all states;
  - Compute fitness for sequence based on that rotamer assignment

Multistate Design

- Implementation:
  - Fixed-backbone design
  - Search through sequence space
    - For each sequence, optimize its rotamers on each “state”
    - Evaluate each sequence based on the state energies after rotamer optimization
- Alternative (Ok) Implementation:
  - Search through sequence space
    - For each sequence, optimize backbone and rotamers
    - (optional: constrained to starting backbone?)

Multistate Design

- Implementation:
  - Fixed-backbone design
  - Search through sequence space
    - For each sequence, optimize its rotamers on each “state”
    - Evaluate each sequence based on the state energies after rotamer optimization
- Alternative (Ok) Implementation:
  - Fixed-backbone, centroid design
  - Search through sequence space
    - For each sequence, thread centroids onto each state
    - Evaluate each sequence based on the state energies

Multistate Design

- Implementation:
  - Fixed-backbone design
  - Search through sequence space
    - For each sequence, optimize its rotamers on each "state"
    - Evaluate each sequence based on the state energies after rotamer optimization
- Alternative (Ok) Implementation:
  - Fixed-backbone, centroid design
  - Optimize sequences for multiple states simultaneously
    - [Grigoryan, Reinke, & Keating, 2009]

Picture

Node 0

"Head Node"

Tells all other nodes what sequence they need to examine

Node 1

Node 2

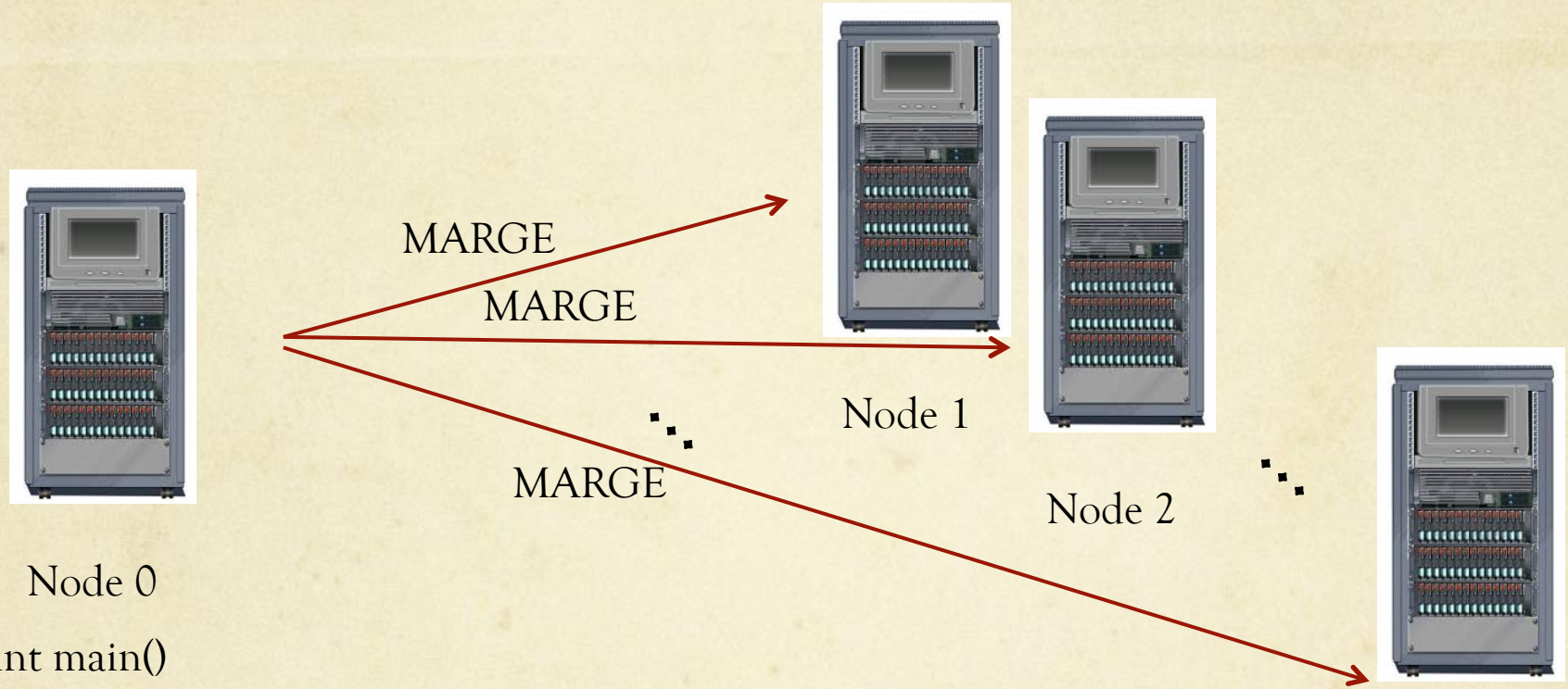
"Worker Node"

