Universitetet i Oslo
Institutt for informatikk

Ittero: multippelt sammenstillings-program for kodende nukleinsyre-sekvenser basert på evolusjonære modeller

Inger Solberg

Hovedfagsoppgave

28. juli 2004

Innhold

1 Innledning **1**
1.1 Oppgavens struktur 3

2 Problemstilling **5**

3 Bakgrunnsstoff for oppgaven **9**
3.1 Parvise sammenstillinger 10
3.2 Multiple sammenstillinger 12
3.3 Ulike programmer og algoritmer/heuristikker for multippel sammenstilling 12
3.3.1 Programmet Clustal 16
3.3.2 Neighbour Joining og andre metoder for å danne et tre . . . 23
3.4 Modeller for evolusjonær forandring 30
3.4.1 Ulike nukleotidemodeller 31
3.4.2 Kodonmodeller 37
3.5 Problemer med den progressive sammenstillingsalgoritmen 41
3.6 BAliScore 42

4 Programmet Ittero **45**
4.1 Beskrivelse av programmet 46
4.2 Programmets struktur/oppbygning 46
4.3 Delen av Ittero som regner ut parameterne 51
4.4 Simulering og parametertolkning 52
4.4.1 Gapstraffer 52
4.4.2 Nukleotide- og kodonfrekvens, π 54
4.4.3 Rate for gjennomsnittlig forandring, μ 54
4.4.4 Transisjons-, transversjonsraten, κ 55
4.4.5 Synonyme, ikke-synonyme raten, ω 55
4.4.6 Parvis likhet 58

5 Datamaterialet **59**
5.1 *mcyABC* 59
5.2 β-globin 60
5.3 Anhydrase 61

6 Resultater **63**
6.1 Valg av gapstraffer 63
6.2 Ulike tellemåter for parvis sammenstilling 65
6.3 Sammenstillingene 69
6.3.1 *mcyABC* 69
6.3.2 β-globin 72
6.3.3 Anhydrase 83

7 Drøftinger og konklusjon **89**
7.1 Ulike sammenstillingsprogrammer og -algoritmer 89
7.2 Generellt om resultatene 90
7.2.1 De to ulike tellemåtene for parvis likhet 91
7.2.2 Valg av gapstraffer 91
7.2.3 Iterasjon 92
7.2.4 Sammenstillingenes lengde 92
7.2.5 BAliScore-verdiene 93
7.2.6 Guidetrær 97
7.2.7 Parametre 97
7.3 Konklusjon 100
7.4 Videre arbeid 101

Bibliografi **103**

A Ordliste **108**

B Genetisk kode **112**

C Resultateksempler **113**
C.1 Clustal 113
C.2 Ittero 115

Forord

Denne hovedoppgaven er skrevet som en del av mitt arbeide for å oppnå Cand.Scient.-graden ved Institutt for informatikk, Universitetet i Oslo.

Jeg vil spesielt takke min veileder Anja Bråthen Kristoffersen for god veiledning og hjelp underveis. Jeg vil også takke Torstein Tengs for ideen om å bruke evolusjonære modeller i multiple sammenstillingsalgoritmer, uten han ville oppgaven sett helt annerledes ut. Desverre ville jobbsituasjon at Torstein Tengs ikke var med under utviklingen av programmet. Jeg vil også takke Kamran Shalchian-Tabrizi for lærerik hjelp og diskusjoner innen biologi, og Bjørg Mikalsen og Kjetil S. Jakobsen for datasettet *mcyABC*. Videre vil jeg takke alle som har lest korrektur og kommet med innspill til oppgaven.

Det forutsettes basiskunnskaper innen biologi i tillegg til informatikk for å kunne få fullt utbytte av oppgaven.

Oslo, 28.07.04
Inger Solberg

Sammendrag

Denne oppgaven ser på multiple sammenstillinger og evolusjonære modeller. Oppgaven er flerdelt ved at jeg først har sett på noen ulike sammenstillingsprogrammer og -algoritmer og noen ulike evolusjonære modeller. Deretter er de evolusjonære modellene implementert i det progressive og iterative programmet Ittero, som er laget i forbindelse med oppgaven. Ittero brukes på kodende nukleinsyresekvenser, og det tillates derfor kun gap av lengde x*kodonlengde. Det vil si at man aldri introduserer gap i et kodon.

Kapittel 1

Innledning

Multippel sammenstilling av biologiske sekvenser har vært en hjørnestein i moderne molekylærbiologi siden teknikken ble innført på 1970-tallet, Lecompte *et al.* (2001). At DNA-sekvenser fra ulike arter ofte er relaterte er et av de viktigste bidragene fra molekylærbiologi ovenfor evolusjonær analyse. Bevarte gener hos ulike arter har ofte lik eller samme funksjon, men de kan også ha mutert og inneha andre funksjoner. Ved å lage sammenstillinger av disse gensekvensene kan man analysere likheter og forskjeller. Ettersom potensialet for å lære om struktur og funksjon i gensekvensene er stort ved bruk av multiple sammenstillinger, har metoder for multippel sammenstilling fått stor oppmerksomhet, Mount (2001).

Innen bioinformatikk er multippel sammenstilling av aminosyresekvenser eller nukleinsyresekvenser den vanligste oppgaven, Notredame (2001). Mange analyser baserer seg på multippel sammenstilling, og disse spenner over et stort område fra databasesøk til prediksjon av sekundær- og tertiærstruktur hos gamle og nye sekvenser. Antall nye sekvenser som oppdages øker stadig, og det er derfor viktig å utvikle effektive og nøyaktige automatiske metoder for multippel sammenstilling, Mount (2001).

> ..there are really only three things that govern the overall accuracy of comparative modelling, alignment quality, alignment quality, and... alignment quality.
>
> — *Jones (1997)*

Multippel sammenstilling ble altså innført tidlig på 1970 tallet da Needleman og Wunsch introduserte den første dynamiske programmeringsalgoritmen for global sammenstilling av to sekvenser, Lecompte *et al.* (2001). Frem til i dag er det utarbeidet mange programmer for multippel sammenstilling av sekvenser. Disse programmene bruker ulike typer algoritmer og parametre for å komme frem til et resultat. De fleste benytter en eller annen form for heuristikker.

Tradisjonelt sett har de mest populære algoritmene vært de progressive som ble introdusert av Needleman og Doolittle i 1987, Lecompte *et al.* (2001). Etterhvert har man funnet andre algoritmer enn dynamisk programmering, og nye evalueringsmetoder har kommet til, Lecompte *et al.* (2001). Blant disse er det en økende bruk av iterative optimeringsstrategier og bruk av funksjoner basert på sammenfallende sekvenser (consistency-based scoring schemes), Notredame (2001).

I oppgaven beskrives kort noen av de multiple sammenstillingsprogrammene som finnes. Problemstillingene er om det er mulig å bruke modeller for evolusjonær forandring i multippel sammenstilling, og om det for multiple sammenstillinger av kodende nukleinsyresekvenser er en fordel å benytte kodonet som enhet. I den forbindelse er programmet Ittero laget. Problemstillingene beskrives mer detaljert i kapittel 2.

Ittero

Ittero er et multippelt sammenstillingsprogram laget i forbindelse med oppgaven, der funksjonen er å sammenstille kodende nukleinsyresekvenser. Kodende nukleinsyresekvenser er posisjonsspesifikke på den måten at tre og tre nukleinsyrer inngår i et kodon, der et kodon koder for en spesiell aminosyre. For disse sekvensene kan man ikke sette inn et gap i et kodon. Hvis dette skjer, får man en leserammemutasjon og sekvensen forandres til å kode for noe annet enn den gjorde i utgangspunktet. Alle tillatte gap blir dermed av lengde x*kodonlengde, altså x*3. Programmer som for eksempel Clustal er ikke spesielt designet for kodende nukleinsyresekvenser og tar derfor ikke hensyn til dette. Programmet Clustal beskrives i avsnitt 3.3.1.
Algoritmen Ittero bruker, er den progressive fremgangsmåten fra Clustal. I tillegg er Ittero gjort iterativt, for å se om det gir en bedre sammenstilling.

Ittero er kodet i programmeringsspråket Java. Java er valgt fordi dette er det språket jeg har best kjennskap til og det er et bra språk da det ikke er lagt vekt på at programmet skal være effektivt med hensyn til tid. Programmet er på ca 3000 linjer med kode og er delt opp i 8 hovedklasser. Kildekoden til programmet er gjort tilgjengelig på min hjemmeside:
http://heim.ifi.uio.no/ingsolbe.

Evolusjonære modeller

Sekvenser skiller seg fra et felles opphav fordi det oppstår mutasjoner. En viss andel av disse mutasjonene forblir i den utviklende populasjonen både ved seleksjon og ved tilfeldigheter, noe som resulterer i at en nukleotide sub-

stitueres av en annen på ulike steder i en sekvens. For å kunne rekonstruere evolusjonære trær, og for å kunne si noe om sekvenslikhet, må man gjøre noen forutsetninger om substitusjonsprosessen og sette opp en modell av disse forutsetningene, Hall (2001)

Modeller for evolusjonær forandring brukes mye innenfor fylogeni. I denne oppgaven brukes noen av disse modellene til å lage multiple sammenstillinger med programmet Ittero. Modellene brukes istedet for en substitusjonsmatrise som er mer normalt innen multippel sammenstilling. De evolusjonære modellene er implemetert i Ittero fordi det i modellene inngår endel biologiske faktorer. De biologiske faktorene inneholder informasjon om sekvensene som man er interessert i å benytte seg av. Vil det gi bedre resultater når man tar hensyn til disse faktorene? Transisjons- transversjonsrate og en rate for gjennomsnittlig forandring, er to av faktorene som inngår i de evolusjonære modellene. Disse faktorene må vurderes før de kan brukes i multippel sammenstilling, noe denne oppgaven har sett på.

1.1 Oppgavens struktur

Følgende kapitteler er med i oppgaven:

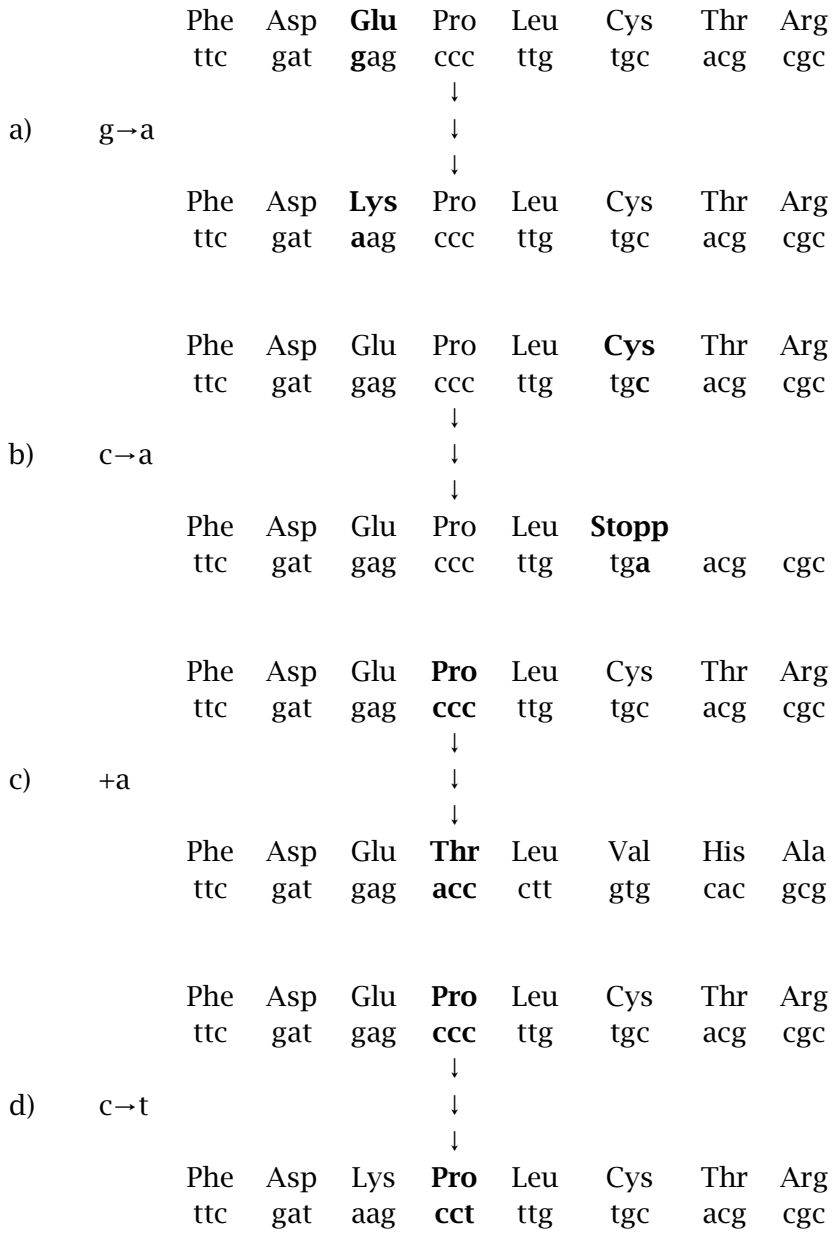
- *Kapittel 2: Problemstilling.* Her presenteres problemstillingen for oppgaven.
- *Kapittel 3: Bakgrunnsstoff for oppgaven.* Dette kapittelet inneholder endel teori og bakgrunnsstoff for temaene i oppgaven. Her presenteres blant annet multippel sammenstilling og evolusjonære modeller.
- *Kapittel 4: Programmet Ittero.* Programmet som er laget i forbindelse med oppgaven beskrives.
- *Kapittel 5: Datamaterialet.* Her presenteres de tre datasettene som er brukt i oppgaven.
- *Kapittel 6: Resultater.* Resultatene fra oppgaven legges frem.
- *Kapittel 7: Drøftinger og konklusjon.* Resultatene diskuteres.

Kapittel 2

Problemstilling

Det finnes mange programmer som lager multiple sammenstillinger av biologiske sekvenser. De fleste programmene er laget for å sammenstille sekvenser på aminosyrenivå og ikke på nukleinsyrenivå, men programmene brukes derimot også på nukleinsyrenivå. For kodende nukleinsyresekvenser er det ikke det samme hvor det introduseres gap i sekvensene siden tre etterfølgende nukleinsyrer koder for en aminosyre, se tabell B.1. Programmer som Clustal tar ikke hensyn til dette og introduserer gap hvor som helst i sekvensene.

Første del av oppgaven er å se kort på hvilke typer sammenstillingsalgoritmer som finnes og ulike programmer som bruker disse. Videre studeres programmet Clustal, som er et av de mest brukte sammenstillingsprogrammer i dag, Ewens & Grant (2001). Neste del av oppgaven ser på problemet med at det ikke er likegyldig hvor gap introduseres i kodende nukleinsyresekvenser. Kan man få bedre resultat ved å tillate kun gap av lengde tre og tre tegn i sekvensene, og kun etter tre og tre tegn? Dette blir forsøkt besvart ved å lage programmet Ittero, hvor denne måten å sette gap på introduseres. Ved å tillate gap som ikke er av lengde tre og tre, eller etter tre og tre tegn, sier man at det har vært en mutasjon som påvirker leserammen til sekvensen. Denne typen mutasjoner vil som oftest føre til store forandringer i sekvensen, og disse mutasjonene vil stort sett ikke overleve. Hvis de overlever gir de opphav til en annen fenotype enn den man opprinnelig hadde, noe som forekommer forholdsvis sjeldent, Winter, Hickey & Fletcher (2002). Figur 2.1 gir eksempler på at sekvenser blir til en annen sekvens enn utgangssekvensen ved denne type mutasjoner. Dette i motsetning til mutasjoner der en nukleinsyre blir skiftet ut med en annen, da denne typen ikke påvirker mer enn et og et kodon. Eksempel på dette vises også i figur 2.1. Figuren er hentet fra Winter, Hickey & Fletcher (2002). En stille mutasjon omtales senere som en synonym substitusjon. En mutasjon der en nukleotide blir skiftet ut med en annen som gjør at kodonet koder for en annen aminosyre kalles ikke-synonym substitusjon.



Figur 2.1: Figuren viser ulike typer mutasjoner. a) er en missense punktmutasjon der en aminosyre går over til en annen. b) viser en nonsense mutasjon der en aminosyre går over til stoppkodon. c) er en leserammemutasjon der resten av sekvensen kommer til å kode for noe annet enn det utgangssekvensen gjorde. d) er en stille mutasjon, overgangen fra c til t fører ikke til noen forandring i aminosyren.

Siste aspekt av oppgaven er å se på evolusjonære modeller. Disse modellene brukes i dag ved fylogenetisk analyse og ikke ved multippel sammenstilling. Modellene tar hensyn til biologiske aspekter ved sammenstillingene, slik som transisjons- transversjonsraten og hvor hyppig de ulike nukleinsyrene forekommer i datamaterialet. Vil det det gi bedre resultater hvis man benytter disse til multippel sammenstilling? Man sier at en sammenstilling med den best mulige verdien etter gitte kriterier er den beste, men det sier ikke alltid noe om det er den beste sammenstillingen biologisk sett, Notredame (1998). Ved å bruke verdier som tar hensyn til biologiske faktorer ved sekvensene når man lager sammenstillinger, kan man kanskje få bedre resultater.

Kapittel 3

Bakgrunnsstoff for oppgaven

I dette kapittelet presenteres endel bakgrunnsstoff. Dette er litt generellt om parvise og multiple sammenstillinger og evolusjonære modeller. Den progressive fremgangsmåten som presenteres for Clustal, benyttes også i Ittero. De evolusjonære modellene som presenteres er de som benyttes i Ittero.

En sammenstilling av to eller flere sekvenser er et resultat av en prosess der man ved hjelp av algoritmer og modeller sier noe om likhet mellom sekvenser.

> An alignment is an hypothesis of positional homology between bases/amino acids.
>
> — *MSA (2002)*

Generelt sammenlikner man sekvenser der utgangspunktet er at sekvensene delvis inneholder likheter, men at en eller flere sekvenser har gjennomgått en evolusjonær forandring, Navarro (2001).
Simultan sammenstilling av mange nukleinsyresekvenser eller aminosyresekvenser er et essensielt redskap innen molekylær biologi, Mount (2001). Sammenstillinger brukes blant annet for å finne diagnostiske mønstre for å karakterisere proteinfamilier, for å oppdage eller vise homologi mellom nye sekvenser og eksisterende familier av sekvenser, og for å forutsi sekundær og tertiær struktur av nye sekvenser. Generellt brukes sammenstilling av sekvenser for å oppdage funksjonell, strukturell og evolusjonær informasjon mellom sekvensene, Notredame (2001).

Formålet med en sekvenssammenstilling er å finne den beste eller mest optimale sammenstillingen av sekvensene etter visse kriterier. Disse kriteriene er blant annet at alle symboler som er i sekvensene må komme i samme rekkefølge i sammenstillingen som i sekvensene. For å lage sammenstillingen kan man innføre et gap, et symbol fra en sekvens sammenstilles med en blank. Det å legge inn et eller flere gap i en sammenstilling vil si at man går ut fra at det

i en eller flere av sekvensene har kommet inn eller forsvunnet ett eller flere tegn gjennom mutasjoner, Giribet & Wheeler (1999). I tillegg kan man også sammenstille to tegn som ikke er like, men som etter visse kriterier er like nok, man sier at det har skjedd en substitusjon på den plassen.

En sammenstilling av sekvenser kan vises ved at man stiller opp sekvensene over hverandre med de sammenstilte tegnene i samme kolonne, se figur 3.1

```
a - t g c t t a a -
a c t g c - - a a c
- c - g c t c - - c
a c c - c t c - a c
- c c a - t c - a c
```

Figur 3.1: Et eksempel på hvordan man kan vise en sammenstilling av fem korte sekvenser ved å sette sekvensene over hverandre slik at de sammenstilte tegnene kommer i samme kolonne.

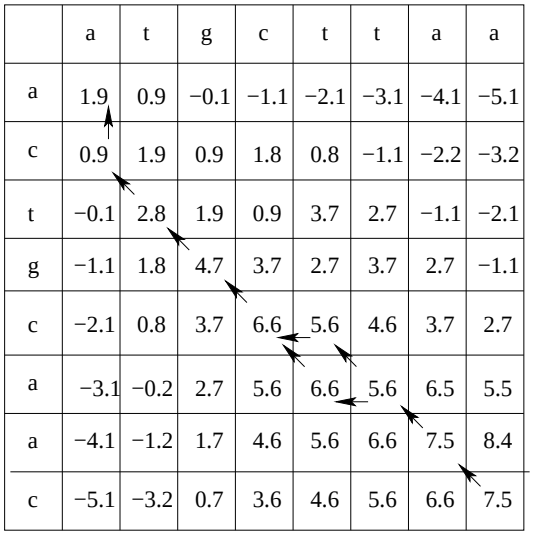
Hovedsakelig skiller man mellom parvis sammenstilling og multippel sammenstilling, disse beskrives under. For både parvis og multippel sammenstilling skiller man mellom global og lokal sammenstilling. Ved global sammenstilling ser man på hele sekvensene, mens ved lokal sammenstilling prøver man å finne likheter i deler av sekvensene.

3.1 Parvise sammenstillinger

En parvis sammenstilling er som ordet sier en sammenstilling av to sekvenser. Parvise sammenstillinger brukes blant annet som første steg i flere algoritmer for multippel sammenstilling. Det mest vanlige er å lage parvise sammenstillinger ved dynamisk programmering, noe som garanterer en matematisk korrekt sammenstilling for en gitt modell, Thompson, Higgins & Gibson (1994*a*). I dynamisk programmering brukes en matrise til å regne seg frem til den parvise sammenstillingen ved å fylle ut matrisen som vist i figur 3.2. For å kunne fylle ut et element $h_{i,j}$ i matrisen må elementene $h_{i-1,j-1}$, $h_{i,j-1}$ og $h_{i-1,j}$ være utfylt. Elementet $h_{i,j}$ fylles ut etter funksjonen

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d \end{cases} \tag{3.1}$$

hvor



	a	t	g	c	t	t	a	a
a	1.9	0.9	−0.1	−1.1	−2.1	−3.1	−4.1	−5.1
c	0.9	1.9	0.9	1.8	0.8	−1.1	−2.2	−3.2
t	−0.1	2.8	1.9	0.9	3.7	2.7	−1.1	−2.1
g	−1.1	1.8	4.7	3.7	2.7	3.7	2.7	−1.1
c	−2.1	0.8	3.7	6.6	5.6	4.6	3.7	2.7
a	−3.1	−0.2	2.7	5.6	6.6	5.6	6.5	5.5
a	−4.1	−1.2	1.7	4.6	5.6	6.6	7.5	8.4
c	−5.1	−3.2	0.7	3.6	4.6	5.6	6.6	7.5

Figur 3.2: Dynamisk programmering for to sekvenser. Man fyller ut en matrise ved først å regne ut den beste verdien for de elementene som går forut for det elementet man nå holder på med. En pil settes tilbake til elementet man hentet verdien fra. Ut fra disse pilene kan man finne tilbake og få den beste sammenstillingen av sekvensene.

