Protein folding via divide-and-conquer optimization

Ricardo Oliva

collaborators

Silvia Crivelli, **Juan Meza**

Computational Sciences Division

Lawrence Berkeley National Laboratory

Protein-folding via numerical optimization

Working assumption:
The "natural" conformation of a protein corresponds to a configuration that minimizes an energy potential.

This premise brings the protein-folding problem into the realm of numerical optimization algorithms (e.g. LBFGS)

Compute an X^* that minimizes $E(X)$,
where X is the vector of atom coordinates,
and E is a potential energy function (e.g. Amber).

This is a challenging problem:
- Potential function E is only a model.
- Large-scale problem (size 10^3--10^6)
- Many local minima.

Amber Energy Potential (Model)

$$E_{AMBER} = E_{Bonds} + E_{Angles} + E_{Dihedrals} + E_{NonBonded}$$

$$E_{Bonds} = \sum_{Bonds} B_i (r_i - \bar{r}_i)^2$$

$$E_{Angles} = \sum_{Angles} A_i \left(\theta_i - \bar{\theta}_i\right)^2$$

$$E_{Dihedrals} = \sum_{Dihedrals} D_i \left(1 + \cos(n_i \phi_i - \delta_i)\right)$$

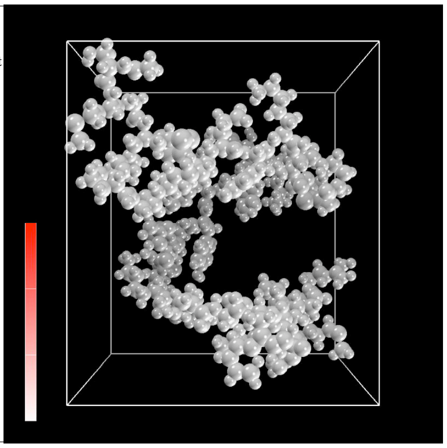
$$E_{NonBonded} = \sum_i \sum_{j>i} \left(\varepsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{r_{ij}} \right)$$

Movie

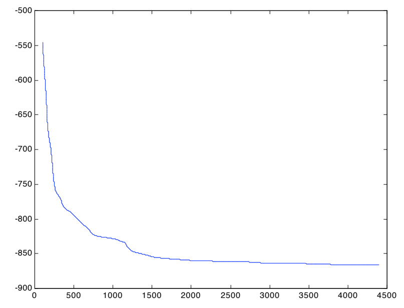
with **Cristina Siegerist** (Visualization Group)

Color ~ ‖ Δ posn. ‖ (speed)

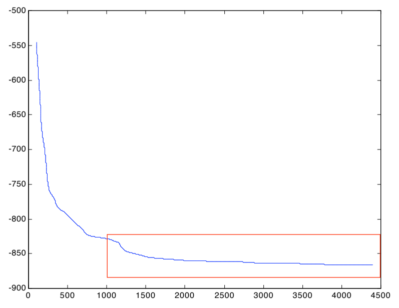
Two observations
- atoms move in clusters.
- slow "adjustment of positions" rather than large displacements



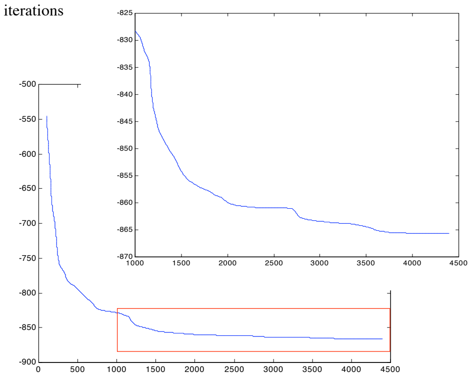
Energy vs iterations



Energy vs iterations



Energy vs iterations



Observation:

• Atoms appear to move slowly and in small clusters during numerical minimization process.

Idea: To “optimize” these clusters in parallel, keeping the other atoms fixed. Is is possible?

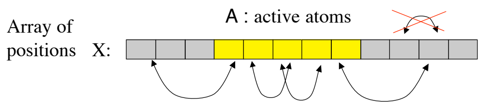
Questions:

• How to define clusters -- i.e. how to divide the atoms ?

• What’s the right energy function wrt these atoms.

