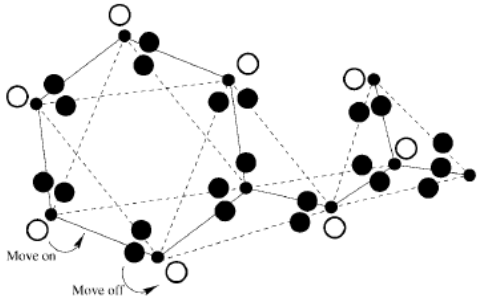
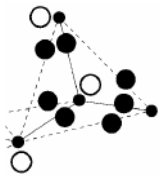


Protein Flexibility Predictions

Using Graph Theory

by D.J. Jacobs, A.J. Rader et. al. from the Michigan State University

Talk by Jan Christoph, 19th January 2007

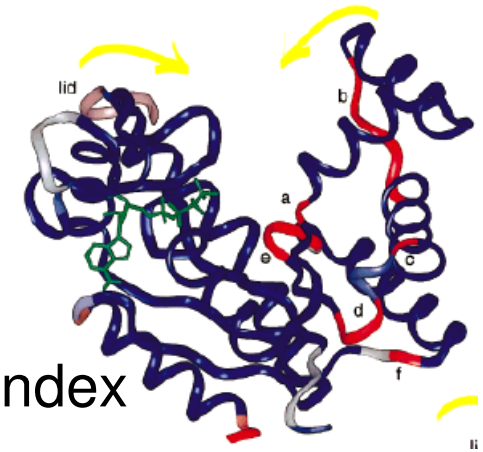


Outline

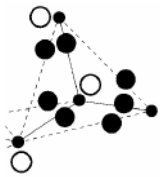
1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References

What do we want?

- Prediction of Protein Flexibilty:
  - Find rigid regions
  - Find strained regions
  - Find flexible regions
  - Do not just cluster: give a smooth index



- Features of method:
  - working with very few determined conformations
  - very fast
  - high correlation with experimental measures

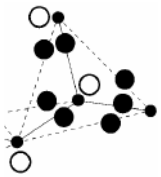


Why is this interesting to know?

The predicted result can be used:

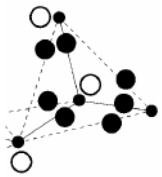
- As help for MD / Monte Carlo methods
- As starting point for Molecular Recognition

- To find important areas/targets



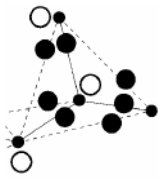
Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



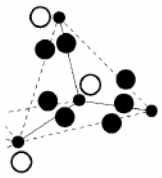
Different basic strategies

1. Compare different conformational states
   - e.g. from crystallographie or NMR
   - Problem: sufficient diversity
2. Simulate protein‘s motion
   - e.g. with molecular dynamics calculations
   - Problem: it requires a lot of time
3. Have a closer look at one structure
   - Build up a mathematic model
   - Identify rigid domains or flexible hinge joints
   - That is what we will do now!



Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



Protein as Bond-Bending Network (1)

- Consider just strong and directional forces:
  - covalent bonds
  - Salt bridges
  - hydrogen bonds
- Model these as distance and angle constraints in a network

Protein as Bond-Bending Network (2)

- Connectivity is defined by nearest-neighbor constraints.
- Angular constraints by next-nearest-neighbor distance constraints
- Fixed dihedral angles by third-nearest-neighbor distance constraints
- Dihedral rotation is elementary flexible element

