Curs 11.

Algoritmi aleatori. Aplicații în bioinformatică

Biblio: cap. 12 din “An introduction to Bioinformatics algorithms”, N.Jones, P. Pevzner

Algoritmi aleatori

- Specific
- Algoritmi de tip Las Vegas
- Algoritmi de tip Monte Carlo
  - Metode aleatoare pentru identificarea șabloanelor
  - Metode aleatoare pentru identificarea structurii proteinelor

Specific

- Algoritmii aleatori folosesc valori aleatoare pe parcursul execuției
- La rulări diferite secvența de prelucrări, numărul de prelucrări efectuate sau chiar rezultatele pot fi diferite
- Se folosesc în special pentru probleme dificile pentru care algoritmii determiniști sunt ineficienți
  - Probleme de căutare în spații de dimensiune mare (căutarea aleatoare evită parcurgerea întregului spațiu de căutare)
  - Probleme de optimizare globală în cazul funcțiilor multimodale (elementele aleatoare permit evitarea blocării în minime locale)

Variante

- Algoritmi de tip Las Vegas
  - Conduc la același rezultat la fiecare rulare
  - Rulările diferă din punct de vedere al numărului de operații efectuate (acest lucru are importanță în evaluarea complexității în cazul mediu – fiind util în special în cazul în care algoritmul este apelat de multe ori în cadrul unei aplicații)
  - Exemplu: QuickSort cu alegere aleatoare a valorii elementului pivot

- Algoritmi de tip Monte Carlo
  - Conduc la rezultate diferite la rulări diferite
  - Aplicații: simularea unor fenomene fizice, aproximarea integralelor multiple, determinarea optimului global al unei funcții

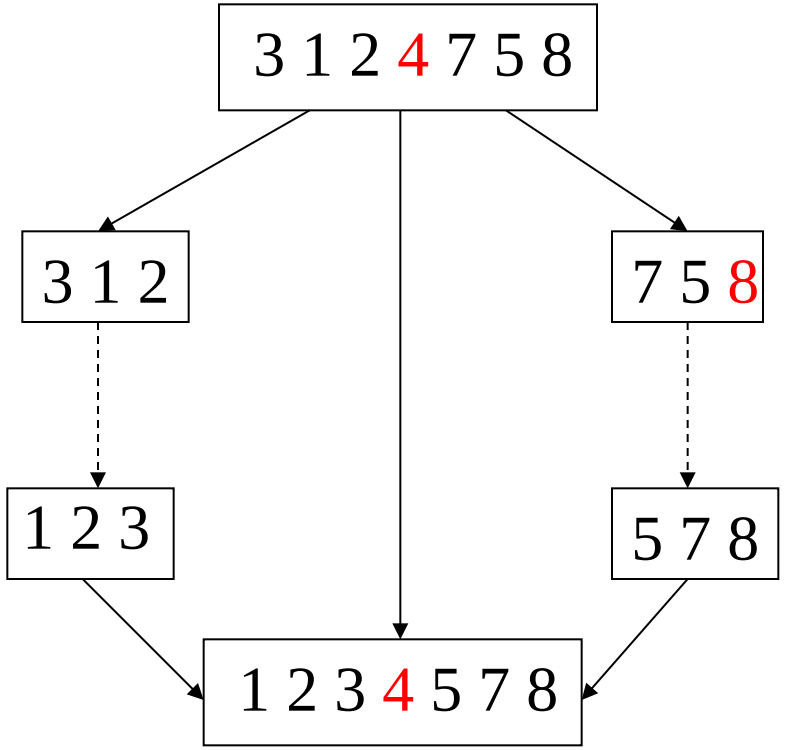
QuickSort cu alegere aleatoare a pivotului

Reminder

Idee: tehnica divizării (divide and conquer)
1. alege o valoare de referință (pivot)

2. reorganizează tabloul astfel ca toate elementele din prima parte a tabloului să fie mai mici decât pivotul iar toate elementele din a doua parte a tabloului să fie mai mari decât pivotul

3.aplică aceeași strategie pentru fiecare subtablou



QuickSort cu alegere aleatoare a pivotului

```
quicksort(x[left..right])
IF le<ri THEN
  q:=pivot(x[left..right])
  x[left..q-1]:=quicksort1(x[left..q-1])
  x[q+1..right]:=quicksort1(x[q+1..right])
ENDIF
RETURN x[left..right]
```

```
pivot(x[left..right])
  index:=random(left,right)
   v:=x[index]
   i:=left-1
   j:=right
   WHILE i<j DO
       REPEAT i:=i+1 UNTIL x[i]>=v
       REPEAT j:=j-1 UNTIL x[j]<=v
       IF i<j THEN x[i]↔x[j] ENDIF
  ENDWHILE
  x[i] ↔ x[right]
  RETURN i
```