- F(i,j) er summen så langt i matrisen av sammenstillingen mellom sekvensene $x_1...x_i$ og $y_1...y_j$
- $s(x, y)$ er kostnaden ved å sammenstille x og y
- d er en enkel gapfunksjon.

En gapfunksjon beregner hvordan et gap i en sekvens straffes. Ved å sette piler for å holde orden på hvilke(n) ruter man brukte til å regne ut verdiene i matrisen, kan man gå tilbake i matrisen og finne de ulike sammenstillingene. I eksemplet i figur 3.2 er bare pilene som har betydning for sammenstillingene tatt med. Følger man pilene tilbake, gir dette eksemplet opphav til to sammenstillinger:

```
a - t g c t t a a        a - t g c t t a a
a c t g c - a a c        a c t g c a - a c
```

Metoden som beskrives her kalles Needleman-Wunsch algoritmen og beregner de(n) optimale global sammenstillingene mellom to sekvenser, Ouyang (2004).

3.2 Multiple sammenstillinger

En multippel sammenstilling er en sammenstilling av mer enn to sekvenser. Ved multippel sammenstilling sammenstilles sekvenser på en optimal måte ved å sette flest mulig like tegn i samme kolonne i sammenstillingen, se figur 3.1. Multiple sammenstillinger kan ikke lages ved hjelp av fullstendig dynamisk programmering for mer enn noen få korte sekvenser, da dette er for tidkrevende. Ved dynamisk programmering øker antall beregninger eksponensielt med antall sekvenser. En sekvens på 300 tegn vil gi:
2 sekvenser: $300^2 = 9 * 10^4$
3 sekvenser: $300^3 = 2.7 * 10^7$ o.s.v.
beregninger. For å lage multiple sammenstillinger benyttes derfor ulike heuristikker.

3.3 Ulike programmer og algoritmer/heuristikker for multippel sammenstilling

De ulike typene algoritmer deles hovedsakelig inn i fire typer, Notredame (2001):

- Eksakte algoritmer
- Progressive algoritmer
- Iterative algoritmer
- Samsvarsbaserte algoritmer (Consistency Based algorithmes)

Hver av algoritmene beskrives kort under, hvor det også blir gitt noen eksempler på programmer som benytter algoritmen.
Skillet mellom iterativ og progressiv algoritme synes ikke å være helt konsis da man i litteraturen enkelte plasser kaller en bestemt algoritme iterativ, mens samme algoritme andre steder blir beskrevet som progressiv.

Eksakte algoritmer

Eksakte algoritmer er basert på Needleman-Wunsch algoritmen, dynamisk programmering. Disse produserer den optimale sammenstillingen ut fra gitte kriterier, Lewis (2003). Problemet med eksakte algoritmer er at tids- og minnekravet til algoritmene vokser eksponensielt som nevnt over, noe som gjør det mulig å benytte de for maksimalt tre sekvenser, Notredame (2001).

Et eksempel på program som bruker en eksakt fremgangsmåte er

- **MSA**
  MSA er et tilnærmet eksakt program, men ikke fullstendig. MSA baserer seg på en korteste vei-algoritme, se Gupta, Kececioglu & Schaffer (1995).

Progressive algoritmer

Progressive algoritmer baserer seg på å lage en multippel sammenstilling ved at man legger til en og en sekvens av gangen. Det vil si at man aldri samtidig sammenstiller mer enn to sekvenser eller to sammenstillinger. Programmer som benytter denne metoden er blant annet Clustal, Thompson, Higgins & Gibson (1994*a*) og DiAlign, Morgenstern *et al.* (1998). Algoritmen til Clustal beskrives i avsnitt 3.3.1.

Eksempel på et annet program som bruker en progressiv fremgangsmåte:

- **PileUp**
  Programmet PileUp er en del av programpakken Wisconsin Sequence Analysis Package, Hickson, Simon &Perrey (2000). PileUp lager en multippel sammenstilling ved progressiv parvis sammenstilling. Algoritmen bruker ikke Neighbour Joining som programmet Clustal, men istedet UPGMA, Unweighted Pair-Group Method with Arithmetic Mean. UPGMA er også en distansemetode, Nei & Kumar (2000). Distansemetoder som Neighbour Joining og UPGMA beskrives i avsnitt 3.3.1. Metoden skiller seg fra Neighbour Joining ved at man starter med de to mest like sekvensene og danner et tre ved å legge til sekvenser, mens Neighbour Joining starter med et stjernetre. Ellers er algoritmen stort sett lik Clustal.

Iterative algoritmer

Iterative algoritmer er basert på ideen om at løsningen til et gitt problem kan beregnes ved å modifisere en allerede eksisterende delløsning. En sammenstilling produseres gjennom en serie av iterasjoner til det ikke er mulig å gjøre flere forbedringer.

Det største problemet med progressive algoritmer er at feil som er gjort i de initielle sammenstillingene aldri forandres, Mount (2001). Iterative algoritmer prøver å gjøre noe med dette ved å re-sammenstille undergrupper av sekvensene og så sammenstille disse delene i en global sammenstilling av alle sekvensene. Poenget er å forbedre en overordnet kostnad, sånn som Sum of Sairs. Se avsnitt 3.6 om Sum of Pairs. Utvelgelsen av disse undergruppene kan baseres på rekkefølgen i et tre, fjerning av en eller to sekvenser, eller en tilfeldig utvelgelse av gruppene.

Eksempler på programmer som bruker en iterativ fremgangsmåte:

- **IterAlign**
  I IterAlign sammenliknes sekvensene lokalt med andre sekvenser og hvert segment som viser stor grad av likhet med andre, byttes ut med sammenfallende sekvenser (an consensus). En runde med denne sammenlikningen er sagt å være første iterasjon. Andre iterasjoner utføres på de sekvensene som er plukket ut helt til samlingen med sammenfallende sekvenser konvergerer. Den ferdige sammenstillingen består av blokker fra samlingen med sammenfallende sekvenser, Notredame (2001).

- **Prrp**
  Prrp algoritmen starter med en initiell multippel sammenstilling. Denne initielle sammenstillingen kan være laget med hvilken som helst annen algoritme. Alle par av sekvenser tillegges så en vekt som avhenger av hvor god sammenstillingen er for disse sekvensene. Vektingen veiledes av guidetreet som er konstruert fra distansene mellom sekvensene i den initielle sammenstillingen. Weighted Sum of pairs (WSP) brukes som et mål på når iterasjonene stoppes. Har WSP konvergert, avsluttes iterasjonene. Verdien for WSP regnes ut ved

$$\text{WSP}=\sum_{i<j} w_{i,j} * S_{i,j} \tag{3.2}$$

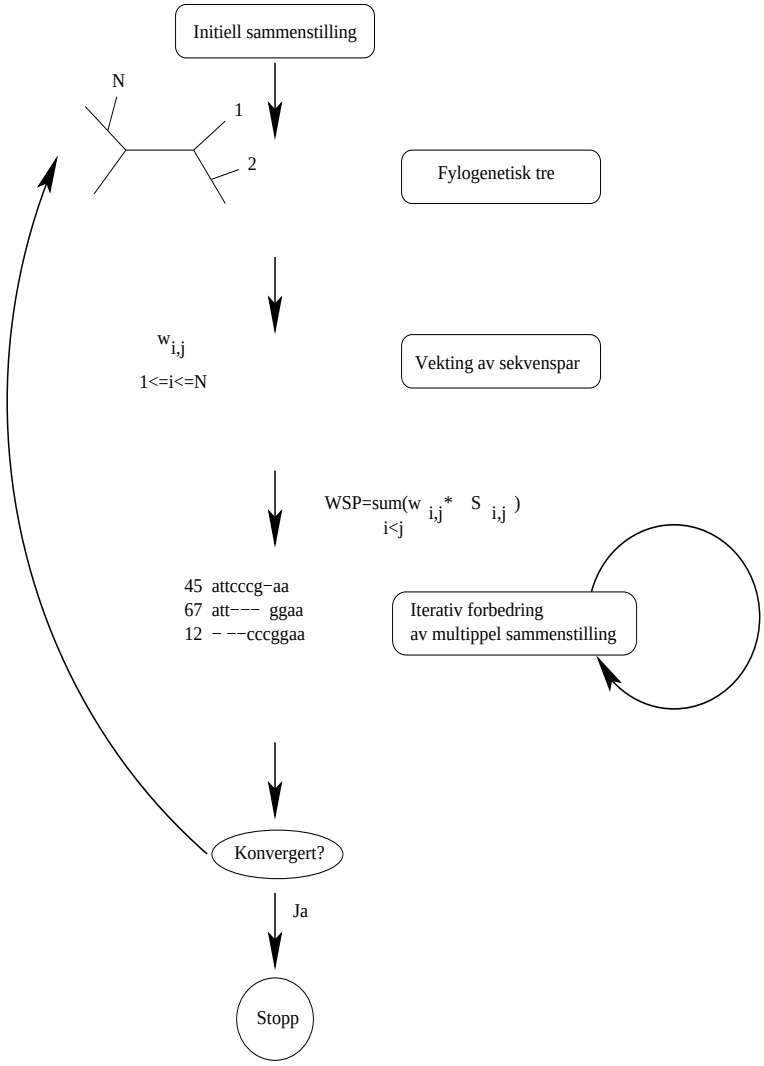
  der $S_{i,j}$ er verdien (score) for sekvensene i og j, og $w_{i,j}$ er deres vekt. Den multiple sammenstillingen forbedres deretter iterativt. Hvis WSP konvergerer, stoppes prosessen. Hvis ikke, beregnes et nytt tre fra den nye multiple sammenstillingen og en ny iterasjon utføres. Figur 3.3 viser fremgangsmåten for programmet. Figuren er hentet fra Gotoh (1996).

Samsvarsbasert (Consistency Based) algoritme

Den optimale sammenstillingen i en samsvarsbasert algoritme er den sammenstillingen som samsvarer best med alle de mulige parvise sammenstillingene. Alle samsvarsbaserte algoritmer er heuristikker da dette er et NP-komplett problem, og de er basert på parvise sammenstillinger. Metoder av denne typen danner en samling av sammenstillinger for hvert par av sekvenser, denne samlingen brukes så til å danne den resulterende multiple sammenstillingen, Lewis (2003).

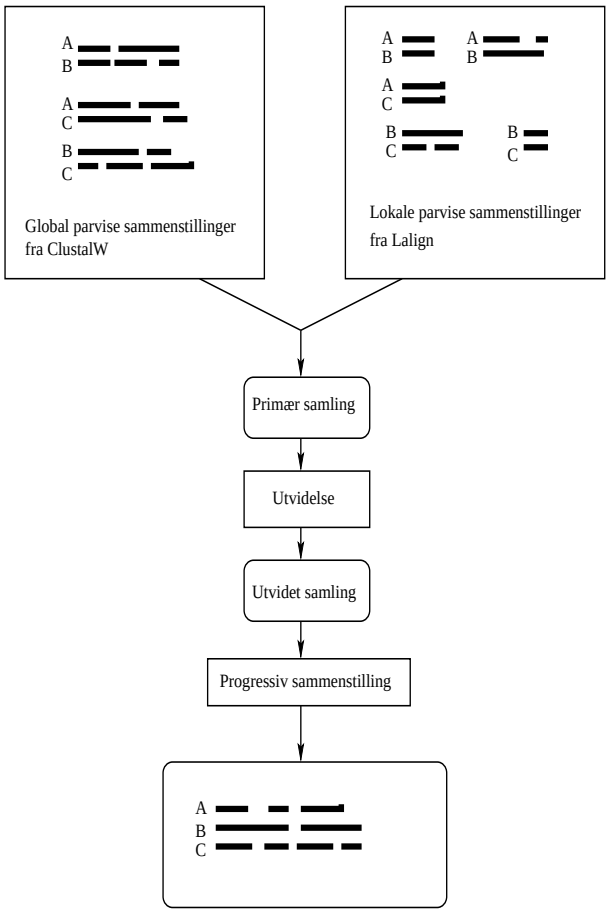
Eksempel på et program som bruker en samsvarsbasert algoritme:

- **T-Coffee** (Tree-based Consistency Objective Function for Alignment Evaluation)
  T-Coffee konstruerer en multippel sammenstilling ved å bruke en samling av parvise sammenstillinger laget av programmene ClustalW og



Figur 3.3: Fremgangsmåten for programmet Prrp. Programmet benytter dobbelt iterasjon.

Lalign. Disse inngår i en primær samling, som igjen forbedres ved å sammenlikne alle sekvensene i samlingen, og man lager en utvidet samling. Sammenstillingene i den utvidete samlingen brukes videre til å lage den multiple sammenstillingen ved en progressiv strategi som ser på informasjon fra alle sekvensene i hvert sammenstillingssteg, ikke bare de som blir sammenstilt i et steg, Notredame, Higgins & Heringa (2000). Fremgangsmåten vises også i figur 3.4.



Figur 3.4: Fremgangsmåten for programmet T-Coffee

3.3.1 Programmet Clustal

Clustal er et av de mest brukte sammenstillingsprogrammer innen molekylærbiologien, Ewens & Grant (2001), og brukes defor her som mal for, og sammenlikningsprogram for programmet Ittero, se kapittel 4. Clustal har gjennomgått mange forandringer siden det første programmet kom for over femten år siden. Programmet slik det er i dag, er stort sett identisk med programmet fra 1992. Da ble det gjort endel forandringer og det fikk navnet ClustalV. I 1994 ble programmet igjen endret, muligheter for å vekte sekvensene ble lagt til, og programmet ble hetende ClustalW. ClustalX er programmet med grafisk brukergrensesnitt, Thompson, Jeanmougin & Gibson (1998).
Clustal bruker en progressiv algoritme. Ordet Clsutal brukes her som en betegnelse på programmene ClustalW og ClustalX da det er de samme algoritmene som benyttes.

Algoritmen

Algoritmen består av tre hovedsteg frem til en multippel sammenstilling, se for eksempel Thompson, Higgins & Gibson (1994*a*), Mount (2001) eller Barton (2001). De tre stegene er:

1. Sammenlikning av alle par av sekvenser der man lager en avstandsmatrise for sekvensene. Et eksempel på en avstandsmatrise er gitt i figur 3.5. Tallene i avstandsmatrisen settes ved at man teller antall like tegn, nukleinsyrer eller aminosyrer, i de parvise sekvensene og får et mål på hvor like de er, eller avstanden mellom de, derav avstandsmatrise.

	Sek.1	Sek.2	Sek.3	Sek.4	Sek.5
Sekvens1	-				
Sekvens2	0.17	-			
Sekvens3	0.59	0.60	-		
Sekvens4	0.59	0.59	0.13	-	
Sekvens5	0.79	0.77	0.75	0.74	-

Figur 3.5: Eksempel på en avstandsmatrise for fem sekvenser.

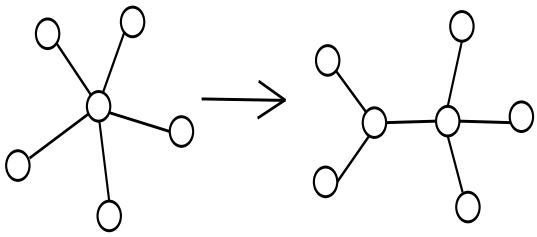
   I det første steget brukes dynamisk programmering til å sammenlikne alle par av sekvenser. I sammenlikningen brukes det to gapstraffer, en for å åpne et gap og en for å utvide et allerede eksisterende gap. I tillegg brukes en fullstendig aminosyresubstitusjonsmatrise eller nukleinsyresubstitusjonsmatrise. En substitusjonsmatrise gir vekt til alle symboler som inngår i datamaterialet. Disse vektene brukes til å bestemme om man skal sammenstille to symboler eller om man skal sette inn et gap. Et eksempel på en substitusjonsmatrise for nukleinsyresekvenser er vist i figur 3.6.

2. Generering av et guidetre ut fra avstandsmatrisen funnet i punkt 1.

   Her produseres et guidetre ut fra avstandsmatrisen som lages i punkt 1. Guidetreet genereres ved hjelp av Neighbour Joining-algoritmen som beskrives i avsnitt 3.3.2, se også figur 3.7.

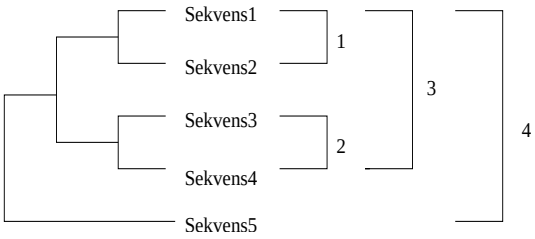
	a	c	g	t
a	1.9	0	0	0
c	0	1.9	0	0
g	0	0	1.9	0
t	0	0	0	1.9

Figur 3.6: Figuren viser et eksempel på en substitusjonsmatrise for nukleinsyresekvenser.



Figur 3.7: Neighbour Joining-algortimen starter med et stjernetre og skiller ut de mest like sekvensene i egne noder helt til det ikke er flere igjen å skille ut.

Algoritmen starter med et stjernetre som er en samling av alle nodene som representerer sekvensene og bygger ut fra dette opp et ikke-rotet fylogenetisk tre, se figur 3.8.



Figur 3.8: Neighbour Joining-tre (guidetre) som lages ved hjelp av avstandsmatrise. På høyre side vises rekkefølgen de multiple sammenstillingene dannes, dette beskrives i trinn 3.

3. Multippel sammenstilling av alle sekvensene

   Det siste trinnet i algoritmen er å danne en multippel sammenstilling av alle sekvensene i datasettet. Dette gjøres ved progressiv sammenstilling. Hvert steg utføres ved dynamisk programmering og består i å sammenstille to sekvenser eller eksisterende sammenstillinger. Rekkefølgen sammenstillingen følger bestemmes av guidetreet generert i trinn 2, se figur 3.8. Det vil si at de to mest relaterte sekvensene sammenstilles først, deretter sammenstiller man to nye sekvenser, eller en sekvens med den sammenstillingen man allerede har. Allerede introduserte gap forandres ikke ved senere sammenstillinger. I basisalgoritmen straffes hvert gap som settes inn ved hvert steg med full åpnings- og utvidelsesstraff, selv om de settes inn i en posisjon der det allerede eksisterer gap. For å beregne kostnad (score) mellom en posisjon i en sekvens, eller en sammenstilling og en annen, brukes gjennomsnittet av alle de parvise substitusjonsmatrisekostnadene i de to settene av sekvenser. For en sammenstilling av to nukleotideblokker som allerede er sammenstilt, regnes kostnad ut som en veiet sum av kryssleddene. Dette er vist under.

   Hvis kolonnene: a a g og t c

   skal sammenliknes så vil man få:

   $$\text{kostnad}=\frac{s(a,t)+s(a,c)+s(a,t)+s(a,c)+s(g,t)+s(g,c)}{6} \tag{3.3}$$

   der $s(x,y)$ er verdien fra substitusjonsmatrisen for x og y.

Parameterene i programmet

Parameterene i programmet kan hovedsakelig deles i tre kategorier:

- Parametre som setter gapstraffer

  To straffer settes for å angi hvor mye det koster å sette inn et gap i en av sekvensene. Den første angir hvor mye det koster å åpne et gap, Gap Open Penalty (gapOpen), den andre angir hvor mye det koster å utvide et allerede eksisterende gap, Gap Extension Penalty (gapExtend).

- Substitusjonsmatrise

  For å kunne sette en verdi for tegnene man sammenstiller trengs en substitusjonsmatrise. Substitusjonsmatriser gir et tall på hvor like to tegn er.

- Andre parametre

  For eksempel DNA transisjonsvekt, og mulighet for å utsette å legge til sekvenser som har en likhet under en viss prosent.

De to delene av Clustal, parvis sammenstilling og multippel sammenstilling har hver sine sett med parametre som kan settes uavhengig av hverandre.

Gapstraffer

De to gapstraffene, gapOpen og gapExtend, settes som to tall mellom null og hundre.

Gapstraffer ved multippel sammenstilling

Initielt brukes to gapstraffer, en for å åpne et gap, gapOpen, og en for å utvide et allerede eksisterende gap, gapExtend. Initielle straffer kan i Clustal settes av brukeren, programmet prøver deretter automatisk å velge den passende gapstraffen for hver sekvenssammenstilling, avhengig av følgende faktorer:

- Substitusjonsmatrisen

  Det er vist at ved å variere gapstraffene brukt med forskjellige substitusjonsmatriser kan nøyaktigheten ved sekvenssammenstilling forbedres. Her brukes den gjennomsnittlige kostnaden for to ikke-like tegn som en skaleringsfaktor for gapOpen, Thompson, Higgins & Gibson (1994*a*).

- Likheten til sekvensene

  Den prosentvise identiteten til de to (gruppene av) sekvenser som skal sammenstilles, brukes for å øke gapOpen for nært relaterte sekvenser, og senke den for mer ulike sekvenser.

- Sekvensenes lengde

  Ettersom skåringen til en sammenstilling er avhengig av lengden på sekvensene, modifiseres gapOpen i forhold til dette. Clustal bruker logaritmen av lengden til den korteste sekvensen for å øke gapOpen med sekvenslengden.

- Forskjellen i lengde mellom de to sekvensene

  GapExtend modifiseres avhengig av forskjellen mellom lengdene av de to sekvensene som skal sammenstilles. Hvis en sekvens er mye kortere

enn den andre, så økes gapExtend for å forhindre for mange lange gap i den korteste sekvensen.

Posisjonsspesifikke gapstraffer

I dynamiske programmeringsapplikasjoner behandles vanligvis de initielle gapOpen og gapExtend likt uansett hvor gapene er lokalisert. Unntaket er gap i begynnelsen og slutten av sekvensene, disse tillates som oftest uten straff. I Clustal konstruseres det en tabell med gapOpen for hver posisjon i de to settene med sekvenser, eller de pre-sammenstilte gruppene av sekvenser før man gjør noen sammenstillinger i det hele tatt. Den initielle gapOpen modifiseres på en posisjonsspesifikk måte for å gjøre gap mindre sannsynlig i forskjellige posisjoner, for eksempel nært eksisterende gap.

Reglene for gapstraffmodifikasjoner brukes på en hierarkisk måte. Detaljene for hver regel beskrives under. Først, hvis det er et gap i en posisjon, så senkes gapOpen og gapExtend, og ingen av de andre reglene gjelder. Dette gjør at gap er mer sannsynlig i posisjoner der det allerede finnes et gap. Hvis det ikke finnes et gap i en posisjon, så økes gapOpen hvis posisjonen er innen åtte tegn borte fra et eksisterende gap. Dette hindrer at gap kommer for tett på hverandre.

