RTMin

Andrew Leaver-Fay

What is RTMin?

- Wang, Furman, & Baker 2005
- Iterate across the residues in the pose in a random order, visiting each residue once
  - Build rotamers for residue i
  - Iterate across each rotamer
    - Minimize the chi dihedrals on rotamer j
  - Take the best minimized rotamer j

Why was it slow in mini?

- Iterate across the residues in the pose in a random order, visiting each residue once
  - ☐Build rotamers for residue i
  - Iterate across each rotamer
    - Minimize the chi dihedrals on rotamer j
  - Take the best minimized rotamer j

1. Replace residue O(N)*
2. Setup for minimization O(N)
   1. MinimizerMap setup
   2. AtomTree traversal
   3. Neighborlist construction
3. Score function evaluation during minimization O(N)*
   1. CD2B terms always evaluated between non-moving residue pairs
   2. All edges of the energy graph traversed
   3. Refold starts at root *

Stumbling block:
Energy methods compute derivatives using a Conformation (a Pose).

Why is it fast now?

- Iterate across the residues in the pose in a random order, visiting each residue once
  - ☐Build rotamers for residue i
  - Iterate across each rotamer
    - Minimize the chi dihedrals on rotamer j
  - Take the best minimized rotamer j

1. ~~Replace residue O(N)*~~
2. Setup for minimization O(1)
   1. SCMinMinimizerMap O(1)
   2. AtomTree traversal O(1)
   3. Neighborlist construction O(1)
3. Score function evaluation during minimization O(1)

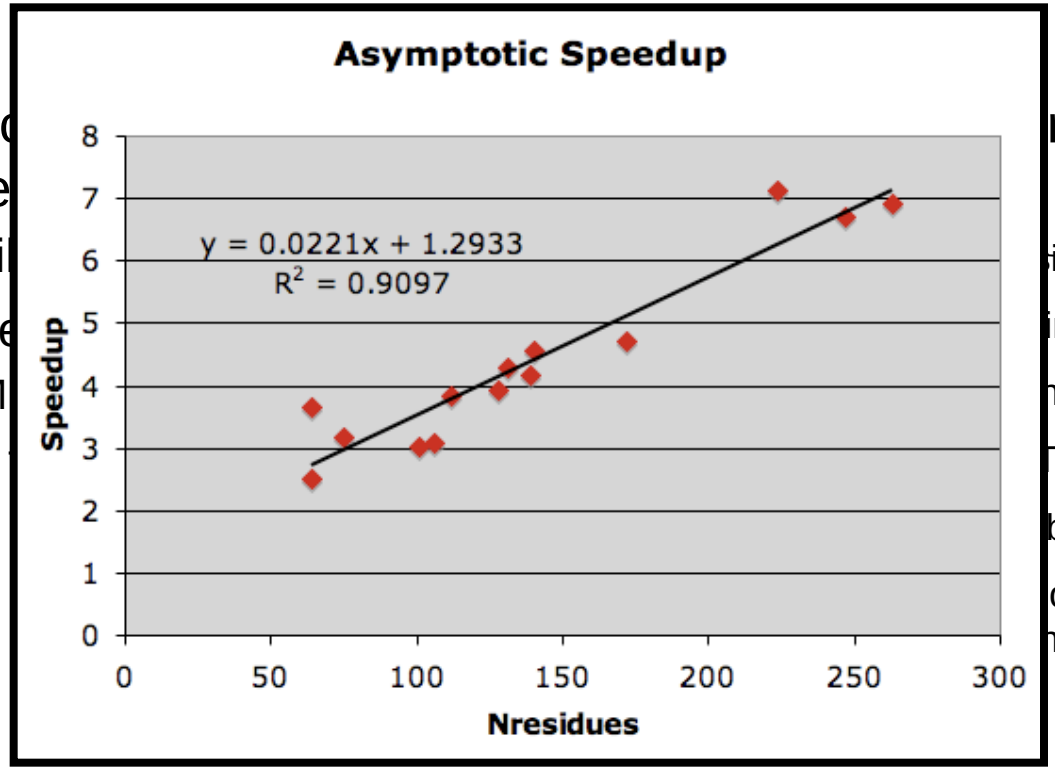
Key difference:
derivative evaluation performed without a Pose

Why is it fast now?

