Vícenásobné zarovnání

Bioinformatika

Tomáš Martínek

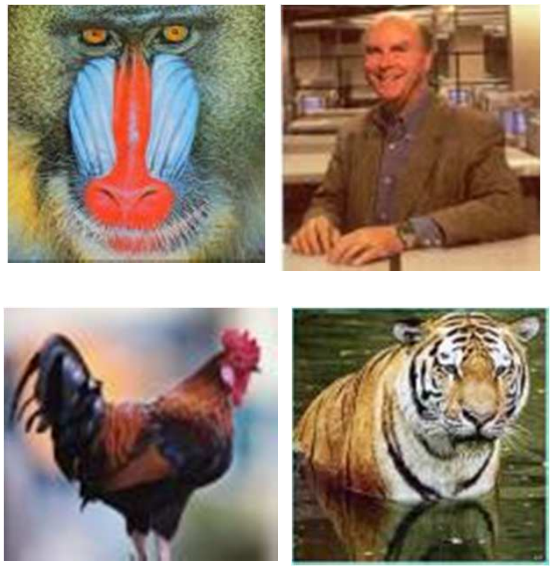
martinto@fit.vutbr.cz

Osnova

- Úvod
- Dynamické programování
- Heuristické metody
  - CLUSTAL
  - MUSCLE
- Shrnutí

Motivace

- Doposud jsme zarovnávali pouze dvě sekvence
- Co takhle zarovnávání více sekvencí?
- Důvody:
  - Části společné u více organizmů ukazují na oblasti, které mohou mít významnou funkci
  - Naopak neshodující se části ukazují na charakteristické rysy, ve kterých se organizmy liší
  - Vícenásobné zarovnání mezi podobnými sekvencemi dokonce zpřesňuje zarovnání, které by u dvojice sekvencí nebylo tak zřetelné
- Použití:
  - Konstrukce fylogenetických stromů
  - Predikce struktury a funkce proteinů



Příklad:

```
A T - G C G -
A - C G T - A
A T C A C – A
A T C G – G A
```

Ohodnocení zarovnání

- Předpokládejme vícenásobné zarovnání generované některým algoritmem
- Jak ohodnotit kvalitu vícenásobného zarovnání?
- Příklad:
  - zarovnání 2 vs. 3 sekvencí?

A	T	-	G	C	G	-
A	-	C	G	T	-	A
1	-1	-1	1	0	-1	-1

Skóre: -2

A	T	-	G	C	G	-
A	-	C	G	T	-	A
A	T	C	A	C	-	A
?	?	?	?	?	?	?

Skóre: ??

Ohodnocení zarovnání

- Počet shodných znaků
- Vlastnosti:
  - Velmi jednoduchý způsob
  - Funguje dobře pouze pro zarovnání s velkým počtem identických znaků
- Příklad:

A	T	C	G	C	G	A
A	-	C	G	T	-	A
A	T	C	G	C	-	A
1	0	1	1	0	0	1

Skóre: 4

A	T	-	G	C	G	-
A	-	C	G	T	-	A
A	T	C	A	C	-	A
1	0	0	0	0	0	0

Skóre: 1

Ohodnocení zarovnání

- Suma párů (Sum of Pairs)
- Myšlenka:
  - Skóre vícenásobného zarovnání je vysoké, pokud je vysoké i skóre zarovnaných dvojic sekvencí
- Výpočet:
  - Pro každý sloupec zarovnání je vypočteno skóre všech dvojic znaků

$$S_{Col}(a_1, a_2, ..., a_n) = \sum_{i \neq j} S(a_i, a_j)$$

$$S(a_i, a_j) = \begin{cases} M(a_i, a_j) & if\ (a_i \,\&\, a_j \neq -) \\ 0 & if\ (a_i \,\&\, a_j = -) \\ -g & otherwise \end{cases}$$

  - kde M(*x*, *y*) je položka skórovací matice a *g* je penalizace za vložení mezery
  - Celkové skóre je suma přes všechny sloupce

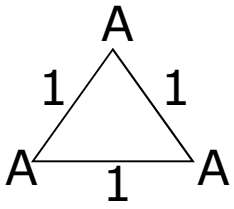
Ohodnocení zarovnání

- Příklad:

