SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**SEMINAR**

# Korištenje skrivenih Markovljevih modela pri poravnavanju niza proteina

*Ana Bulović*
Voditelj: *Doc. dr. sc. Mile Šikić*

Zagreb, srpanj 2011.

# SADRŽAJ

# 1. Uvod

Protein se može predstaviti pomoću njemu svojstvenog niza aminokiselina. Različitih aminokiselina je 20 i one se standardno zapisuju pomoću 20 slova engleske abecede, sukladno njihovim nazivima. Proteinsku strukturu je uvijek moguće deterministički odrediti i zapisati pomoću relativno malog niza simbola, što posebno pogoduje računalnoj reprezentaciji. Znanje o funkciji proteina moguće je dobiti uspoređujući njegov slijed aminokiselinskog ostatka sa sljedovima proteinima već poznate ili pretpostavljenje funkcije. Pretpostavka na kojoj se zasniva ideja da iz sličnosti slijeda proteina možemo pričati i o njihovoj funkcionalnoj sličnosti leži u postupku nastanka novih proteina u prirodi. Novi sljedovi su rezultat modifikacije starih sljedova proteina nastalih mutacijom i križanjem, ali nikada potpuno novog, slučajnog stvaranja čitavog slijeda. Poravnanje proteina u najjednostavnijem slučaju možemo promatrati kao na poravnanje običnih stringova, ne obazirući se na pozadinsku semantiku. Predloženi su brojni algoritmi koji se bave ovim problemom, od onih koji se zasnivaju na dinamičkom programiranju, do onih koji su heurističke prirode. Ključna stvar svih algoritama, bez obzira na njihovu prirodu, je funkcija cijene: koliko će koštati izmjena simbola u slijedu, koliko brisanje iz slijeda te koliko umetanje. Određivanje funkcije cijene je bitan, ako ne i najbitniji, korak u razvoju algoritma poravnanja i njena preciznost, osjetljivost i temeljenost za stvarnoj biološkoj pozadini u velikoj će mjeri odrediti konačnu uspješnost algoritma. Osim određivanja funkcije cijene, glavni dio poravnanja je tehnički - za to postoje razvijeni, poznati algoritmi, često i s gotovim implementacijama u brojnim programskim jezicima.

Na internetu je dostupno mnogo baza proteina u kojima se oni mogu dohvaćati po dogovornim imenima u formatu slijeda simbola te se može doznati njihova biološka funkcija. Korištenjem upravo algoritama za određivanje sličnosti sljedova proteina, iz ovih je baza omogućen dohvat informacija kao što su moguća vrsta kojoj nepoznati protein pripada, njegova domena te generiranje evolucijskog stabla. Najpoznatiji primjer takvog algoritma je BLAST. Algoritmi pogodni za ovakvo pretraživanje velikog broja podataka se najčešće zasnivaju na različitim heuristikama i u cilju brzog

izvođenja žrtvuju optimalnost. Zbog tog se najčešće koriste u inicijalnim ispitivanjima prirode proteina s kojima se radi, više da se 'dobije osjećaj' s kakvim se podacima radi i kako im najbolje pristupiti.

Iako su pristupi za poravnanje bioloških sljedova slični, u ovom radu će se pretpostaviti da se radi o proteinskim sljedovima. U sljedećim poglavljima bit će pojašnjen pristup određivanju funkcije cijene, te kako su te funkcije uobličene u javno dostupnim matricama, kao i pozadinska matematika na kojoj počiva izračun tih cijena. Bit će obrađeni i različiti načini određivanja cijene umetanja praznine, te njihove prednosti i nedostaci. U narednom je poglavlju objašnjeni princip na kom se zasniva poravnanje sljedova, te kakvim se jednostavnim modifikacijama može osnovni algoritam izmijeniti za pojedinu upotrebu. U svrhu toga pojašnjen je algoritam dinamičkog programiranja. Osim tog, pojašnjen je i heuristički pristup, te zbog čega i u kojim primjenama je on potreban. Na samom kraju objašnjeno je kako odrediti statističku važnost postignutog poravnanja.

# 2. Poravnanje dva slijeda

Najjednostavniji način kojim se može saznati jesu li dva proteina povezana je poravnavajući njihove sljedove aminokiselinskih ostataka. Nakon što je poravnanje postignuto, potrebno je na neki način odrediti kolika je postignuta sličnost i je li ona rezultat slučajnosti ili stvarne biološke povezanosti. Za dva slijeda, u ovisnosti o korištenom algoritmu ili definiciji optimalnosti, moguće je postići više različitih poravnanja. Za primjer se mogu uzeti dva niza: `GAATTCAGTT` i `GGATCGA`. Jedno od mogućih poravnanja ta dva niza dano je kao:

```
G G A T T C A G T T A
|   |   | |   |     |
G A A - T C - G - - A
```

Kako se odlučiti za pojedino poravnanje i proglasiti ga najboljim? Za to je potrebno odgovoriti na sljedeća pitanja

- Kakva poravnanja treba razmatrati te kakve događaje možemo koristiti kao sredstva poravnanja?
- Kakva će biti funkcija cijene kojom će se ocijeniti i međusobno rangirati različita poravnanja?
- Kojim se algoritmom mogu pronaći optimalna poravnanja?
- Kakvim se statističkim metodama može ocijeniti kvaliteta i značaj postignutih poravnanja?