Beskrivelse av reglene:

- Lavere gapstraff ved allerede eksisterende gap:

  Hvis det allerede eksisterer et gap i en posisjon, så reduseres gapOpen i forhold til hvor mange sekvenser det er i denne posisjonen som inneholder et gap, og gapExtend halveres. Den nye gapOpen beregnes ved:

$$\text{gapOpen} \rightarrow \text{gapOpen} * 0.3 * \left(\frac{\text{antall sekvenser uten gap}}{\text{antall sekvenser}}\right)$$

- Økt gapstraff nær eksisterende gap:

  Hvis en posisjon ikke har noen gap, men posisjonen er innen åtte tegn fra et gap, så økes gapOpen ved:

$$\text{gapOpen} \rightarrow \text{gapOpen} * (2 + ((8\text{-distanse fra gap}) * 2)/8)$$

Substitusjonsmatriser

Endel substitusjonsmatriser er innebygget i programmet, de ulike valgene for aminosyresekvenser er:

- BLOSSUM series
- PAM series
- Gonnet series
- Identity matrix

For nukleinsyresekvenser er valgene:

- IUB
  Denne benyttes i Ittero. IUB-substitusjonsmatrisen er en enkel matrise der alle like IUB-symboler vektes med 1.9, og alle ulike med 0.0, Thompson, Higgins & Gibson (1994*a*). IUB-symbolene vises i tabell 3.1. Hvis det

Symbol	Oversettelse	Nukleinsyre
a	a	Adenin
c	c	Cytosin
g	g	Guanin
t	t	Tymin
m	a eller c	
r	a eller g	
w	a eller t	
s	c eller g	
y	c eller t	
k	g eller t	
v	a eller c eller g	
h	a eller c eller t	
d	a eller g eller t	
b	c eller g eller t	
x	g eller a eller t eller c	
n	g eller a eller t eller c	

Tabell 3.1: Figuren viser alle IUBsymbolene og hva de ulike tegnene betyr.

  ikke er andre tegn i sekvensene enn a, g, c og t blir substitusjonsmatrisen her som vist i figur 3.6.

- ClustalW(1.6)
  Her vektes alle like symboler med 1.0 og ulike symboler med 0.

Man kan også laste inn egne substitusjonsmatriser.

Andre parametre

For parvis sammenstilling kan man velge om sammenstillingene skal skje ved full dynamisk programmering som er en nøyaktig, men ikke veldig rask metode, eller om man skal bruke en rask metode som ikke gir et nøyaktig svar.

For den multiple sammenstillingen kommer i tillegg parameterne:

- Mulighet til å utsette å legge til mer distante sekvenser til senere i sammenstillingen. Dette angis i prosent ulikhet over sekvensene.
- DNA transisjonsvekt.
  Her kan man ta inn informasjon om transisjoner og transversjoner.

3.3.2 Neighbour Joining og andre metoder for å danne et tre

Det andre steget i den progressive algoritmen hos blant annet Clustal, er en metode for å danne et tre. Tre metoder presenteres kort her:

1. Maximum Parsimony
2. Maximum Likelihood
3. Distansemetoder

De to første metodene skiller seg fra den siste ved at de ser på alle mulige trær ut fra det gitte datasettet, og velger tre eller trær ut fra gitte kriterier. Distansemetoder derimot danner ett tre.

1. Maximum Parsimony
   Dette er en metode som finner det evolusjonære treet, eller trærne, som minimerer antall steg som trengs for å komme frem til den observerte variasjonen i sekvensene. Denne metoden refereres derfor ofte til som "minimum evolusjonsmetode". Metoden passer best for sekvenser som er forholdsvis like, og for et lite antall sekvenser. Algoritmen som følges er ikke særlig komplisert og garanterer å finne det beste treet for de gitte betingelsene fordi alle trær som kan gi den gruppen av sekvenser som studeres også vurderes. Metoden er derfor veldig tidkrevende da alle de ikke-rotete trærne vurderes i analysen. Sekvensvariasjonen på hver plass i sammenstillingen plasseres i de eksterne nodene av treet, og det treet som krever færrest forandringer for å forklare variasjonen velges ut. Denne analysen gjentas for hver informative posisjon (informative site), og treet eller trærne med færrest forandringer totalt blir funnet.
   En posisjon er informativ når det er minst to forskjellige nukleotider i posisjonen, og i tillegg må det være minst to sekvenser som har hver av de to ulike nukleotidene, Mount (2001). Et eksempel på dette vises i figur

3.9. I figuren ser man at i posisjon 3, 5 og 7 er det to ulike nukleotider, og det er to av hver av nukleotidene. Ved Maximum Parsimony ønsker man

Sekvensnummer	Posisjon i sekvensene og nukleotide									
	1	2	3	4	5	6	7	8	9	10
1	a	a	c	g	t	t	c	a	g	g
2	a	g	c	t	c	a	c	a	g	g
3	a	g	t	t	t	a	g	a	g	a
4	a	t	t	t	c	a	g	g	c	c

Figur 3.9: Figuren viser de informative posisjonene i noen korte sekvenser. Posisjonene 3, 5 og 7 er informative da det er minst to av sekvensene som har ulike tegn i disse posisjonene.

altså å finne det treet med færrest mulig forandringer ut fra de dataene man har. Generelt opererer Parsimony-metoder for fylogeni ved å velge trær som minimerer den totale trelengden, Swofford *et al.* (1996):

> antall evolusjonære steg-transformasjoner fra et tegn til et annet som trengs for å forklare et gitt sett data.

2. Maximum Likelighood Likelihood
sannsynlighetsberegning for å finne det treet som best samsvarer med variasjonene i et sett av sekvenser. Metoden er lik Maximum Parsimony på den måten at analysen utføres på hver kolonne i en multippel sammenstilling. Alle mulige trær vurderes, derfor egner metoden seg best for et lite antall sekvenser. For hvert tre vurderes antall forandringer som kan ha foregått for å gi den observerte variasjonen over sekvensene. Fordi mutasjonsraten er veldig lav, vil et tre som krever mange mutasjoner for å forklare de gitte sekvensene, være mindre sannsynlig enn et tre med få mutasjoner, Mount (2001). For å finne treet eller trærne med høyest sannsynlighet, brukes en gitt sannsynlighetsmodell for evolusjonær forandring. Evolusjonære modeller beskrives i avsnitt 3.4. Her tas endel forutsetninger, blant annet at de ulike karakterene forandres uavhengig av hverandre, og at de ulike artene forandrer seg uavhengig av hverandre. Det at det brukes en modell for evolusjonær forandring, og ikke som i Maximum Parsimony der man ser på antall forandringer, gjør at metoden kan brukes til å utforske relasjoner blant mer ulike sekvenser, noe Maximum Parsimony ikke takler like bra. Den største ulempen med Maximum Likelihood-metoder er at de krever store ressurser til beregning av treet. Algoritmen er som følger, Mount (2001):

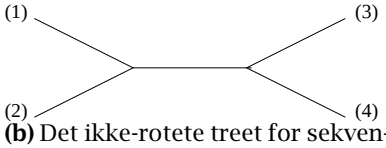
> For enhver posisjon i en multippel sammenstilling beregnes sannsynligheten for ethvert tre mulig for denne posisjonen. For

å finne sannsynligheten for hele treet, beregnes produktet av sannsynligheten til alle posisjonene.

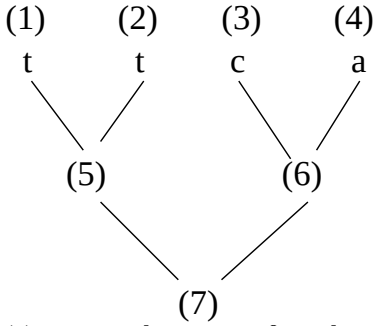
se også figur 3.10.

	1			j						N
(1)	a	a	g	t	c	c	g	a	a	c
(2)	g	g	c	t	g	g	g	t	a	c
(3)	g	t	a	c	a	t	c	t	a	a
(4)	a	t	a	a	t	t	c	g	a	a

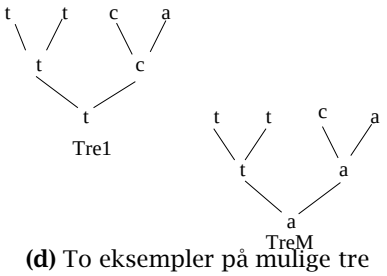
(a) Del av sammenstilling



(b) Det ikke-rotete treet for sekvensene



(c) Et generelt rotet tre for sekvensene



(d) To eksempler på mulige tre

$$L_j = \text{Prob}(Tre1) + \text{Prob}(Tre2) + ... + \text{Prob}(TreM)$$

(e) Utregning av sannsynligheten for en posisjon i sammenstillingen

$$L = L_{(1)} * L_{(2)} * .... * L_{(N)} = \Pi_{j=1}^{N} L_{(j)}$$

(f) Sannsynligheten til hele treet er produktet av sannsynlighetene til alle de mulige trærne for de N ulike posisjonene

$$\ln L = \ln L_{(1)} + \ln L_{(2)} + ... + \ln L_{(N)} = \sum_{j=1}^{N} \ln L_{(j)}$$

(g) Log Likelihood

Figur 3.10: Figuren viser fremgangsmåten for Maximum Likelihood-metoden

3. Distansemetoder
   Distansemetoder bruker antall forandringer mellom hvert par i en gruppe av sekvenser for å produsere et fylogenetisk tre. De sekvensparene som har færrest antall forandringer, kalles naboer. I et tre deler disse sekvensene en felles opphavsnode. Målet for distansemetoder er å identifisere et tre som plasserer naboene korrekt, og som har grenlengder som reproduserer de originale dataene så godt som mulig, Mount (2001).

   Fordelene ved denne metoden er at den er rask, ettersom den i motsetning til Maximum Likelihood og Maximum Parsimony ikke undersøker alle trær, men danner et tre. En annet fortrinn er at man ikke trenger å ha en multipell sammenstilling på forhånd. Problemet med denne metoden er at den ikke garanterer å finne det beste treet, ettersom man aldri undersøker alle mulige trær. Ved å bruke distansemetoder får man altså et tre raskt, mens ved å bruke en av de andre metodene ville man kanskje ha fått et bedre guidetre og et bedre resultat.

   Eksempler på distansemetoder er Neighbour Joining-algoritmen som beskrives i neste avsnitt og Unweighted Pair-Group Method with Arithmetic Mean (UPGMA), som ikke beskrives her, men blant annet i Hall (2001).

Neighbour Joining-algortimen

Neighbour Joining er en distansemetode for å konstruere fylogenier, Hall (2001). Algoritmen for Neighbour Joining beskrives flere steder, denne beskrivelsen følger Swofford *et al.* (1996). Utgangsdataene for Neighbour Joining er en avstandsmatrise, og det initielle treet er et stjernetre. Fra det initielle treet lages et ikke-rotet tre. Dette ikke-rotete treet får satt en rot ved algoritmen beskrevet under. Kort forklart regnes det ut hvilke av sekvensene som er mest "like", disse fjernes fra den avstandsmatrisen man hadde, og man regner ut nye avstander fra de gjenværende sekvensene til denne noden. Dette gjøres helt til man har tatt alle sekvensene.

Stegene i algoritmen er som følger:

- Gitt en matrise med parvise distanser D. For hver av sekvensene/-bladnodene i vil man beregne dens "avstand" r_i fra alle de andre ved å bruke formelen:
  $$r_i = \sum_{k=1}^{N} D_{ik}$$
  hvor N er antall sekvenser/bladnoder i matrisen. En antakelse er at $D_{ii} = 0$.

- Lag en "rate-corrected" avstandsmatrise M der elementene er definert av:
$$M_{ij} = D_{ij} - (r_i + r_j)/(N-2)$$
for alle i og med $j > i$.
- Definer en ny node u som binder sammen nodene i, j og resten av treet. Definer lengden av grenene fra u til i og j som:
$$v_{iu} = D_{ij}/2 + (r_i - r_j)/[2(N-2)]$$
$$v_{ju} = D_{ij} - v_{iu}$$
- Definer avstanden fra u til hver av de andre terminale nodene (for alle $k \neq$ i eller j) ved:
$$D_{ku} = (D_{ik} + D_{jk} - D_{ij})/2$$
- Fjern avstandene fra nodene i og j fra matrisen og senk N med en.
- Hvis det er mer enn to noder igjen, så gå til steg 1. Hvis ikke, så er treet nå fullstendig definert bortsett fra lengden på grenen som forbinder de to gjenværende nodene i og j. La denne lengden være:
$$v_{ij} = D_{ij}$$
Hvert steg har generert en intern node og har estimert lengden til to av grenene som er forbundet til den noden. Treet kan nå tegnes fra disse dataene.

En "midtpunkt"-metode brukes til å sette rota. Denne metoden beskrives under.

Algoritme for å sette rot i treet fra Neighbour Joining-algoritmen

Denne algoritmen beskrives i Thompson, Higgins & Gibson (1994*b*) og brukes av Clustal for å sette rot på treet som lages av Neighbour Joining-algoritmen. De ulike stegene er:

- La P være ethvert punkt i treet sånn at P deler treet i to deler, høyre og venstre subtre. La de n_l sekvensene på venstre side av P være gitt ved $L_1, L_2, ..., L_{n_l}$ og la de n_r sekvensene på høyre side tilsvarende være gitt ved $R_1, R_2, ..., R_{n_r}$.
- For en sekvens S_l, la avstanden fra sekvensen til P være gitt ved d_{p,S_l}.

- Forskjellen Δp mellom gjennomsnittlig grenlengde på hver side av P er definert ved:

$$\Delta p = \frac{\sum_{i=1}^{n_l} d_{p,L_i}}{n_l} - \frac{\sum_{i=1}^{n_r} d_{p,R_i}}{n_r}$$

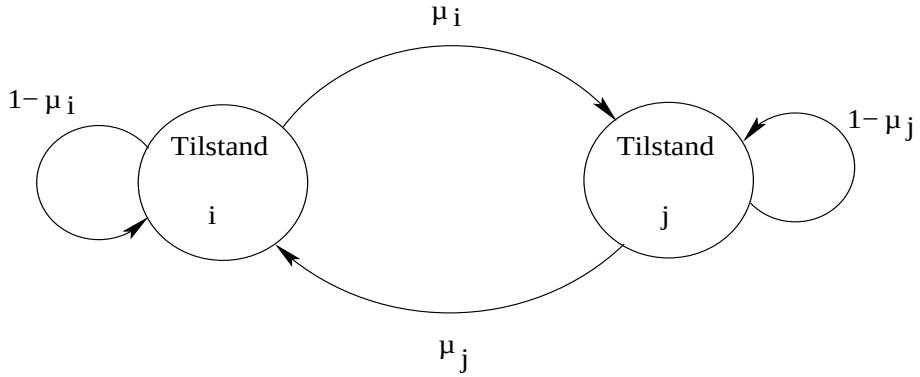
  En rot for treet er definert som ethvert punkt der $\Delta p = 0$
- For å finne alle røtter til et tre, finner man først alle noder som har en positiv Δp hvor foreldrenoden har en negativ Δp, noe som viser at et sted på greina mellom disse to nodene er Δp null.
- Fra settet av alle mulige røtter velges den som minimerer den lengste avstanden fra enhver sekvens til rota.

3.4 Modeller for evolusjonær forandring

En modell for evolusjonær forandring , også kalt en substitusjonsmodell, forteller hvordan man tenker seg at evolusjonen har foregått. Modellene viser hvordan man går ut fra at sekvensene har forandret seg ved mutasjon, og de ulike forandringene kan dermed gis verdier. For å kunne se på relasjoner mellom sekvenser er det en forutsetning at alle sekvensene stammer fra en opphavssekvens. Ettersom man ikke vet hvordan evolusjonen har foregått, er det ikke mulig å sette opp en korrekt modell, og endel antagelser gjøres. Disse antagelsene sier noe om hvor fort man antar at et tegn forandres, substitusjonsraten, og hvilke overganger mellom tegn som er tillatt, eller hvilke som er mest sannsynlige. Resultater fra evolusjonær analyse avhenger sterkt av hvor korrekt den evolusjonære modellen, som er brukt som basis for estimeringen, er, Cavalli-Sforza & Edwards (1967). Dette gjelder spesielt ved estimering med dataprogrammer der den evolusjonære modellen, og andre parametre slik som gapstraffer, utgjør den biologiske basisen.

> Mathematical models of nucleotide substitution are important for the estimation of phylogentic trees and for understanding the evolution of DNA sequences. (...) Better models should lead to more accurate estimates of the evolutionary history of the species concerned and to a better understanding of the forces that affected the evolution of the sequences.
>
> — *Goldman & Yang (1994b)*

Evolusjonære modeller presenteres ofte som en Markov-modell hvor sannsynligheten til en forandring fra tilstand *i* til tilstand *j* er uavhengig av hva som har skjedd før tilstand *i*. En generell Markov-modell for to tilstander vises i figur 3.11. Matematisk uttrykkes en evolusjonær modell som en matrise med



Figur 3.11: Generell Markov-modell for to tilstander *i* og *j*

substitusjonsrater per posisjon per enhet evolusjonær avstand. Modeller for evolusjon hos proteinkodende sekvenser baserer seg enten på mononukleotisk nivå i nukleinsyresekvenser eller på aminosyrenivå i aminosyresekvenser, Goldman & Yang (1994*a*). Mononukleotisk nivå for nukleinsyresekvenser vil si at man ser på en og en nukleotide, og at disse er uavhengige av hverandre. Uavhengigheten gjelder også på aminosyrenivå.

Statistiske tester av nøyaktigheten til ofte brukte modeller indikerer at enkle modeller ikke er gode nok, Goldman & Yang (1994*b*). Modeller som tar hensyn til biologiske faktorer, sånn som for eksempel transisjons- transversjonsraten er ofte bedre modeller.

3.4.1 Ulike nukleotidemodeller

Den enkleste modellen man kan se på er en modell som sier at sannsynlighetene for at en nukleotide kan gå over til hvilken som helst annen nukleotide er den samme. For å beregne sannsynligheten for at en spesiell nukleotide vil forandres til en annen i et gitt tidsintervall, trenger man bare å vite nukleotidesubstitusjonsraten. Altså hvor hvor ofte, eller hvor fort en spesiell nukleotide forandres, Hall (2001). Denne enkleste modellen har da bare en parameter, nemlig substitusjonsraten. Metoden kalles Jukes Cantor, og beskrives senere.

For evolusjonære modeller brukes som sagt ofte en Markov-modell hvor sannsynligheten til en forandring fra tilstand i til en tilstand j ikke er avhengig av hva som har skjedd før tilstand i. Matematisk utrykkes en substitusjonsmodell som en matrise med substitusjonsrater. For nukleinsyresekvenser kan det gis i en 4*4 matrise Q, hvor hvert element Q_{ij} representerer raten for forandring fra nukleotide i til nukleotide j. Den mest generelle formen for denne matrisen er:

$$Q = \begin{pmatrix} -\mu(a\pi_C + b\pi_G + c\pi_T) & \mu a\pi_C & \mu b\pi_G & \mu c\pi_T \\ \mu g\pi_A & -\mu(g\pi_A + d\pi_G + e\pi_T) & \mu d\pi_G & \mu e\pi_T \\ \mu h\pi_A & \mu j\pi_C & -\mu(h\pi_A j\pi_C + f\pi_T) & \mu f\pi_T \\ \mu i\pi_A & \mu k\pi_C & \mu l\pi_G & -\mu(i\pi_A + k\pi_C + l\pi_G) \end{pmatrix} \tag{3.4}$$

hvor radene og kolonnene korresponderer til basene A, C, G og T, og hvor

- μ er den gjennomsnittlige substitusjonsraten for nukleotidene
- π_a, π_g, π_c og π_t frekvensparameterne for nukleotidene
- a, b, c,....,l relative rateparameter som tilhører hver av overgangene fra en nukleotide til en annen

Sannsynligheten for overgangene finnes ved $P_{ij}(t) = e^{Q_{ij}t}$. Hvordan sannsynligheten beregnes, beskrives senere.
Ofte dekomponeres **Q** i to matriser **R** og Π, hvor

$$\mathbf{R} = \begin{pmatrix} - & \mu a & \mu b & \mu c \\ \mu g & - & \mu d & \mu e \\ \mu h & \mu j & - & \mu f \\ \mu i & \mu k & \mu l & - \end{pmatrix} \tag{3.5}$$

og

$$\Pi = \begin{pmatrix} \pi_a & 0 & 0 & 0 \\ 0 & \pi_c & 0 & 0 \\ 0 & 0 & \pi_g & 0 \\ 0 & 0 & 0 & \pi_t \end{pmatrix} \tag{3.6}$$

Elementene utenom diagonalen til Q er lik elementene utenom diagonalen til matriseproduktet RΠ. Elementene på diagonalen til Q er den negtive summen til elementene i samme rad.

Nesten alle nukleinsyresubstitusjonsmodeller som er foreslått er en utgave av matrisen 3.4. Det er vanlig å anta at substitusjonsraten fra nukleotide *i* til nukleotide *j* i et gitt tidsintervall er den samme som raten fra *j* til *i*. Slike modeller kalles tidsreversible. Dette tilsvarer at rateparameterne settes til *g*=*a*, *h*=*b*, *i*=*c*, *j*=*d*, *k*=*e* og *l*=*f*. Altså ser man bare på symmetriske **R** matriser som:

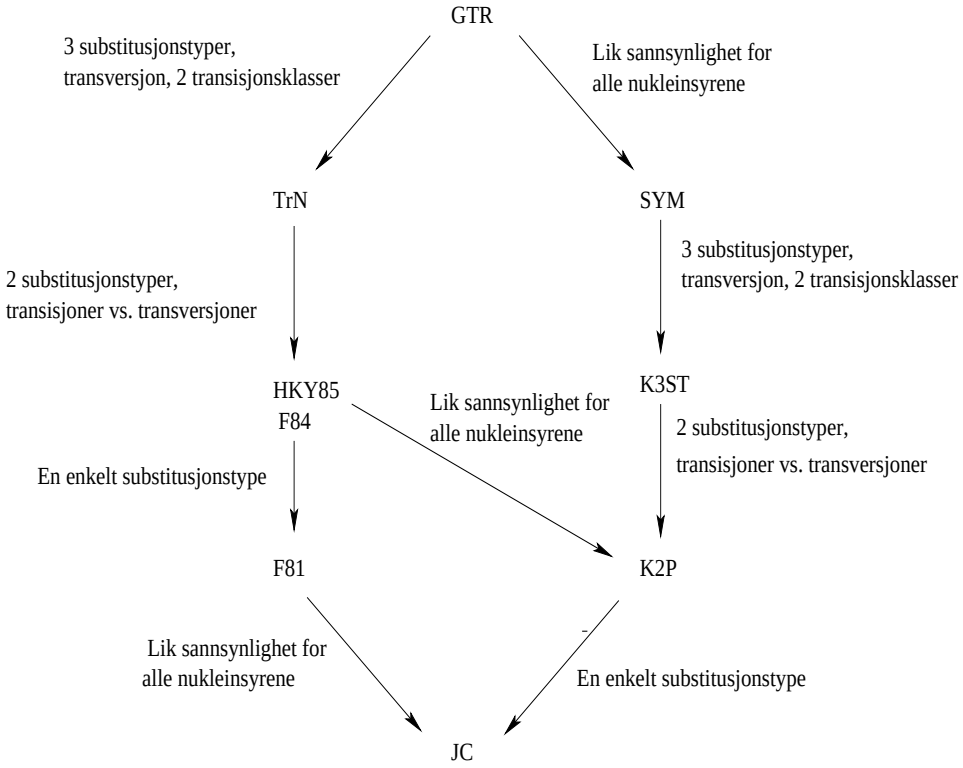
$$R = \begin{pmatrix} - & \mu a & \mu b & \mu c \\ \mu a & - & \mu d & \mu e \\ \mu b & \mu d & - & \mu f \\ \mu c & \mu e & \mu f & - \end{pmatrix} \tag{3.7}$$

Den mest generelle tidsreversible (GTR) modellen representeres da ved:

$$Q = \begin{pmatrix} -\mu(a\pi_C + b\pi_G + c\pi_T) & \mu a\pi_C & \mu b\pi_G & \mu c\pi_T \\ \mu a\pi_A & -\mu(a\pi_A + d\pi_G + e\pi_T) & \mu d\pi_G & \mu e\pi_T \\ \mu b\pi_A & \mu d\pi_C & -\mu(h\pi_A]\pi_C + f\pi_T) & \mu f\pi_T \\ \mu c\pi_A & \mu e\pi_C & \mu f\pi_G & -\mu(c\pi_A + e\pi_C + f\pi_G) \end{pmatrix} \tag{3.8}$$

Ved å sette enda flere parametre lik hverandre i GTR-modellen får vi Jukes Cantor-modellen, se figur 3.12. Denne figuren viser at man ved å sette noen parametre lik hverandre kan gå fra spesifikk modell til mer generelle modeller. Figuren er hentet fra Swofford *et al.* (1996).