A	T	-	G	C	G	-
A	-	C	G	T	-	A
A	T	C	A	C	-	A
3	-1	-1	1	1	-2	-1

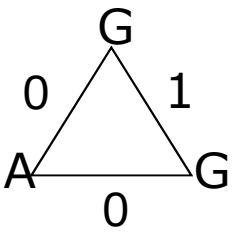
Skórovací matice

	A	T	C	G
A	1	0	0	0
T	0	1	0	0
C	0	0	1	0
G	0	0	0	1



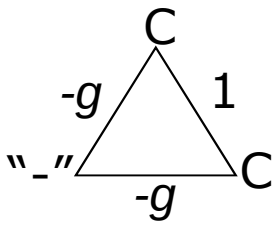
Sloupec 1

Score=3



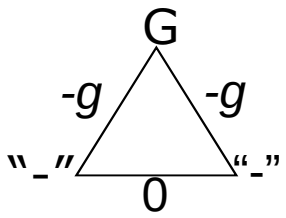
Sloupec 4

Score = 1



Sloupec 3

Score = 1–2g=-1



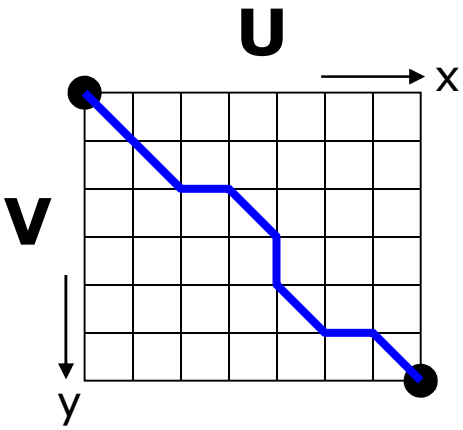
Sloupec 3

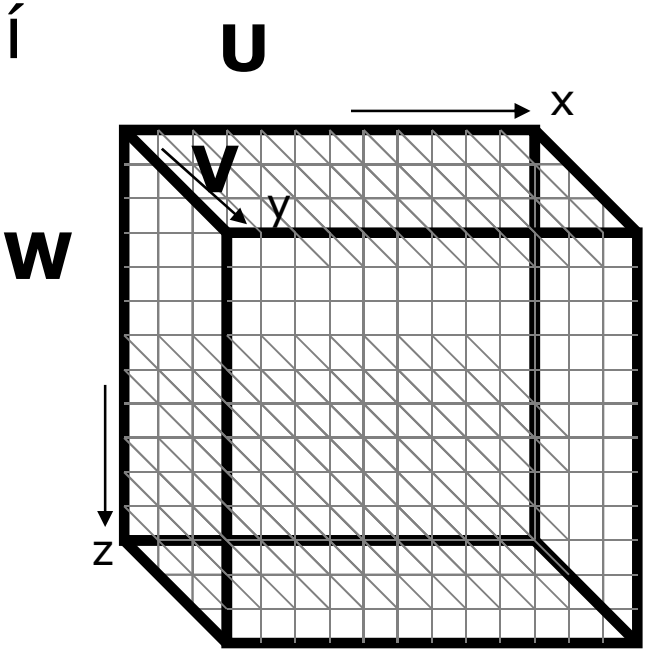
Score = –2g=-2

- Důležitá poznámka:
  - pro *n* sekvencí je potřeba porovnat $\binom{n}{2}$ párů

Dynamické programování

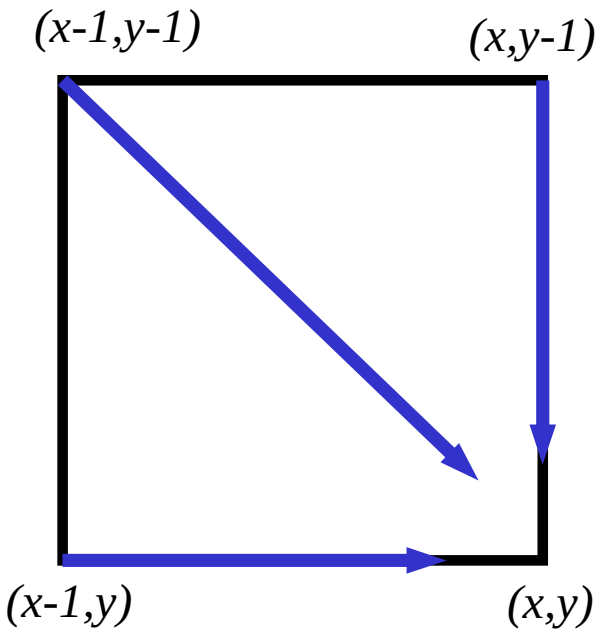
- Zobecnění přístupu ze zarovnání dvou sekvencí na zarovnání *n* sekvencí
- Výpočet *n*-rozměrné matice, každá sekvence reprezentuje jednu souřadnou osu
- Příklad: 2D vs. 3D zarovnání