  => **Degrees of freedom**

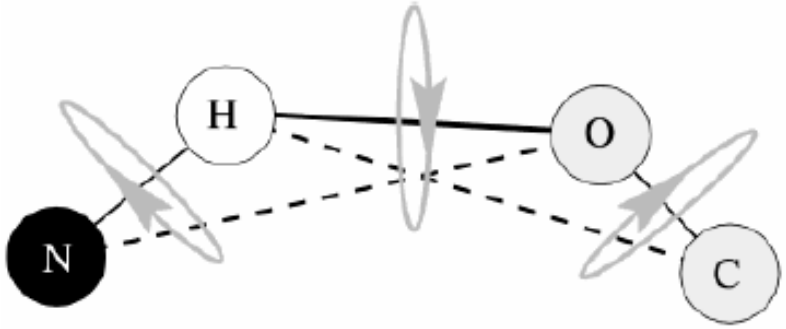
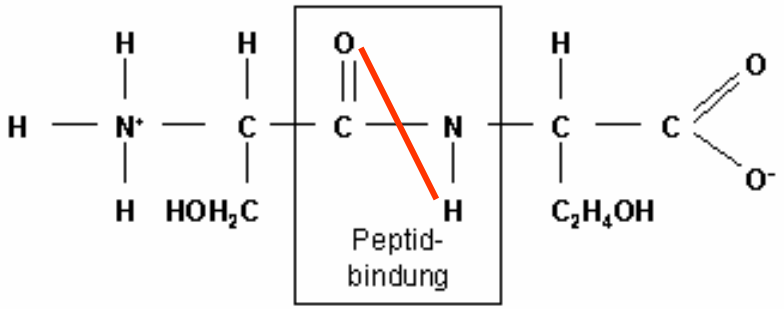
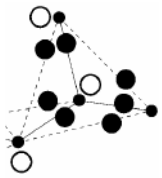


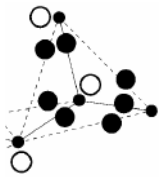
Fig. 1. Model of a hydrogen bond, involving donor and acceptor atoms, shown as nitrogen and oxygen, respectively. Covalent bonds are shown as thin black lines. The hydrogen bond is modeled as three distance constraints, consisting of a nearest-neighbor central-force constraint shown as a thick solid line (top center), and two next-nearest-neighbor bond-bending force constraints (constraining the donor–hydrogen–acceptor angle), shown as dashed lines. Each hydrogen bond is also associated with three a priori rotatable dihedral angles, indicated by the arrows.





Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



Matrix Diagonalization: brute-force (1)

- Fundamental: test whether a constraint is
  - **redundant:** breaking has no effect on flexibility
  - or **independent:** breaking does affect flexibility
- Test of all bonds (constraints)
  1. Define a network of atoms and distance constraints
  2. Replace each distance constraint by a spring
  3. Construct a dynamical matrix, calculate and count eigenvalues
  4. Add the bond to be tested and repeat 3.
  5. If number of zero eigenvalues remains, the bond is redundant, otherwise independent.

Bonus: Calculation on the blackboard: Eigenvalues

$$(A - \lambda E) \cdot \vec{x} = \vec{0}$$

$$A = \begin{pmatrix} 0 & 2 & -1 \\ 2 & -1 & 1 \\ 2 & -1 & 3 \end{pmatrix}$$

$$A - \lambda \cdot E = \begin{pmatrix} 0-\lambda & 2 & -1 \\ 2 & -1-\lambda & 1 \\ 2 & -1 & 3-\lambda \end{pmatrix}$$

$$det\,(A - \lambda E) = 0$$

$$\alpha_n \cdot \lambda^n + \alpha_{n-1} \cdot \lambda^{n-1} + \ldots + \alpha_1 \cdot \lambda + \alpha_0 = 0$$

Ausrechnen der Determinante dieser Matrix (mit Hilfe der Regel von Sarrus):
det(A – λE) = (0 – λ)(– 1 – λ)(3 – λ) + 4 + 2 – (2λ + 2 + λ + 12 – 4λ)
= – λ3 + 2λ2 + 4λ – 8
= – (λ – 2)(λ – 2)(λ + 2)

Die Eigenwerte entsprechen den Nullstellen des Polynoms:
λ1 = 2, λ2 = – 2

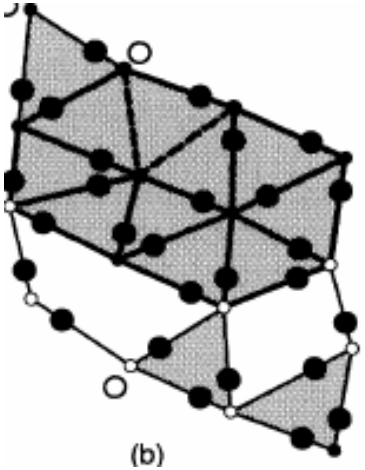
Der Eigenwert 2 hat algebraische Vielfachheit 2, da er doppelte Nullstelle des charakteristischen Polynoms ist.