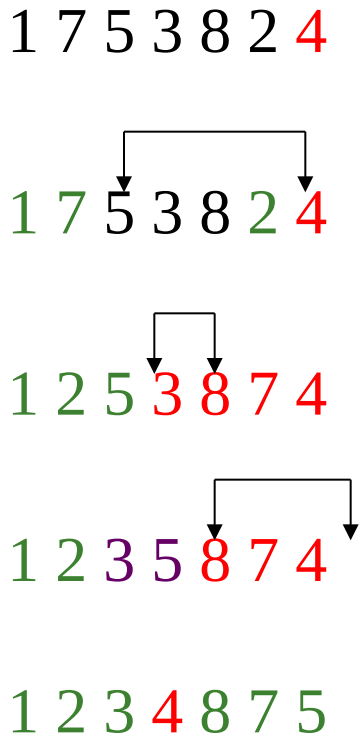
Obs: index este indicele unui element selectat aleator din x

QuickSort cu alegere aleatoare a pivotului

Analiza complexității:

Exemplul 1:

1 7 5 3 8 2 4



Partiționare echilibrată

Exemplul 2:

1 7 5 3 8 2 4

Exemplul 3:

1 7 5 3 4 2 8

Partiționări dezechilibrate

QuickSort cu alegere aleatoare a pivotului

Analiza complexității: Numărul de operații efectuate depinde de tipul partiționării:

- partiționare dezechilibrată (cazul cel mai defavorabil): $O(n^2)$

- partiționare echilibrată (cazul cel mai favorabil): O(nlogn)

Cazul mediu: O(nlogn)

Alegerea aleatoare a valorii pivotului permite reducerea numărului de situații în care se ajunge la partiționare dezechilibrată

Obs. pentru ca o partiționare sa fie echilibrată e suficient ca pivotul (poziția de partiționare) să fie plasat în vecinătatea lui n/k sau a lui n(1-1/k) cu k<10

Algoritmi aleatori pentru identificarea sabloanelor

Enunț (reminder curs 5):

Pornind de la un set de t secvențe ADN, fiecare de lungime n, să se găsească un set de subsecvențe de lungime dată l, câte una din fiecare secvență astfel încât scorul de potrivire asociat sa fie maxim

$$scor(s, set) = \sum_{i=1}^{l} \max_{k \in \{A,C,G,T\}} P(k,i)$$

$P(k,i)$ = frecventa nucleotidei k in pozitia i a sablonului

Obs: set=setul de t secvențe
s=(s_1,s_2,…s_t) pozițiile de start ale șablonului în cele t secvențe

Set de secvențe

a G g t a c T t
C c A t a c g t
a c g t T A g t
a c g t C c A t
C c g t a c g G

Matrice profil

A	3	0	1	0	3	1	1	0
C	2	4	0	0	1	4	0	0
G	0	1	4	0	0	0	3	1
T	0	0	0	5	1	0	1	4

Sablon consensual

a c g t a c g t

Scor: 3+4+4+5+3+4+3+4=30

Algoritmi aleatori pentru identificarea sabloanelor

Reminder curs 5

Branch and bound

```
Gen(k)
IF k=t+1 THEN "prelucrează s[1..t]"
                    actualizează maxScor
ELSE
   inițializare lista candidati L
   FOR i:=1, m DO
       s[k]:=i
       IF scor(s[1..k])+l*(t-k)>maxScor
           THEN adauga s[k] la lista L
   sortează lista descrescător după scor
   FOR fiecare element i din lista L DO
      s[k]=i
      Gen(k+1)
```

obs: m=n-l+1

Greedy:

Se determină s_1 și s_2
FOR i:=3,t
Determină s_i ce maximizează scorul în raport cu alinierea parțială constituită din primele i șabloane selectate

Algoritmi aleatori pentru identificarea sabloanelor

Idee:

- se pornește de la un set aleator de puncte de start $s=(s_1,s_2,\ldots,s_t)$
- se modifică succesiv componentele vectorului s fie aplicând o strategie greedy fie folosind o distribuție de probabilitate

Specific:

- se folosește o probabilitate de potrivire dintre o subsecvență $a=a_1\ldots a_l$ și un profil P (matrice 4xl care conține frecvențele relative de apariție ale nucleotidelor în setul de secvențe aliniate)

$$prob(a \mid P) = \prod_{i=1}^{l} p_{a_i,i}$$

Frecvența relativă a nucleotidei a_i pe poziția i în setul de secvențe aliniate

Obs: dacă a este apropiată de șablonul consensual atunci prob(a|P) va fi mare, altfel prob(a|P) va fi mică