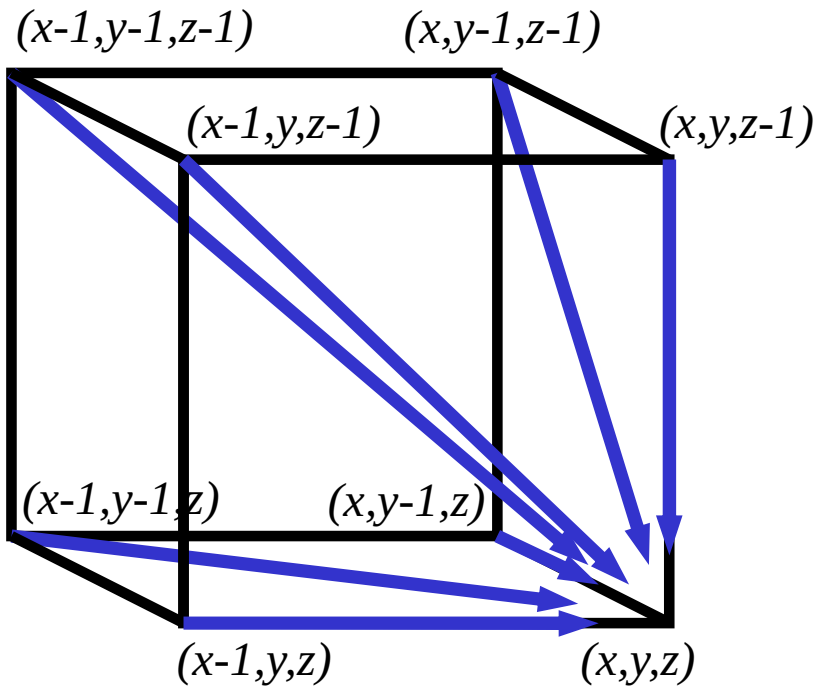


Dynamické programování

- Příklad: 2D vs. 3D zarovnání
  - Pravidlo výpočtu dynamického programování



2D: zarovnání ze 3 možných směrů

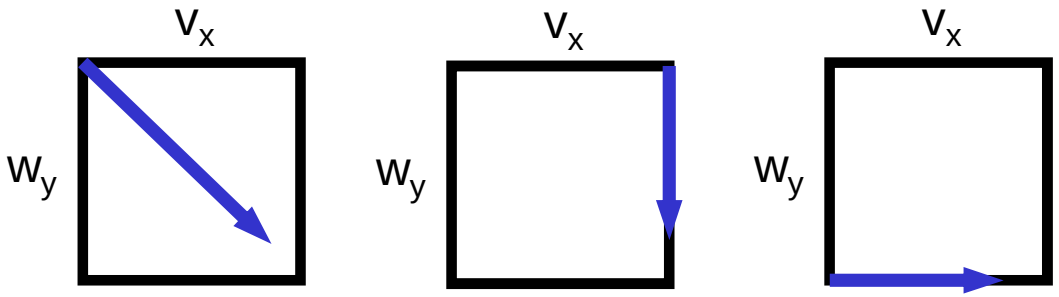


3D: zarovnání ze 7 možných směrů

Dynamické programování

- 2D: zarovnání ze 3 možných směrů

1	2	3
v_x	v_x	-
w_y	-	w_y



- Výpočet skóre
  - $S_{x,y} = \max \begin{cases} S_{x-1,y-1} + M(v_x, w_y) \\ S_{x,y-1} - g \\ S_{x-1,y} - g \end{cases}$
  - kde $M(x, y)$ je položka skórovací matice a g je penalizace za vložení mezery

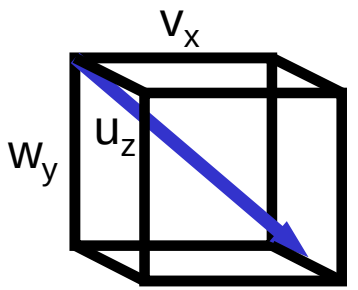
	A	T	C	G
A	1	0	0	0
T	0	1	0	0
C	0	0	1	0
G	0	0	0	1

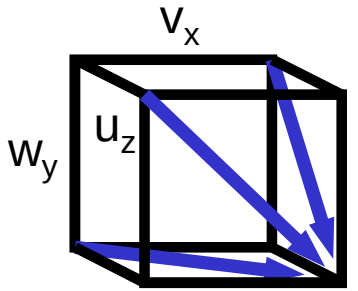
Skórovací matice

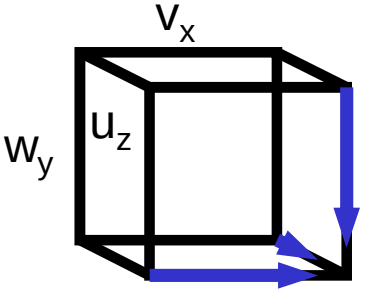
Dynamické programování

- 3D: zarovnání ze 7 možných směrů

1	2	3	4	5	6	7
v_x	-	v_x	v_x	-	v_x	-
w_y	w_y	-	w_y	-	-	w_y
u_z	u_z	u_z	-	u_z	-	-







- Výpočet skóre:

$$S_{x,y,z} = \max \begin{cases} S_{x-1,y-1,z-1} + M(v_x, w_y) + M(v_x, u_z) + M(w_y, u_z) \\ S_{x-1,y-1,z} + M(v_x, w_y) - 2*g \\ S_{x-1,y,z-1} + M(v_x, u_z) - 2*g \\ S_{x,y-1,z-1} + M(w_y, u_z) - 2*g \\ S_{x-1,y,z} - 2*g \\ S_{x,y-1,z} - 2*g \\ S_{x,y,z-1} - 2*g \end{cases}$$

Založeno na principu sumy párů

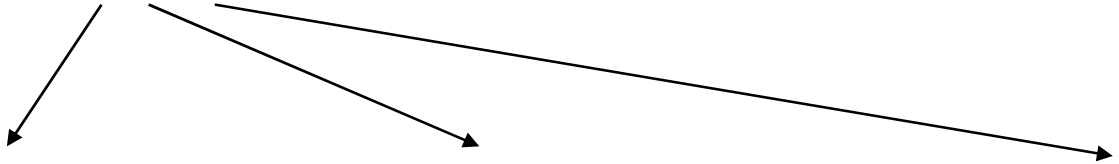
Dynamické programování

- Časová složitost:
  - Pro 3 sekvence délky n je potřeba vypočítat $7n^3$ součtů, časová složitost je $O(n^3)$
  - Pro obecně k sekvencí je sestavena k-dimensionální matice a vypočteno $(2^k-1)(n^k)$ součtů, časová složitost je $O(2^k n^k)$
- Zhodnocení:
  - Časová složitost roste exponenciálně s počtem sekvencí
  - Přístup je použitelný do nejvýše 20 (relativně) krátkých sekvencí