En beskrivelse av noen modeller følger under, og er i hovedsak hentet fra



Figur 3.12: Figur som viser hvordan noen modeller for evolusjonær forandring kan forenkles, og dermed bli lik en annen modell. Fra spesifikk til mer generell modell.

Swofford *et al.* (1996).

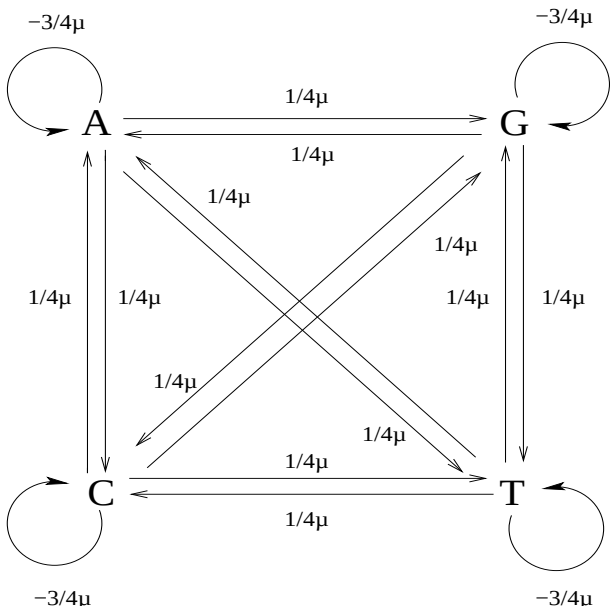
- Jukes Cantor (JC)
  Ved å anta at det er like stor sannsynlighet for alle nukleinsyrene, altså at det finnes like mange av hver: $\pi_a = \pi_c = \pi_g = \pi_t$, og at alle substitusjoner skjer med samme rate, så har man Jukes Cantor-modellen (JC):

$$Q = \begin{pmatrix} -\frac{3}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu \\ \frac{1}{4}\mu & -\frac{3}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{1}{4}\mu & -\frac{3}{4}\mu & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{1}{4}\mu & \frac{1}{4}\mu & -\frac{3}{4}\mu \end{pmatrix} \tag{3.9}$$

Sannsynligheten for hver nukleinsyre, μ, og substitusjonsraten $\frac{1}{4}$, blir som oftest kombinert i en parameter α: $\alpha = \frac{\mu}{4}$. Matrisen kan da skrives på formen:

$$Q = \begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix} \tag{3.10}$$

Figur 3.13 viser Markov-modellen for JC-modellen.



Figur 3.13: Figuren viser overgangene i matrisen for JC-modellen

- Kimuras to-parameter modell (K2P)
  Kimuras to-parameter modell (K2P), skiller seg fra JC-modellen ved at den tar hensyn til at transversjoner og transisjoner ikke skjer med samme rate, men den antar samme sannsynlighet for alle nukleinsyrene. I den generelle modellen blir da $a = c = d = f = 1$ og $b = e = \kappa$, og man får:

$$Q = \begin{pmatrix} -\frac{1}{4}\mu(\kappa+2) & \frac{1}{4}\mu & \frac{1}{4}\mu\kappa & \frac{1}{4}\mu \\ \frac{1}{4}\mu & -\frac{1}{4}\mu(\kappa+2) & \frac{1}{4}\mu & \frac{1}{4}\mu\kappa \\ \frac{1}{4}\mu\kappa & \frac{1}{4}\mu & -\frac{1}{4}\mu(\kappa+2) & \frac{1}{4}\mu \\ \frac{1}{4}\mu & \frac{1}{4}\mu\kappa & \frac{1}{4}\mu & -\frac{1}{4}\mu(\kappa+2) \end{pmatrix} \tag{3.11}$$

  Hvis transisjonsraten settes til $\alpha = \frac{\mu\kappa}{4}$ og transversjonsraten til $\beta = \frac{\mu}{4}$, kan matrisen omskrives til:

$$Q = \begin{pmatrix} -\alpha-2\beta & \beta & \alpha & \beta \\ \beta & -\alpha-2\beta & \beta & \alpha \\ \alpha & \beta & -\alpha-2\beta & \beta \\ \beta & \alpha & \beta & -\alpha-2\beta \end{pmatrix} \tag{3.12}$$

  Her er $\kappa = \frac{\alpha}{\beta}$ transisjons- transversjonsraten, sånn at når $\kappa = 1$, reduseres modellen til JC-modellen. Men siden det er dobbelt så mange transversjonsformer som transisjonsformer er den forventede transisjons- transversjonsraten 1:2.

- HKY85
  K2P-modellen kan generaliseres ved å tillate ulike sannsynligheter for de ulike nukleinsyrene. Man får da HKY85-modellen gitt ved:

$$Q = \begin{pmatrix} -\mu(\kappa\pi_g + \pi_y) & \mu\pi_c & \mu\kappa\pi_g & \mu\pi_t \\ \mu\pi_a & -\mu(\kappa\pi_t + \pi_r) & \mu\pi_g & \mu\kappa\pi_t \\ \mu\kappa\pi_a & \mu\pi_c & -\mu(\kappa\pi_a + \pi_y) & \mu\pi_t \\ \mu\pi_a & \mu\kappa\pi_c & \mu\pi_g & -\mu(\kappa\pi_c + \pi_r) \end{pmatrix} \tag{3.13}$$

Hvor

- $\alpha = \mu$
- $\beta = \mu\kappa$
- $\pi_r = \pi_a + \pi_g$
- $\pi_y = \pi_c + \pi_t$

Alle disse modellene, og også andre modeller, kan altså generaliseres eller spesifiseres ved å sette inn eller fjerne restriksjoner.

Beregning av sannsynlighetene

For å beregne sannsynligheten for en overgang fra en tilstand, her en nukleotide, til en annen tilstand ved tiden t, brukes $P_{ij}(t) = e^{Q_{ij}t}$, se for eksempel Schadt, Sinsheimer & Lange (2002) og Swofford *et al.* (1996).
For modellene som brukes i denne oppgaven får man da:

- JC

$$P_{ij}(t) = \begin{cases} \frac{3}{4} + \frac{3}{4}e^{-\mu t} & \text{for i=j} \\ \\ \frac{1}{4} - \frac{1}{4}e^{-\mu t} & \text{For i} \neq \text{j} \end{cases} \tag{3.14}$$

- K2P

$$P_{ij}(t) = \begin{cases} \frac{1}{4} + \frac{1}{4}e^{-\mu t} + \frac{1}{2}e^{-\mu t\left(\frac{\kappa+1}{2}\right)} & \text{For i=j} \\ \\ \frac{1}{4} + \frac{1}{1}e^{-\mu t} - \frac{1}{2}e^{-\mu t\left(\frac{\kappa+1}{2}\right)} & \text{For i} \neq \text{j, transisjon} \\ \\ \frac{1}{4} - \frac{1}{4}e^{-\mu t} & \text{For i} \neq \text{j, transversjon} \end{cases} \tag{3.15}$$

- HKY85

$$P_{ij}(t) = \begin{cases} \pi_j + \pi_j \left(\frac{1}{\Pi_j} - 1\right) e^{-\mu t} + \left(\frac{\Pi_j - \pi_j}{\Pi_j}\right) e^{-\mu t(1+\Pi_j(\kappa-1))} & \text{for i=j} \\ \pi_j + \pi_j \left(\frac{1}{\Pi_j} - 1\right) e^{-\mu t} - \left(\frac{\pi_j}{\Pi_j}\right) e^{-\mu t(1+\Pi_j(\kappa-1))} & \text{for i} \neq \text{j, transisjon} \\ \pi_j \left(1 - e^{-\mu t}\right) & \text{for i} \neq \text{j, transversjon} \end{cases} \tag{3.16}$$

Hvor

- $\Pi_j = \pi_a + \pi_g$ hvis nukleotide j er en purin, a eller g.
- $\Pi_j = \pi_c + \pi_t$ hvis nukleotide j er en pyrimidin, c eller t.

3.4.2 Kodonmodeller

Kodonmodeller skiller seg fra de andre nevnte modellene fordi de bruker kodonet som enhet og ikke nukleinsyren. I et kodon er det ikke uavhengighet mellom posisjonene da et kodon koder for en aminosyre, se tabell B.1. Kodonmodeller representerer en viktig forandring innen evolusjonære modeller fordi de eksplisitt tar hensyn til den genetiske koden, Lewis (2001). Statistiske tester viser at ved fylogenetisk analyse gir modeller som tar mer hensyn til den biologiske informasjonen i sekvensene, som for eksempel transisjons-transversjonsraten bedre resultater, Whelan, Li & Goldman (2001).

Her presenteres to kodonmodeller, Muse and Gaute-modellen, Lewis (2001) og Muse & Gaute (1994) og Goldman and Yang-modellen, Goldman & Yang (1994*a*).

- Muse and Gaute-modellen (M&G)
  Denne modellen antar at det bare kan være en mutasjon per steg, altså at sannsynligheten for aaa→acg er null. Modellen tar hensyn til synonyme og ikke-synonyme substitusjoner, men ikke transversjon og transisjon. Modellen:

$$Q_{ij} = \begin{cases} 0 & \text{Hvis to eller tre av parene } (i_1 j_1), (i_2 j_2), (i_3 j_3) \text{ er ulike} \\ \alpha\pi_n & \text{Synonyme forandringer} \\ \beta\pi_n & \text{Ikke-Synonyme forandringer} \end{cases} \tag{3.17}$$

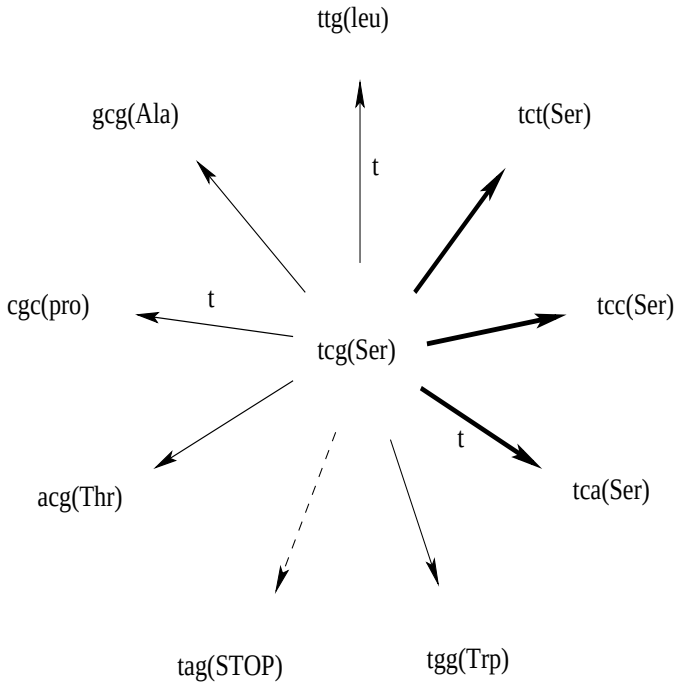
	ttt(Phe)	ttc(Phe)	tta(Leu)	ttg(Leu)	ctt(Leu)	ctc(Leu)	...	ggg(Gly)
ttt(Phe)	—	$\alpha\pi_c$	$\beta\pi_a$	$\beta\pi_g$	$\beta\pi_c$	0	...	0
ttc(Phe)	$\alpha\pi_t$	—	$\beta\pi_a$	$\beta\pi_g$	0	$\beta\pi_c$	...	0
tta(Leu)	$\beta\pi_t$	$\beta\pi_c$	—	$\alpha\pi_g$	0	0	...	0
ttg(Leu)	$\beta\pi_t$	$\beta\pi_c$	$\alpha\pi_a$	—	0	0	...	0
ctt(Leu)	$\beta\pi_t$	0	0	0	—	$\alpha\pi_c$	...	0
ctc(Leu)	0	$\beta\pi_t$	0	0	$\alpha\pi_t$	—	...	0
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
ggg(Gly)	0	0	0	0	0	0	...	—

Tabell 3.2: Substitusjonsmatrisen for Muse and Gaute modellen. Ved $i = j$, har man at Q_{ij} er den negative summen av raden.

Ved $i = j$ har man at Q_{ij} er den negative summen av raden.

Deler av substitusjonsmatrisen for denne modellen presenteres i tabell 3.2. Her er π_a, π_c, π_g og π_t frekvensene til nukleotidene. α og β er henholdsvis synonyme og ikke-synonyme substitusjonsrate. Ettersom hvert kodon består av tre nukleotider, er det 64 mulige kodoner, og substitusjonsmatrisen for denne modellen blir dermed mye større enn substitusjonsmatrisene for de tidligere nukleotidemodellene.
I artikkelen Muse & Gaute (1994) brukes nukleotidefrekvens, men det sies også at det ville vært bedre å bruke kodonfrekvens. På grunnlag av dette brukes begge de to frekvensene. De er adskilt ved at de kalles M&G nukleinsyre og M&G aminosyre.

- Goldman and Yang-modellen (G&Y)
  Goldman and Yang-modellen tar hensyn til substitusjonsraten, transisjons-transversjonsraten, synonyme og ikke-synonyme raten og aminosyreulikheter. Modellen er laget ved en Markov-modell, der tilstandene er de 61 sensekodonene. De tre siste kodonene, som koder for stoppkodonene, er ikke tatt hensyn til, da en slik mutasjon som oftest ikke vil overleve. Også for denne modellen sier man at det bare kan være en forandring for hvert kodon. Ethvert kodon har da på det meste ni "naboer", som representerer mulige kodonoverganger, se figur 3.14.



Figur 3.14: Eksempel på naboer for Goldman and Yang-modellen der transisjoner er merket med t, og de tykke pilene markerer synonyme substitusjoner

Parameterne i modellen er:

- μ, rate for gjennomsnittlig forandring
- π_j, kodonfrekvens
- κ, transisjons- transversjonsraten
- $\exp(-d_{aa_i aa_j}/V)$
  En faktor som sier om kodonene koder for ulike aminosyrer. Her er
  - aa_i representerer aminosyren kodon i koder for
  - d_{aa_i,aa_j} er hvor ulike aminosyre aa_i og aminosyre aa_j er
  - V er en faktor som sier noe om genets tendens til å forandres

For å sette $\exp(-d_{aa_i aa_j}/V)$, brukes en aminosyresubstitusjonsmatrise. Matrisen for to ulike kodoner $i = i_1 i_2 i_3$ og $j = j_1 j_2 j_3$ blir da som følger:

$$Q_{ij} = \begin{cases} 0, & \text{Hvis to eller tre av parene } (i_1 j_1), (i_2 j_2), (i_3 j_3) \text{ er ulike} \\ \mu\pi_j * \exp(-d_{aa_i, aa_j}/V), & \text{Hvis nøyaktig et av parene } (i_1 j_1), (i_2 j_2), (i_3 j_3) \text{ er ulike, og forskjellen er en transversjon} \\ \mu\kappa\pi_j * \exp(-d_{aa_i, aa_j}/V), & \text{Hvis nøyaktig et av parene } (i_1 j_1), (i_2 j_2), (i_3 j_3) \text{ er ulike, og forskjellen er en transisjon} \end{cases} \tag{3.18}$$

En forenkling av modellen beskrives i artiklene Yang & Nielsen (1998) og Nielsen & Yang (1998). Det er denne forenklete modellen som benyttes i denne oppgaven. Modellen er :

$$Q_{ij} = \begin{cases} 0 & \text{Hvis to eller tre av parene } (i_1 j_1), (i_2 j_2), (i_3 j_3) \text{ er ulike} \\ \mu\pi_j & \text{Ved synonym transversjon} \\ \mu\kappa\pi_j & \text{Ved synonym transisjon} \\ \mu\omega\pi_j & \text{Ved ikke-synonym transversjon} \\ \mu\omega\kappa\pi_j & \text{Ved ikke-synonym transisjon} \end{cases} \tag{3.19}$$

hvor ω er synonyme, ikke-synonyme raten.

Over gjelder det at ikke mer enn et av parene $(i_1 j_1), (i_2 j_2), (i_3 j_3)$ er ulike.

Her er også ved $i = j$ den negative summen av raden for i. Altså $Q_{ij} = -\sum$(raden for i).

3.5 Problemer med den progressive sammenstillingsalgoritmen

Det er to hovedproblemer som beskrives i litteraturen når det gjelder den progressive fremgangsmåten blant annet brukt i Clustal, se Thompson, Higgins & Gibson (1994*a*) og MAPA (2002). De to problemene er det lokale minimum problemet (the local minimum problem) og problemet med valg av parametre (the alignment parameter choice problem).

- Det lokale minimum problemet
  Ved sammenstilling av to sekvenser får man nesten alltid flere mulige sammenstillinger. I den progressive fremgangsmåten velges en av disse. Dette valget gjøres uten å sjekke om den man velger er den beste. Guidetreet dannes etter de parvise sammenstillingene, og det er ikke sikkert at rekkefølgen i guidetreet er den beste. En av de parvise sammenstillingene som ikke ble valgt, kunne ha gitt et bedre resultat. De første sammenstillingene forandres aldri, dermed påvirkes hele sammenstillingen.

- Problemet med valg av parametre
  Et og samme program vil kunne gi ulike svar etter hvordan parameterne er satt, Vingron & Waterman (1994) og Giribet, Wheeler & Muona (2002). Ved for eksempel å velge høye gapstraffer vil man prioritere sammenstillinger med få gap og flere sammenstillinger av ulike tegn.

I Ittero, programmet som lages her, gjøres det to forsøk på å motvirke problemet med valg av parametre. For noen av datasettene testes et spenn av gapverdier. Man ser på hvor bra de multiple sammenstillingene blir for gapverdiene, og velger gapstraffer etter det. En annen måte som prøves for å motvirke problemet er iterasjon. Det regnes ut parametre fra de multiple sammenstillingene fra hver iterasjon. Iterasjonen stopper når parameterne ikke lenger forandres.

3.6 BAliScore

BAliScore er en del av BAliBASE som definerer to verdier for sammenstillinger. BAliBASE er en database som inneholder multiple sammenstillinger laget spesielt for evaluering av multiple sammenstillingsprogrammer, Thompson, Plewniak & poch (1999*a*). De multiple sammenstillingene er sagt å være "riktige", og kan dermed benyttes som en referanse eller et mål på hvor bra sammenstillinger et program lager. Ettersom databasen bare har aminosyresekvenser brukes ikke disse sammenstillingene i oppgaven, bare BAliScore delen. BAliScore kan også brukes på nukleinsyresekvenser. Verdiene som beregnes i BAliScore er Sum of Pairs Score, SPS og Total Columns Score, TC, disse er beskrevet under. Programmet er hentet fra
ftp-igbmc.u-strasbg.fr/pub/BAliBASE/BAliScore
og kjørt lokalt. BAliScore er blant annet beskrevet og brukt i artiklene Edgar & Sjølander (2003) Thompson, plewniak & Poch (1999*b*). Fremstillingen her følger nøyaktig den som er gitt i den andre av de to artiklene over.

For å beregne en verdi ved hjelp av BAliScore må man ha en referansesammenstilling for datasettet. Verdiene fra BAliScore, og en referansesammenstilling for hvert datasett, brukes i oppgaven som et mål på hvor bra en sammenstilling er, det vil si hvor lik sammenstillingen er referansesammenstillingen, Thompson, Plewniak & Poch (1999*b*). De to verdiene som regnes ut og sjekkes er som følger:

- Sum of Pairs Score, SPS

  For en sammenstilling av N sekvenser og M kolonner, setter man kolonne nummer i som $A_{i1}, A_{i2}, ..., A_{iN}$. For hvert par av symboler A_{ij} og A_{ik} definerer man $p_{ijk} = 1$ hvis symbolene A_{ij} og A_{ik} er sammenstilt likt som i referansesammenstillingen, ellers $p_{ijk} = 0$. Verdien S_i for kolonne nummer i er da som følger:

  $$S_i = \sum_{j=1, j\neq k}^{N} \sum_{k=1}^{N} p_{ijk} \tag{3.20}$$

  SPS-verdien for hele sammenstillingen blir:

  $$\text{SPS} = \sum_{i=1}^{M} S_i / \sum_{i=1}^{Mr} S_{ri} \tag{3.21}$$

  hvor Mr er antall kolonner i referansesammenstillingen, og S_{ri} er S_i for kolonne nummer i i referansesammenstillingen. Denne verdien sier hvor mange sekvenser som er sammenstilt, og er en skaleringsfaktor for beregningen. SPS-verdien er dermed alltid et tall mellom null og en.

SPS-verdien øker med antall sekvenser som er riktig sammenstilt. SPS-verdien sier noe om hvor stor andel av sammenstillingen man ser på som har noen, om ikke alle sekvensene, riktig sammenstilt.

- Column Score, TC

For den kolonne nummer i i sammenstillingen er $C_i = 1$ hvis alle symbolene i kolonnen er sammenstilt som i referansesammenstillingen. Hvis ikke settes $C_i = 0$. TC-verdien for sammenstillingen er da som følger:

$$\text{TC} = \sum_{i=1}^{M} C_i / M \tag{3.22}$$

TC-verdien sier om alle sekvensene i en sammenstilling er sammenstilt riktig.

Kapittel 4

Programmet Ittero

Dette kapittelet presenterer programmet Ittero. Hensikten med å lage et program for multippel sammenstilling er å sette seg godt inn i noen av algoritmene og prinsippene for et slikt program, og å teste ut problemstillingene i oppgaven.