Promatrajući navedeni primjer, može se dati odgovor na prvo pitanje. Događaji pomoću kojih možemo modelirati poravnanje su podudaranje, zamjena, umetanje i brisanje simbola. Problem kog adresira sljedeće pitanje je kako nagraditi (ili kazniti) svaki od ovih događaja. Kako ocijeniti podudaranje, a kako zamjenu? Je li zamjena 'skuplja' ili jeftinija od umetanja simbola, i u kakvom je to odnosu s brisanjem simbola? Pri otvaranju praznine u slijedu (kao što za dva predzadnja simbola `T T` u drugom nizu imamo - -), je li jednako skupo otvoriti prazninu kao i proširiti je? Nakon što

je praznina jednom otvorena, treba li sniziti cijenu njenog proširenja? Poravnanje u navedenom primjeru dobiveno je postupkom dinamičkog programiranja, koji će biti opisan dalje u tekstu, gdje nagrada za podudaranje iznosi 1, dok se svi ostali događaji ocjenjuju nulom. U počecima izučavanja problema poravnanja sljedova, predložene su brojne funkcije cijene koje, poput ovdje upotrebljene, nisu imale biološko uporište i čiji autori nisu mogli ponuditi dobro opravdanje za njihovo korištenje. Zbog važnosti ovog problema, potrebno je pogledati kakve funkcije cijene danas stoje na raspolaganju bioifnormatičarima te za kakve su upotrebe one prikladne.

# 3. Funkcije cijene

Kao što je već rečeno, pri poravnanju važno je voditi računa o tome da rezultat nosi informaciju o stvarnoj sličnosti dvaju proteina - o njihovoj trodimenzionalnoj strukturi, o njihovoj funkciji, o njihovoj evolucijskoj povezanosti... Pažljivim odabirom načina ocjenjivanja poravnanja može se, ako ne u potpunosti izbjeći, onda barem smanjiti mogućnost postizanja velike sličnosti nizova koji nemaju biološke poveznice. Kratkim pretraživanjem tematike na internetu svatko će vrlo brzo doći do rezultata kao što su matrice BLOSUM ili PAM. To su već izračunate funkcije cijene koje su integrirane u mnoge bioinformatičke alate. Prije no što ih se neoprezno počne koristiti, potrebno je znati kako su te matrice izračunate (Durbin (1998)) i koje pretpostavke inherentno činimo kada ih koristimo za ocjenu poravnanja.

Ako je poznata cijena poravnanja svakog od simbola (za slučaj proteina to su aminokiseline), cijena ukupnog poravnanja može se izračunati kao suma poravnanja svakog od simbola. Željeni učinak funkcije cijene bio bi da je iz konačnog rezultata moguće vidjeti je li sličnost poravnatih nizova rezultat slučajnosti ili stvarne biološke povezanosti. Slučaj kada važi pretpostavka da su poravnanja nastala bez biološke pozadine možemo modelirati modelom $R$. U tom modelu za svaki simbol $a$ definiramo vjerojatnost $q_{a_i}$ pojavljivanja na $i$-tom mjestu u nizu. Ova vjerojatnost se može interpretirati i kao frekvencija pojavljivanja simbola. Tako se vjerojatnost pojavljivanja dvaju sljedova definira kao umnožak nezavisnih vjerojatnosti pojave pojedinih aminokiselina u svakom od sljedova:

$$P(x, y|R) = \prod_i q_{x_i} \prod_j q_{y_j} \tag{3.1}$$

Za razliku od slučajnog modela $R$, model $M$ pretpostavlja da poravnanje nije rezultat puke slučajnosti. Zbog tog se definiraju vjerojatnosti $p_{ab}$ da se će simboli a i b pojaviti na točno određenom mjestu u poravnatim sljedovima. Iz tog se može zapisati:

$$P(x, y|M) = \prod_i p_{x_i y_i} \tag{3.2}$$

Konačni rezultat poravnanja trebao bi nositi informaciju kolika je vjerojatnost da su sljedovi zaista povezani u odnosu na vjerojatnost da je postignuta sličnost rezultat slučajnosti. Omjer tih vjerojatnosti zapisuje se kao:

$$\frac{P(x,y|M)}{P(x,y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} \tag{3.3}$$

Zbog monotono rastuće prirode logaritamske funkcije, a u svrhu jednostavnijeg računa, kao rezultat se može koristiti logaritam umnoška iz 3.3:

$$S(x,y) = \sum_i s(x_i, y_i) \tag{3.4}$$

gdje je:

$$s(a,b) = log(\frac{p_{ab}}{q_a q_b}) \tag{3.5}$$

Time je ne samo pojednostavljen račun, već se ide u prilog algoritmu koji se koristi za optimalno poravnanje nizova. Takvi se algoritmi zasnivaju na dinamičkom programiranju, te će u sljedećem poglavlju postati jasno zašto je baš ovaj oblik sumacije pogodan za njegovu implementaciju. Sada je poznato kako se pomoću vjerojatnosnog modela mogu dobiti parametri za ocjenu kvalitete poravnanja. Postavlja se sljedeće pitanje: odakle su poznate te vjerojatnosti ili kako ih se može procijeniti?

Opisani problem ne bi bio težak kada bi za procjenu ovih vjerojatnosti bilo na raspolaganju mnoštvo ispravno poravnatih sljedova. Iz takvih bi se sljedova moglo izbrojati koliko puta se aminokiseline (a i b) nalaze u paru i iz toga izračunati vjerojatnosti $p_{ab}$. Moglo bi se izračunati kolike su frekvencije pojavljivanja pojedinih aminokiselina i iz toga dobiti pojedine $q_a$, te iz broja praznina izračunati $f(g)$. Prvi problem koji se može uočiti u ovako jednostavno zamišljenom rješenju problema je pretpostavka da je baza poravnatih sljedova na raspolaganju za procjenu parametara. Na koji način doći do značajne količine sljedova u čije se poravnanje može pouzdati dovoljno da se iz njih procjenjuju statistički parametri? Drugi problem možda ne postaje uočljiv odmah, ali je svejedno značajan. Ako se uspoređuju nizovi koji su se evolucijom udaljili od zajedničkog pretka prije kraćeg evolucijskog vremena, onda se može s pravom očekivati da će sve vjerojatnosti $p_{ab}$ za $a \neq b$ biti izrazito male, te će njihove logaritamske vrijednosti u odnosu na $q_a q_b$ biti izrazito negativne. Svaka razlika u nizovima bit će strogo kažnjena. Ako se to razdvajanje od zajedničkog pretka dogodilo (u evolucijskim udaljenostima) davno, svaka razlika u nizu se ne bi smjela tako strogo kažnjavati. Što je veća ta evolucijska udaljenost sljedova koje se uspoređuje, logično je za očekivati da će vjerojatnost $p_{ab}$ polako težiti u $q_a q_b$, te će svaki par poravnatih aminokiselina pridonositi za približno $0$.