Algoritmi aleatori pentru identificarea sabloanelor

Exemplu (l=6):

A	**1/2**	**7/8**	**3/8**	0	1/8	0
C	1/8	0	1/2	**5/8**	**3/8**	0
T	1/8	1/8	0	0	1/4	**7/8**
G	1/4	0	1/8	3/8	1/4	1/8

Probabilitate șablon consensual:

Prob(**aaacct|P**) = 1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8 = .033646

Probabilitatea altui sir:

Prob(**atacag|P**) = 1/2 x 1/8 x 3/8 x 5/8 x 1/8 x 1/8 = .001602

Obs: de regulă se caută subsecvențe dintr-o secvență care maximizează probabilitatea de potrivire cu matricea profil

Algoritmi aleatori pentru identificarea sabloanelor

Abordare greedy (hill climbing) bazată pe probabilitățile de potrivire:

Pas 1. Selectează aleator poziții de start $s=(s_1,s_2,\ldots,s_t)$ și stabilește scorul corespunzător

Pas 2. Creează matricea de profil corespunzătoare configurației s

Pas 3. Identifică în fiecare secvență cea mai probabilă subsecvență (în raport cu profilul curent)

Pas 4. Calculează un nou profil și scorul corespunzător

Pas 5. Dacă s-a obținut o îmbunătățire a scorului se trece la Pas 3
alfel STOP

Algoritmi aleatori pentru identificarea sabloanelor

Observații:

1. Fiind metodă de tip hill climbing se poate bloca în optime locale

2. O variantă îmbunătățită care nu se bazează pe o alegere greedy ci pe una aleatoare (în conformitate cu o distribuție de probabilitate care favorizează configurațiile de scor mare) este cea bazată pe selecție de tip Gibbs (Gibbs sampling)

Gibbs sampling pentru identificarea sabloanelor

Particularități ale metodelor de tip Gibbs sampling:

1. la fiecare iterație se modifică punctul de start corespunzător unei singure subsecvențe (în varianta greedy era posibil să se modifice mai multe poziții de start simultan)

2. modificarea are caracter aleator ceea ce permite să se evite optimele locale (varianta greedy se blochează frecvent în optime locale)

Gibbs sampling pentru identificarea sabloanelor

Etape:

Pas 1. Selectează aleator poziții de start $s=(s_1,s_2,\ldots,s_t)$
Pas 2. Se alege uniform aleator una dintre secvențe (pentru care se va modifica poziția de start a subsecvenței)

Pas 3. Se construiește profilul corespunzător celorlalte t-1 subsecvențe

Pas 4. Pentru fiecare poziție din secvența selectată se calculează probabilitatea de potrivire cu profilul construit la Pasul 3; pornind de la cele m=n-l+1 probabilități se construiește o distribuție de probabilitate pentru poziții de start (din mulțimea {1,2,…,m})

Pas 5. Se generează în secvența selectată un punct de start în conformitate cu distribuția de probabilitate de la Pasul 4
Pas 6. Dacă nu este satisfacută condiția de oprire se reia de la Pasul 2

Gibbs sampling pentru identificarea sabloanelor

Obs:

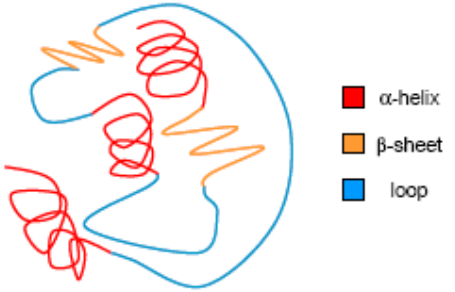
1. condiția de oprire se poate referi la un număr maxim de iterații sau la faptul că nu s-au mai obținut îmbunătățiri ale soluției în ultimele k iterații (k = parametru stabilit de către utilizator)

2. ca orice algoritm aleator necesită rularea de mai multe ori și interpretarea rezultatelor în context statistic

Determinarea structurii proteinelor (protein folding)

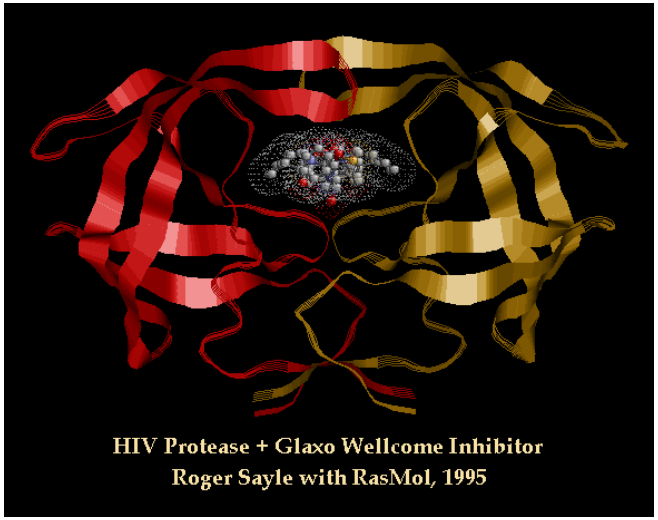
Structura primară: secvență de aminoacizi