http://de.wikipedia.org/wiki/Eigenwerte

Matrix Diagonalization: brute-force (2)

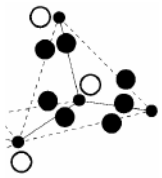
To identify all rigid clusters:

- Place test constraint between each pair of atoms
- Redundant constraints are added to the network
- Each contiguous path of face-sharing tetrahedrons forms a rigid cluster



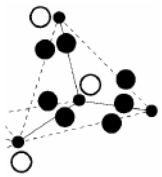
Problems:

- Scales as $O(N^5)$, N = #(atoms)
- Numerical round-off errors



Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



Pebble Game (1)

- Based on Laman‘s theorem for 2D framework

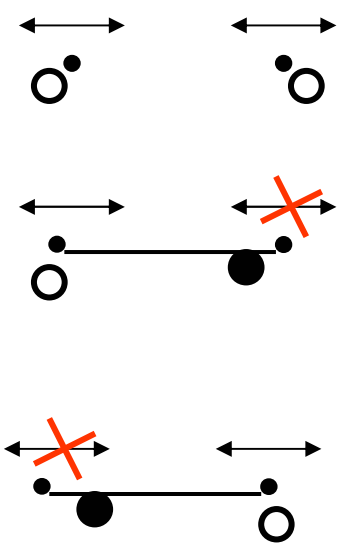
  “On Graphs and Rigidity of Plane Skeletal Structures.“ (1970)
- Directly applying: scales exponentially
- Recursively, efficient integer algorithm derived: „2D pebble game“
- Generalisation of Laman‘s theorem to 3D

- General idea for our application:
- **free pebbles** symbolizes **degrees of freedem**

Pebble Game (2)

- Pebbles are associated with nodes
- They have two possible states:
  - be free => Atom can move (independently)
  - cover a bond of „their“ node => Atom is fixed
- Once a bond is covered, it must remain covered
- Rearrangement allowed within above rules

1D-Example:



Free pebbles represent the independent degrees of freedom of a atom

Pebble Game (3): 3D – Independence-Test

- Every node has 3 pebbles
- Test of edges on independence/redundance:
  - Test whether a pebble can be moved to the new constraint from its node.
  - If a move is possible
    - Edge independence

      result: pebble on edge
  - else
    - Edge is redundant

      result: edge pebbleless

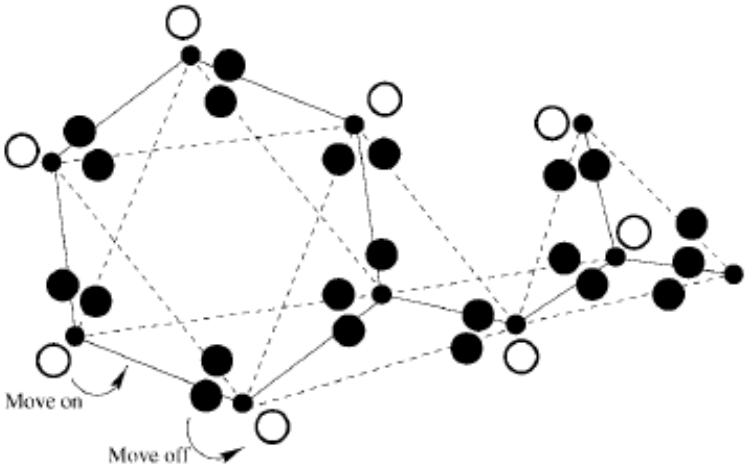
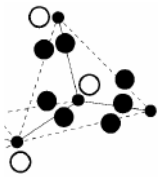


Fig. 2. Diagrammatic representation of the final pebble covering of a simple network. Free pebbles are placed directly on vertices and denote degrees of freedom (○). Pebbles covering a bond are placed directly on the edges of the graph and represent distance constraints (●). The pebble covering is not unique, because pebbles can be rearranged according to a few simple rules as explained in the text. An example is shown of how an elementary pebble exchange works (two arrows). A free pebble can be moved onto a covered edge (adjacent to a vertex), provided that a corresponding pebble, presently covering the edge and associated with the neighboring vertex, is moved off.