Defining *E* w.r.t. a subset of "active atoms"

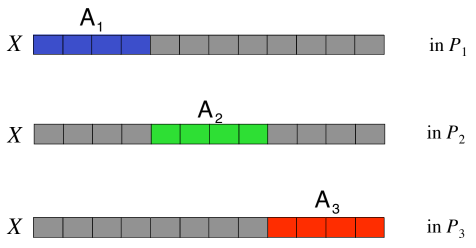
E(A;*X*) = Sum of all energy terms in *E(X)* that involve *at least one* atom in A



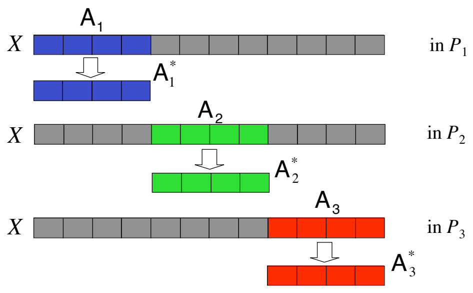
Basic "Divide and conquer" (parallel) optimization approach:

1. Distribute atoms among *P* processors:
   Subset A_i is *active* on P_i
2. In parallel, each P_i minimizes A_i using $E_i = E(A_i\,;X)$
3. Combine the results of each P_i.

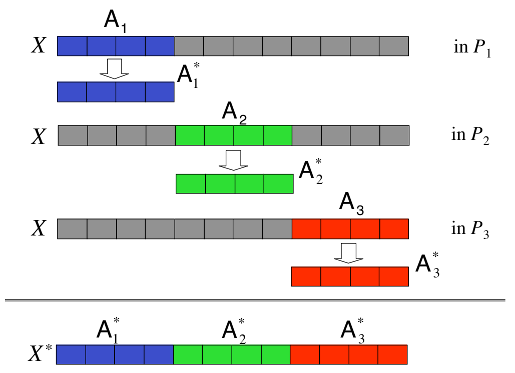
Basic "Divide and conquer" (parallel) optimization approach:



Basic "Divide and conquer" (parallel) optimization approach:



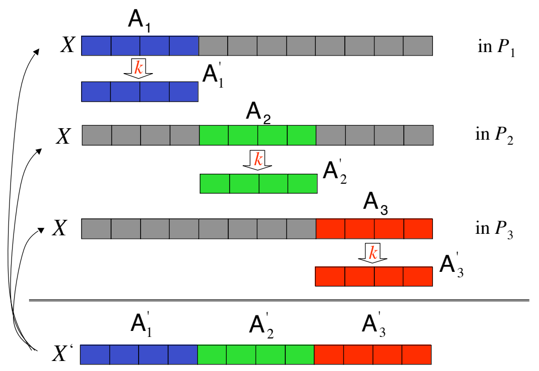
Basic "Divide and conquer" (parallel) optimization approach:



"Divide and conquer" (parallel) optimization with global updates:

1. Distribute atoms among P processors:
   Subset A_i is *active* on P_i
2. In parallel, each P_i lowers the energy of A_i (i.e. $E(A_i ; X)$ by performing a small number k of optimization iterations.
3. Combine results of each P_i on each process ("all-gather").
4. Stop upon convergence, else go to step 2 and repeat.

"Divide and conquer" (parallel) optimization with updates:

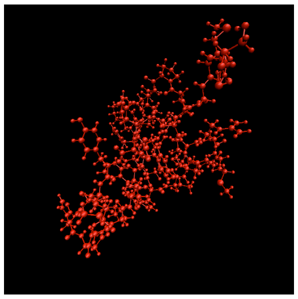


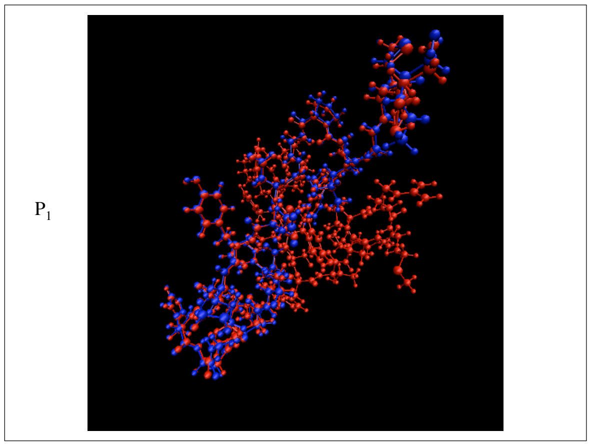
Example 1

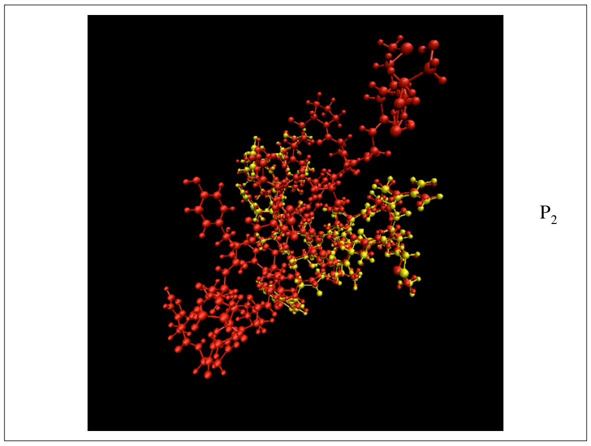
Protein 1e0m
593 Atoms
Initial E > 1e+6

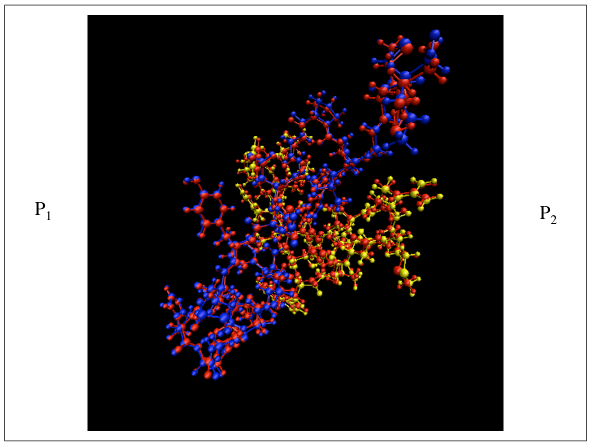
$k = 50$

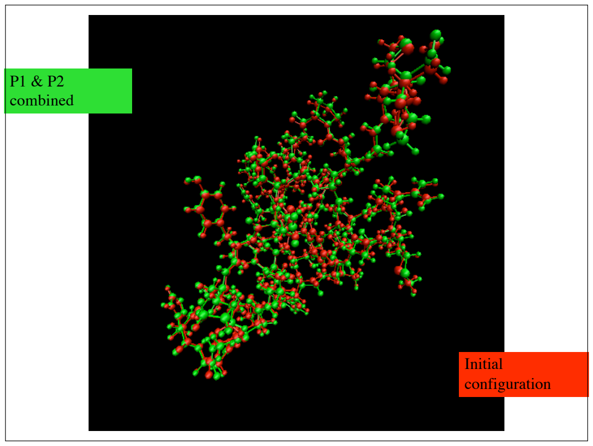
P = 2




P_1

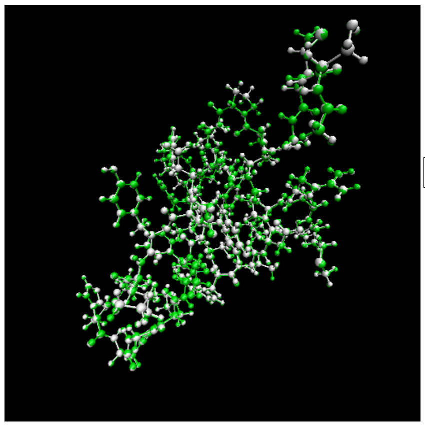

P_2


P_1
P_2


P1 & P2 combined
Initial configuration

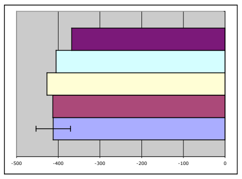
After 50it

with 2P



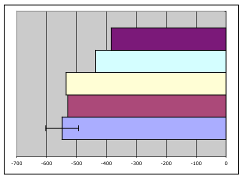
With 1P

How do energy values compare when parallel results are combined?



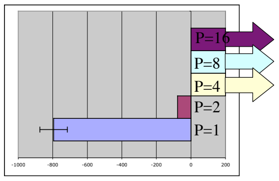
Energy @ 50 iterations

How do energy values compare when parallel results are combined?



Energy @ 100 iterations

How do energy values compare when parallel results are combined?

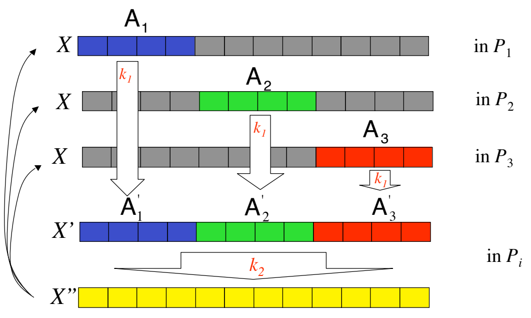


Energy @ 500 iterations

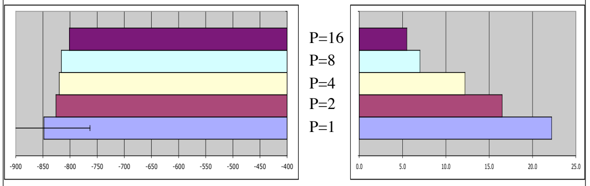
Divide and conquer optimization with correction steps:

1. Distribute atoms among P processors:
   Subset A_i is *active* on P_i
2. In parallel, each P_i lowers A_i using $E_i = E(\mathsf{A}_i\ ; X)$ by performing a small number k_1 of optimization iterations.
3. Combine the results of each P_i.
4. Correction Step: Carry on a small number k_2 of optimization iterations using on the full system $E(X)$.
5. Stop upon convergence, else go to step 2 and repeat.