Waits for the head node to tell it what sequence to examine; then repacks that sequence on one (or more) states.

That is, each state is paired with a single node

Node n-1

Picture



int main()
  Setup_phase
  Genetic algorithm across sequence space
    Generate next generation of sequences
    Broadcast each sequence to worker nodes
    (Optimize rotamers for its states)
    Listen for worker nodes to return state energies
    Evaluate fitness for sequence based on state energies

Picture

"Pack Daemon"

Node 0

MARGE

MARGE

MARGE

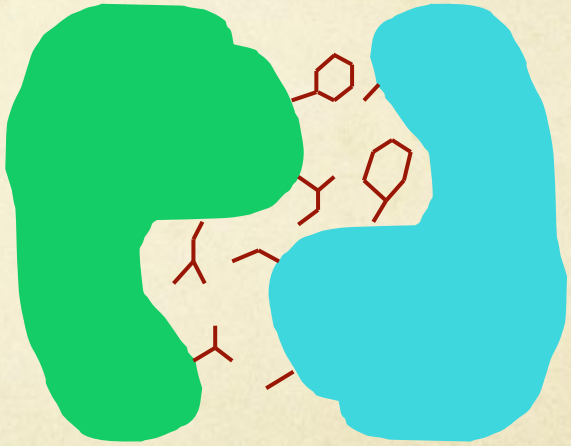
Node 1

Node 2

int main()
  Setup_phase
  while (true)
    Listen for sequences from node 0
    When sequence arrives,
      Optimize this sequence on each state
      (pack_rotamers)
    Send energies back to node 0

Example 1: Heterodimerization

- Take WT/WT homodimer
  - 4 residues on each side of the interface
- Design AB heterodimer (neither AA nor BB form)
- Model
  - A monomer
  - B monomer
  - AA homodimer
  - BB homodimer
  - AB heterodimer



Example 1: Heterodimerization

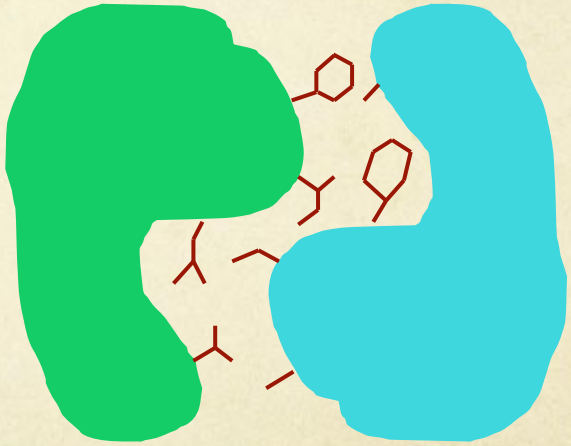
- Question: How long is the sequence string that node 0 will have to broadcast?
- Answer: 8
- Question: How will the node repacking the AB heterodimer know how to map between the sequence string that node 0 broadcasts and residues on the AB backbone?
- Answer: A *correspondence* must be provided