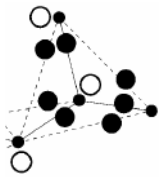


Pebble Game (4): Algorithm for 2D

0. Give any vertex 2 free pebbles, no single edge is set.

1. Collect 2 free pebbles on vertex 1.
2. Try to collect *2* free pebbles on vertex 2, which is connected to vertex 1 by a constraint, while holding the 2 pebbles on vertex 1 fixed.
3. If you can collect 2 pebbles on vertex 2, then the constraint between 1 and 2 is independent. Place one pebble from vertex 2 onto the constraint between 1 and 2.
4. If you fail to collect *2* pebbles on vertex 2, then the constraint is redundant..

To remind look at the overhead projector… Here now a game!



Pebble Game (5): Example

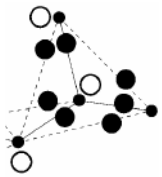
- http://flexweb.asu.edu/software/pebble_game/2D_interactive/
- Place 4 nodes
- Connect 1-2, 2-3, 2-4, 4-1, 1, 3:

  pebble search allways successful =>

  all edges are independent:
- Connect 2-3:

  Pebble search fails =>

  edge is redundant: stress!

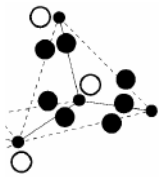


Pebble Game (6): Algorithm for 3D

0. Give any vertex 3 free pebbles, no single edge is set.

1. Collect 3 free pebbles on vertex 1.
2. Try to collect *3* free pebbles on vertex 2, which is connected to vertex 1, while holding the 3 pebbles on vertex 1 fixed.
3. If you can only collect 2 pebbles on vertex 2, then the constraint between 1 and 2 is redundant.
4. (a)If you can collect 3 pebbles on vertex 2, then hold all the 6 pebbles on vertices 1 and 2 fixed. Using all sites connected to either vertices 1 or 2 by a constraint, check each of these neighbors, one at time, for a seventh pebble. If it is not possible to find the seventh pebble for any one of these neighbors, then the constraint is redundant
   (b) Otherwise place one pebble from vertex 2 onto the constraint between 1 and 2.
5. Next test *all* the angular constraints involving sites 1 and 2 as one arm, in a similar way to that described in steps 1 through 4 above.

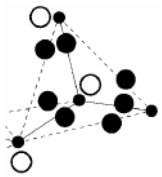
edge is independent



Bonus: Pebble Game (7): Further Rules

No random placement of the constraints!

1. place at first a central-force distance constraints
2. Place all its angular (next-nearest-neighbor) distance constraints in a arbitrary order
3. Repeat 1. with the next arbitrary central force.
4. After all central-force and angular constraints are set: place their torsional constraints (third-nearest-neighbors)



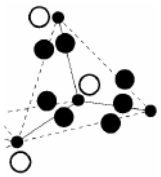
Pebble Game (8): Free Pebbles

- After all distance constraints have been placed:

  Number of free pebbles gives the total number of **degrees of freedom** required to describe the motion of the network.

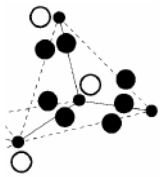
  Including the six trivial rigid body translational and rotatinal degrees of freedom of the whole network.

- What can we do with this information?



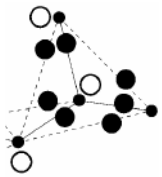
Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



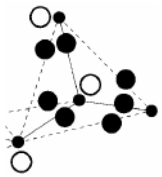
Identify strained=overconstrained regions

- Redundant bond is identified when pebble search failed
- Pebble search failed if a set of nodes have no extra free pebbles to give up
- This physically means: placing an additional distance constraint between atoms that have already a predefined fixed distance => stress
  - Failed pebble search identifies overconstrained regions
  - Overlapping overconstrained regions merge together
  - Overconstrained regions are more stable than „just“ rigid regions (question to the audience: why?)