Více-násobné zarovnání

- Z více-násobného zarovnání můžeme jednoduše sestrojit zarovnání dvojic (i když ne zcela optimální)
- Příklad:

```
x: AC-GCGG-C
y: AC-GC-GAG
z: GCCGC-GAG
```



```
x: ACGCGG-C;   x: AC-GCGG-C;   y: AC-GCGAG
y: ACGC-GAC;   z: GCCGC-GAG;   z: GCCGCGAG
```

Více-násobné zarovnání

- Pokud máme k dispozici zarovnání všech dvojic:

```
x: ACGCTGG-C;   x: AC-GCTGG-C;   y: AC-GC-GAG
y: ACGC--GAC;   z: GCCGCA-GAG;   z: GCCGCAGAG
```

- Můžeme získat více-násobné zarovnání?

  => Bohužel, ne vždy

Více-násobné zarovnání

- Příklad:
  - zarovnání sekvencí:
    - AAAATTTT
    - TTTTGGGG
    - AAAAGGGG
- Kompatibilní sekvence

Více-násobné zarovnání

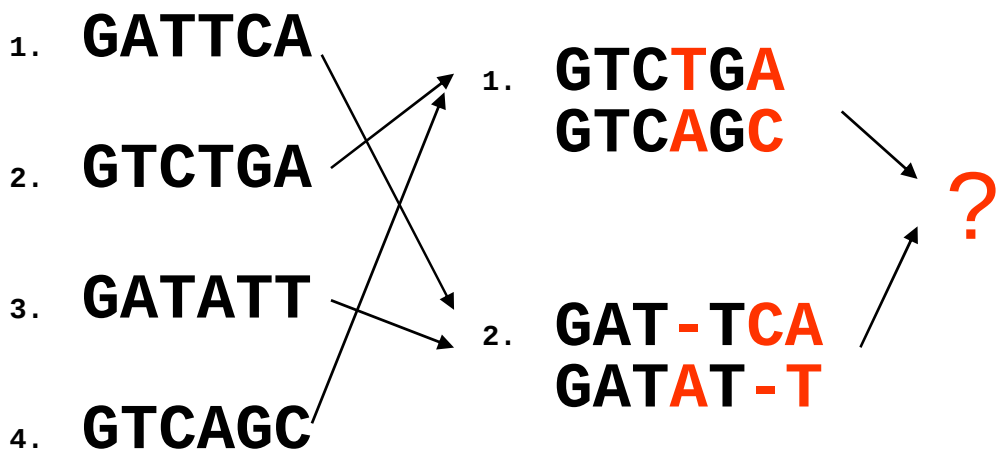
- Příklad:
  - zarovnání sekvencí:
    - AAAATTTT
    - TTTTGGGG
    - GGGGAAAA
- Nekompatibilní sekvence

Progresivní metody

- Postup:
  1. Z množiny *k* sekvencí zarovnat dvě a získat tak množinu *k-1* sekvencí
  2. Opakovat, dokud nejsou zarovnány všechny sekvence
- Problémy:
  1. Jak zarovnat sekvenci k zarovnání nebo zarovnání k zarovnání?
  2. Jak vybrat pořadí, ve kterém se budou jednotlivé dvojice zarovnávat?

Zarovnání sekvence a zarovnání

- Jak zarovnat sekvenci k zarovnání nebo zarovnání k zarovnání?
- Příklad:



	G	T	C	t/a	G	a/c
	0	-1	-2	-3	-4	-5	-6
G	-1	1	0	-1	?		
A	-2	0	1	0	?		
T	-3	-1	1	0	?		
-/a	-4	?	?	?	?		
T	-5						
c/-	-6						
a/t	-7						

Zarovnání sekvence a zarovnání

- Příklad:

		G	T	C	t/a	G	a/c
	0	-1	-2	-3	-4	-5	-6
G	-1	1	0	-1	$x_{1,4}$		
A	-2	0	1	0	$x_{2,4}$		
T	-3	-1	1	0	$x_{3,4}$		
-/a	-4	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$		
T	-5						
c/-	-6						
a/t	-7						

$$x_{3,4} = \max \begin{cases} x_{2,4} + (-1) \\ 0 + (-1) \\ 0 + \text{cmp}(T,t/a) \end{cases}$$

$$x_{4,4} = \max \begin{cases} x_{3,4} + (-1) \\ x_{4,3} + (-1) \\ 0 + \text{cmp}(-/a,t/a) \end{cases}$$

cmp(T,t/a) = (M(T,T) + M(T,A))/2 = (1 + 0)/2 = 0,5

cmp(-/a,t/a) = (g + g + M(A,T) + M(A,A))/4 = ((-1)+(-1)+0+1)/4 = -0,25

Zarovnání sekvence a zarovnání

- Čím více je zarovnaných sekvencí, tím horší je přehlednost popisu sloupců a řádků DP matice => používá se zápis ve formě profilu, tj. procentuální zastoupení jednotlivých znaků
- Příklad:

```
 -  A  G  G  C  T  A  T  C  A  C  C  T  G
 T  A  G  –  C  T  A  C  C  A  -  -  -  G
 C  A  G  –  C  T  A  C  C  A  -  -  -  G
 C  A  G  –  C  T  A  T  C  A  C  –  G  G
 C  A  G  –  C  T  A  T  C  G  C  –  G  G
```

```
A    1                 1        .8
C   .6           1        .4  1    .6 .2
G         1 .2                  .2        .4  1
T   .2                 1    .6             .2
-   .2         .8                      .4 .8 .4
```

Profil množiny zarovnaných sekvencí

- Jak zarovnávat profily mezi sebou?