- Iterate ac rder, visiting e
  - ☐Bui idue ~~O(N)*~~
  - Iterate inimization O(1)
    - M nMinimizerMap O(1)
  - Take Tree traversal O(1)

borlist construction O(1)

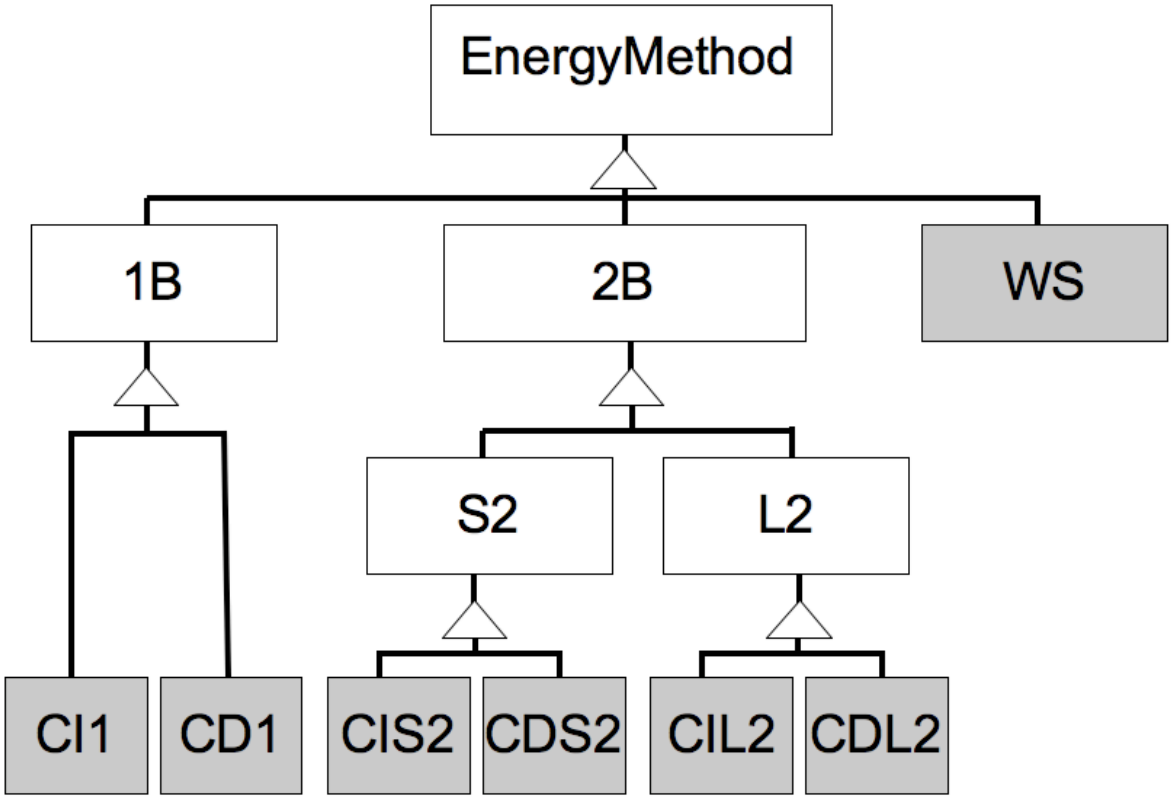
on evaluation during n O(1)



Key difference:
derivative evaluation performed without a Pose

How?

- Review: EnergyMethod hierarchy



How?

- Packer: requires residue-pair decomposable scoring
  - OneBodyEnergy::eval_residue_energy
  - TwoBodyEnergy::residue_pair_energy

- RTMin: requires residue-pair decomposable derivative evaluation
  - OneBodyEnergy::eval_atom_deriv_for_residue
  - TwoBodyEnergy::eval_atom_deriv_for_residue_pair

What does this change?

- EnergyMethods
  - Derivative evaluation for all ~170 terms
  - Score12 implemented
- Minimization
- Symmetric Minimization

MinimizationGraph

- MinimizationNodes cache residue data
- MinimizationEdges cache residue-pair data
  - HBonds for hbond derivative evaluation
  - Etable residue-pair neighbor list
    - (Auto-update on by default, but no new edges)
  - Constraints for a particular residue pair
- Fixed-sequence assumptions allowed

Residue-pair decomposable derivative evaluation

- TwoBodyEnergy::eval_atom_deriv_for_residue_pair(

```
    Size atom_index,
    Residue const & rsd1,
    Residue const & rsd2,
    ResSingleMinimizationData const & r1dat,
    ResSingleMinimizationData const & r2dat,
    ResPairMinimizationData const & respairdat
    EnergyMap const & weights,
    Vector & F1,
    Vector & F2
) const;
```