Structura secundară: ansamblu de structuri locale caracterizate prin maniera de grupare a aminoacizilor vecini:



- elice (alfa)
- panglici (beta)
- bucle

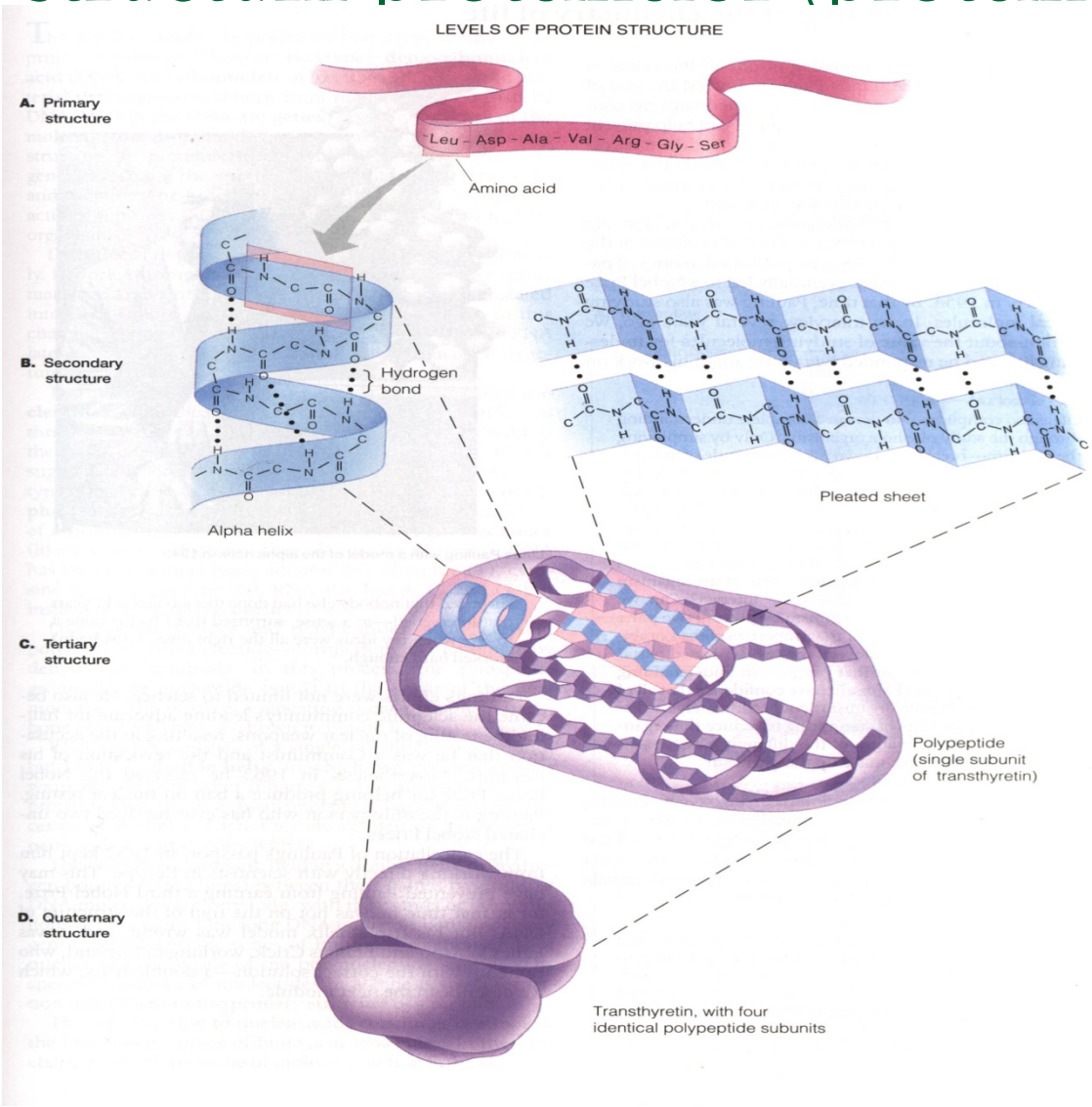
Structura terțiară: aranjarea spațială a ansamblului de aminoacizi astfel ca aminoacizi aflați la distanță mare în secvență ajung să fie apropiați spațial