## 3.1. PAM matrice

Uzimajući ove probleme u obzir, (Dayhoff M. O. (1978)) su razvili matrice zamjene nazvane PAM (Point Accepted Mutation). Svaka PAM matrica nosi na kraju i broj - poput PAM1 ili PAM250. Taj broj govori o tome koliko se mutacija dogodilo na 100 aminokiselinskih ostataka. Da bi dobili ove matrice, (Dayhoff M. O. (1978)) su prvo izračunali pretpostavljeno evolucijsko stablo za proteine visoke sličnosti - razlika nije smjela biti veća od $15\%$ siljeda. Iz tog su stabla izračunali frekvenciju $A_a b$ kojom se tokom evolucije aminokiselina $a$ zamijeni aminokiselinom $b$. Nisu uzimali u obzir smjer zamjene, pa se $A_{ab}$ povećava pri $a \to b$ i pri $b \to a$. Zbog tog što im je cilj bio izračunati matrice zamjene za veće evolucijske udaljenosti, nije im od interesa bila već opisana vjerojatnost zamjene $p_{ab}$, već vjerojatnost da je $a$ zamijenjena sa $b$ u nekakvom vremenu $t$ - $P(b|a,t) = p_{ab}(t)/q_a$. Za kratke evolucijske udaljenosti, ova uvjetna vjerojatnost može se procijeniti iz matrice $A_{ab}$ kao:

$$P(b|a) = B_{a,b} = A_{ab} / \sum_c A_{ac} \tag{3.6}$$

Ove vrijednosti je potrebno skalirati na način da nose informaciju o očekivanom broju evolucijskih promjena, odnosno zamjena aminokiselina. Kako je očekivana frekvencija zamjena u proteinu $\sum_{a \neq b} q_a q_b B_{a,b}$, ovu sumu je potrebno skalirati na željeni broj izmjena. Ako je taj broj jednak $1\%$, onda se traži takav koeficijent $\sigma$ koji će sumu svesti na $0.01$. Tako se može definirati 1-PAM matrica $C$ kao:

$$C_{ab} = \sigma B_{ab} C_{aa} = \sigma B_{aa} + (1 - \sigma) \tag{3.7}$$

Nek je 1-PAM matrica označena sa $S(1)$. Elementi te matrice odgovaraju vjerojatnostima $P(b|a, t = 1)$. Sukladno teoriji Markovljevih lanaca [1], matrica $S(n)$ može se dobiti kao $S(n) = S(1)^n$. Iz poznate formule za uvjetnu vjerojatnost vrijedi $P(b|a) = p_{ab}/q_a$, te će vrijednosti matrice zamjena biti izračunate kao

$$s(a, b|t) = log \frac{P(b|a,t)}{q_a} \tag{3.8}$$

Zbog jednostavnosti i brzine računanja, ove vrijednosti se skaliraju na najbližu cjelobrojnu vrijednost.

## 3.2. BLOSUM matrice

PAM matrice su često korištene pri računanju poravnanja. Ipak, podizanje matrice $P(1)$ na $n$-tu potenciju nije jednako proteku $n$ puta više vremena, budući da su u kra-

[1] Chapman - Kolmogorovljeve jednadžbe

ćim vremenskim periodima najčešće zamjene aminokiselina koje su rezultat izmjene jedne dušične baze[2], dok su pri duljim vremenskim periodima prisutne raznovrsnije izmjene aminokiselina. Kako su od nastanka PAM matrica postale dostupne baze evolucijski udaljenijih poravnatih proteina (s manjim postotkom sličnosti), to je omogućilo direktniji način računa matrica zamjena za udaljenije proteine, bez potrebe za postavljanjem bilo kakvih pretpostavki o odnosu evolucijski bliskih i udaljenih proteina (kao što je to model Markovljevih lanaca u slučaju PAM matrica). BLOSUM matrice (Henikoff (1992)) izračunate su iz skupa poravnatih sljedova proteinskih familija bez praznina. Sljedovi su grupirani na način da su grupe činili oni proteini čiji je postotak istih aminokiselina prešao razinu od $L\%$. Potom su iz tako grupiranih podataka izračunate frekvencije $A_{ab}$, koje označavaju broj puta kada je uočeno da je simbol $a$ iz jedne grupacije poravnat sa simbolom $b$ iz druge. Vrijednosti su skalirane faktorom $1/n_1n_2$, gdje su $n_1$ i $n_2$ veličine odgovarajućih grupacija. Iz ovako izračunatih frekvencija lako je dobiti matrice zamjene kako je to opisano u uvodnom dijelu ovog poglavlja, formulom $s(a,b) = logp_{ab}/(q_aq_b)$ (3.5). Pojedine vjerojatnosti se računaju kao:

$$q_a = \sum_b A_{ab} / \sum_{cd} A_{cd} \tag{3.9}$$

$$p_{ab} = A_{ab} / \sum_{cd} A_{cd} \tag{3.10}$$

Broj koji se nalazi uz naziv BLOSUM matrice (od kojih su primjerice poznate BLOSUM50 i BLOSUM62) ima značenje postotka sličnosti $L$ po kom se grupiraju podaci iz kojih se računaju vjerojatnosti. Niži postotak sličnosti odgovara većoj evolucijskoj udaljenosti.

## 3.3. Cijene praznina

Iako je početkom poglavlja rečeno da su pri poravnanju dozvoljene tri vrste događaja: zamjena (podudaranje i nepodudaranje se može svrstati pod zamjenu), brisanje i umetanje, nigdje u daljnjem tekstu nije bilo riječi o tome kako se kažnjava uvođenje praznine u poravnanje.