Example 1: Heterodimerization

- Correspondence file:
  - Broadcast sequence index → PDB id
    - PDB = chain, residue #, insertion code
    - E.g. “4 A 323”
  - A broadcast sequence index can be used multiple times
  - Not all broadcast sequence indices need to be used
- AB correspondence file:

1 A 321	5 A 325
2 B 321	6 B 325
3 A 323	7 A 327
4 B 323	8 B 327

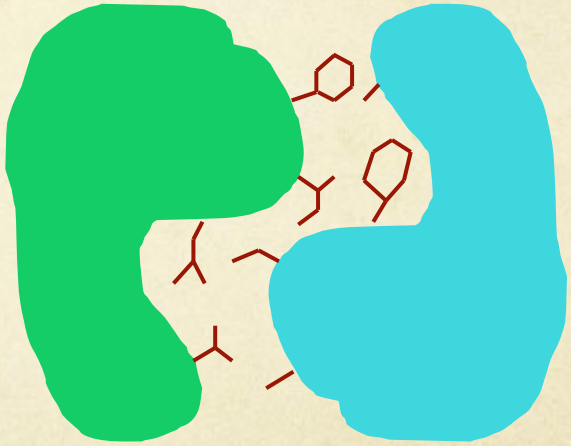
…



Example 1: Heterodimerization

- Correspondence file:
  - Broadcast sequence index → PDB id
    - PDB = chain, residue #, insertion code
    - E.g. “4 323 A”
  - A broadcast sequence index can be used multiple times
  - Not all broadcast sequence indices need to be used
- AB correspondence file:

1 321 A	5 321 B
2 323 A	6 323 B
3 325 A	7 325 B
4 327 A	8 327 B
…	



Example 1: Heterodimerization

- AB correspondence file:

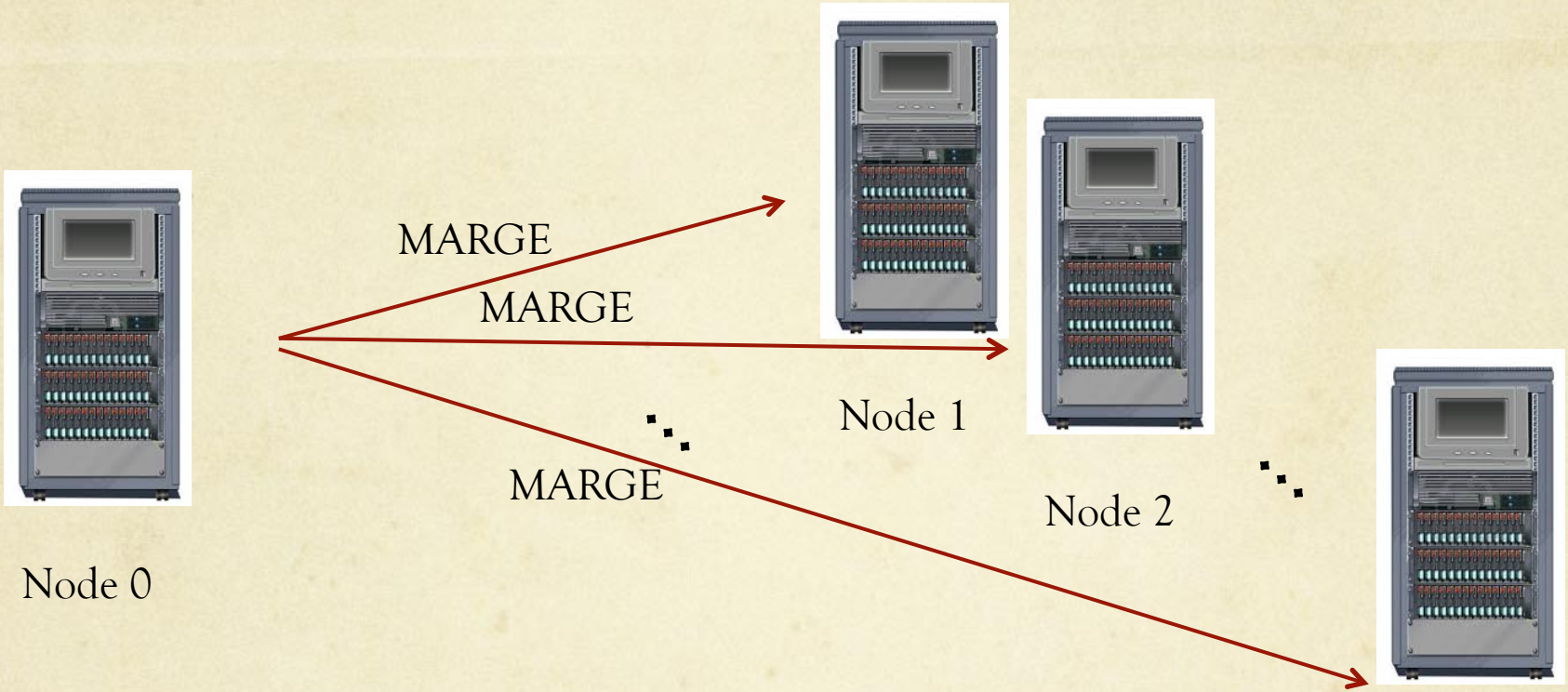
1 321 A	5 321 B
2 323 A	6 323 B
3 325 A	7 325 B
4 327 A	8 327 B