Determinarea structurii proteinelor (protein folding)

Structura cuaternară: gruparea mai multor secvențe împachetate într-un complex de proteine

Exemplu:



Determinarea structurii proteinelor (protein folding)

Problema împachetării proteinelor = determinarea structurilor alfa, beta și a buclelor

Este echivalentă cu o problemă de optimizare (dpdv fizic se urmărește minimizarea energiei libere a structurii)

Estimarea energiei libere este dificil de realizat – din acest motiv se utilizeaza modele simplificate

Variante simplificate ale problemei:
- modelul de tip contact direct
- modelul HP (hidrofobic – polar (hidrofilic))

Ambele variante presupun ca aminoacizii sunt plasați în nodurile unei grile sau latici (bi sau tridimensionale)

Determinarea structurii proteinelor (protein folding)

Modelul de tip contact direct:

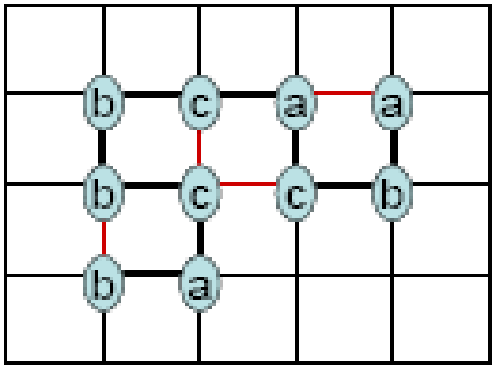
- pentru o secvență $s_1 s_2 \ldots s_n$ se caută configurația C care maximizează

$$f(C) = \sum_{i,j} w_{ij} \Delta(i,j) \quad \text{cu } \Delta(i,j) = 1 \text{ d.n.d } s_i \text{ este vecin cu } s_j \text{ in } C$$

(dar nu sunt vecine in secventa initiala)

Obs. Factorii w_{ij} măsoară gradul de interacțiune dintre s_i și s_j. In cazul cel mai simplu w_{ij} este nenul doar dacă $s_i = s_j$

Exemplu: bacbbcacba
f(C)=4



Determinarea structurii proteinelor (protein folding)

Modelul HP (K. Dill, 1985): se bazează pe următoarele proprietăți ale aminoacizilor
- aminoacizii hidrofobici (H – non-polari) au tendința să se concentreze în interiorul structurii
- aminoacizii hidrofilici (P - polari) se distribuie la suprafața structurii
- scorul configurației este similar celui de la modelul de tip contact direct:

$$f(C) = \sum_{i,j} w_{ij}\Delta(i,j) \quad \text{cu } \Delta(i,j) = 1 \text{ d.n.d } s_i \text{ este vecin cu } s_j \text{ in } C$$

(dar nu sunt vecine in secventa initiala)