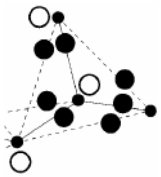
Identify rigid cluster

- Even in rigid regions, up to 6 free pebbles are always possible (question to the audience: why?)
- Select a node n_1 and two of its bonded neares-neighbor nodes n_2&n_3
- Collect 3 pebbles on n_1 and 2 resp. 1 on n_2&n_3
- Check in an **breadth-first** search all unmarked neighbors, if a free pebble can be obtained
- If **no** pebble can be obtained, mark the new node and note it as part of the same rigid cluster



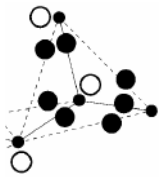
Identify hinge joints and understrained regions

- Hinge joint = rotatable bond dihedral angle
- If the two incident nodes of a bond belong to different rigid clusters, the bond is marked as a hinge joint (draft on blackboard)
- Hinge joints can be
  - independent
  - dependent: Collective Motions
- Algorithm for finding underconstrained regions is similar to finding strained regions



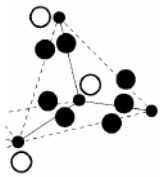
Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



Flexibility Index: Why necessary?

- A flexible region consisting of many interconnected rigid clusters may define a collective motion having only a few independent degrees of freedom
- An isostatically rigid region without redundant constraints is probably less stable than a overconstrained region
- Because of this **continuum** between rigidity and flexibility, a **continuous index** is useful

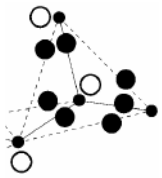


Flexibility Index

- f_i := flexibility index of the *i*th central-force bond
- In the *k*th underconstrained region:
  - H_k := #(hinge joints)
  - F_k := #(independent dihedral rotations)
- In the *j*th overconstrained region:
  - C_j := #(central-force bonds)
  - R_j := #(redundant constraints)

$$f_i \equiv \begin{cases} \dfrac{F_k}{H_k} & \text{in an underconstrained region} \\ 0 & \text{in an isostatically rigid region} \\ \dfrac{-R_j}{C_j} & \text{in an overconstrained region} \end{cases}$$