- Question: What does the AA correspondence file look like?
- Answer: AA correspondence file:

1 321 A	1 321 B
2 323 A	2 323 B
3 325 A	3 325 B
4 327 A	4 327 B

If AB is “A D E R F R A G”, then AA is “A D E R A D E R”

Picture



Example 1: Heterodimerization

- AB correspondence file:

1 321 A	5 321 B
2 323 A	6 323 B
3 325 A	7 325 B
4 327 A	8 327 B

- Question: What does the BB correspondence file look like?
- Answer: BB correspondence file:

5 321 A	5 321 B
6 323 A	6 323 B
7 325 A	7 325 B
8 327 A	8 327 B

Example 1: Heterodimerization

- AB correspondence file:

1 321 A	5 321 B
2 323 A	6 323 B
3 325 A	7 325 B
4 327 A	8 327 B

- Question: What does the A correspondence file look like?
- Answer: A correspondence file:

1 321 A
2 323 A
3 325 A
4 327 A

Example 1: Heterodimerization

- AB correspondence file:

1 321 A	5 321 B
2 323 A	6 323 B
3 325 A	7 325 B
4 327 A	8 327 B

- Question: What does the B correspondence file look like?
- Answer: B correspondence file:

5 321 A
6 323 A
7 325 A
8 327 A

Example 2: Multiple Loop Conformation Compatibility

- Let's say you have a loop that's crystalized in two different conformations and want to know…
- What other sequences support both conformations?
- (Let's say it's residues 10 through 20)
- Question: How many correspondence files do you need?
- Answer: 1
- Question: How many positions are being designed?

Example 2: Multiple Loop Conformation Compatibility

- Let's say you have a loop that's crystalized in two different conformations and want to know…
- What other sequences support both conformations?
- (Let's say it's residues 10 through 20)
- Question: What does the correspondence file look like?
- Answer:

1 10 A

2 11 A

3 12 A

…

Example 3: Ubiquitin Transfer Pathway

- Redesign ubiquitin and E1 so that it will transfer a mutant UBQ to an E2, but so that neither the mutant UBQ nor the mutant E1 cross react with their wild-type analogs
- A: Ubiquitin
- B: E1
- C: A particular E2

Example 3: Ubiquitin Transfer Pathway

- Redesign 5 residues on UBQ, 8 residues on E1
- Model:
  - Amut
  - Bmut
  - Amut Bmut
  - Awt Bmut
  - Amut Bwt
  - Bmut C