Problemstillingene som Ittero tar hensyn til:

- Ved sammenstilling av proteinkodende nukleinsyresekvenser kan man, som beskrevet tidligere, ikke ha gap av lengde en eller to nukleinsyrer, eller gap som ikke er av lengde x*kodonlengde, hvis man vil beholde leserammen. Dette løses i Ittero ved at man istedetfor en og en nukleinsyre slår sammen tre og tre nukleinsyrer til en enhet før man starter sammenstillingen. I den dynamiske programmeringen for parvise sammenstillinger er det da kodoner som sjekkes mot hverandre og ikke nukleinsyrer. I nukleinsyremodellene, JC, K2P og HKY85, legges verdiene fra de tre nukleinsyrene sammen. Når man så introduserer gap, settes disse av lengde x*kodonlengde. Det samme gjøres ved den multiple sammenstillingen.
- Ittero er også laget for å kunne teste ut modeller for evolusjonær forandring fra fylogeni for multiple sammenstillinger. Dette er så langt jeg har funnet ikke brukt i andre sammenstillingsprogrammer på samme måte. Modellene er kodet inn i programmet og brukeren velger modell. Hvordan de ulike parameterne for modellene er settes, beskrives i avsnitt 4.4.

I forbindelse med valg av parametre i modellene, velges det å gjøre programmet iterativt. Først brukes IUB-verdiene for å få en initiell sammenstilling. Deretter regnes parameterne ut fra denne initielle sammenstillingen, og disse brukes for å regne ut en ny sammenstilling. Iterasjonene gjentas til parmeterne ikke lenger forandres.

Ittero består av to hoveddeler, en del for å gjøre den multiple sammenstillingen, og en som regner ut parameterne som igjen brukes til å lage en ny

sammenstilling.

4.1 Beskrivelse av programmet

Den progressive fremgangsmåten i programmet følger i stor grad Clustal, altså en progressiv algoritme i tre deler:

- Parvise sammenstillinger dannes og en avstandsmatrise lages ut fra dette.
- Et guidetre lages på bakgrunn av avstandsmatrisen. Her benyttes Neighbour Joining algoritmen. Roten settes med algoritmen beskrevet i avsnitt 3.3.2.
- Progressiv multippel sammenstilling av sekvensene etter rekkefølgen i guidetreet.

Clustal er et program med åpen kildekode og derfor lett å få tilgang til. Det er derimot ikke like enkelt å finne ut hvilke valg som er gjort. Algoritmene beskrives i artikler, men ved flere mulige valg i en algoritme er det som oftest ikke beskrevet hvilke valg som er gjort. Et slikt valg er ved parvis sammenstilling. I dynamisk programmering for to sekvenser vil det ofte være flere mulige sammenstillinger som oppnår høyeste verdi. Clustal gir likevel bare en sammenstilling som resultat her. Hvordan dette velges er at man alltid følger diagonalen der dette er en mulighet. Når man har valg mellom å sette gap i begge sekvensene, velges konsekvent den ene sekvensen som den man setter et gap inn i.

4.2 Programmets struktur/oppbygning

Den delen av Ittero som lager de multiple sammenstillingene består av:

- **Hoveddel**
  Dette er hoveddelen som starter programmet og kontrollerer det andre som skjer. Den leser sekvensene fra fil og lager de parvise sammenstillingene ved hjelp av de forskjellige metodene i Avstandsmatrise og Metoder. Det opprettes et objekt av typen Sammenlikning for hver parvis sammenlikning som gjøres. Disse kastes etterhvert for å ikke fylle opp minnet i maskinen. Filformatet programmet leser, er ikke et standarisert filformat, men et filformat som kun brukes i denne oppgaven. I utgangspunktet var planen å ha et filformat som kunne brukes igjen i flere programmer, men det ble det ikke lagt vekt på da det omtrent ikke finnes programmer som bruker samme filformat. Formatet til filene som

leses i oppgaven vises i figur 4.1

Antall sekvenser
ID sekvensens navn
sekvensens lengde
sekvensen
ID Sekvensens navn
sekvensens lengde
Sekvensen
.
.
.

Figur 4.1: Oppsettet til filformatet programmet leser.

Resultatet skrives til fil i Hoveddel. Figur 4.2 viser eksempel på resultatfil. Denne filen gjøres om til filformatet som BAliScore leser ved et lite program.

- **Avstandsmatrise**
  Denne klassen har matrisen som brukes til å regne seg frem til de parvise sammenstillingene ved dynamisk programmering. Hvis det er flere mulige parvise sammenstillinger for to sekvenser, følges fremgangsmåten i Clustal ved at man først velger diagonalen. Hvis dette ikke er et alternativ, velges konsekvent en av sekvensene som den det introduseres gap i. Avstandsmatrisen, som er grunnlaget for guidetreet, regnes også ut her. Disse regnes i Ittero ut ved at man ser på hvor mange like som er sammenstilt i forhold til totalt antall sammenstilte.

- **Multiple**
  Klassen Multiple lager de multiple sammenstillingene ved hjelp av metoder i klassen Modeller. Den plukker sekvenser etter rekkefølgen i guidetreet gitt ved Neighbour Joining.

- **Modeller**
  Her finnes alle modellene i programmet. De ulike modellene er presentert i kapittel 3.4. Klassen benytter klassene Kodonmodeller, Nukleotidmodeller og Oversettelse for å gjøre beregningene, se figur 4.3.

- **NeighbourJoining**
  Neighbour Joining-algoritmen for å lage guidetreet utføres her. Guidetreet bruker objekter av klassen Trenoder til å lagre og regne ut opplysninger om hver node i treet.

```
gapOpen: -10.0 gapExtend: -0.2
Multiple: gapOpen: -10.0 gapExtend: -0.1

ID cya228
ID hub524
ID cya57
ID cya143
ID cya169
ID cya324
ID K-139c
ID cya264
ID PCCc
ID grsA
ID cya31
ID cya161
ID PCC
ID cya118
ID K-139
1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cca gaa aat tta gct tat gtt ata tac act tct ggt tca aca gga aaa cct aaa ggt gtg
cct gag aat tta gcc tat gtt ata tac act tct ggt tca acg gga aaa cct aaa ggt gtg
aca gaa aat tta gct tat gtt ata tac act tcg ggt tca acg gga aaa cct aaa ggt gtg
tca acc gat ctt gct tat gtt att tat act tct ggt aca aca ggc aat cca aaa ggt aca
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggg gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta


21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
atg aat att cat aga gga --- att tgt aat act ctg aca tat act att --- ggt cat tat
atg aat att cat aga gga --- att tgt aat act ctg aca tat act att --- ggt cat tat
atg aat att cat aga gga --- att tgc aat act ata aaa tat gct att --- ggt cat tat
atg aat att cat aga gga --- att tgt aat act ctg aca tat act att --- ggt cat tat
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
.   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
```

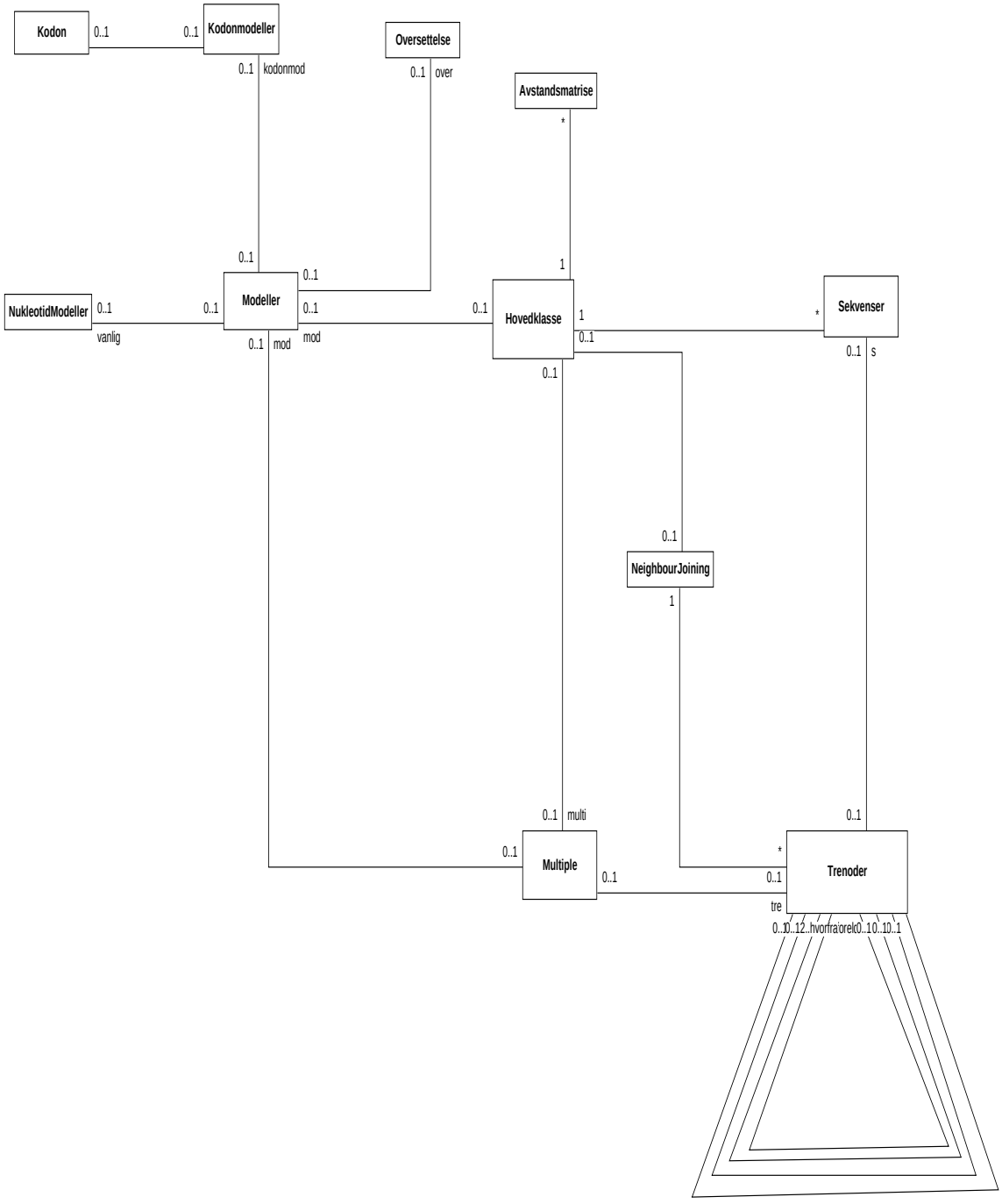
Figur 4.2: Eksempel på filformatet det skrives til i oppgaven

- **Trenoder**
  Dette er nodene i guidetreet som bygges i Neighbour Joining, all informasjon om de ulike nodene befinner seg her.

- **Sekvenser**
  Informasjonen om sekvensene legges her når de leses fra fil i klassen Hoveddel. Et objekt for hver sekvens.

- **Kodon**
  Denne brukes til å holde informasjon om de ulike kodonene som inngår i datamaterialet. Dette er stort sett hvilket kodon det er, og hvor mange kodoner av denne typen det er i datamaterialet.

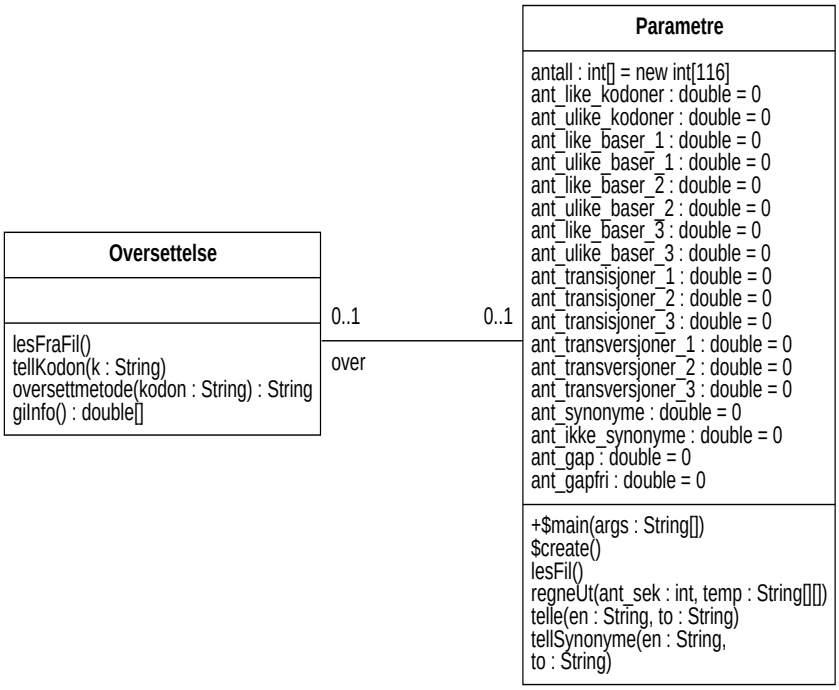
Se også figur 4.3 som viser UML klassediagram for programmet, UML (2004).



Figur 4.3: Figuren viser UML klassediagram for delen av programmet som lager de multiple sammenstillingene. Figuren er laget med programmet Telelogic Tau UML Suite.

4.3 Delen av Ittero som regner ut parameterne

Alle parameterne regnes ut fra en multippel sammenstilling, bortsett fra nukleotide- og kodonfrekvens. Nukleotide- og kodonfrekvens telles ved innlesing av sekvensene. Denne delen inneholder metoder for å telle antall forandringer på nukleotidenivå, transisjoner, transversjoner, synonyme posisjoner, ikke-synonyme posisjoner, synonyme og ikke-synonyme forandringer og regner ut gjennomsnittlig forandring, transisjons- transversjonsrate og synonym, ikke-synonym rate. Alle disse parameterne forklares i avsnitt 4.4. Figur 4.4 viser UML klassediagram for denne delen, UML (2004).



Figur 4.4: Figuren viser UML klassediagram for delen av programmet som regner ut parameterne. Figuren er laget med programmet Telelogic Tau UML Suite.

4.4 Simulering og parametertolkning

Alle de evolusjonære modellene inneholder parametre man må finne ved hjelp av en allerede eksisterende sammenstilling. Dette løses ved hjelp av en initiell sammenstilling som lages med Ittero og IUB-substitusjonsmatrisen. Like tegn får verdi 1.9 ved sammenstilling, mens ulike får verdi 0.0. Dette omtales som den initielle modellen. Fra denne initielle sammenstillingen regnes parameterne til de andre modellene ut. Etter at de initielle parameterne er regnet ut fra den initielle sammenstillingen gjøres iterasjonene for en modell. En første sammenstilling for denne modellen dannes, og fra denne regnes parameterne til andre iterasjon ut. Denne prosessen med sammenstillinger og utregning av parametre fortsetter til parameterne ikke lenger forandres.

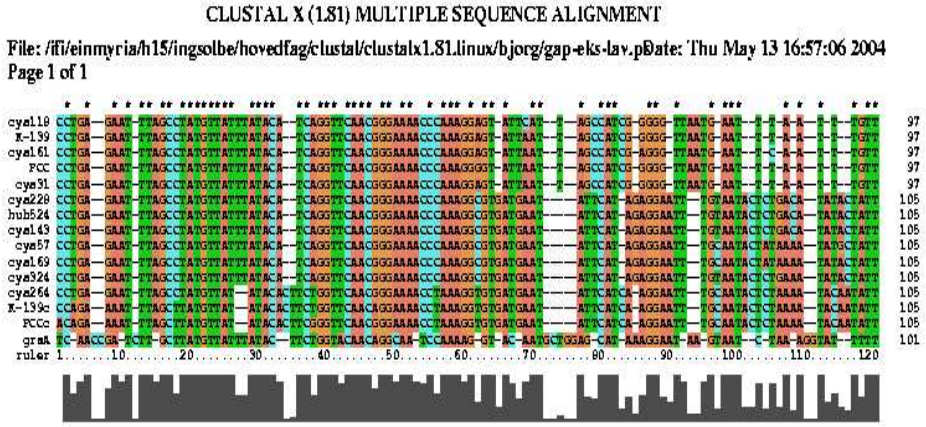
Tolkning av parametre

Modellene som implementeres i Ittero er i utgangspunktet tenkt for fylogenetisk analyse, og ut fra dette må endel parametre sees nøye på. Generellt sett er valg av parametre til multippel sammenstilling en komplisert oppgave, Vingron & Waterman (1994).

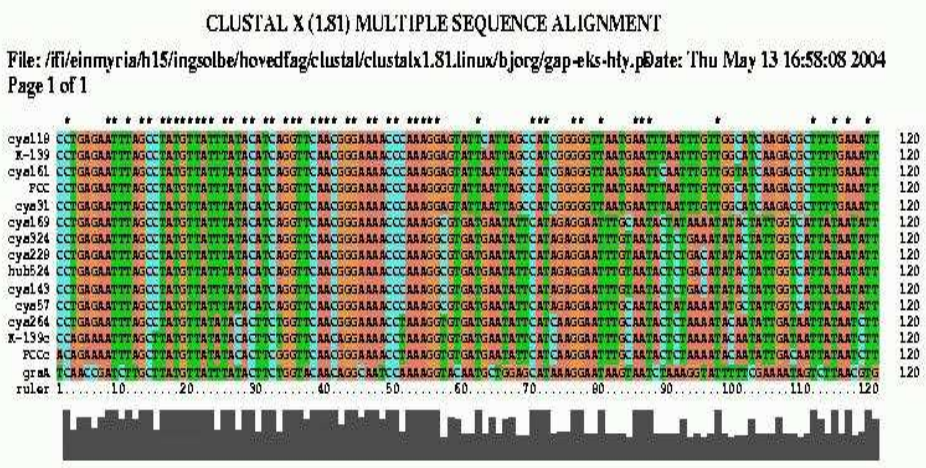
4.4.1 Gapstraffer

Valg av gapstraffer har mye å si for resultatet ettersom de direkte styrer hvor gap settes inn i sekvensene og hvor mange gap som settes inn. Ved å sette gapstraffer veldig høyt vil man få få gap, mens ved å sette gapstraffer lavt vil man få mange gap. Hva som er høyt og lavt påvirkes selvfølgelig av de andre parameterne som settes, Thompson, Higgins & Gibson (1994*a*). Hvis man for eksempel velger en lav gapstraff og en høy verdi for substitusjon, vil man få en sammenstilling med mange gap og få sammenstilte nukleotider. En slik sammenstilling er ikke realistisk, Nei & Kumar (2000).

Et eksempel på hvordan sammenstillinger påvirkes av gapstraffer vises i figur 4.5. Her er det i eksempel (a) satt gapstraffer høyt og i eksempel (b) satt lavt. Gapstraffene i Clustal kan være i området 0-100. Verdiene som brukes i eksempelet vises i tabell 4.1. Her ser man at ved høy gapstraff innføres det ikke gap, mens ved lav gapstraff innføres det flere gap. Gapstraffene velges litt forskjellig for de ulike datasettene. I datasettet hvor gapverdiene til referansesammenstillingen er kjent, velges disse. Der de er kjent er de oppgitt av forfatter. For de andre datasettene velges gapstraffer ut fra hva som gir beste verdi i BAliScore etter at de andre parameterne er satt.



(a) Lav gapstraff



(b) Høy gapstraff

Figur 4.5: Figuren viser eksempler på resultat fra sammenstillinger av *mcyABC*datasettet. Sammenstillingen er laget i Clustal. Gap-parameterne vises i tabell 4.1.

	Parvise		Multiple	
	Gap open	Gap extend	Gap open	Gapextend
Høye	100	20	100	10
Lave	1.00	0.02	1.00	0.01

Tabell 4.1: Gapverdiene som brukes i eksempelet i figur 4.5

4.4.2 Nukleotide- og kodonfrekvens, π

Nukleotide- eller kodonfrekvens er i modellene angitt ved π. Dette er en parameter for frekvensene av de ulike nukleotidene eller kodonene i datamaterialet. Parameteren finner man ved å telle antall ulike nukleotider og kodoner for så å dele på det totale antallet. Det er ikke helt enkelt å sette denne parameteren. I evolusjonære modeller er π frekvensen til "target nucleotide", der frekvensen hentes fra dataene man har, Muse & Gaute (1994). Muse & Gaute angir sannsynligheten for at kodonet AGG forandres til AGA som $\alpha\mu_a$. For fylogenetisk analyse er dette greit, da man der ser på sannsynligheten til et tre. For hver node finner man hva sannsynligheten blir for hver av de fire nukleotidene. Se dannelsen av tre ved Maximum Likelihood, avsnitt 3.3.2. I multippel sammenstilling vil man derimot ha et mål for hvor sannsynlig det er at en nukleotide er substituert med en annen, ikke hva som kan ha vært forgjengeren til de to. Man kan ikke si at det er den ene sekvensen som har forandret seg til den andre da man ikke vet hvilken av de som i er foreldresekvensen. I modeller som tar hensyn til ulike nukleotide- og kodonfrekvenser vil man få ulikt resultat hvis man bare tar en av sekvensene tilfeldig og sier at det er denne som er foreldresekvensen. Dette gjelder spesielt hvis frekvensene er svært ulike. I Ittero løses dette ved å multiplisere frekvensene til de to nukleotidene, man tar da hensyn til begge to. I et område med mange overganger mellom for eksempel c og g, vil dette gjenspeiles i den nye frekvensparameteren. Istedet for en frekvensparameter for en nukleotide, får man da en parameter for begge nukleotidene.

4.4.3 Rate for gjennomsnittlig forandring, μ

Parameteren μ forteller hvor mange forandringer som faktisk har funnet sted i datamaterialet. Man teller totalt antall forandringer i den multiple sammenstillingen mellom alle par av sekvenser og deler på totalt antall mulige forandringer. Antall mulige forandringer er antall forandringer pluss antall ikkeforandringer.

4.4.4 Transisjons-, transversjonsraten, κ

Det er fire transisjonsmuligheter a→g, g→a, c→t og g →t, og det er åtte transversjonsmuligheter, a→c, a →t, g→c, g →t, c→a, c→g, t →a, og t →g. Teoretisk vil det være flere transversjoner enn transisjoner. Transisjoner skjer likevel oftere enn transversjoner i reelle data. Transisjons- transversjonsraten ligger gjerne mellom 0.5-2 for nukleært DNA, Nei & Kumar (2000). Parameteren regnes ut fra hvor mange transisjoner og transversjoner som finnes i sammenstillingen.