Greatest flexibility: $f_i = 1$
Negativ values: very rigid

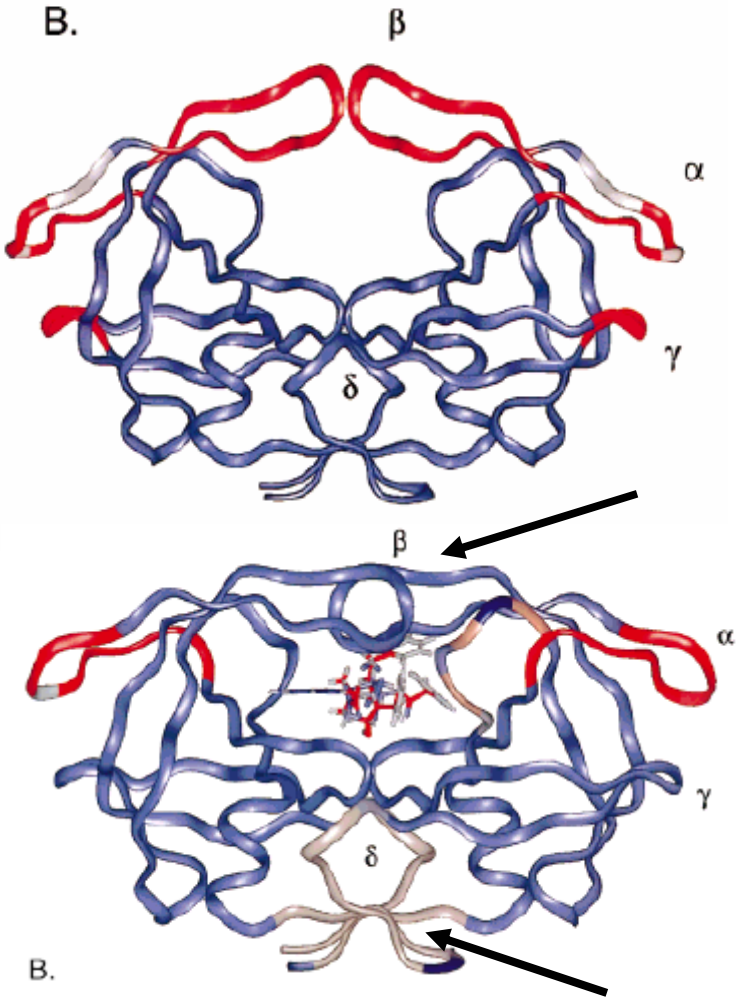


Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References

Some results (1): internal comparison

- Prediction based on a single 3D structure:
  - Conformational flexibility depends on whether the structure is an open (ligand-free) or a closed (ligand-bound) form.
  - However, flexible and rigid predicted regions are „*remarkably consistent*“ *(?)*
  - Example: HIV Protease (HIVP) (PDB code 1htg)