Example 3: Ubiquitin Transfer Pathway

- Amut Bmut correspondence

1 10 A	6 38 B	11 43 B
2 11 A	7 39 B	12 44 B
3 12 A	8 40 B	13 45 B
4 13 A	9 41 B	
5 14 A	10 42 B	

- Amut Bwt correspondence

1 10 A
2 11 A
3 12 A
4 13 A
5 14 A

Example 3: Ubiquitin Transfer Pathway

- Amut Bmut correspondence

1 10 A	6 38 B	11 43 B
2 11 A	7 39 B	12 44 B
3 12 A	8 40 B	13 45 B
4 13 A	9 41 B	
5 14 A	10 42 B	

- Awt Bmut correspondence

6 38 B	11 43 B
7 39 B	12 44 B
8 40 B	13 45 B
9 41 B	
10 42 B	

Example 3: Ubiquitin Transfer Pathway

- Amut Bmut correspondence

1 10 A	6 38 B	11 43 B
2 11 A	7 39 B	12 44 B
3 12 A	8 40 B	13 45 B
4 13 A	9 41 B	
5 14 A	10 42 B	

- Bmut C correspondence

6 38 B	11 43 B
7 39 B	12 44 B
8 40 B	13 45 B
9 41 B	
10 42 B	

Entity Resfile

- Entity:
  - The string that node-0 broadcasts
  - Justin Ashworth & Colin Smith's nomenclature
  - (I stole their genetic algorithm code)
- Entity Resfile
  - The resfile that describes the sequence space for these strings
  - A resfile except for one line at the beginning giving the length of the entity strings
  - Also describes rotamer sampling behavior

Example 1: Heterodimerization

- AB correspondence file:

  1 321 A 5 321 B
  2 323 A 6 323 B
  3 325 A 7 325 B
  4 327 A 8 327 B

- Entity Resfile:

  8
  ALLAAxC EX 1 EX ARO 2
  start
  # exclude C & H at 325
  3 A PIKAA ADEFGIKLMNPQRSTVWY EX1 EX ARO 2
  7 A PIKAA ADEFGIKLMNPQRSTVWY EX1 EX ARO 2

State

- A state is defined by three things:
  - A PDB file
  - A correspondence file
  - A secondary resfile
- The residues listed in the correspondence file take their instructions from the entity resfile
  - Disagreement on allowed AAs for such residues would not make sense, e.g. imagine
    - State 1: residue 10 corresponds to entity element 3
      - Allowed AAs: ADE
    - State 2: residue 10 corresponds to entity element 3
      - Allowed AAs: FGH
    - How does the GA decide what to assign entity element 3?
  - If the user needs to provide information, they ought only to provide it once

State

- A state is defined by three things:
  - A PDB file
  - A correspondence file
  - A secondary resfile
- Secondary Resfile
  - Resfile for all the other residues
  - “Design the core, repack the periphery”
    - List the core residues in the correspondence file
    - List the periphery residues in the secondary resfile
  - BEWARE: default PackerTask/resfile instruction is “redesign all residues with all amino acids”

Example 1: Heterodimerization

- AB Secondary resfile:

NATRO #very important: do not redesign the rest of the protein!
start
#repack the shell around the designed residues
322 NATAA EX 1 EX 2
328 NATAA EX 1 EX 2

Example 3: Ubiquitin Transfer Pathway

- AmutBmut.2res

NATRO
start
22 A NATAA
26 A NATAA
53 A NATAA
55 A NATAA
37 B NATAA
46 B NATAA
103 B NATAA
105 B NATAA
107 B NATAA

Reminder:
redesigning 10-14 on A
redesigning 38-45 on B

- AmutBwt.2res

NATRO
start
22 A NATAA
26 A NATAA
53 A NATAA
55 A NATAA
37 B NATAA
38 B NATAA
39 B NATAA
40 B NATAA
41 B NATAA
42 B NATAA
43 B NATAA
44 B NATAA
45 B NATAA
46 B NATAA
103 B NATAA
105 B NATAA
107 B NATAA

Example 3:
Ubiquitin Transfer Pathway

- AmutBmut.2res
  - NATRO
  - start
  - 22 A NATAA
  - 26 A NATAA
  - 53 A NATAA
  - 55 A NATAA
  - 37 B NATAA
  - 46 B NATAA
  - 103 B NATAA
  - 105 B NATAA
  - 107 B NATAA