4.4.5 Synonyme, ikke-synonyme raten, ω

En synonym forandring er en forandring i et kodon som fører til at kodonet forandres, men at det fortsatt koder for samme aminosyre. Et eksempel er aaa→aag som begge koder for aminosyren lysin. En ikke-synonym forandring er en forandring som fører til at kodonet ikke lenger koder for samme aminosyre. Et eksempel er aaa→aac, der aac koder for asparagin.
Parameteren regnes ut ved, Yang (2004):

$$\omega = \frac{d_N}{d_S}$$

der

- ω er synonyme, ikke-synonyme raten
- d_N er antall synonyme substitusjoner per synonym posisjon
- d_S er antall ikke-synonyme substitusjoner per ikke-synonym posisjon

En synonym posisjon er for hvert kodon hvor mange mulige forandringer av en nukleotide som fører til en synonym forandring. Ikke-synonym posisjon er da hvor mange forandringer som fører til ikke-synonyme forandringer. I begge tilfeller ser man bort fra de forandringer som fører til at kodonet blir et stoppkodon. For hver av posisjonene i kodonet $i = 1, 2, 3$ har man mulighet for tre forandringer. For antall mulige synonyme forandringer, s, og ikke-synonyme forandringer n, har man da

$$s = \sum_{i=1}^{3} f_i$$

og

$$n = 3 - s$$

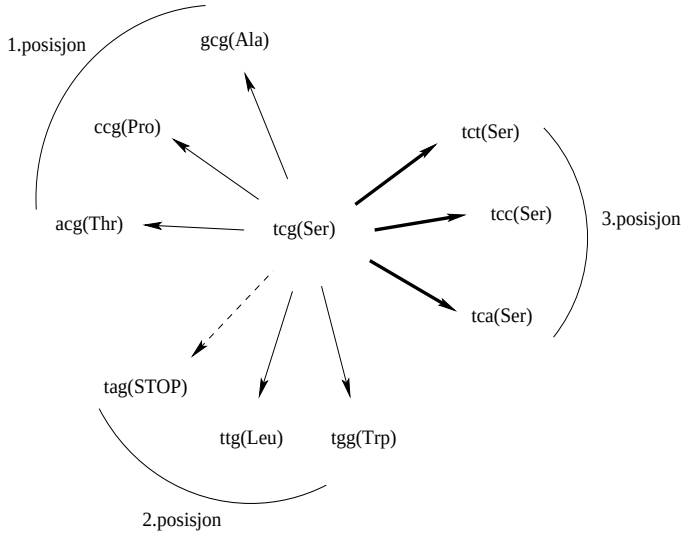
der f_i er antall forandringer for hver plass i kodonet.
For eksempel for kodonet tcg har man:

$$s = \frac{0}{3} + \frac{0}{2} + \frac{3}{3}$$

og

$$n = 3 - \frac{3}{3} = \frac{6}{3}$$

Nei & Kumar (2000). Se også figur 4.6. Antall synonyme og ikke-synonyme po-



Figur 4.6: Eksempel på alle mulige overganger for kodonet tcg. De tykke pilene markerer synonyme forandringer, de tynne pilene markerer ikke-synonyme forandringer. I posisjon 3 har man tre av tre mulige synonyme forandringer, mens begge de to andre posisjonene bare gir ikke-synonyme forandringer. I posisjon to er det mulighet for en forandring til et stoppkodon, denne muligheten telles ikke med i antall muligheter, og her er det dermed bare to mulige overganger, hvor begge er ikke-synonyme.

sisjoner i datamaterialet regnes ut ved at man teller hvor mange av hvert kodon som finnes. Antallet multipliseres med kodonets synonyme posisjoner og ikke-synonyme posisjoner. Antall synonyme og ikke-synonyme forandringer i datamaterialet telles ut fra sammenstillingene, og man kan dermed regne ut raten for synonyme og ikke-synonyme forandringer. Tabell 4.2 viser utregning av s og n for noen kodoner.

Kodon	Amninosyre	1.pos	2.pos	3.pos	S	N=3-S
ttt	Phe	0	0	1/3	1/3	8/3
tta	Leu	1/3	0	1/3	2/3	7/3
ttc	Phe	0	0	1/3	1/3	8/3
ttg	Leu	1/3	0	1/3	2/3	7/3
tat	Tyr	0	0	1/1	1/1	2
tct	Ser	0	0	3/3	3/3	6/3
tgt	Cys	0	0	1/2	1/2	5/2
att	Ile	0	0	2/3	2/3	7/3
ctt	Leu	0	0	3/3	3/3	6/3
gtt	Val	0	0	3/3	3/3	6/3
acg	Thr	0	0	3/3	3/3	6/3
agc	Ser	0	0	1/3	1/3	8/3
act	Thr	0	0	3/3	3/3	6/3
atc	Ile	0	0	2/3	2/3	7/3
agt	Ser	0	0	1/3	1/3	8/3
atg	Met	0	0	0	0	3
cag	Glu	0	0	1/3	1/3	8/3
cga	Arg	1/3	0	3/3	4/3	5/3
cat	His	0	0	1/3	1/3	8/3
cta	Leu	0	0	3/3	3/3	6/3
cgt	Arg	0	0	3/3	3/3	6/3
ctg	leu	1/3	0	3/3	4/3	5/3
tac	Tyr	0	0	1/1	1/1	2
tca	Ser	0	0	3/3	3/3	6/3
tag	stop					
tga	stop					
tcg	Ser	0	0	3/3	3/3	6/3
tgc	Cys	0	0	1/1	1/1	2

Tabell 4.2: Utregning av synonyme og ikke-synonyme posisjoner for noen kodoner.

4.4.6 Parvis likhet

Guidetreet lages fra en avstandsmatrise bygget på hvor like sekvensene er. Dette gjøres ved at man lager alle de parvise sammenstillingene og teller hvor mange like som er sammenstilt. Den parvise sammenstillingen som har flest like, er sagt å inneholde de to mest like sekvensene. Her er det flere muligheter for å si hva det vil si at to tegn er like. I Clustal telles antall like nukleotider som er sammenstilt. I Clustal er dette naturlig da nukleotiden er enheten i programmet. I Iterro er det kodonet som er enheten, og det er da naturlig å bruke kodonet som kriterium for likhet. Her er det to muligheter. En mulighet er at man har likhet når kodonene er like, den andre muligheten er at man har likhet når de to kodonene koder for samme aminosyre. Vi vil se på begge måtene å telle likhet på for å se om en av de gir klare fordeler.

Kapittel 5

Datamaterialet

Fordi et sammenstillingsprogram lager en sammenstilling uansett hva man gir av sekvenser som innputt, må valg av sekvenser være bevisst. Programmene gir bare fornuftige svar hvis sekvensene man ser på er fornuftige og homologe, Notredame (2001). Ved globale sammenstillinger, som her, må også sekvensene være relativt like over det hele. Sekvensene som brukes i oppgaven velges ut fra følgende kriterier:

- Sekvensene må være kodende nukleinsyresekvenser fordi for ikke-kodende sekvenser er ikke leserammen alltid beholdt.
- Det må for sekvensene finnes en referansesammenstilling som kan brukes i BAliScore, se avsnitt 3.6.
- Tilslutt bør de ikke være for ulike da det ved sekvenser på nukleinsyrenivå kan bli for mye ulikhet, og det kan være en fordel å sammenstille disse på aminosyrenivå, Goldman & Yang (1994*a*).

De tre datsettene som brukes i oppgaven presenteres i de følgende avsnittene.

5.1 *mcyABC*

Dette datasettet er brukt og beskrevet i de to artiklene Baumbusch *et al.* (2001) og Jakobsen *et al.* (2003). Referansesammenstillingen for sekvensene er laget i Clustal med parametre som vist i tabell 5.1. Sammenstillingen er modifisert manuelt i ettertid av artikkelforfatterne ved hjelp av programpakken McClade, Maddison & Maddison (2004).

Sekvensene er fra genet *mcyABC* hos 15 nært relaterte Cyanobakterier *Microcystis*, se tabell 5.2. Sekvensene har en likhet fra 17.7% til 96.8% med gjennomsnittlig likhet på 56.4%.

Parvise		Muliple	
gap opening	10.00	gap opening	10.00
gap extension	0.20	gap extension	0.10
		delay divergent seq	30%
		DNA transition weight	0.50
DNA weight matrix:	IUB	DNA weight matrix:	IUB

Tabell 5.1: Clustalparameterne brukt i referansesammenstillingen.

Kortnavn	Bakterie	Sekvenslengde
cya31	*Microcystis aeruginosa*	930
cya57	*Microcystis aeruginosa*	927
cya143	*Microcystis aeruginosa*	930
hub524	*Microcystis aeruginosa*	933
PCC	*Microcystis aeruginosa*	933
K-139	*Microcystis aeruginosa*	933
cya228	*Microcystis aeruginosa*	933
cya264	*Microcystis botrys*	936
cya324	*Microcystis* sp.	927
cya118	*Microcystis* sp.	930
cya161	*Microcystis botrys*	930
cya169	*Microcystis viridis*	933
PCCc	*Microcystis aeruginosa*	936
K-139c	*Microcystis aeruginosa*	933
grsA	*B. brevis*	924

Tabell 5.2: Sekvensene i datasettet *mcyABC*

5.2 β-globin

Dette datamaterialet er hentet fra artikkelen Yang *et al.* (2000). Et av ti datasett brukt i artikkelen velges, nemlig datasettet D2. Sekvensene er β-globin gener fra 17 ulike vertebrater (virveldyr), se tabell 5.3. Alle sekvensene har en lengde på 432 nukleotider. Sekvensene har en likhet fra 16.0% til 93.0% med et gjennomsnitt på 49.5%. Referansesammenstillingen er sammenstilt manuelt uten at artikkelen beskriver dette nærmere. Referansesammenstillingen for dette datasettet inneholder ingen gap.

Kortnavn	Fullt navn	Beskrivelse
human	human	Homo sapiens hemoglobin beta chain (HBB) mRNA
tarsier	tarsier	T.syrichta beta globin gene, complete cds.
bushbaby	bush baby	Otolemur crassicaudatus epsilon-, gamma-, delta-,and beta-globin genes, complete cds, and eta-globin pseudogene
hare	hare	Lepus europaeus adult beta-globin gene
rabbit	rabbit	Rabbit beta-globin mRNA
cow	cow	Bovine adult beta-globin gene
sheep	sheep	Sheep beta-B globin gene
pig	pig	S.scrofa beta-globin gene
elephseal	elephant seal	Mirounga angustirostris mRNA, partial cds.
rat	rat	Rat major beta-globin mRNA, complete cds.
mouse	mouse	Mouse gene for beta-1-globin.
hamster	hamster	Cricetinae mRNA for beta major globin chain
marsupial	opposum	Opossum beta-hemoglobin beta-M gene, complete cds.
duck	duck	Duck rearranged beta-globin gene for beta-globin (hemoglobin alpha-2 beta-2)
chicken	chicken	Gallus gallus gene coding for beta-globin.
xenlaev	African claved frog (Xenopus laevis)	Xenopus laevis mRNA for larval beta II globin
xentrop	western cloved frog (Xenopus tropicalis)	Xenopus tropicalis larval beta-globin gene

Tabell 5.3: Sekvensene i β-globin datasettet

5.3 Anhydrase

Karbonanhydrase er et enzym i røde blodceller som hjelper til å omdanne karbondioksid og vann til karbonsyre, protoner og bikarbonationer, Dutta & Goodsell (2004). Dette datasettet er hentet fra BAliBASE. Det tilhører referanse 1, og kalles her 2cba. I BAliBASE finnes sekvensene kun som aminosyresekvenser. For å finne nukleotidsekvensene er SwissProt:
http://www.ebi.ac.uk/swissprot/
benyttet. SwissProt Accessionnumber for sekvensene vises i tabell 5.4. Datasettene i BAliBASE er valgt fra databasene FSSP og HOMSTRAD, eller de kan være hentet fra litteraturen. VAST webserver er brukt for å fastslå at sekvensene i hver

Sekvensens navn	SwissProt Accession number	Sekvenslengde
CAH1_HUMAN	P00915	711
CAH4_RAT	P48284	726
CAH6_HUMAN	P23280	711
CAH_DUNSA	P54212	717
CAH2_CHLRE	P24258	759

Tabell 5.4: Tabellen viser navn brukt på sekvensene, deres SwissProt Accession number og sekvensenes lengde i nukleotider.

sammenstilling er relaterte og dermed kan sammenstilles. Sammenstillingene blir også manuelt verifisert. Dette er hentet fra Thompson, Plewniak & Poch (1999*a*). Årsaken til at ikke flere av datasettene fra BAliBASE brukes, er at det er altfor tidkrevende å finne de tilsvarende nukleotidsekvensene. Sekvensene har en likhet fra 19.5% til 31.9% med gjennomsnitt 25.4%. Disse sekvensene har en noe lav likhet, men de brukes for å se hvordan modellene i Ittero takler dette. Ifølge artikkelen Thompson, Plewniak & Poch (1999*b*) har likhet for aminosyresekvenser en grenseverdi rundt 10-20% der sekvensene er for ulike til å få bra resultater.

Nukleotidefrekvens

For datamaterialet vises nukleotidfrekvensene i tabell 5.5.

Nukleotide	Frekvens		
	Datasett		
	mcyABC	βglobin	Anhydrase
a	0.30	0.21	0.26
c	0.18	0.26	0.28
g	0.20	0.29	0.25
t	0.32	0.24	0.21

Tabell 5.5: Tabellen viser nukleotidefrekvenser for datasettene.

Kapittel 6

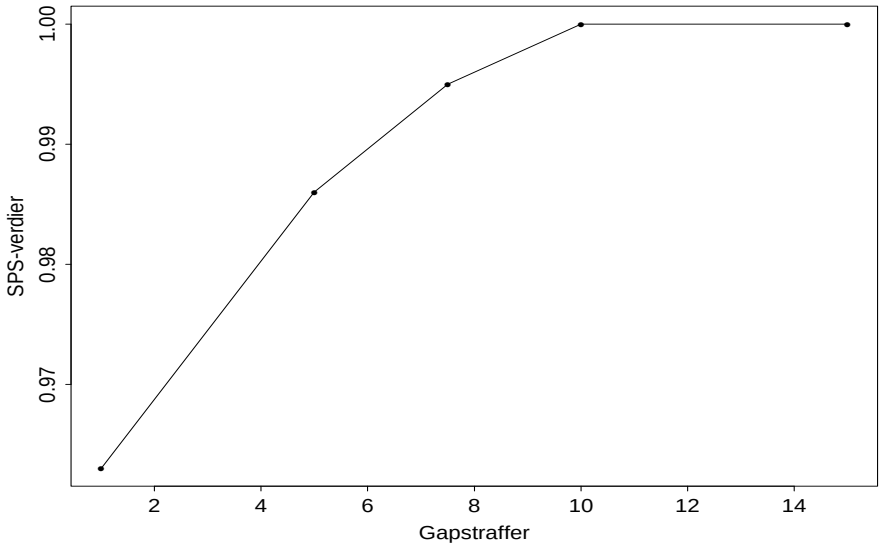
Resultater

I dette kapittelet presenteres alle resultatene fra de multiple sammenstillingene med Ittero og Clustal. I tillegg til dette presenteres også endel andre punkter, som resultatene fra valg av gapstraffer.

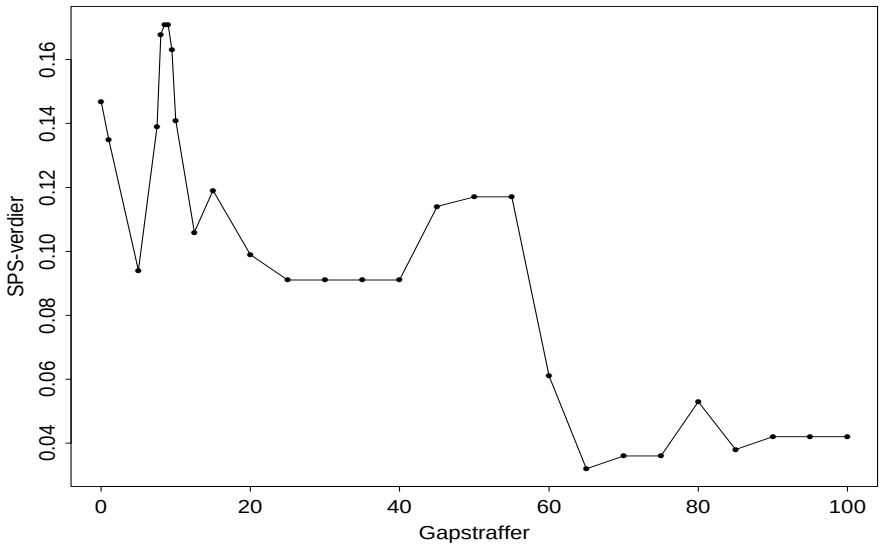
6.1 Valg av gapstraffer

For datasettet *mcyABC* settes gapstraffer likt de som ble satt da referansesammentsillingen ble laget. For de andre datasettene velges gapstraff etter hvilken gapstraff som gir beste SPS-verdi for den initielle modellen. Ordet gapstraffer henviser heretter til gapOpen gapstraffer da det er disse det testes for. Verdien for gapExtend settes til en fast verdi da endel tester viser at disse ikke fører til forandringer i resultatene. Alle testene utføres med verdiene gapExtend=0.2 for parvise sammenstillinger og gapExtend=0.1 for multiple sammenstillinger. Figur 6.1 viser hvordan SPS-verdiene varierer for datasettet β-globin. Gapstraffene er satt mellom 1.0 og 15.0. Ved gapstraffer på 10.0 og høyere innføres det ikke gap i sekvensene i dette datasettet, og SPS-verdien er 1.0. Ved lavere gapstraffer innføres det gap, og SPS-verdien synker. Datasettet β-globin testes med gapstraffene 10.0, 7.5, 5.0 og 2.5. Her vil man få SPS-verdi=1.0 uansett ved gapstraff=10.0 og høyere, og noen lavere gapstraffer testes derfor for å se på hva som vil skje med resultatene ved lavere gapstraffer.

For datasettet Anhydrase testes den initielle modellen i Ittero med gapstraffer i området 0.0-100.0. Hvordan SPS-verdien varierer i dette området vises i figur 6.2. Fra figuren kan man se at SPS-verdien har en topp på gapstraff=9.0. SPS-verdien ligger også forholdsvis høyt for verdier rundt 0.0 og 1.0, og for verdier fra 40.0 til 60.0. Gapstraff velges til 9.0 ettersom denne verdien gir høyest SPS-verdi.



Figur 6.1: Variasjon i SPS-verdi for datasettet β-globin ved gapstraffer 1.0 til 15.0. Ved gapstraffer høyere enn 10.0 vil ingen gap bli innført i sekvensene.



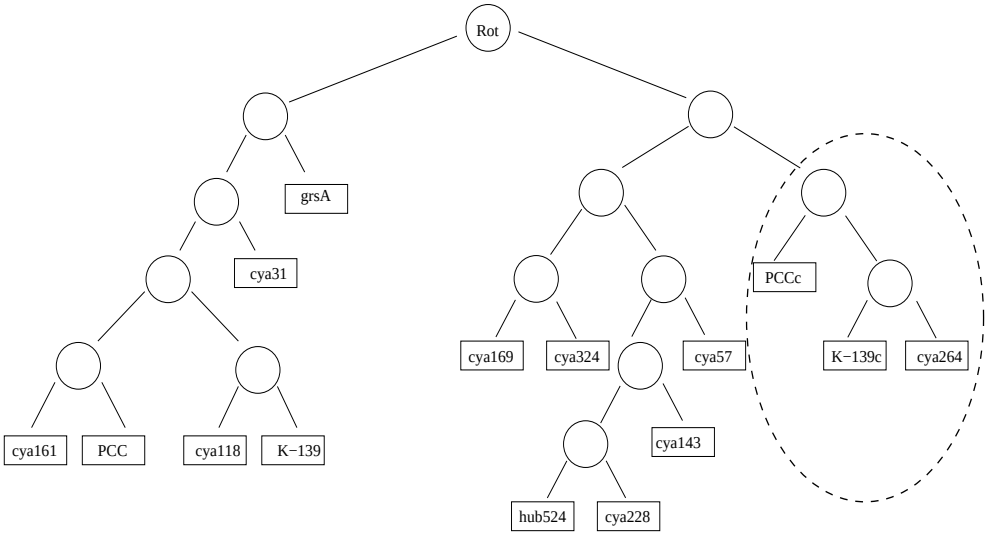
Figur 6.2: Variasjon i SPS-verdi for datasettet Anhydrase.

6.2 Ulike tellemåter for parvis sammenstilling

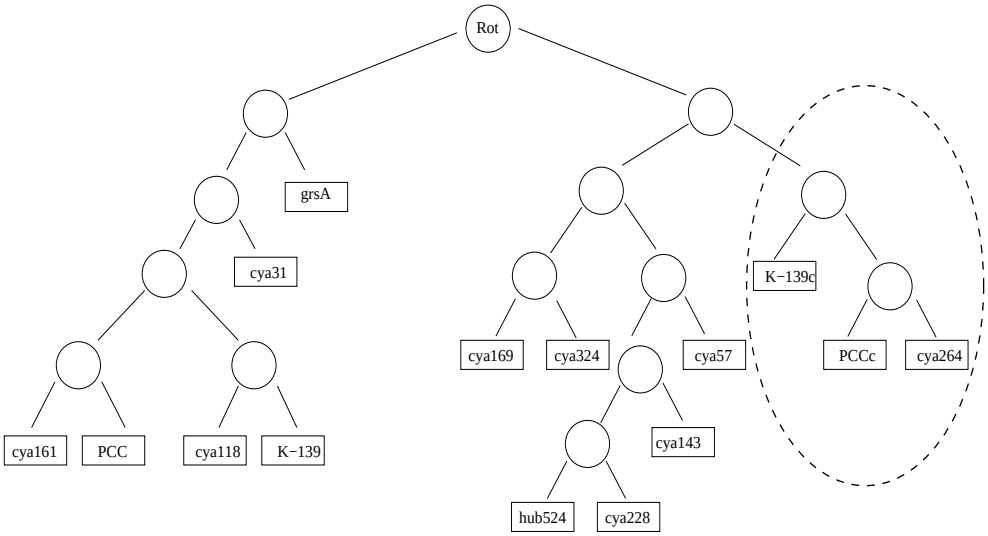
De to ulike tellemåtene for å sjekke likhet ved parvis sammenstilling testes ut på de tre datsettene, og for alle brukes den initielle metoden i Ittero. Resultater for datasettene:

- *mcyABC*
  Ved kodonlikhet spenner den parvise likheten fra 17.7% til 96.8% med gjennomsnitt på 56.4%. Når kodonene koder for samme aminosyre spenner likheten fra 40.0% til 98.1% med gjennomsnitt på 69.6%. Likheten når de koder for samme aminosyre ligger som ventet over kodonlikhet.

  Guidetrærne for de to ulike tellemåtene vises i figur 6.3. Figur 6.3a) viser kodonlikhet og figur 6.3b) viser aminosyrelikhet. Figuren viser at det er liten forskjell for de to guidetrærne, bare tre av sekvensene er stokket noe om. Dette er sannsyligvis ikke nok ulikhet til å påvirke sammenstillingen, og sammenstillingene er da også like.