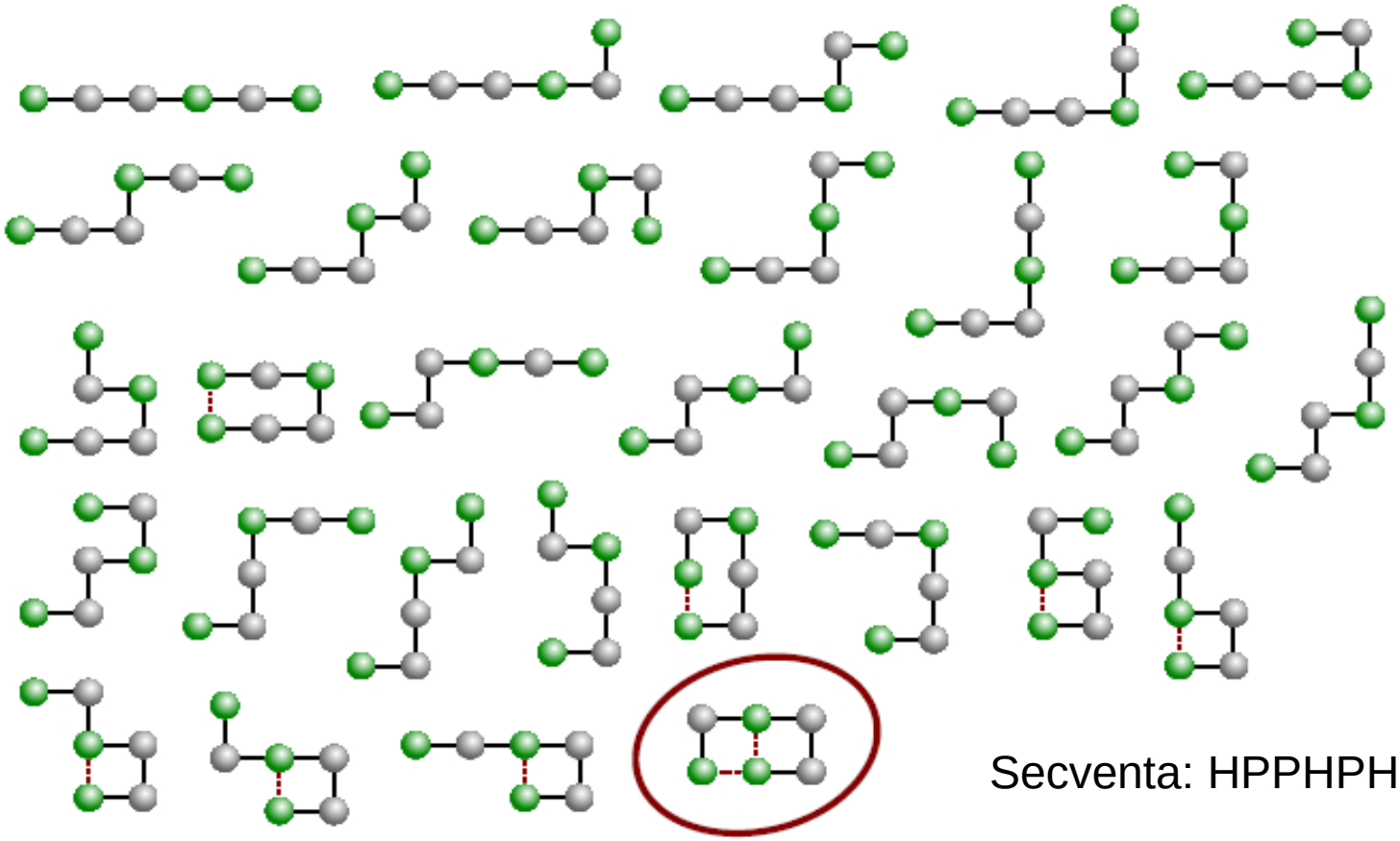
Obs.

- Factorii w_{ij} măsoară gradul de interacțiune dintre s_i și s_j. In cazul cel mai simplu w_{ij} este nenul doar dacă s_i și s_j sunt ambii hidrofobici
- Despre 2 aminoacizi de tip H care sunt vecini in cadrul grilei dar nu sunt vecini in cadrul secvenței se spune că formează un contact

Determinarea structurii proteinelor (protein folding)

Modelul HP: ilustrare configuratii.

Conventie de reprezentare: verde: H (hidrofobic), gri: P (hidrofilic),



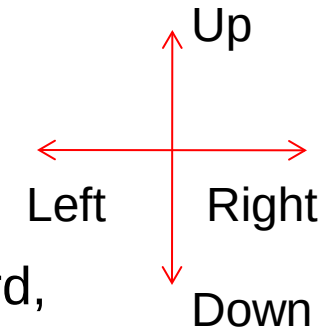
Secventa: HPPHPH

Determinarea structurii proteinelor (protein folding)

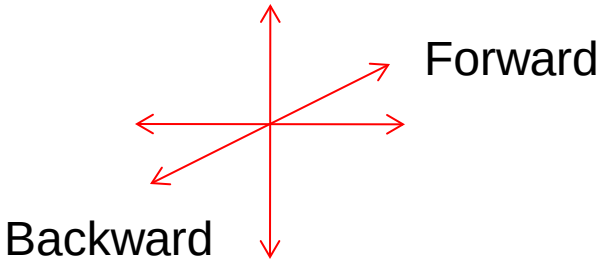
Modelul HP: reprezentarea unei configurații corespunzătoare unei secvențe de n aminoacizi

Idee: pentru fiecare element din secvență se specifică direcția de deplasare pentru a ajunge la poziția corespunzătoare următorului element