Postoje dva uobičajena načina određivanja cijene za uvođenje praznine. Prvi je linearno ovisan o duljini praznine. Ako je duljina praznine $g$, cijena će biti:

$$\gamma(g) = -gd \tag{3.11}$$

[2]tri dušične baze kodiraju jednu aminokiselinu

Druga vrsta je afina funkcija cijene, gdje se različito kažnjava uvođenje praznine i njeno produljenje. Može se zapisati kao:

$$\gamma(g) = -d - (g-1)e \tag{3.12}$$

gdje je $d$ cijena otvaranja praznine, a $e$ cijena njenog produljenja. Funkcija cijene praznine također se može opisati vjerojatnosnim modelom, točnije, može se opisati istim vjerojatnosnim modelom koji je opisan ranije u poglavlju. Vjerojatnost otvaranja praznine na pojedinom mjestu u slijedu bit će funkcija njene duljine, dakle $f(g)$, te kombinacije vjerojatnosti umetnutih simbola.

$$P(gap) = f(g) \prod_{i\ u\ praznini} q_{x_i} \tag{3.13}$$

Iz ove formule je jasno da je pretpostavljeno da simboli koji se umeću u prazninu nisu ovisni o njenoj duljini. Ta pretpostavka može, ali i ne mora biti ispravna. Ako se, zbog jednostavnosti, i dalje radi pod tom pretpostavkom, konačnu cijenu praznine dobijemo kao i u jednadžbi , dijeljeći sa produktom vjerojatnosti pojavljivanja umetnutih simbola. Kako je taj produkt prisutan i u vjerojatnosti pojavljivanja praznine, oni se krate, te se kao cijena praznine dobije $\gamma(g) = log(f(g))$. Cijena praznine je logaritamska vrijednost vjerojatnosti da će praznina biti upravo duljine $g$.

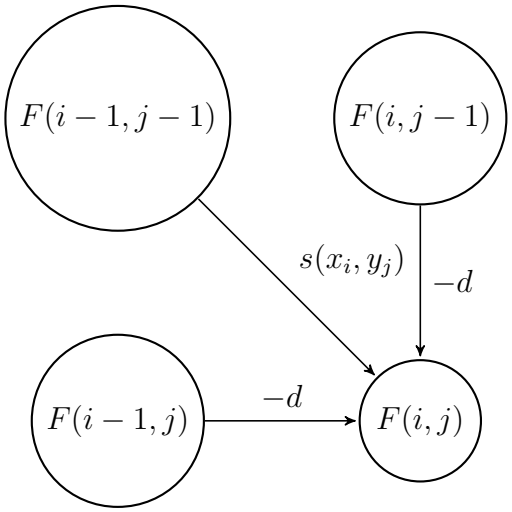
# 4. Algoritmi poravnanja

Uz pretpostavku da je potrebno poravnati dva niza duljina $n$, uz dopušteno umetanje praznina, bit će moguće napraviti $\binom{2n}{n}$ poravnanja. Takva je složenost u praktičnim rješenjima nedopustiva pa očito neće biti moguće napraviti svako od mogućih poravnanja i odlučiti se za najbolje. Dinamičko programiranje daje algoritam pomoću kojeg se najbolje rješenje može pronaći u $O(n^2)$. Osim te očite prednosti, algoritam dinamičkog programiranja se može izrazito lako modificirati tako da se za dva slijeda pronađe najbolje globalno ili lokalno rješenje, da se u jednom nizu identificira motiv sadržan u drugom i slično.

## 4.1. Globalno poravnanje

Osnovni algoritam dinamičkog programiranja je jednostavan (Neil Jones (2004)). Ovo je olakotna okolnost za sve bioinformatičare budući da se gotovo svi algoritmi za poravnavanje sljedova oslanjaju na dinamičko programiranje. Ideja dinamičkog programiranja je da se uvijek može brzo i jednostavno izračunati optimalno rješenje za jednostavne slučajeve, a onda iz tih jednostavnih (i optimalnih) rješenja moguće je konstruirati rješenje za kompliciranije slučajeve. Dinamičko programiranje implementira se na način da se napravi tablica, kojoj retke definira jedan, a stupce drugi slijed kog se poravnava.

Na slici 4.1 se može ilustrirati jednostavan princip popunjavanja tablice dinamičkog programiranja. Neka je trenutni položaj u tablici $F(i, j)$. Na čas se mogu zanemariti rubni slučajevi, to jest prvi redak i prvi stupac za koje ovaj graf nije ilustrativan. U svaku ćeliju matrice može se doći na tri načina, kao što se i vidi na slici. Put iz stanja $F(i-1, j)$ u $F(i, j)$ odgovara umetanju praznine u 'lijevi' niz, dok put iz $F(i, j-1)$ u $F(i, j)$ odgovara umetanju praznine u 'gornji' niz. Oba ova puta su težine $-d$, što je već spomenuto kao cijena umetanja praznine u niz. Put iz $F(i-1, j-1)$ u $F(i, j)$ odgovara zamjeni aminokiselina i njegova težina je jednaka elementu matrice zamjene za aminokiseline $x$ i $y$ - $s(x_i, y_j)$. Kako će optimalno rješenje uvijek biti maksimalno




**Slika 4.1:** Prikaz principa dinamičkog programiranja

rješenje zbog 3.5, u svakoj ćeliji traži se maksimalna od ove tri vrijednosti. To se može zapisati kao:

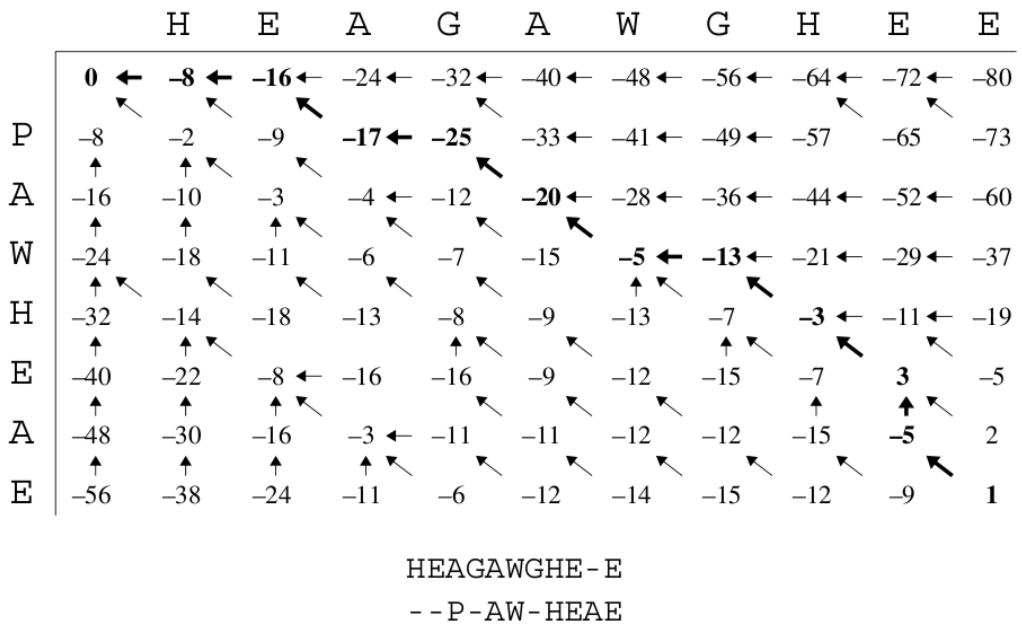
$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i,y_i) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \tag{4.1}$$

Na slici se može vidjeti rezultat primjene formule 4.1. Sada je vrijeme da se objasne rubni elementi. $F(0,0)$ se uvijek postavlja na $0$. Prvi redak i stupac je umetanje praznina u 'lijevi', odnosno 'gornji' niz. Zbog toga se svaki pomak prema dolje u prvom stupcu i prema desno u prvom retku kažnjava sa $-d$, što je cijena umetanja praznine. Za svaku od ostalih ćelija može se primijeniti formula 4.1.

Što je bitno u dinamičkom programiranju, a može se uočiti i na slici **??** je da je potrebno iz svake ćelije pamtiti pokazivače na ćeliju iz koje se došlo. Time je omogućena rekonstrukcija najboljeg poravnanja. Naravno, ovakvih poravnanja može biti i više, ako se u ćeliji najbolji rezultat može postići iz više susjedih ćelija, a ne samo jedne.

## 4.2. Lokalno poravnanje

Svaka je dva slijeda moguće globalno poravnati. Ipak, time se inzistira na sličnosti sljedova koja se proteže njihovom cijelom duljinom. U stvarnim slučajevima ovo često nije istina. Zbog toga bi bilo dobro kada bi se moglo omogućiti da se između dva




**Slika 4.2:** Primjer poravnanja dva proteinska slijeda upotrebom osnovnog algoritma dinamičkog programiranja

slijeda nađe lokalno optimalno poravnanje, dok ostatak slijeda ne utječe na kvalitetu poravnanja. Ovo je najbolji način otkrivanja sličnosti sljedova proteina koji su evolucijski udaljeni. Modifikacija opisanog algoritma za globalno poravnanje u algoritam prikladan za lokalno poravnanje je izrazito jednostavna i može se opisati sljedećom formulom:

$$F(i,j) = max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_i) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \tag{4.2}$$

Značenje nule koja nije bila prisutna u formuli 4.1, ali je se može pronaći u 4.2 je početak novog lokalnog poravnanja. Ako je najbolje moguće poravnanje do neke točke $(i, j)$ negativno, bolje je započeti novo poravnanje nego nastaviti započeto. Zbog toga će i rubni elementi matrice dinamičkog programiranja biti popunjeni nulama, ne više cijenom umetanja praznina, $-id$ ili $-jd$. Još jedna razlika u odnosu na globalno poravnanje je ta da nam se sada najbolji rezultat neće nalaziti u donjoj desnoj ćeliji matrice, već se može nalaziti na bilo kojem mjestu u matrici.

Uvjet da bi ovakav algoritam dao kvalitetne rezultate je taj da su očekivani rezultati slučajnog poravnanja negativni. U protivnom bi slučajna poravnanja dominirala

samo zbog njihove duljine. Matematička interpretacija ovog uvjeta za poravnanje bez praznina je:

$$\sum_{a,b} q_a q_b s(a,b) < 0$$

Nažalost, formalna analiza utjecaja praznina na lokalno u odnosu na globalno poravnanje ne postoji. Zbog toga je potrebno posebno obratiti pažnju na postavljenje cijena praznina.

Globalno i lokalno poravnanje su samo dva primjera upotrebe algoritma dinamičkog programiranja. Osnovni algoritam se može modificirati da pronalazi poravnanja podsljedova koja se ponavljaju ili podsljedova koji se u potpunosti preklapaju.

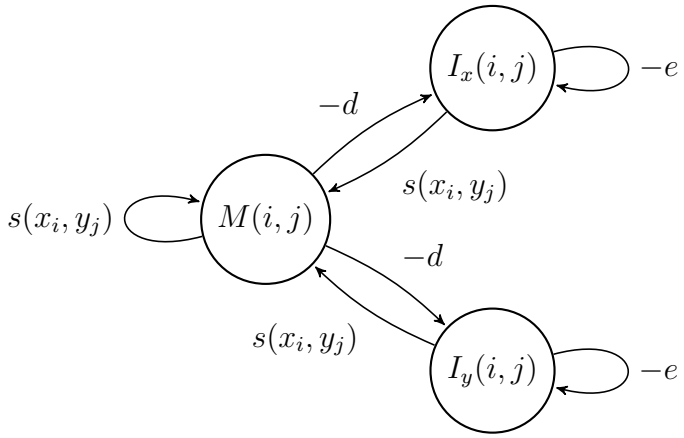
## 4.3. Korištenje afine funkcije cijene praznine u dinamičkom programiranju