Zarovnání sekvence a zarovnání

- Příklad:

	Profil 1	Profil 2
A	0,6	
C		0,8
G	0,2	
T		
-	0,2	0,2

- Skóre =

  0,6*0,8*$M(A,C)$ + 0,6*0,2*(-g) +
  0,2*0,8*$M(G,C)$ + 0,2*0,2*(-g) +
  0,2*0,8*(-g) + 0,2*0,2*0

Výběr dvojic pro zarovnání

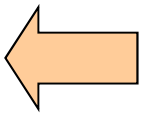
- Všimněte si, že postupným přidáváním sekvencí se výsledné zarovnání neustále rozšiřuje, přičemž mezivýsledek v kroku *n* nelze zpětně upravovat v následujících krocích *n+1*, *n+2*, atd.
- Pořadí výběru dvojic sekvencí proto významně ovlivňuje kvalitu výsledného zarovnání
- Základní pravidlo:
  - Výběr takových dvojic sekvencí, které jsou si nejvíce podobné
- Postup:
  1. sestavení matice podobnosti všech sekvencí
  2. sestavení pomocného stromu (Guided Tree)

Sestavení matice podobnosti

- Postup:
  1. Vypočítej zarovnání všech kombinací dvojic řetězců
  2. Ohodnoť každou dvojici např. podle vzorce:

     Skóre = počet shodných znaků / délka zarovnání
  3. Hodnoty ulož do trojúhelníkové matice
- Příklad:
  - sekvence: GATTCA, GTCTGA, GATATT, GTCAGC

	s_1	s_2	s_3	s_4
s_1	-			
s_2	.57	-		
s_3	.71	.42	-	
s_4	.50	.66	.42	-



- Výpočet zarovnání

s2 **GTC**T**G**A
s4 **GTC**A**G**C (4/6 = 0.66)

s1 **G**A**T**-**T**C**A**
s2 **G**-**T**C**T**G**A** (4/7 = 0.57)

s1 **GAT**-**T**C**A**
s3 **GAT**A**T**-**T** (5/7 = 0.71)

s1 **GAT**T**CA**--
s4 **G**-**T**-**CA**GC (4/8 = 0.5)

s2 **G**-**T**C**T**GA
s3 **G**A**T**A**T**-T (7/3 = 0.42)

s3 **G**A**T**-**A**TT
s4 **G**-**T**C**A**GC (3/7 = 0.42)

Sestavení pomocného stromu (UPGMA)

- Postup:
  - Vyber z tabulky podobnosti sekvenci s_A a s_B s nejvyšší mírou identity
  - Dvojici spoj do jedné sekvence s_{AB} a přepočítej vzdálenosti v tabulce podobnosti podle vztahu: $D_{AB,C} = (D_{A,C}+D_{B,C})/2$
  - Spojení obou sekvencí zakresli do pomocného stromu
  - Opakuj, dokud nejsou spojeny všechny dvojice

- Příklad:
  - Mějme následující tabulku podobnosti

	v_1	v_2	v_3	v_4
v_1	-			
v_2	.57	-		
v_3	.71	.42	-	
v_4	.50	.66	.42	-

  - Sloučíme sekvence v_1 a v_3 a přepočítáme tabulku

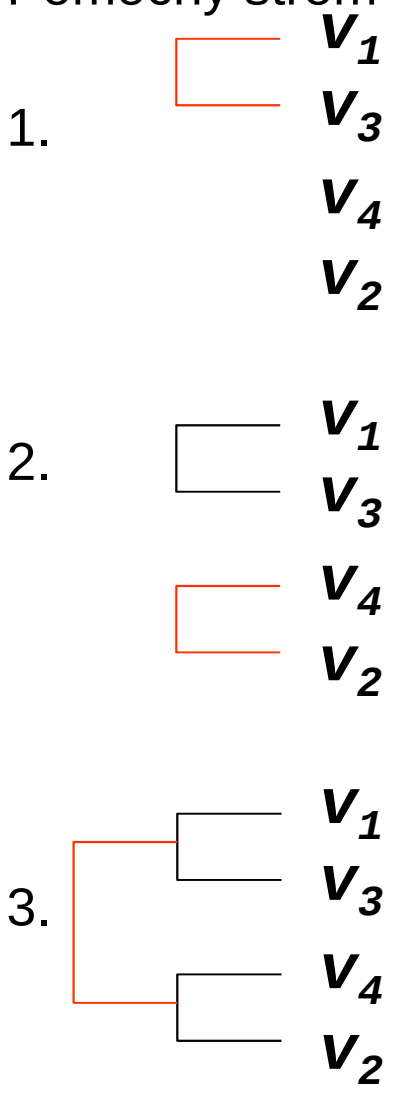
Sestavení pomocného stromu

- Přepočet vzdáleností
  - $D_{13,2} = (D_{1,2} + D_{3,2})/2 = (0.57+0.42)/2 = 0.49$
  - $D_{13,4} = (D_{1,4} + D_{3,4})/2 = (0.50+0.42)/2 = 0.46$
- Upravená tabulka:

	$v_{1,3}$	v_2	v_4
$v_{1,3}$	-		
v_2	.49	-	
v_4	.46	.66	-

- Výsledný postup výběru dvojic:
  1. ***v1,3*** = alignment***(v1, v3)***
  2. ***v2,4*** = alignment***(v2,v4)***
  3. ***v1,2,3,4*** = alignment***((v1,3),(v2,4))***

Pomocný strom



Algoritmus CLUSTAL

- Doposud popsané techniky jsou základem jednoho z nejrozšířenějších algoritmů – CLUSTAL [D. Higgins, P. Sharp, 1988]
- Postup:
  1. Porovnání všech dvojic sekvencí mezi sebou, sestavení tabulky podobnosti
  2. Sestavení pomocného stromu
  3. Na základě pomocného stromu postupné zarovnávání dvojic sekvencí
- Vylepšená (často používaná) verze se nazývá CLUSTALW

CLUSTAL vs. CLUSTALW

- CLUSTAL
  - při sestavení tabulky podobnosti používá princip BLASTu/FASTA (slovo 1-2 znaky pro proteiny, 2-4 pro nukleotidy)
  - při sestavování stromu se používá spojování sekvencí s nejvyšší mírou identity (metoda UPGMA)
  - při výpočtu zarovnání se bere v úvahu zastoupení jednotlivých znaků (profil)
  - jako skórovací matice se používají pouze vybrané verze PAM a BLOSUM matic
- CLUSTALW (Weighted)
  - při sestavení tabulky podobnosti používá dynamické programování
  - při sestavování stromu používá princip Neighbour-Joining (viz. fylogenetické stromy)
  - při výpočtu zarovnání je každá sekvence navíc váhována podle své významnosti (váhovaný profil)
  - skórovací matice se vybírá dynamicky na základě míry identity zarovnávaných sekvencí (BLOSUM-XX)
  - vylepšený způsob penalizace za vložení mezer

Problémy progresivních metod

1. Jsou pomalé, výpočetně nejnáročnější část je porovnání všech dvojic sekvencí a sestavení tabulky podobnosti
   - Příklad: mějme 100 sekvencí délky *N*
     - je nezbytné provést 4950 porovnání dvojic (100 * 99/2)
     - pokud použito DP, potom pro každou dvojici výpočet N^2 položek v tabulce
   - Pro sestavení tabulky podobnosti nepotřebujeme ve skutečnosti znát přesné zarovnání, stačí nám pouze hodnota ukazující míru podobnosti – můžeme použít méně výpočetně náročnou metodu
2. Výsledné zarovnání již nelze změnit a ani nejlepší možný výběr pořadí zarovnávání nemusí vést k optimálnímu výsledku => výsledek je potřeba iterativně zpřesňovat

Zrychlení porovnání dvojic sekvencí

- Princip metody počítání k-tic (k-mer counting)
  1. Obě porovnávané sekvence *X* a *Y* se rozdělí do k-tic použitím posouvajícího se okénka
     - množina všech možných *k*-tic W_k je tvaru:

       $W_k = \{w_1, w_2, \ldots, w_n\}$
     - kde $n = r^k$ je počet všech možných *k*-tic a r je počet znaků abecedy (4 pro nukleotidy, 20 pro aminokyseliny)
  2. Pro každou *k*-tici z W_k se zaznamená počet (frekvence) jejich slova w_i v sekvencích *X* resp. *Y*
     - množina frekvence výskytů je ve tvaru:

       $c_X = \{c_{X1}, c_{X2}, \ldots, c_{Xn}\}$ resp.

       $c_Y = \{c_{Y1}, c_{Y2}, \ldots, c_{Yn}\}$

Zrychlení porovnání dvojic sekvencí

3. Na základě počtu výskytů slov se vypočte vzdálenost obou sekvencí *X* a *Y* např. pomocí Euklidovské vzdálenosti:

$$d(X,Y) = (c_X - c_Y).(c_X - c_Y) = \sum_{i=1}^{n} (c_{Xi} - c_{Yi})^2$$

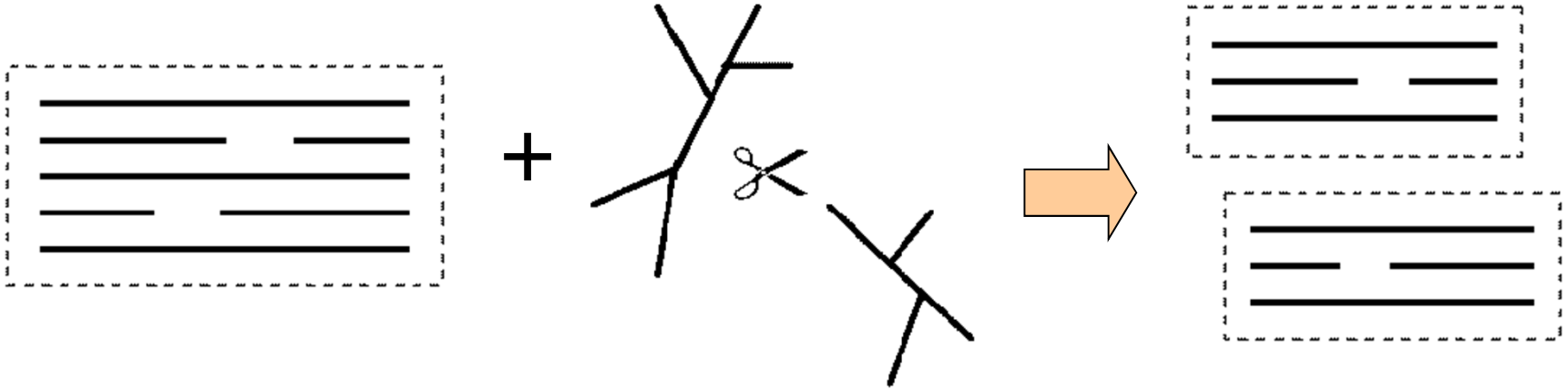
- Poznámky:
  - Je statisticky dokázáno, že čím více jsou si sekvence X a Y podobné, tím více si odpovídají i četnosti výskytů slov
  - Tento jednoduchý princip se začíná používat při prohledávání rozsáhlých databází jako základní filtr
  - Existuje celá řada přístupů pro výpočet vzdálenosti, liší se svou přesností