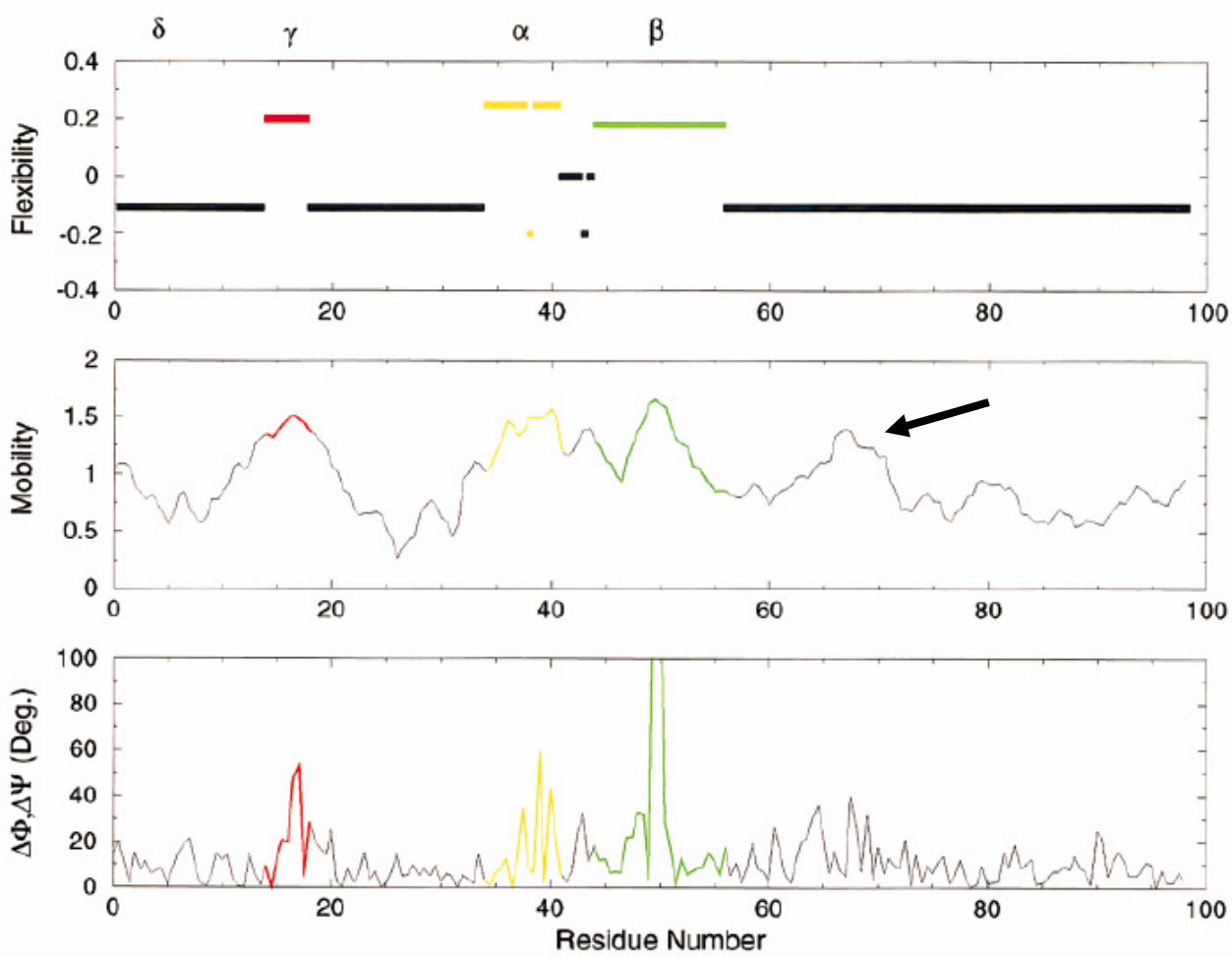


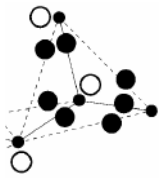
Some results (2): experimental comparison

Prediction of FIRST

B-value, measured crystallographically

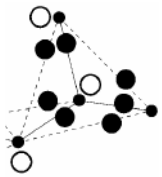
main-chain dihedral angle changes: crystal structures of open and closed conformations





Outline

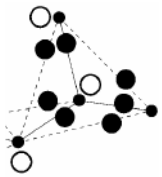
1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



FIRST ==> ROCK ==> SLIDE

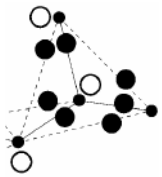
- Patented implementation of the shown algorithms based on the pebble game
- http://flexweb.asu.edu/software/first/

FIRST (**F**loppy **I**nclusion and **R**igid **S**ubstructure **T**opography) is a graph-theoretical approach to identify rigid and flexible regions based on the protein bond network consisting of covalent / hydrogen bonds and hydrophobic interactions



FIRST ==> ROCK ==> SLIDE

- ROCK (**R**igidity **O**ptimized **C**onformational **K**inetics)
- Uses restricted random-walk sampling to search the conformational space availible to proteins
- Ouput of FIRST (detected flexible and rigid regions) is Input for ROCK
- Conformational space is divided into allowed and disallowed regions: Sampling just with allowed regions
- Most distinct conformers produced by ROCK are selected for output



FIRST ==> ROCK ==> SLIDE

- SLIDE (**S**creening for **L**igands by **I**nduced-Fit **D**ocking, **E**fficiently)
- Software for docking known ligands into the ligand-free conformation of a binding site
- Output of ROCK (generated conformations) is input for SLIDE
- SLIDE models protein-ligand interactions based on steric complementary combined with hyrophobic and hydrogen-bonding interactions

Results of FIRST=>ROCK=>SLIDE

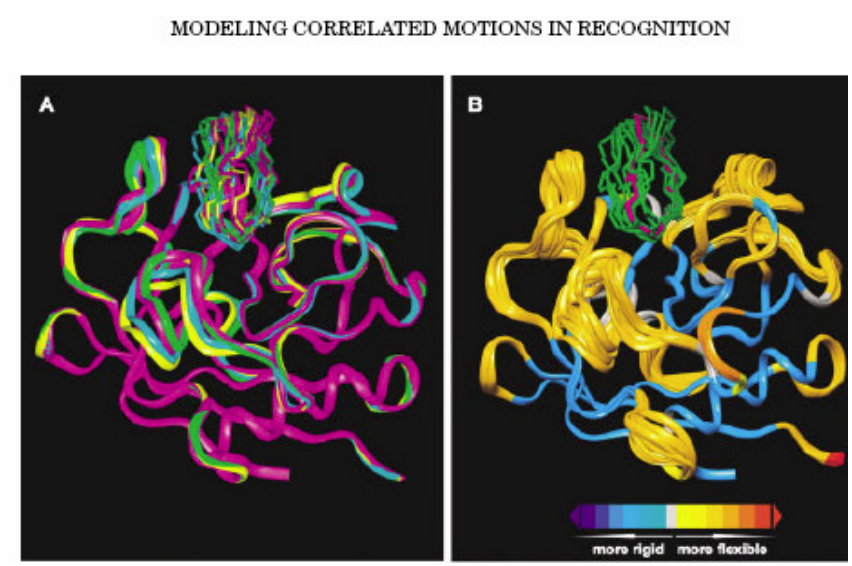
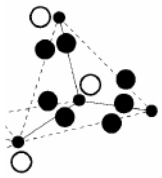


Fig. 8. (**A**) Cyclosporin conformers (showing only the main chains as tubes) docked into the binding sites of the ROCK-generated CypA conformers (shown as ribbons corresponding to the protein main chain).