(a) For like kodoner



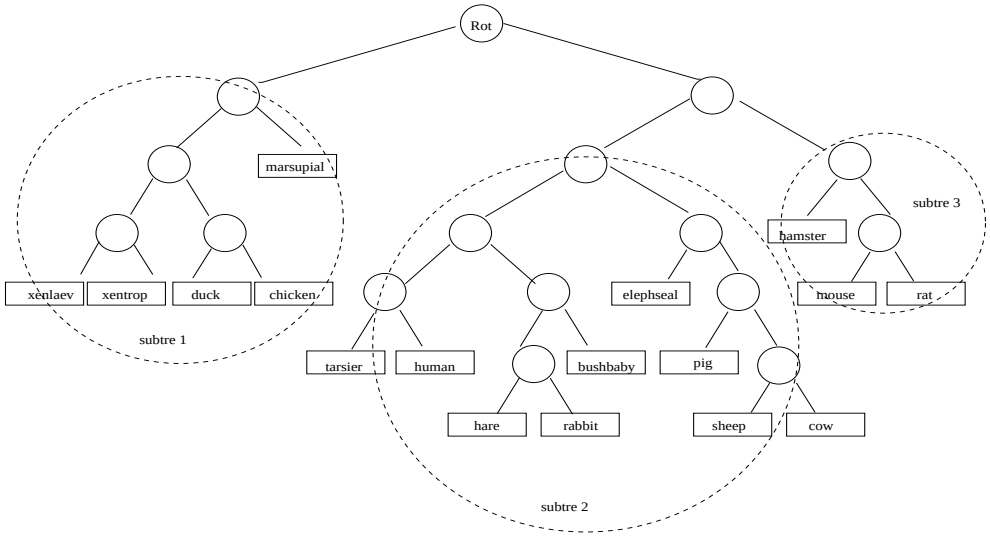
(b) For kodoner som koder for samme aminosyre

Figur 6.3: Figuren viser guidetrærne for *mcyABC* for de to ulike tellemåtene for parvis likhet. Tre (a) viser for like kodoner. Tre (b) viser når kodonene koder for samme aminosyre. De delene av trærne som er innringet, viser hvor de ikke er like.

- β-globin
  Datasettet testes med gapstraffene 10.0, 7.5, 5.0 og 2.5. Resultatene for de ulike gapstraffene:

  - gapstraffer 10.0:

Sekvensene har for kodonlikhet en likhet fra 16.0% til 93.1% med gjennomsnitt 49.5%. For aminosyrelikhet spenner likheten fra 46.5% til 97.2% med gjennomsnitt 71.3%. Begge metodene gir BAliScore-verdiene SPS-verdi=1.0 og TC-verdi=1.0. Guidetrærne for de to ulike metodene er også like noe som vises i figur 6.4



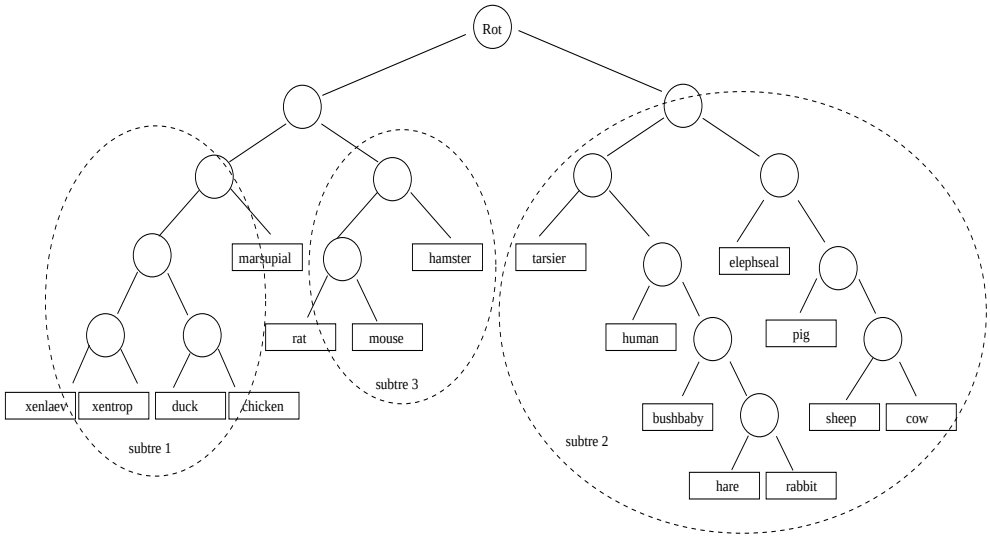
Figur 6.4: Figuren viser guidetreet for de to ulike tellemåtene for gapstraffer 10, 7.5 og 2.5 for datasettet β-globin

– gapstraff 7.5:

  Sekvensenes likhet er tilsvarende som for gapstraff på 10.0. Det samme gjelder guidetrærne, se figur 6.4. Verdiene fra BAliScore endret seg derimot fra gapstraff=10.0. For kodonlikhet har man nå SPS-verdi=0.995 og TC-verdi=0.963. For aminosyrelikhet har man SPS-verdi=0.995 og TC-verdi=0.979. At sekvensene har samme parvise likhet som for gapstraff=10.0, men ulike BAliScore-verdier kan tyde på at det ikke innføres gap ved de parvise sammenstillingene.

– gapstraffer 5.0:

  Også her er sekvensenes likhet tilsvarende som for gapstraff på 10.0. Guidetrærne for de to måtene skiller seg derimot fra hverandre. Guidetreet for aminosyrelikhet er likt som treet vist i figur 6.4. Guidetreet for kodonlikhet vises i figur 6.5. Forskjellen på de to trærne er at subtre 3 er flyttet fra siden av rota med subtre 2 til å være på samme side av roten som subtre 1. Den andre forskjellen er at innad i subtre 2 er ikke lenger sekvensene human og tarsi-



Figur 6.5: Figuren viser guidereet for kodonlikhet ved gapstraffer 5.0.

er samlet i en node. Her er verdiene fra BAliScore høyere for aminosyrelikhet enn for kodonlikhet. For kodonlikhet er SP-verdi=0.982 og TC-verdi=0.924. For aminosyrelikhet er SP-verdi=0.986 og TC-verdi=0.938.

- gapstraffer 2.5:
  Likheten for kodonlikhet spenner her fra 16.8% til 93.1% med gjennomsnitt 49.6%. For aminosyrelikhet har man likhet fra 46.5% til 97.2% med gjennomsnitt 71.4%. Guidetrærne er like som for gapstraff 10.0. BAliScore-verdiene har sunket noe fra gapstraff 5.0, men også her er verdiene for aminosyrelikhet høyere enn for kodonlikhet. For kodonlikhet er SPS-verdi=0.979 og TC-verdi=0.910. For aminosyrelikhet er SPS-verdi=0.983 og TC-verdi=0.924.

Ettersom de ulike gapstraffene ikke fører til stor forandring i likhet over sekvensene kan det se ut som om at gapstraffene ikke har noe særlig innvirkning på de parvise sammenstillingene.

- Anhydrase
  Dette datasettet gir likt resultatet for de to tellemåtene. Likheten spenner fra 19.5% til 31.9% med gjennomsnitt 25.4%. SPS- og TC-verdiene ble de samme for begge metodene med SPS-verdi=0.170 og TC-verdi=0.034. Begge metodene ga en sammenstilling med lengde 852 nukleotider.

Selv om det ser ut som at det er en fordel å benytte aminosyrelikhet ved gapstraffer under 10.0 for datasettet β-globin, velges det her kodonlikhet. Dette er en strengere form for likhet, men samtidig mister man ikke den informasjonen som finnes i at ulike kodoner koder for samme aminosyre. De sekvensene

som er mest like på kodonnivå, og ikke på aminosyrenivå, settes som mest like.

6.3 Sammenstillingene

Resultatene fra sammenstillingene vurderes på tre måter:

- Lengden av den endelige sammenstillingen
- Rekkefølgen i guidetreet
- De to verdiene fra BAliScore

Verdiene på parameterne til hver av modellene i Ittero oppgis også i en tabell.

6.3.1 *mcyABC*

Lengde

Lengden av sammenstillingene vises i tabell 6.1. Lengdene gis i nukleotider. Lengden på referansesammenstillingen er 969 nukleotider. Den av modellene

Modell	Lengde i nukleotider
Clustal	997
Referansesammenstilling	969
Initielle sammenstilling	1026
JC	1014
K2P	972
HKY85	1014
Goldman&Yang	1047
Muse&Gaut, kodon	1047
Muse&Gaut, base	1047

Tabell 6.1: Tabellen viser lengden av den endelige sammenstillingen for hver modell i Ittero og Clustal i nukleotider for datasettet *mcyABC*.

som kommer nærmest i lengde er K2P med 972 nukleotider. Deretter følger Clustal med 997 nukleotider, JC og HKY85 med 1014 nukleotider, den initielle med 1026 nukleotider og alle kodonmodellene med samme lengde, 1047 nukleotider.

Resultatene fra BAliScore

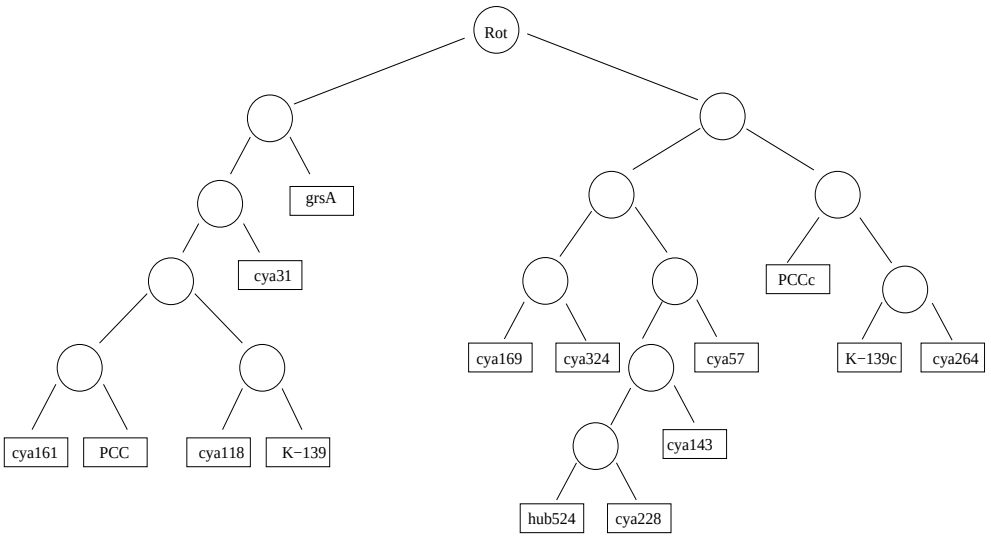
Resultatene fra BAliScore vises i tabell 6.2. Felles for alle modellene er at SPS- og TC-verdiene forandres ved første iterasjon. De multiple sammenstillingene endres altså etter den initielle sammenstillingen. Fra tabellen kan man se at bare K2P-modellen har høyere SPS-verdier enn Clustal, mens de andre nukleotidemodellene og kodonmodellene gir lavere verdier. Alle kodonmodellene gir samme verdi med SPS-verdi=0.833. For nukleotidemodellene er det K2P-modellen som kommer best ut for SPS-verdi, med SPS-verdi=0.900. JC og HKY85 er jevne for SPS-verdi med henholdsvis SPS-verdi=0.863 og SPS-verdi=0.865. Den initielle har SPS-verdi=0.856, bedre en kodonmodellene, men lavere enn Clustal og nukleotidemodellene. For TC-verdiene er forholdene omtrent de samme som for SPS-verdiene. Kodonmodellene gir også her dårligst verdier med TC-verdi=0.638 for alle. Clustal gir en TC-verdi på 0.683. HKY85 og JC har TC-verdi=0.678. Den initielle gir TC-verdi=0.654, også her lavere enn Clustal og nukleotidemodellene, men høyere enn kodonmodellene. K2P gir også den høyeste TC-verdien med TC-verdi=0.734.

	Clustal	Initiell	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
SPS	0.867	0.856	0.863	0.900	0.865	0.833	0.833	0.833
TC	0.683	0.654	0.678	0.734	0.678	0.638	0.638	0.638

Tabell 6.2: Tabellen viser BAliScore-verdiene for *mcyABC* datasettet.

Guidetreet

Alle modellene i Ittero gir lik rekkefølge på sekvensene i guidetreet. Guidetreet vises i figur 6.6. Guidetreet fra Clustal vises også i figur 6.6. Det uttegnede treet med rot er likt som treet for modellene i Ittero. Dette viser at guidetreet ikke kan være direkte årsak til variasjonene i SPS- og TC-verdier og variasjon i lengdene for de ulike modellene og Clustal.



(a) Guidetre fra Iterro og det uttegnede fra Clustal

(((((((cya118 K-139)(cya161 PCC))cya31)grsA)((cya264 K-139c)PCCc))
(cya169 cya324))((cya228 hub524)cya143)cya57)

(b) Guidetre fra Clustal

Figur 6.6: Figuren viser guidetrærne for *mcyABC*. Tre a) viser treet for alle modellene i Ittero og det uttegnede treet med rot fra Clustal. Tre b) viser guidetreet slik det oppgis i Clustal.

Parameterne

Hvordan de ulike parameterne i modellene endrer seg for iterasjonene etter den initielle sammenstillingen vises i tabell 6.3. I tabellen betyr - at denne parameteren ikke inngår i denne modellen. Fra tabellen kan man se at rate for gjennomsnittlig forandring, μ er konstant eller øker etter første iterasjon for de modellene som har denne parameteren. Transisjons- transversjonsraten κ forandres også, men denne synker for JC og HKY85 og øker for G&Y. Synonyme, ikke-synonyme raten ω for G&Y synker også. For de to M&G-modellene øker parameteren for synonym, α, fra 0.087 til 0.091. Parameterne for ikke-synonyme forandringer, β, er konstant på 0.041. Ettersom parameterne ikke forandres etter andre iterasjon har parameterne konvergert.

	Initielle	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
1.iterasjon	μ=0.056	0.056	0.059	0.057	0.056	-	-
	κ=0.788	-	0.683	0.783	0.827	-	-
	ω=0.469	-	-	-	0.456	-	-
	α=0.087	-	-	-	-	0.91	0.091
	β=0.041	-	-	-	-	0.041	0.041
2.iterasjon	μ	0.056	0.059	0.057	0.056	-	-
	κ	-	0.683	0.783	0.827	-	-
	ω	-	-	-	0.456	-	-
	α	-	-	-	-	0.091	0.091
	β	-	-	-	-	0.041	0.041

Tabell 6.3: Tabellen viser parameterne for de ulike modellene for *mcyABC*-datasettet. - betyr at denne parameteren ikke inngår i denne modellen.

6.3.2 β-globin

Dette datasettet testes med fire gapstraffer. Dette fordi alle gapstraffer høyere enn 10.0 gir helt like sammenstillinger som referansen. I forhold til referansesammenstillingen vil det være best å velge gapstraff så man ikke får innført gap i sekvensene. Det er likevel ikke interessant å se på for høye gapstraffer da det uansett datamateriale ikke vil innføres gap ved for høye gapstraffer. Her ses det istedet på hvor lavt man kan sette gapstraffene før det innføres gap i sekvensene og man får avvik fra referansesammenstillingen. Modellene i Ittero gir samme sammenstilling som referansesammenstillingen helt til man kommer ned i gapstraff på 2.5. For Clustal må man opp i gapstraff mellom 15.5 og 17.5 før det ikke lenger innføres gap i sekvensene. Gapstraffene som det ble testet på er 10.0, 7.5, 5.0 og 2.5.
Resultatene:

- For gapstraffer=10.0:

Lengde

Lengden av sammenstillingene vises i tabell 6.4. Tabellen viser at alle modellene i Ittero gir like lengder, 432 nukleotider. Dette er samme lengde som referansesammenstillingen og sier dermed at ingen gap innføres i sekvensene. Clustal har en lengde på 433 nukleotider noe som viser at det her innføres et gap i hver av sekvensene.

Modell	Lengde i nukleotider
Clustal	433
Referansesammenstilling	432
Initielle sammenstilling	432
JC	432
K2P	432
HKY85	432
Goldman&Yang	432
Muse&Gaut, kodon	432
Muse&Gaut, base	432

Tabell 6.4: Tabellen viser lengden av den endelige sammenstillingen for hver modell i nukleotider for datasettet β-globin og gapstraff 10.0.

Resultatene fra BAliScore

Resultatene fra BAliScore vises i tabellen 6.5. Tabellen viser at det bare er

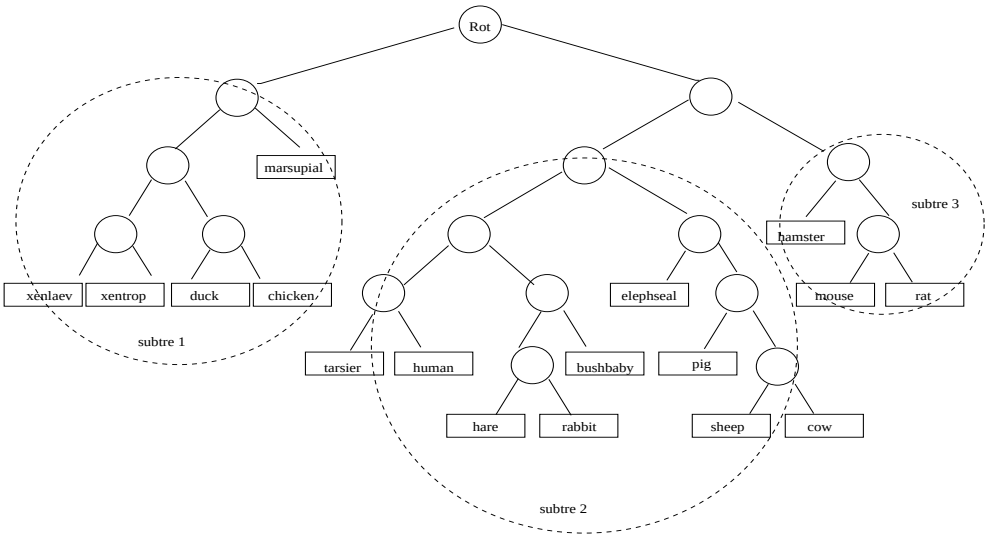
	Clustal	Initiell	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
SPS	0.998	1.00	1.00	1.00	1.00	1.00	1.00	1.00
TC	0.981	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Tabell 6.5: Tabellen viser verdiene fra BAliScore for datasettet β-globin, gapstraff=10.0.

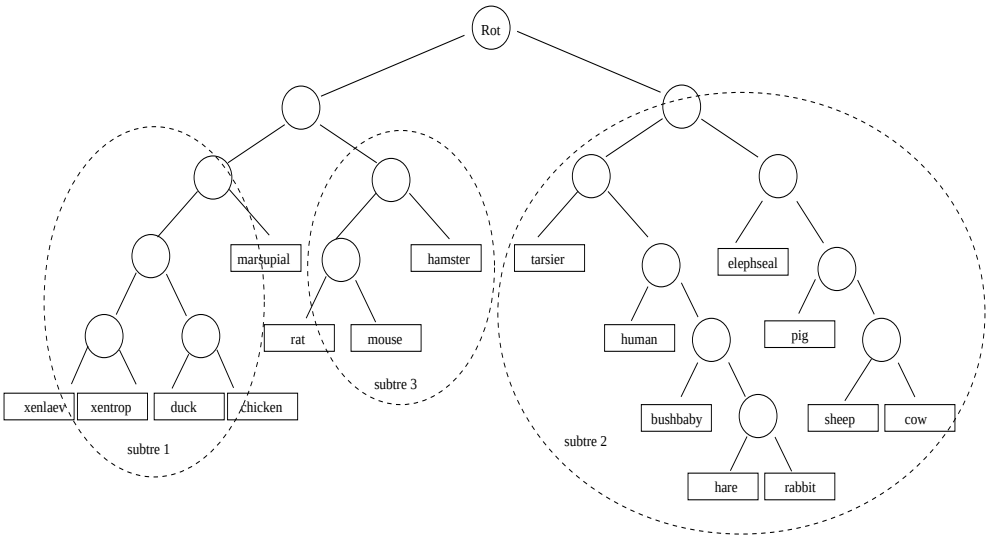
resultatet fra Clustal som avviker fra referansesammenstillingen. Modellene i Ittero får verdier på 1.0 for både SPS-verdi og TC-verdi, og det vil si at sammenstillingene herfra er helt like som referansesammenstillingen.

Guidetreet

Alle modellene i Ittero unntatt den initielle gir lik rekkefølge på sekvensene i guidetreet. Trærne vises i figur 6.7. De to trærne, for den initielle modellen og de andre modellene i Ittero, gir stort sett de samme grupperingene i subtrærne. Den største forskjellen for disse to guidetrærne er at subtre 3 er på hver sin side av rota. Det vil si at for den initielle vil disse sekvensene sammenstilles med subtre 2 i figuren, mens for de andre modellene vil de sammenstilles med sekvensene i subtre 1, før alle sammenstilles til slutt. I tillegg er rekkefølgen i subtre 2 noe ulik for de to trærne da sekvensene human og tarsier sammenstilles i den initielle



(a) Guidetre for initiell modell i Ittero



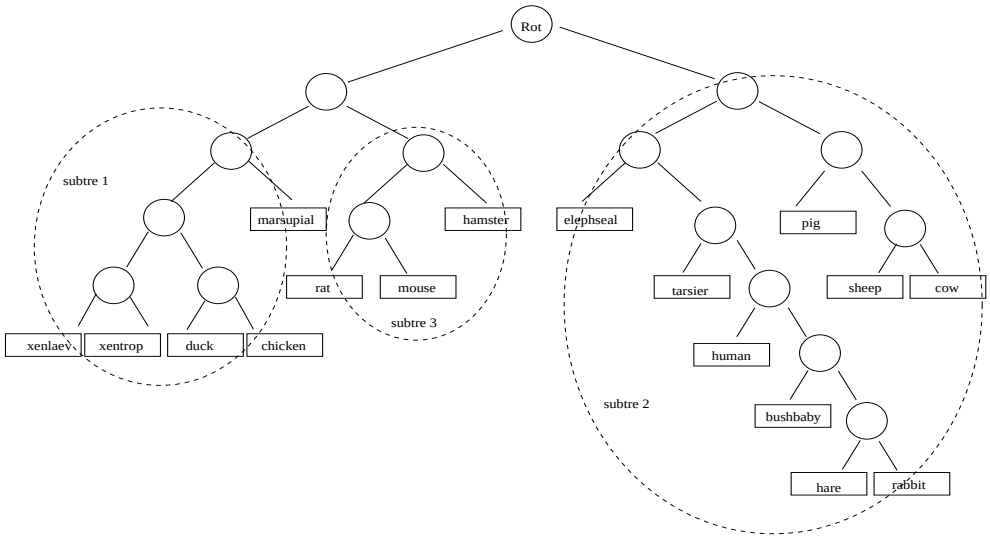
(b) Guidetre for de andre modellene i Ittero

Figur 6.7: Figuren viser guidetrærne for βglobin. Tre a) viser treet for den enkle modellene i Ittero. Tre b) viser guidetreet for den andre modellene i Ittero.

modellen, mens i de andre modellene sammenstilles human først med de andre sekvensene i subtreet, deretter sammenstilles tarsier med denne sammenstillingen. Guidetreet fra Clustal vises i figur 6.8. Dette treet har en forskjell fra guidetreet for modellene i Ittero unntatt den initielle. Denne forskjellen er at elephseal flyttes innad i subtre 2.

(((((((chicken duck)(xenlaev xentrop))marsupial)((mouse rat) hamster))
((cow sheep) pig)) elephseal)(((hare rabbit) bushbaby) human) tarsier)

(a) Guidetre fra Clustal



(b) Guidetre fra Clustal, uttegnet med rot

Figur 6.8: Figuren viser guidetreet for Clustal slik det oppgis i programmet (a), og uttegnet med rot (b).