Zrychlení porovnání dvojic sekvencí

- Příklad:
  - Vstupní sekvence:
    - X = ATATAC
    - Y = ATATAG
  - Množina všech trojic:
    - W_3 = {ATA, TAT, TAC, TAG, ...}
  - Frekvence výskytů jednotlivých slov v X a Y:
    - c_X = {2, 1, 1, 0,...}
    - c_Y = {2, 1, 0, 1, ...}
  - Výpočet Euklidovské vzdálenosti:
    - $d(X,Y) = (2-2)^2+(1-1)^2+(1-0)^2+(0-1)^2 = 2$

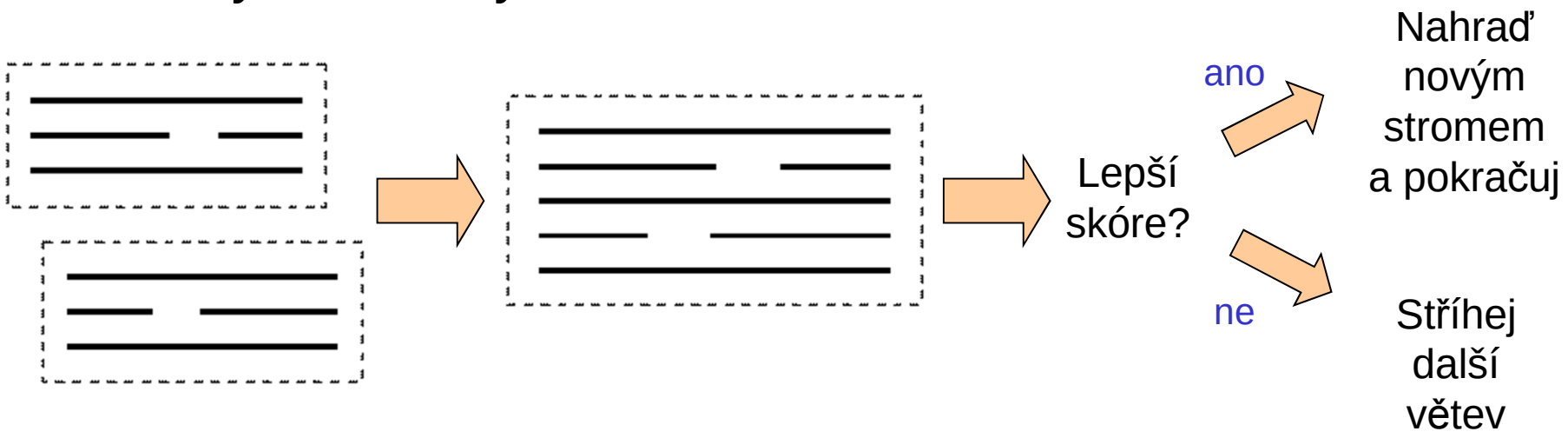
Iterativní zpřesňování zarovnání

- Na vstupu předpokládejme pomocný strom a výsledek více-násobného zarovnání
- Postup:
  1. Strom se rozstřihne na dvě části a pro každou část se odděleně sestaví vícenásobné zarovnání (jako návod se použijí vzniklé podstromy)