Reminder:
redesigning 10-14 on A
redesigning 38-45 on B

- AwtBmut.2res
  - NATRO
  - start
  - 10 A NATAA
  - 11 A NATAA
  - 12 A NATAA
  - 13 A NATAA
  - 14 A NATAA
  - 22 A NATAA
  - 26 A NATAA
  - 53 A NATAA
  - 55 A NATAA
  - 37 B NATAA
  - 46 B NATAA
  - 103 B NATAA
  - 105 B NATAA
  - 107 B NATAA

Example 3: Ubiquitin Transfer Pathway

- AmutBmut.2res
  - NATRO
  - start
  - 22 A NATAA
  - 26 A NATAA
  - 53 A NATAA
  - 55 A NATAA
  - 37 B NATAA
  - 46 B NATAA
  - 103 B NATAA
  - 105 B NATAA
  - 107 B NATAA

Reminder:
redesigning 10-14 on A
redesigning 38-45 on B

- BmutC.2res
  - NATRO
  - start
  - 37 B NATAA
  - 46 B NATAA
  - 103 B NATAA
  - 105 B NATAA
  - 107 B NATAA
  - 120 C NATAA
  - 122 C NATAA
  - 124 C NATAA
  - 125 C NATAA
  - 127 C NATAA
  - 130 C NATAA
  - 131 C NATAA
  - 132 C NATAA
  - 135 C NATAA

State

- Three things define a state:
  - PDB file
  - Correspondence file
  - Secondary resfile
  - (Entity resfile)

Fitness Function

- Great, now you can define a state and an entity resfile. What should you do with them?
- Describe what makes a good sequence.
- ".daf" file:
  - Dynamic aggregate function
    - Aggregate function [Ashworth & Smith] aggregates the state energies
    - "Dynamic" as in, defined at runtime
  - Declares states
  - Define arbitrarily complicated fitness functions

Fitness Function

- Example 2: Multiple Loop Compatibility

STATE loop1 1abc.pdb loop.corr 1abc.2res
STATE loop2 1def.pdb loop.corr 1def.2res
FITNESS loop1 + loop2

- When you declare a state, you name it
  - Acts as a variable in later expressions
  - Assigned the value given by its energy under a particular amino acid assignment

Fitness Function

- Example 1: heterodimerization

```
STATE A 1bmf_chA.pdb A.corr A.2res
STATE B 1bmf_chA.pdb B.corr A.2res
STATE AB 1bmf.pdb AB.corr AB.2res
STATE AA 1bmf.pdb AA.corr AB.2res
STATE BB 1bmf.pdb BB.corr AB.2res

SCALAR_EXPRESSION dGAB = AB – A + B
SCALAR_EXPRESSION dGAA = AA – 2 * A
SCALAR_EXPRESSION dGBB = BB – 2 * B

FITNESS AB + dGAB – dGAA – dGBB
```

Fitness Function

- "scalar expression"
  - Creates a new variable
    - Scalar variable, as opposed to a vector variable
  - For each sequence examined, this variable's value will be computed from the expression on the right-hand side of the "=" sign
  - RHS can be an arbitrarily complicated mathematical expression
    - +, -, *, /
    - Math functions: sqrt, abs, exp, ln, min, max
    - Logical functions: < (lt), > (gt), <= (lte), gte, equals, not, and, or
    - Turnary function (if,then,else), "ite"
  - Evaluated in the order they are declared

Fitness Function

- Example 1: heterodimerization

```
STATE A 1bmf_chA.pdb A.corr A.2res
STATE B 1bmf_chA.pdb B.corr A.2res
STATE AB 1bmf.pdb AB.corr AB.2res
STATE AA 1bmf.pdb AA.corr AB.2res
STATE BB 1bmf.pdb BB.corr AB.2res

SCALAR_EXPRESSION dGAB = AB – A + B
SCALAR_EXPRESSION dGAA = AA – 2 * A
SCALAR_EXPRESSION dGBB = BB – 2 * B

FITNESS AB + dGAB – dGAA – dGBB
```