2D: există 4 direcții posibile: Left, Right, Up, Down



3D: există 6 direcții posibile: cele patru de mai sus + Forward, Backward

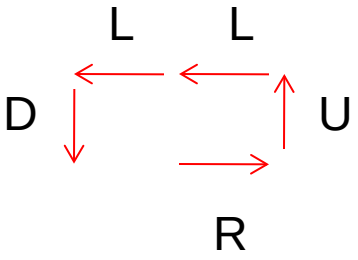


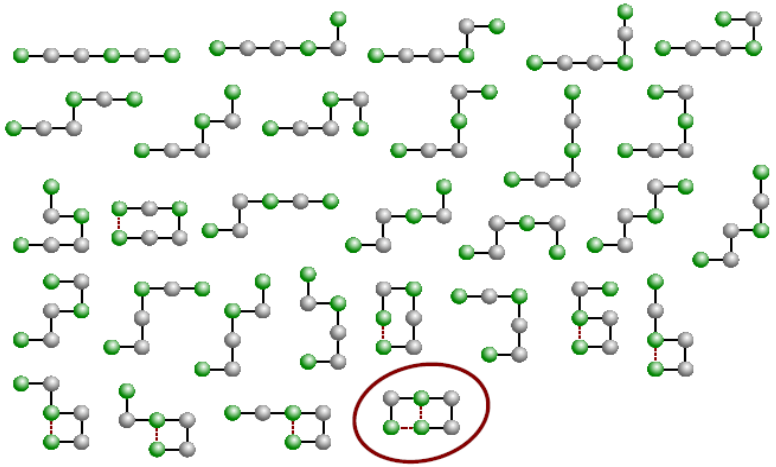
Determinarea structurii proteinelor (protein folding)

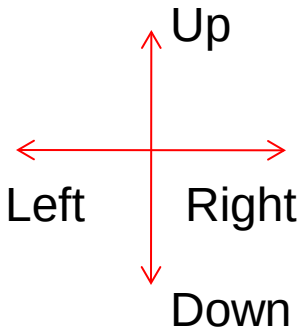
Modelul HP: reprezentarea unei configurații corespunzătoare unei secvențe de n aminoacizi

Exemplu: (pt configurația încadrată):

RULLD







Determinarea structurii proteinelor (protein folding)

Rezolvarea problemei de optimizare:

- Algoritmi determiniști
  - Exacți (similari cu branch and bound) – garantează obținerea soluției optimale însă sunt de complexitate exponențială
  - Aproximativi – conduc la soluții sub-optimale în timp polinomial (există rezultate teoretice privind calitatea soluției: pt minimizarea unei funcții f valoarea soluției aproximative f(aprox) nu depășește a*f(exact), a fiind un raport de aproximare)

Determinarea structurii proteinelor (protein folding)

Algoritmi aproximativi:

- Permit determinarea unor configuratii pentru care numărul de contacte este cuprins între a*nrmaxContacte și nrmaxContacte
- In cazul modelelor cu grila pătratică numărul maxim de contacte este 2*min(oddH(S),evenH(S)) unde
  - oddH(S) reprezintă numărul de poziții impare din secvență pe care se află aminoacizi de tip H
  - evenH(S) reprezintă numărul de poziții pare din secvență unde se află aminoacizi de tip H
- Unul dintre cei mai simpli algoritmi aproximativi (cu rata de aproximare a=0.25) este algoritmul Hart-Istrail

Determinarea structurii proteinelor (protein folding)

Ideea algoritmului Hart-Istrail:

- Se determină un punct de balansare (p) = indice cu proprietatea că cel puțin jumătate dintre aminoacizii H de pe poziții impare se află de o parte a lui p iar de cealaltă parte a lui p se află cel puțin jumătate din aminoacizii de tip H de pe poziții pare
- Un astfel de indice există întotdeauna, fiind suficient să se determine indicele cu proprietatea că în stânga sa se află jumătate dintre aminoacizii H de pe poziții pare iar în dreapta cealaltă jumătate
- Pentru determinarea unei împachetări se construiește o buclă în dreptul punctului de balansare și se stabilesc contacte între aminoacizii de tip H aflați pe poziții pare pe o parte a lui p și cei cu poziții impare aflați pe cealaltă parte a lui p (vezi exemple Lab 7)

Determinarea structurii proteinelor (protein folding)

Ideea algoritmului Hart-Istrail:

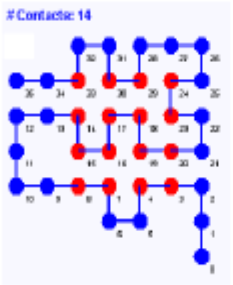


Figure 1: Optimal fold of a protein of length 36

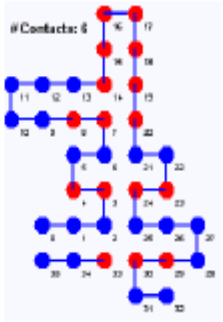


Figure 2: A fold constructed by the Hart-Istrail algorithm

Obs: rosu – hidrofobic, albastru - hidrofilic

S. Istrail, F. Lam, Combinatorial Algorithms for Protein Folding in Lattice Models: A Survey of Mathematical Results, 2009

Determinarea structurii proteinelor (protein folding)

Rezolvarea problemei de optimizare:

- Algoritmi probabilişti din familia euristicilor (similari cu algoritmii Monte Carlo) – conduc la soluții sub-optimale însă în general nu există rezultate teoretice privind calitatea soluției
  - simulated annealing
  - algoritmi evolutivi
  - algoritmi bazați pe modele de inteligență colectivă etc.