Linearna funkcija cijene praznine ne opisuje dobro način i vjerojatnost nastajanja praznine u biološkom slijedu. Kada je riječ o biološkim sljedovima, veća je vjerojatnost jedne duže praznine nego nekolicine malih. Zbog tog je pri poravnanju dobro koristiti afinu funkciju cijene praznine (jednadžba 3.3) Opisani algoritam (4.1) je jednostavno modificirati da koristi takvu funkciju cijene praznine:

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_i) & \\ F(k,j) - \gamma(i-k) & \text{k = 0,...,i-1} \\ F(i,k) - \gamma(j-k) & \text{k = 0,...,j-1} \end{cases} \tag{4.3}$$

Ovakav bi pristup povećao složenost algoritma s $O(n^2)$ na $O(n^3)$ jer je u svakom poraku potrebno provjeravati $i + j + 1$ prethodnika ćelije, ne samo tri, kao što je to u slučaju globalnog poravnanja s linearnom funkcijom cijene. Srećom, model se može modificirati da radi u $O(n^2)$. Umjesto jedne varijable po ćeliji, $F(i,j)$, potrebno je pamtiti 3 varijable. Jedna će i dalje pamtiti dosad najbolje postignuto poravnanje, dok će druge dvije pamtiti je li se u prethodnom koraku dogodilo otvaranje praznine u jednom, odnosno drugom nizu. Varijable su prikladno imenovane $I_x$ i $I_y$ i simboliziraju umetanje (Insert) u niz $x$, odnosno $y$.

Umjesto standardnog jednog stanja kog je trebalo računati za svaki korak po formuli (4.1), sada će u svakom koraku trebati obnavljati vrijednosti triju varijabli - $M(i,j)$, $I_x(i,j)$ i $I_y(i,j)$. Cijene prijelaza se mogu iščitati iz automata stanja prikazanog na slici 4.3.




**Slika 4.3:** Dijagram stanja za implementaciju poravnanja dva niza pomoću dinamičkog programiranja uz afinu cijenu praznine

$$
\begin{aligned}
\text{M(i,j)} &= \max \begin{cases} M(i-1,j-1) + s(x_i, y_i) \\ I_x(i-1,j-1) + s(x_i, y_i) \\ I_y(i-1,j-1) + s(x_i, y_i) \end{cases} \\
\text{I}_x(i,j) &= max \begin{cases} M(i-1,j) - d \\ I_x(i-1,j) - e \end{cases} \\
\text{I}_y(i,j) &= max \begin{cases} M(i,j-1) - d \\ I_y(i,j-1) - e \end{cases}
\end{aligned}
\tag{4.4}
$$

Ovaj model moguće je dodatno pojednostaviti uklanjanjem jedog od $I$ stanja. Preostalo $I$ stanje preuzima ulogu proširenja praznine i u jedan i u drugi niz i semantički odgovara tome da se poravnanje trenutno nalazi u praznini, neovisno o kojem je nizu riječ. Iako ovaj model ima veću mogućnost pogreške, ta pogreška je lokalizirana na dijelove poravnanja u kom se nalaze praznine, što ionako odgovara dijelovima niza koji nisu slični. Greška na takvom mjestu nije od velike važnosti za ukupno poravnanje, pa je stoga takav model prihvatljiv.

Kako je računska složenost ovakvog načina poravnanja još uvijek iznimno velika, potrebno je razviti načine poravnanja koji će biti prikladni za pretraživanje velikih baza bioloških sljedova. Takvi algoritmi koriste pretpostavke o prirodi problema i uobličuju ih u heurističke funkcije, te su po tom i poznati kao heuristički algoritmi.

## 4.4. Heuristički algoritmi

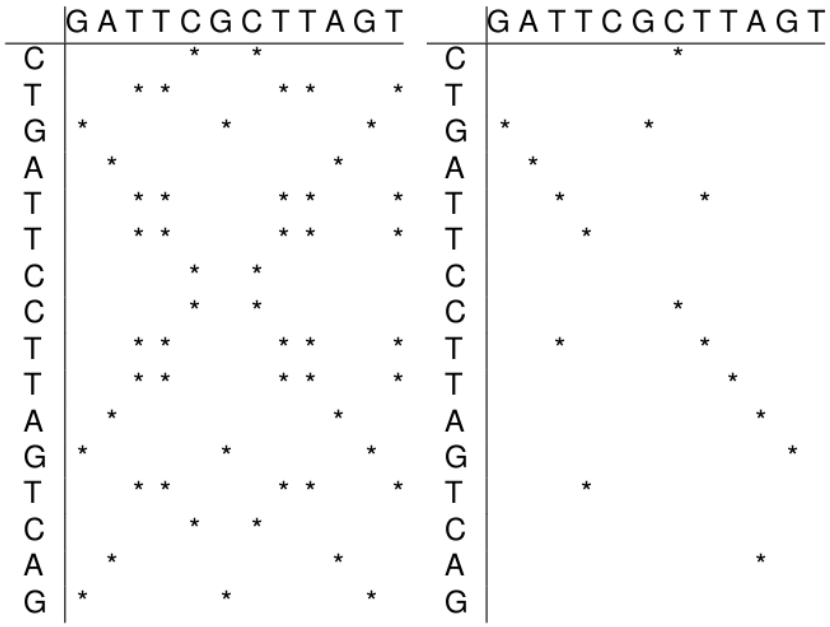
Priroda problema kog se pokušava riješiti uvijek nalaže smjer u kom se treba kretati. Ono što danas interesira znanstvenike koji bi htjeli sakupiti informacije o tek sekvenciranom proteinu ili genu je postoje li sekvencirani proteini koji se nalaze u dostupnoj bazi bioloških sekvenci čija je funkcija poznata i koji pokazuju značajnu sličnost s novo sekvenciranim proteinom. Prethodno opisani algoritmi dinamičkog programiranja su možda bili prikladni kada se broj otkrivenih proteina kretao oko nekoliko tisuća. Danas je taj broj znatno veći i za pretraživanje takvih baza vremenska složenost od $O(n^2)$ nije prihvatljiva. Iz nužde su nastali algoritmi koji ne daju garanciju pronalaska optimalnog rješenja, ali su njihovi rezultati dovoljno dobri, posebice uzme li se u obzir reducirana vremenska složenost. Primjeri heurističkih algoritama za poravnavanje bioloških sekvenci su FASTA i BLAST.