Note: binding energy can never be positive

Fitness Function

- Example 1: heterodimerization

```
STATE A 1bmf_chA.pdb A.corr A.2res
STATE B 1bmf_chA.pdb B.corr A.2res
STATE AB 1bmf.pdb AB.corr AB.2res
STATE AA 1bmf.pdb AA.corr AB.2res
STATE BB 1bmf.pdb BB.corr AB.2res

SCALAR_EXPRESSION dGAB = AB – A + B
SCALAR_EXPRESSION dGAA = min( AA – 2 * A, 0 )
SCALAR_EXPRESSION dGBB = min( BB – 2 * B, 0 )

FITNESS AB + dGAB – dGAA – dGBB
```

Fitness Function

- Example 1: heterodimerization
- Output:
  - At conclusion, MSD outputs a PDB for each structure that contributes to the fitness

SCALAR_EXPRESSION dGAB = AB – A + B
SCALAR_EXPRESSION dGAA = min(AA – 2 * A, 0)
SCALAR_EXPRESSION dGBB = min(BB – 2 * B, 0)

FITNESS AB + dGAB – dGAA – dGBB

Fitness Function

- Example 1: heterodimerization
- Output:
  - At conclusion, MSD outputs a PDB for each structure that contributes to the fitness

SCALAR_EXPRESSION dGAB = AB – A + B
SCALAR_EXPRESSION dGAA = min(AA – 2 * A, 0 * AA)
SCALAR_EXPRESSION dGBB = min(BB – 2 * B, 0 * BB)

FITNESS AB + dGAB – dGAA – dGBB

Fitness Function

- STATE_VECTOR
  - Declare multiple states in one file
  - Convenience
  - Multiple conformations for one chemical species
- STATE_VECTOR <variable_name> <list_file>
  - List file
    - Each line declares a state
    - <pdbname> <corrfilename> <2resfilename>
- Helpful functions: vmin, vmax
  - Get the lowest energy out of a vector variable

Extra Features

- Entity constraint file
  - Allows a score based purely on the sequence
  - Useful when interested in biasing towards native
- Hooks for arbitrary after-packing score calculations
  - NPD_PROPERTY <varname> <statename> <property>
  - E.g. How many buried unsatisfied polars are there?
  - Pose given to such calculators; score returned.
  - Returned score will be assigned to a variable and can then be used as part of the fitness function

Extra Features

- Entity constraint file
  - Set containment for “entity elements” (string positions)

SET_CONDITION ee15nat = ee_15 in { K }

SET_CONDITION ee15charged = ee_15 in { D, E, K, R }

  - ee_* variables defined automatically, one for as many positions there are in the entity strings
  - Assigned the 1-letter amino acid code
  - Sub expressions (scalar expressions) valid:

SUB_EXPRESSION nnat = ee1nat + ee2nat + ee3nat …

SUB_EXPRESSION nmut = 22 – nnat;

  - Must conclude with a “SCORE”

#pentaly for >4 mutations

SCORE ite(gt(nmut, 4), nmut – 4, 0)

Recommended flags

- (Write this down)

-use_input_sc
-preserve_c_beta
-ms::fraction_by_recombination 0.02
-mute core.pack.annealer.MultiCoolAnnealer
-unmute protocols.pack_daemon
-msd:double_lazy_ig_mem_limit 1024

Job Management

- mpi_msd
  - Tedious to manually create its input files, but
  - Offers a very regular input file structures
- For sufficient sampling, write a python wrapper around your MSD jobs
  - Treat mpi_msd like pack_rotamers
  - Job control is programming
  - Make sure you have a job-submission system (e.g. LSF) that allows you to chain jobs together.
- Use GIT to version control your job-control scripts

Job Management

- Organize your jobs by what kinds of things will change
  - Fitness file structure
  - Weights (constants) in your fitness file
  - State set
    - PDB and which species map to which PDBs
  - Design set
    - What positions are being mutated
    - Correspondence files
    - Secondary resfiles
    - Entity constraint files