Each docked cyclosporin conformer is colored the same as the CypA conformer to which it docked best.

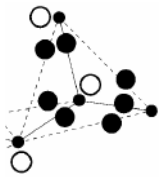
(**B**) Ribbon diagrams of the 12 most distinct CypA conformers colored by flexibility index with the docked cyclosporin conformers (backbones shown as green tubes), illustrating the range of conformational changes during the protein–ligand recognition process as modeled by the FIRST–ROCK–SLIDE combined method.

comparison to other methods or usable RMSD-Values are missing



Outline

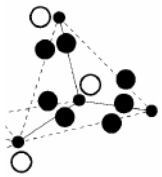
1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References



Discussion: matrix diagonalization, FIRST

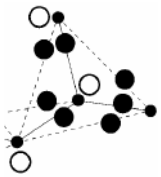
- Matrix diagonalization is too slow, has got round-off errors
- FIRST some results as above, but very fast and because of integer counting exact
- Ability to determine coupled motions is advantage over other methods
- Results correlate well with experimental data

- Problems: Setup of Constraints
  - Threshold of hydrogen bonds
  - inherent to the system (just taking on structure): connectivity



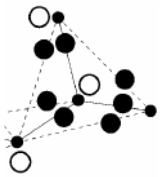
Discussion / personal criticism

- Matrix diagonalization:
  - Why testing the „normal“ bonds whether they are independent or redundant if it does not matter?
  - No detailed record about complexity of testing for independency and searching face-sharing tetrahedrons
- FIRST
  - Explanation for algorithm that find underconstrained regions at the end a little bit dubious
  - Comparison between FIRST results an experimental data could be better quantified
- Papers in themselves:
  - Readably and mostly comprehensible, but unfortunately with very few example or illustrations.



Outline

1. Introduction / Motivation
2. Different possibilities to recognize flexible regions
3. Modelling a protein as a graph
4. Approach I: Matrix Diagonalization
5. Approach II: Pebble Game
    I. Theory and Example
    II. Identify (Under-)strained Regions, Rigid Cluster
    III. Flexibility Index
6. Some results
7. FIRST ==> ROCK ==> SLIDE
8. Discussion
9. References

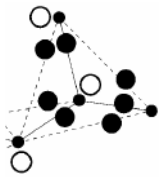


References

- Protein flexibility predictions using graph theory (2001)
  http://www3.interscience.wiley.com/cgi-bin/fulltext/82003580/PDFSTART
- Modeling correlated main-chain motions in proteins for flexible molecular recognition (2004)
  http://www3.interscience.wiley.com/cgi-bin/fulltext/109075321/PDFSTART
- Generic Rigidity Percolation: The Pebble Game
  http://prola.aps.org/pdf/PRL/v75/i22/p4051_1
- http://flexweb.asu.edu/software/pebble_game/
- http://flexweb.asu.edu/software/first/first_online/

- http://www.uni-bayreuth.de/departments/ddchemie/umat/peptidsynthese/peptidbindung.gif
- http://de.wikipedia.org/wiki/Eigenwerte
- http://mathworld.wolfram.com/LamansTheorem.html
- http://flexweb.asu.edu/software/pebble_game/2D_CF_only/
- http://flexweb.asu.edu/software/pebble_game/3D_CF_and_AF/

- *http://dict.leo.org (used for translation)*



Knowledge you **must** take home

- There are methods to determine flexibility of proteins
- Graphtheory can be very helpful
- Pebble game is an interesting approach to determine degrees of freedom of a network

The End

Special thanks go to Susanne Eyrisch for the good mentoring.

Questions please!