Prvi korak u reduciranju vremena potrebnog za pretraživanje baze je organizacija sadržanih podataka u učinkovite strukture koje će omogućiti upotrebu učinkovitijih algoritama od onih do sada predstavljenih. Primjer struktura koje omogućavaju pronalazak motiva u čitavoj bazi u linearnom vremenu (ovisnom samo o duljini motiva) su sufiks-stablo i tablice raspršenog adresiranja.

Heuristički algoritmi se zasnivaju na pronalasku istih (ili sličnih) kratkih podsljedova (duljine 2 ili 3 za proteine i 8,9 za DNK) sekvenci koje se uspoređuju, koje je potom cilj proširiti i pronaći lokalno maksimalno poravnanje. Što je duljina tog podniza veća, pretraživanje je brže, ali je vjerojatnost pronalaska optimalnog rješenja tim manja. Za podslijed duljine jedan, riječ je o algoritmu dinamičkog programiranja.

Poravnanje se definira kao lokalno maksimalno ako ga nije moguće proširiti ni u jednom smjeru. Ideja koja se uvijek proteže pri poravnanju sljedova je da nije uvijek potrebno postići identično poravnanje, potrebno je dopustiti neki od 3 događaja: umetanje, brisanje ili zamjenu. Neka je $l$ značajna duljina poravnatih podsljedova. Ako se unutar tog poravnanja želi dopustiti jedno nepodudaranje, ono će se nalaziti ili u prvoj ili u drugoj polovici slijeda duljine $l$. To znači da je potrebno naći upola kraće sljedove duljine $l/2$ ili $(l-1)/2$ koji će u potpunosti odgovarati duž cijelog niza te na osnovu njihove razdiobe u nizu zaključiti jesu li nastali slučajno ili ne.

FASTA (fast-All) algoritam za takvo što koristi matrice kakve se mogu vidjeti na slici **??**. Nakon što su pronađena sva lokalna poravnanja (zvjezdice u matrici), potrebno je naći podslijed koji bi odgovarao stvarnom lokalno maksimalnom poravnanju. Takvi podsljedovi pronalaze se u regijama gdje se u dijagonali nalazi mnogo zvjezdica. Ostala lokalna poravnanja mogu se smatrati šumom. Nakon što su lokalno maksi-




**Slika 4.4:** Primjer matrice generirane uspoređujući nizove GATTCGCTTAGT i CTGATTCCTTAGTCAG pomoću FASTA algoritma. Zvjezdice označavaju da je u toj točki postignuto željeno poravnanje. Na lijevoj se slici mogu vidjeti sva postignuta poravnanja, a na desnoj važna poravnanja. To je matrica iz koje je uklonjen šum.

malna poravnanja pronađena, FASTA će najbolje rezultate ponovno obraditi koristeći algoritam dinamičkog poravnanja i time dati potpuno ispravan rezultat (u terminima definirane funkcije cijene).

Problem kod FASTA algoritma je način na koji on određuje inicijalno poravnanje podnizova (obično duljine $k = 2, 3 \ldots$ za proteine). On uspoređuje jednakost prisutnih aminokiselina. Ovakav pristup će preskočiti biološki značajna poravnanja zbog razlike u jednoj aminokiselini slične funkcije onoj koja se traži.

BLAST (Basic Local Alignment Search Tool) nudi odgovor na ovaj problem. U BLAST-u se za ocjenu poravnanja koriste matrice zamjena poput onih opisanih u poglavlju 3. Time je vjerojatnost pronalaska biološki značajnih poravnanja povećana. Rezultat poravnanja inicijalnih segmenata definira se kao i kod dinamičkog programiranja, kao suma cijena zamjena pojedinih aminokiselina $\sum_{i=1}^{l} s(x_i, y_i)$. Veće vrijednosti odgovaraju boljem poravnanju. Složenost izvornog BLAST algoritma može se opisati formulom (prema Altschul (1990)):

$$aW + bN + cNW/20^w \tag{4.5}$$

gdje je $W$ broj generiranih riječi pomoću kojih se radi podudaranje, te sukladno tome prvi član $aW$ odgovara vremenu potrebnom za generiranje liste $k$-torki rezidua. $N$

je broj rezidua i drugi član $bN$ odgovara vremenu potrebnom za pretraživanje baze. Treći član odgovara vremenu potrebnom za produljenje pronađenih preklapanja sljedova. $a$, $b$ i $c$ su konstante. BLAST je jedan od najkorištenijih bioinformatičkih alata (Casey (2005)) i njegove izvedbe nude, među ostalim, usporedbu sljedova aminokiselinskih ostataka, sljedova gena, imunoglobulina, pronalazak homolognih proteina te usporedbu mišjih i čovječjih sljedova DNK.

Pri pretraživanju baze proteina ili DNK, nije cilj pronaći samo jedno najbolje poravnanje. Dakle, potrebno je odrediti koja su od postignutih poravnanja od interesa. To se može napraviti koristeći neki od statističkih modela opisanih u narednom poglavlju.

# 5. Određivanje statističke važnosti postignutog poravnanja

Jednom kada su postignuta sva željena poravnanja, ona kao izlaz daju brojčani rezultat, najčešće cjelobrojni. Postavlja se pitanje kakvu informaciju nosi taj rezultat i može li se iz njega odrediti vjerojatnost da su nizovi zaista biološki povezani ili je za ocjenu poravnanja potrebno možda koristiti drugu mjeru osim samog rezultata? Postoji više načina ocjene značaja postignutog rezultata te će u tekstu biti opisana dva statistička modela - Bayesov i klasični statistički pristup koji funkcionira na principu usporedbe rezultata poravnanja slučajno odabranih nizova s onima koje je potrebno ocijeniti.