Iterativní zpřesňování zarovnání

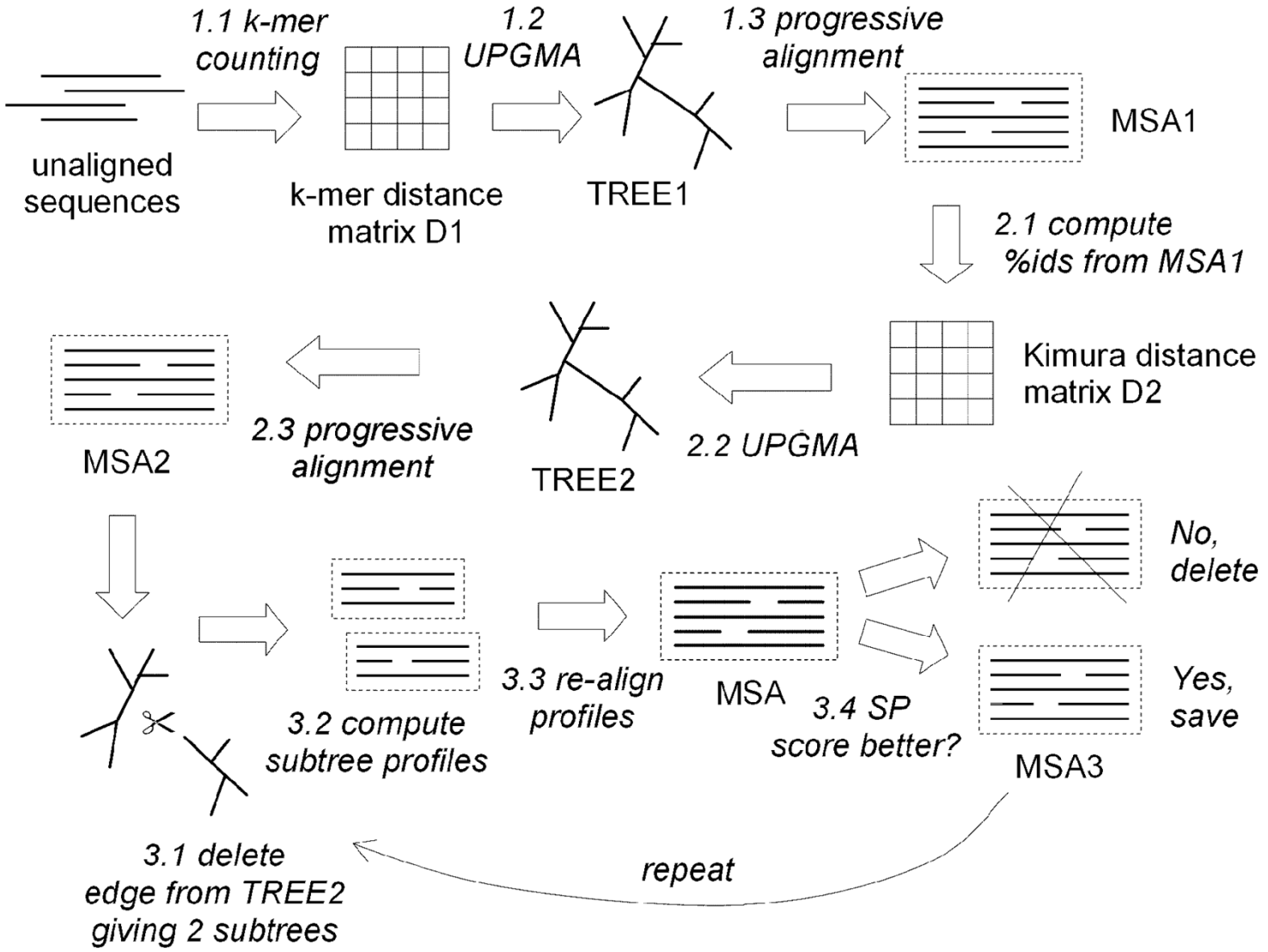
2. Z obou zarovnání se sestrojí výsledné zarovnání a ohodnotí se jeho skóre
3. Pokud bylo dosaženo lepší skóre, původní strom se nahradí novým a zpřesňování pokračuje dokud jsou nalézána lepší skóre nebo již byly vyzkoušený všechny větve. Při volbě větve pro rozstřižení se postupuje systematicky od kořene k uzlům



Metoda MUSCLE

- V současnosti jedna z nejpřesnějších metod [Robert C. Edgar, 2004]
- Princip:
  1. Výpočet předběžného zarovnání
     - Matice podobnosti se sestrojí skrze počítání k-tic
     - Vytvoří se první pomocný strom (TREE1) metodou UPGMA
     - Vytvoří se první zarovnání sekvencí (MSA1)
  2. Přepočet zarovnané sekvence
     - Z MSA1 se vypočte nová matice podobnosti skrze přesnější výpočet (Kimurova vzdálenost)
     - Sestaví se druhý pomocný strom (TREE2) a vytvoří se nové zarovnání (MSA2)
  3. Aplikace iterativního zpřesňování na TREE2 a MSA2 (viz předchozí popis)

Metoda MUSCLE

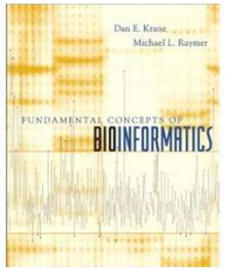


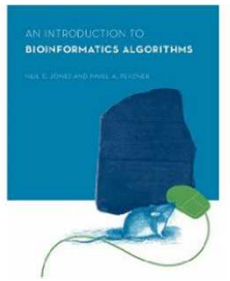
Shrnutí

- Vícenásobné zarovnání nám umožňuje získat daleko více informací o zkoumaných organizmech
- Optimální algoritmus dynamického programování v n-rozměrném prostoru nelze použít pro svou složitost
- Dobré výsledky dávají progresivní metody založené na sestavení pomocného stromu a postupném zarovnání dvojic sekvencí (UPGMA)
- Ani tyto metody však nedávají dostatečně přesné výsledky, pokud nejsou iterativně zpřesňovány např. prostříháváním pomocného stromu

Literatura

- Dan K. Krane, Michael L. Raymer: ***Fundamental Concepts of Bioinformatics***, ISBN: 0-8053-4633-3, Benjamin Cummings 2003.
- Neil C. Jones, Pavel A. Pevzner, ***An Introduction to Bioinformatics Algorithms***, ISBN-10: 0262101068, The MIT Press, 2004





Konec

Děkuji za pozornost