Example Conversion to Poseless derivatives

```
///
Real
OmegaTetherEnergy::eval_dof_derivative(
   id::DOF_ID const &,// dof_id,
   id::TorsionID const & tor_id,
   pose::Pose const & pose,
   ScoreFunction const &,// sfxn,
   EnergyMap const & weights
) const
{
   Real deriv(0.0);
   if ( tor_id.valid() && tor_id.type() == id::BB ) {
      conformation::Residue const & rsd( pose.residue( tor_id.rsd() ) )
      if ( rsd.is_protein() &&
             tor_id.torsion() == 3 ) {
         Real omega_score, dscore_domega;
         potential_.eval_omega_score_residue( rsd, omega_score,
            dscore_domega );
         deriv = dscore_domega;
      }
   }
   // note that the atomtree Oomega dofs are in radians
   // use degrees since dE/dangle has angle in denominator
   //return numeric::conversions::degrees( weights[ omega ] * deriv );
   return numeric::conversions::degrees( weights[ omega ] * deriv );
}
```

```
Real
OmegaTetherEnergy::eval_residue_dof_derivative(
   conformation::Residue const & rsd,
   ResSingleMinimizationData const &,
   id::DOF_ID const &,
   id::TorsionID const & tor_id,
   pose::Pose const &,
   ScoreFunction const &,
   EnergyMap const & weights
) const
{
   Real deriv(0.0);
   if ( tor_id.valid() && tor_id.type() == id::BB &&
         tor_id.torsion() == 3  && rsd.is_protein() ) {
      Real omega_score, dscore_domega;
      potential_.eval_omega_score_residue( rsd, omega_score, dscore_dome
      deriv = dscore_domega;
   }
   return numeric::conversions::degrees( weights[ omega ] * deriv );
}
```

Opt-in bifrication of scoring

- TwoBodyEnergy::residue_pair_energy(…)
- TwoBodyEnergy::residue_pair_energy_ext(…)
  - EneryMethods may opt-in by answering:
    bool use_extended_residue_pair_energy_interface()

Awesome future functionality

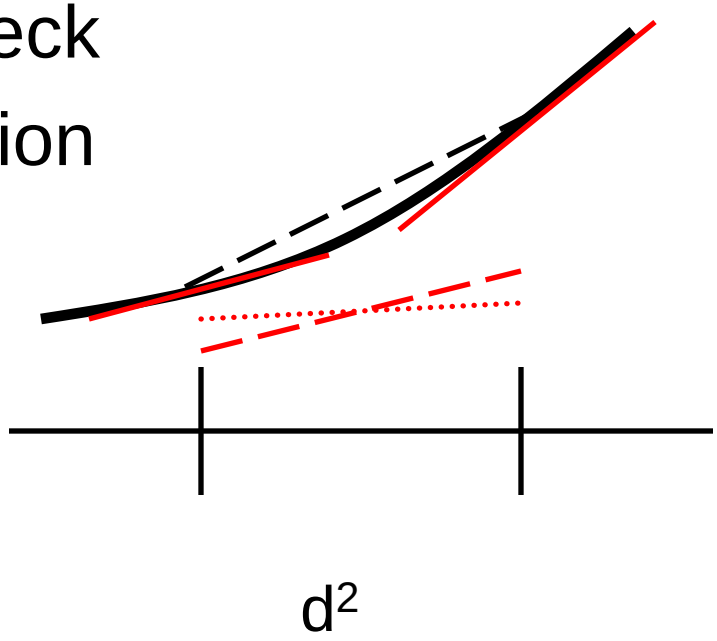
- per-residue reweighting
  - Turn off a residue (Mike)
  - Trp, tyr, phe upweighted
- per-residue-pair reweighting
  - Upweight inter-chain interactions
- Pack-minimizer
  - Minimize after each sidechain substitution

The Plan

- Convert -correct terms to the new scheme
  - Score12 + constraints are already done
- Grandfather remaining energy methods using the old eval_atom_deriv() interface
  - still used by whole-structure energies
- Merge to trunk
  - All integration tests will break
  - Tons of new unit tests added showing norm==norm_numeric
- Gradually modernize remaining energy terms
  - Terms incompatible with rtmin until modernization

Random thoughts

- dljatr_
  - Screws up deriv check
  - Improves minimization
    - Faster
    - Better energies



Acknowledgements

- Phil Bradley & Jim Havranek
- Frank DiMaio & Ingemar Andre
- Brian Kuhlman