Bayesova formula je jedna od poznatijih u području vjerojatnosti. Bayesov teorem daje rješenje pri traženju uvjetnih vjerojatnosti i glasi:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5.1}$$

U poglavlju 3 opisano je kako se računaju uvjetne vjerojatnosti $p(x, y|M)$. Ipak, $p(M|x, y)$ je ta koja daje informaciju o tome kolika je vjerojatnost da su nizovi povezani u odnosu na vjerojatost da nisu. Gledajući Bayesovu formulu postaje očigledno da iz prethodno izračunatih vjerojatnosti $p(x, y|M)$ je moguće jednostavno dobiti i vjerojatnost $p(M|x, y)$. Informacija koja nam nedostaje su apriorne vjerojatnosti pretpostavljenih modela $M$ i $R$. Ove vjerojatnosti se ne mogu odrediti iz podataka, već će one predstavljati očekivanje onoga tko obavlja poravnanje da će sljedovi zaista biti povezani. Za apriorne vjerojatnosti vrijedit će $P(R) = 1 - P(M)$. Sada je moguće izračunati aposteriornu vjerojatnost kao:

$$P(M|x, y) = \frac{P(x, y|M)P(M)}{P(x, y)} = \frac{P(x, y|M)P(M)}{P(x, y|M)P(M) + P(x, y|R)P(R)} \tag{5.2}$$

$$= \frac{P(x, y|M)P(M)/P(x, y|R)P(R)}{1 + P(x, y|M)P(M)/P(x, y|R)P(R)}$$

Već su opisane prednosti korištenja logaritamskog računa, pa se sukladno tome mogu uvesti oznake:

$$S' = S + log(\frac{P(M)}{P(R)}) \tag{5.3}$$

gdje je $S$:

$$S = log(\frac{P(x, y|M)}{P(x, y|R)}) \tag{5.4}$$

Vjerojatnost $P(M|x, y)$ se može opisati sigmoidalnom funkcijom:

$$P(M|x, y) = \sigma(S') = \frac{e^{S'}}{1 + e^{S'}} \tag{5.5}$$

Iz ovog je vidljiva interpretacija dobivene vjerojatnosti. Kada $S'$ teži u $-\infty$, tada će vjerojatnost $P(M|x, y)$ biti nula. Kada pak $S'$ teži u $+\infty$, vjerojatnost će biti jedan. Kada je $S' = 0$, vjerojatnost $P(M|x, y) = 1/2$.

$S'$ je već korišteni logaritamski omjer $\frac{P(x,y|M)}{P(x,y|R)}$, ali pomnožen omjerom apriornih vjerojatnosti $\frac{P(M)}{P(R)}$, što ima i intuitivnog smisla.

Ono što je ključno pri korištenju Bayesovog modela je da korištene vrijednosti zadovoljavaju i uvjete koji se na vjerojatnosti postavljaju - moraju se kretati u intervalu $(0, 1)$ i njihova suma mora biti jednaka jedan. Ako ovi uvjeti nisu zadovoljeni, dobiveni rezultati neće biti indikativni.

Ono što je možda jednostavnije za razumjeti i primijeniti od Bayesovog pristupa je standardni statistički pristup. Potrebno je izračunati optimalna poravnanja $N$ slučajno odabranih nizova. Ako je vjerojatnost da će rezultat ciljnog poravnanja biti lošiji od maksimuma slučajno poravnatih nizova mala, onda se s određenom dozom sigurnosti može reći da je postignuto poravnanje značajno, barem gledano statistički. Matematički model korišten za ovakav proračun je složen pa će na uvid biti predstavljen samo konačni kriterij - da bi rezultat $S$ bio značajan, treba biti zadovoljeno:

$$S > T + \frac{logmn}{\lambda} \tag{5.6}$$

gdje su $m$ i $n$ duljine nizova, a $\lambda$ se može izračunati iz uvjeta da vrijedi

$$\sum_{a,b} q_a q_b e^{\lambda s(a,b)} = 1 \tag{5.7}$$

# 6. Zaključak

Poravnanje bioloških sljedova je jedan od osnovih alata današnjeg bioinformatičara. Zbog prirode nastanka novih sljedova, moguće se pouzdati u rezultate o sličnosti i povezanosti bioloških sekvenci dobivene na temelju njihovog poravnavanja. Pri određivanju poravnanja potrebno je znati ocijeniti težinu zamjene, umetanja i brisanja pojedinog simbola iz slijeda. Ove ocjene uobličene su u funkcije cijene, a funkcije cijene u javno dostupne matrice zamjena aminokiselina. Algoritam korišten za pronalazak globalno optimalnog rješenja pri poravnanju dva slijeda naziva se dinamičko programiranje. U dinamičko se programiranje vrlo lako mogu integrirati različite funkcije cijene, te je osim toga lako sitnim izmjenama postići različite ciljeve - pronalazak lokalno optimalnog rješenja, pronalazak motiva u slijedu i slično. Kako je vremenska složenost ovih algoritama kvadratna, pri pretraživanju velikih baza sljedova koriste se heuristički algoritmi, poput BLAST-a, koji ostvaruju značajne vremenske uštede. Nakon izračunatih poravnanja, potrebno je odrediti jesu li ona rezultat slučajnosti ili biološke povezanosti. Postoje razni vjerojatnosni modeli pomoću kojih je moguće izračunati vjerojatnost da je dobiveni rezultat zaista biološki značajan. Primjeri takvih modela su Bayesov i standardni vjerojatnosni model.

# 7. Literatura


Gish Miller Myers Lipman Altschul. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, listopad 1990.

R. M. Casey. Blast sequences aid in genomics and proteomics. *Business Intelligence Network*, 2005.

Schwartz R. M. Orcutt B. C. Dayhoff M. O. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978.

Krogh Mitchison Durbin, Eddy. *Biological Sequence Analysis - Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

Henikoff J. G. Henikoff, S. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

Pavel Pevzner Neil Jones. *An Introduction to Bioinformatics Algorithms*. Massachusetts Institute of Technology, 2004.