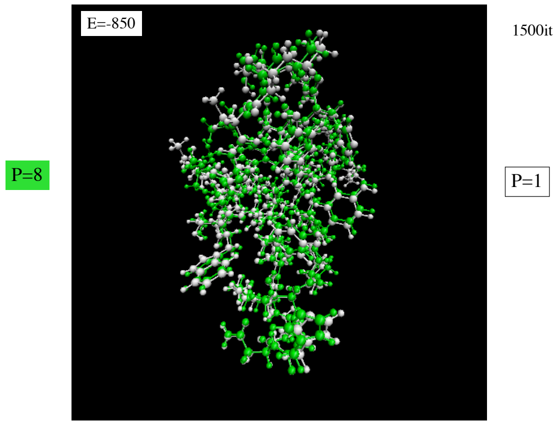
"Divide and conquer" (parallel) optimization with corrections:



Results on 1e0m (same protein as before)
using k_1=30, $k_2 = 3$:



Energy @ 3300 iter   Time (min)

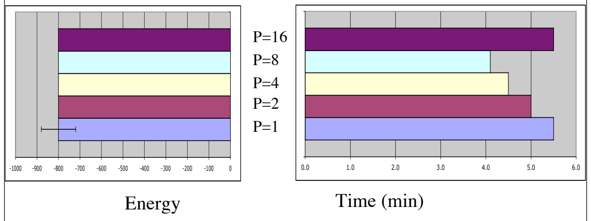


A caveat:

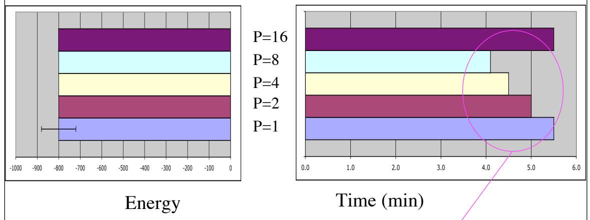
In parallel step, time of per iteration is reduced, but (total) energy drop per iteration is also lowered.

Q: can we balance these two effects and get significant reduction in time for a given energy value?

Time to reach E = -800



Time to reach E = -800

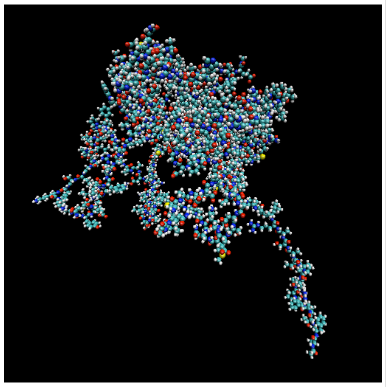


Gain can be significant for larger proteins…

Example 2

Large protein (T146)

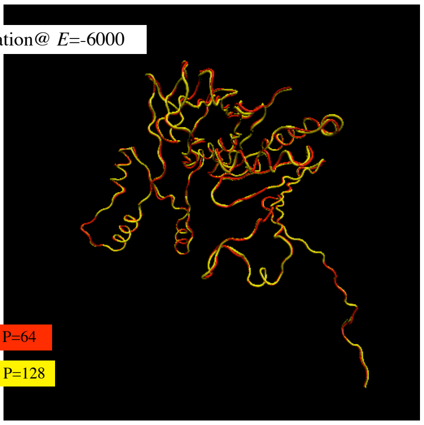
5053 atoms



Time to E=-6000 with k_1=30, k_2=3 :

P=128	51 min
P=64	49 min
P=1	**> 9 hrs!**

Configuration@ E=-6000



🟥 with P=64

🟨 with P=128

Conclusions:

• A parallel divide-and-conquer scheme with global corrections can significantly reduce the computational time required for lowering the (Amber) energy of some protein configurations.

• A few full-size optimization corrections appear to keep the parallel optimization in line with its serial equivalent, even for proteins as large as 5000 atoms.

• In general, the approach has two opposites effects:
1. Reducing the time per iteration, and
2. Reducing the energy drop per iteration,

with increasing number of processors (parallel scale issue).

Improvements & future work:

• More testing! (results are preliminary --only a few examples)

• Grouping atoms according to structure (by amino, or per coils, alpha-helix, or beta sheets) --should improve parallel E reduction.

• Using clusters of "active atoms" (e.g. using ||gradient||) --motivating idea.

• Partitioning protein by spatial location --some proteins come in multiple "lumps" of atoms.

• Developing better strategy for setting the parameters k_1, k_2 (possibly adapting these during optimization).

END