Determinarea structurii proteinelor (protein folding)

Simulated Annealing = algoritm iterativ de căutare aleatoare în spațiul soluțiilor (configurațiilor) caracterizat prin:

- pornind de la configurația curentă se generează o configurație nouă prin perturbare aleatoare

- daca noua configurație este mai bună atunci este acceptată necondiționat; dacă nu este mai bună atunci este acceptată cu o anumită probabilitate

- probabilitatea de acceptare a configurațiilor mai puțin bune descrește de la o iterație la alta

Determinarea structurii proteinelor (protein folding)

Sursa de inspiratie:

- procesul de reorganizare a structurii unui solid supus unui tratament termic:
  - Solidul este încalzit (topit): particulele sunt distribuite într-o maniera aleatoare
  - Solidul este răcit lent: particulele se reorganizează pentru a se ajunge la o configurație de energie cât mai mica (cu cât energia internă a sistemului este mai mică cu atât el este mai stabil)

Terminologie:

simulated annealing = tratament termic simulat = călire simulată

Istoric: Metropolis(1953), Kirkpatrick, Gelatt, Vecchi (1983), Cerny (1985)

Determinarea structurii proteinelor (protein folding)

Algoritmul Metropolis (1953) - probabilitate constantă de acceptare – problema de minimizare

Init C(0)

k:=0

REPEAT

C':=perturb(C(k))

IF f(C')< f(C(k)) THEN C(k+1):=C' (neconditionat)

ELSE C(k+1):=C'

cu probabilitatea exp(-(f(C')-f(C(k))/T)

k:=k+1

UNTIL "este indeplinita o conditie de iesire"

Obs: pt o problemă de maximizare condiția f(C')< f(C(k)) se înlocuiește cu f(C') >f(C(k)) și se schimba semnul de la expresia de calcul a probabilității de acceptare

Determinarea structurii proteinelor (protein folding)

Parametrul T joacă rolul temperaturii din procesul fizic de tratare termică a aliajului

- Valori mari ale lui T (T tinde la infinit): argumentul lui exp este aproape 0 => configurațiile sunt echiprobabile (fiecare poate fi acceptată cu aceeași probabilitate)

- Valori mici ale lui T (T tinde la 0): vor avea probabilitate nenulă doar configurațiile de scor minim (pt o problema de minimizare) sau maxim (pentru o problema de maximizare)

Simulated annealing = aplicarea repetată a algoritmului Metropolis pentru diferite valori ale temperaturii (inițial T este mare și treptat este micșorat)

Determinarea structurii proteinelor (protein folding)

Structura generală Simulated Annealing

Init C(0), T(0), k:=0

REPEAT

C':=perturb(C(k))

IF f(C')< f(C(k)) THEN C(k+1):=C' (neconditionat)

ELSE C(k+1):=C'

cu probabilitatea exp(-(f(C')-f(C(k))/T)

calcul T(k+1)

k:=k+1

UNTIL T(k)<eps

Problema: alegerea schemei de modificare a temperaturii ("cooling scheme")

Determinarea structurii proteinelor (protein folding)

Scheme de “răcire”:

$T(k)=T(0)/(k+1)$

$T(k)=T(0)/\ln(k+k0)$

$T(k)=aT(k-1)$ $(a<1$, ex: $a=0.995)$

Obs. T(0) se alege astfel încât la primele iterații să fie acceptate aproape toate configurațiile generate (explorarea spațiului soluțiilor)

Determinarea structurii proteinelor (protein folding)

Proprietăți de convergență:

Dacă sunt satisfăcute proprietățile:

- Pg(C(k+1)=C'|C(k)=C)>0 pentru orice C și C' (probabilitatea de trecere între oricare două configurații este nenulă)
- Pa(C(k+1)=C'|C(k)=C)=min{1,exp(-(f(C')-f(C))/T)} (probabilitate de acceptare de tip Metropolis)
- T(k)=K/lg(k+k0) (schema logaritmică de răcire)

Atunci P(f(C(k))=f(C*)) -> 1 (C(k) tinde în probabilitate la minimul global C* când k tinde la infinit)

Determinarea structurii proteinelor (protein folding)

Alte metaeuristici inspirate de natura:

- Algoritmi evolutivi
- Algoritmi inspirati de modelul coloniilor de furnici (Ant Colony Optimization)
- Algoritmi inspirati de modelul stolurilor de pasari (Particle Swarm Optimization)

Detalii: Algoritmi metaeuristici (an II)