Parametre

Tabell 6.6 viser endringene i parameterene etter iterasjonene. Parameterene endres etter første iterasjon, men etter andre iterasjon er det ingen forandring. Tabellen viser og at parameterne er de samme uansett modell.

	initielle	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
1.iterasjon	μ = 0.119	0.122	0.122	0.122	0.122	-	-
	κ=1.046	-	1.019	1.019	1.019	-	-
	ω=0.320	-	-	-	0.320	-	-
	α=0.243	-	-	-	-	0.242	0.242
	β=0.078	-	-	-	-	0.077	0.077
2.iterasjon	μ	0.122	0.122	0.122	0.122	-	-
	κ	-	1.019	1.019	1.019	-	-
	ω	-	-	-	0.320	-	-
	α	-	-	-	-	0.242	0.242
	β	-	-	-	-	0.077	0.077

Tabell 6.6: Tabellen viser parameterne for de ulike modellene for datasettet β-globin med gapstraff 10.0

- For gapstraffer=7.5:

Lengde

Sammenstillingenes lengder vises i tabell 6.7. Tabellen viser at for Clus-

Modell	Lengde i nukleotider
Clustal	440
Referansesammenstilling	432
Initielle sammenstilling	435
JC	432
K2P	432
HKY85	432
Goldman&Yang	432
Muse&Gaut, kodon	432
Muse&Gaut, base	432

Tabell 6.7: Tabellen viser lengden av den endelige sammenstillingen for hver modell i Ittero i nukleotider for datasettet β-globin og gapstraff 7.5..

talsammenstillingen øker lengden i forhold til gapstraff=10.0. Lengden på den initielle sammenstillingen øker også, men ikke lengdene for de andre modellene i Ittero.

Resultatene fra BAliScore

Resultatene fra BAliScore vises i tabell 6.8. I forhold til gapstraff på 10.0 er SPS- og TC-verdiene for Clustal nå enda lavere. Her er de henholdsvis 0.983 og 0.900. I tillegg innføres det i den initielle sammenstillingen gap, da SPS- og TC-verdiene for denne er lavere enn 1.0.

	Clustal	Initiell	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
SPSS	0.983	0.995	1.00	1.00	1.00	1.00	1.00	1.00
TC	0.900	0.963	1.00	1.00	1.00	1.00	1.00	1.00

Tabell 6.8: Tabellen viser verdiene fra BAliScore for datasettet β-globin og gapstraff 7.5.

Guidetreet

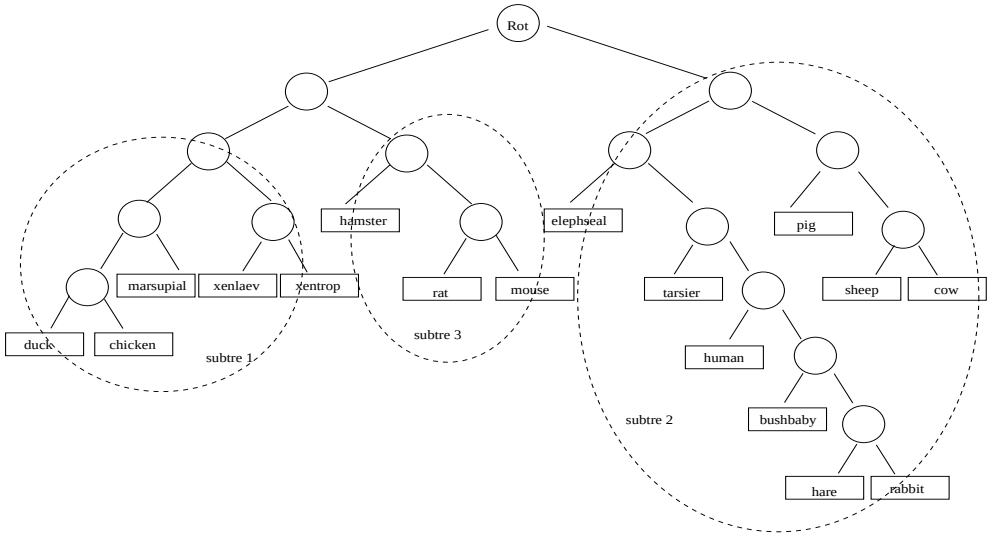
Alle modellene i Ittero, unntatt den initielle, gir lik rekkefølge på sekvensene i guidetreet. Guidetrærne er de samme som ved gapstraffer på 10.0. Disse guidetrærne vises i figur 6.7(a) og (b). Guidetreet fra Clustal endres noe fra gapstraff 10.0 og vises i figur 6.9. Guidetreet fra Clustal skiller seg fra treet for Itteromodellene på to steder i treet. Innad i subtre 2 flyttes elephseal som for gapstraff 10.0. Innad i subtre 1 sammenstilles nå marsupial med duck-chicken noden, før denne noden igjen sammenstilles med xenlaev-xentrop noden.

Parametre

Parameterene ved denne gapstraffen endrer seg på samme måte, og har like verdier, som for gapstraff=10.0. Tabell 6.6 viser at parameterene forandres etter den første iterasjonen, og at det ikke er noen forandringer etter den andre iterasjonen. Også her er parameterne de samme i alle modellene de inngår i.

(((((((chicken duck) marsupial)(xenlaev xentrop))((mouse rat) hamster))
((cow sheep) pig)) elephseal)(((hare rabbit) bushbaby) human) tarsier)

(a) Guidetre fra Clustal



(b) Guidetre fra Clustal, uttegnet med rot

Figur 6.9: Figuren viser guidetreet fra Clustal for datasettet β-globin og gapstraff 7.5.

- For gapstraffer=5.0:

Lengde

Lengdene til sammenstillingene vises i tabell 6.9. Fra tabellen kan man se at lengden øker ytterligere for både Clustalsammenstillingen og den initielle sammenstillingen fra Ittero. For nukleotidemodellene og kodonmodellene i Ittero innføres det heller ikke her gap i sekvensene.

Resultatene fra BAliScore

BAliScore-verdiene vises i tabell 6.10. Fra tabellen kan man og se at det i nukleotide- og kodonmodellene fra Ittero ikke innføres gap, mens SPS- og TC-verdiene for Clustal og den initielle fra Ittero er enda lavere enn ved gapstraff på 7.5.

Modell	Lengde i nukleotider
Clustal	452
Referansesammenstilling	432
Initielle sammenstilling	441
JC	432
K2P	432
HKY85	432
Goldman&Yang	432
Muse&Gaut, kodon	432
Muse&Gaut, base	432

Tabell 6.9: Tabellen viser lengden av den endelige sammenstillingen for hver modell Ittero og Clustal i nukleotider for datasettet β-globin og gapstraff 5.0.

	Clustal	Initiell	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
SPS	0.957	0.982	1.00	1.00	1.00	1.00	1.00	1.00
TC	0.808	0.924	1.00	1.00	1.00	1.00	1.00	1.00

Tabell 6.10: Tabellen viser verdiene fra BAliScore for datasettet β-globin for gapstraff=5.0.

Guidetreet

Alle modellene unntatt den initielle, har lik rekkefølge på sekvensene i guidetreet. Guidetrærne er de samme som ved gapstraffer på 10.0, og vises i figur 6.7(a) og (b). Guidetreet fra Clustal vises i figur 6.10. Det uttegnede treet med rot er likt som i figur 6.8b). Guidetrærne forandres altså ikke ved å senke gapstraffene ytterlig.

Parametre

Tabell 6.6 viser også parameterne for gapstraff=5.0 da de er like som for de foregående gapstraffene på 10.0 og 7.5.

(((bushbaby (hare rabbit)) human) ((((((chicken duck) (xenlaev xentrop)) marsupial)
(hamster (mouse rat))) ((cow sheep) pig)) elephseal) tarsier)

Figur 6.10: Figuren viser guidetreet fra Clustal for datasettet β-globin og gapstraff 5.0.

- For gapstraffer=2.5:

Lengde

Lengden av sammenstillingene vises i tabell 6.11. Her ser man at det for alle modellene i Ittero nå innføres gap i sekvensene da alle har en lengde større enn 432 nukleotider. Nukleotidemodellen K2P kommer best ut med en lengde på 435 nukleotider. Det innføres her et gap av lengde ett kodon i hver av sekvensene.

Modell	Lengde i nukleotider
Clustal	490
Referansesammenstilling	432
Initielle sammenstilling	444
JC	438
K2P	435
HKY85	441
Goldman&Yang	441
Muse&Gaut, kodon	441
Muse&Gaut, base	441

Tabell 6.11: Tabellen viser lengden av den endelige sammenstillingen for hver modell i Ittero og Clustal i nukleotider for datasettet β-globin og gapstraff 2.5.

Resultatene fra BAliScore

Resultater fra BAliScore vises i tabell 6.12. BAliScore-verdiene understreker at det nå innføres gap i sekvensene da alle verdiene er lavere enn 1.0. I samsvar med lengden av sammenstillingen er det nukleotidemodellen K2P fra Ittero som har de høyeste BAliScore-verdiene. Dette viser også at man kan tillate lave gapstraffer før man får innført gap med modellene i Ittero for dette datasettet.

	Clustal	Initiell	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
SPS	0.899	0.979	0.991	0.995	0.982	0.987	0.987	0.987
TC	0.625	0.910	0.965	0.979	0.924	0.951	0.951	0.951

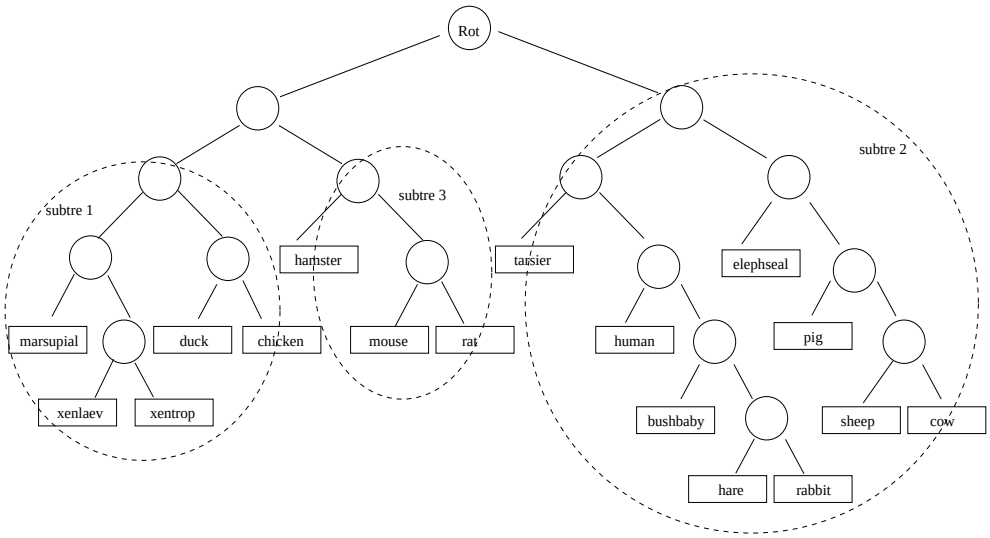
Tabell 6.12: Tabellen viser verdiene fra BAliScore for datasettet β-globin og gapstraff 2.5.

Guidetreet

Alle modellene unntatt den initielle gir lik rekkefølge på sekvensene i guidetreet. Guidetrærne er de samme som ved gapstraffer på 10.0 og vises i figur 6.7(a) og (b). Guidetreet fra Clustal vises i figur 6.11. Guidetreet fra

((((((chicken duck)(marsupial(xenlaev xentrop)))((mouse rat)hamster))
(elephseal((cow sheep)pig)))(((hare rabbit)bushbaby)human)tarsier)

(a) Guidetre fra Clustal



(b) Guidetre fra Clustal, uttegnet med rot

Figur 6.11: Figuren viser guidetreet fra Clustal for datasettet β-globin og gapstraff 2.5.

Clustal skiller seg fra guidetreet for modellene i Ittero ved at marsupial i subtre 1 sammenstilles med xenlav-xentrop noden i Clustal-treet, mens i Ittero-treet sammenstilles marsupial med noden for alle sekvensene i dette subtreet. Subtre 2 er nå likt som subtre 2 for modellene i Ittero unntat den initielle for gapstraff 10.0, se figur 6.7(b).

Parametre

Parameterene for de ulike modellene i Ittero vises i tabell 6.13. Som ved de foregående får man her og en forandring etter første iterasjon, men ikke etter andre. Forskjellen fra tidligere er at her er ikke parameterene de samme for alle modellene. For eksempel er transisjons- transversjons-raten 0.997 for K2P-modellen og 1.013 for G&Y-modellen.

	initielle	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
1.iterasjon	μ = 0.119	0.121	0.121	0.120	0.120	-	-
	κ=1.011	-	0.997	1.011	1.013	-	-
	ω=0.312	-	-	-	0.317	-	-
	α=0.241	-	-	-	-	0.241	0.241
	β=0.075	-	-	-	-	0.076	0.076
2.iterasjon	μ	0.121	0.121	0.120	0.120	-	-
	κ	-	0.997	1.011	1.013	-	-
	ω	-	-	-	0.317	-	-
	α	-	-	-	-	0.241	0.241
	β	-	-	-	-	0.076	0.076

Tabell 6.13: Tabellen viser parameterne for de ulike modellene i Ittero for datasettet β-globin med gapstraff 2.5.

6.3.3 Anhydrase

For dette datasettet konvergerer ikke Ittero med JC-modellen. Modellen M&G nukleinsyre konvergerer etter tre iterasjoner, mens de andre modellene konvergerer etter en som for de andre datasettene.

Datasettet testes og ut med gapstraff på 55.0 da dette er i området der den initielle har en høy SPS-verdi, men disse resultatene er mye dårligere enn ved gapstraff på 9.0, og presenteres ikke.

Lengde

Sammenstillingenes lengder vises i tabell 6.14. Lengden gis i nukleotider. Fra

Modell	Lengde i nukleotider
Clustal	935
Referansesammenstilling	945
Initielle sammenstilling	852
JC	-
K2P	762
HKY85	843
Goldman&Yang	771
Muse&Gaut, kodon	771
Muse&Gaut, base	831

Tabell 6.14: Tabellen viser lengden av den endelige sammenstillingen for hver modell i Ittero og Clsutal i nukleotider for datasettet Anhydrase.

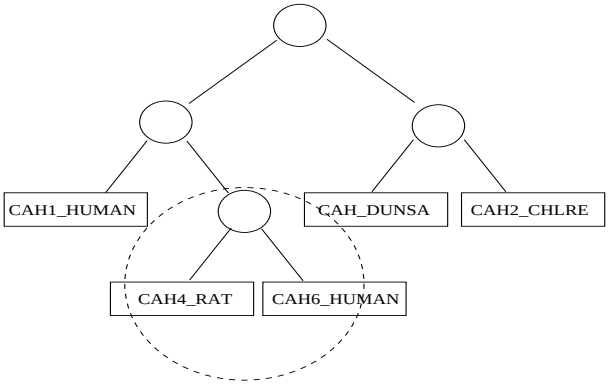
tabellen kan man se at alle modellene i Iterro gir en sammenstilling som er kortere enn referansesammenstillingen. Referansesammenstillingen har en lengde på 945 nukleotider, mens Ittero-modellene gir lengder mellom 771 og 879 nukleotider. Sammenstillingen fra Clustal er også kortere enn referansesammenstillingen med en lengde på 935 nukleotider.

Guidetrærne

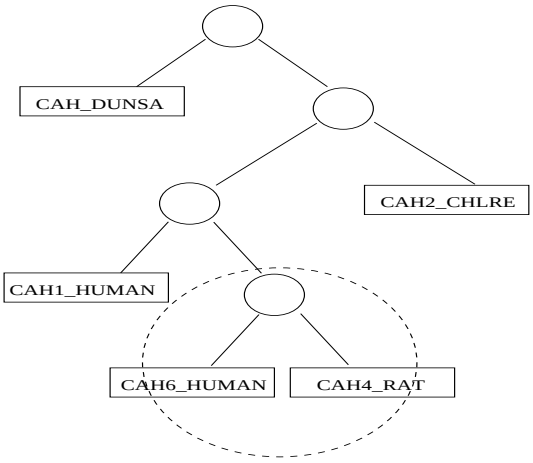
Dette datasettet resulterer i endel ulike guidetrær. Disse vises i figurene 6.12 til 6.16. For modellene i Ittero gir G&Y og M&G aminosyre, og den initielle og M&G nukleinsyre, det samme treet, de andre gir ulike trær. For alle trærne unntatt treet fra Clustal og treet for K2P-modellen er sekvensene P48284 og P23280 samlet i en node. Bortsett fra dette er det ingen spesielle mønster å se i trærne. Treet fra Clustal er ikke tegnet ut med rot ettersom det ikke er noe mønster hos trærne fra Ittero.

((CAH_DUNSA HAH2_CHLRE) (CAH4_RAT CAH1_HUMAN) CAH6_HUMAN)

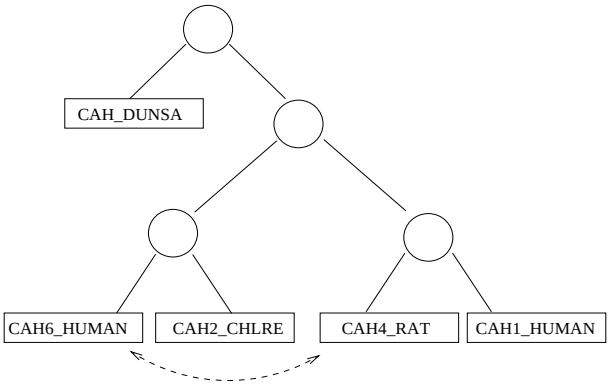
Figur 6.12: Figuren viser guidetreet fra Clustal for datasettet Anhydrase.



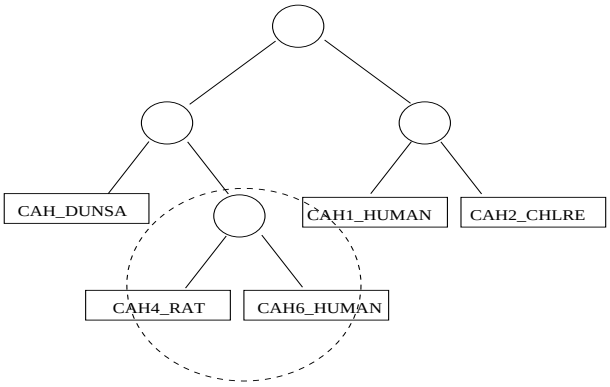
Figur 6.13: Figuren viser guidetreet for den initielle modellen og M&G nukleinsyre for datasettet Anhydrase.



Figur 6.14: Figuren viser guidetreet for modellen HKY85 for datasettet Anhydrase.



Figur 6.15: Figuren viser guidetreet for modellen K2P for datasettet Anhydrase.



Figur 6.16: Figuren viser guidetreet for modellene G&Y og M&G aminosyre for datasettet Anhydrase.

BAliScore verdiene

SPS-verdiene og TC-verdiene fra BAliScore vises i tabell 6.15. Tabellen viser at det er den initielle modellen som kommer best ut for SPS-verdien med SPS-verdi=0.170. I forhold til de to andre datasettene er dette en lav verdi, men sekvensene her er mindre like enn sekvensene i de to andre datasettene. Etter den initielle følger HKY85-modellen med en SPS-verdi=0.126. Denne modellen har og den høyeste TC-verdien, nemlig TC-verdi=0.046. Disse to modellene er de eneste som har høyere verdier enn Clustal. Clustal gir en SPS-verdi=0.090. I motsetning til de andre datasettene kommer modellen K2P her forholdsvis dårlig ut med SPS-verdi=0.082. Dette er dårligere enn både G&Y og M&G aminosyre som begge har SPS-verdi=0.087. M&G nukleinsyre er den modellen som er dårligst med SPS-verdi=0.062.

	Clustal	Initiell	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
SPS	0.090	0.170	-	0.082	0.126	0.087	0.087	0.062
TC	0.044	0.034	-	0.000	0.046	0.000	0.000	0.000

Tabell 6.15: Tabellen viser verdiene fra BAliScore for datasettet Anhydrase.

Parameterne

Parameterne for JC er tatt med i resultatene her selv om denne ikke konvergerer. Rate for gjennomsnittlig forandring, μ, varierer mellom 0.396 og 0.397. Dette er ikke stor forskjell, men modellen konvergerer uansett ikke. Parameterne for de ulike modellene i Ittero vises i tabell 6.16. Her kan man se at for alle modellene som inneholder μ, øker denne etter den initielle sammenstillingen. Transisjons- transversjonsraten går noe ned, mens synonyme, ikke-synonyme raten for G&Y-modellen går forholdsvis mye opp, fra 1.481 til 2.816. For de to M&G-modellene øker også rate for synonym forandring og rate for ikke-synonym forandring.

	initielle	JC	K2P	HKY85	G&Y	M&G aminosyre	M&G nukleinsyre
1.iterasjon	μ = 0.367	0.391	0.427	0.385	0.430	-	-
	κ=0.545	-	0.443	0.543	0.508	-	-
	ω=1.481	-	-	-	2.816	-	-
	α=0.166	-	-	-	-	0.101	0.112
	β=0.246	-	-	-	-	0.283	0.274
2.iterasjon	μ	0.396	0.427	0.385	0.430	-	-
	κ	-	0.443	0.543	0.508	-	-
	ω	-	-	-	2.816	-	-
	α	-	-	-	-	0.101	0.126
	β	-	-	-	-	0.283	0.273
3.iterasjon	μ	0.397				-	-
	κ	-				-	-
	ω	-	-	-		-	-
	α	-	-	-	-		0.123
	β	-	-	-	-		0.274
4.iterasjon	μ	0.396				-	-
	κ	-				-	-
	ω	-	-	-		-	-
	α	-	-	-	-		0.123
	β	-	-	-	-		0.274

Tabell 6.16: Tabellen viser parameterne for de ulike modellene i Ittero for datasettet Anhydrase. De tomme rutene betyr modellene har konvergert.

Kapittel 7

Drøftinger og konklusjon

I dette kapittelet drøftes resultatene i kapittel 6, og litt om sammenstillingprogrammer og -algoritmer. Siden jeg ikke finner noe litteratur rundt bruk av evolusjonære modeller i multippel sammenstilling er det ikke mulig å sammenlikne resultatene med tilsvarende arbeider. Jeg har søkt etter artikler i flere av journaler som dekker bioinformatikk og også Google og Bibsys.

Når man skal lage en multippel sammenstilling av et gitt sett sekvenser, har man ikke en referansesammenstilling av sekvensene. SPS- og TC-verdier kan dermed ikke benyttes. Disse verdiene brukes til å sjekke hvor bra ulike sammenstillingsprogrammer gjør det i forhold til en referansesammenstilling. Når man lager multiple sammenstillinger kan man se på for eksemple lengden av sammenstillingen i forhold til lengden av sekvensene man sammenstiller. Hvis sammenstillingen har en veldig høy verdi for lengden i forhold til lengden av sekvensene, er det introdusert mange gap i sekvensene. Slike sammenstillinger er ikke realistiske, Nei & Kumar (2000). En annen verdi som kan benyttes er Sum of Pairs, Kilpelainen (2004). Man ser da på hvor mange like som er sammenstilt i forhold til hvor mange tegn som er sammenstilt totalt. Dette tallet vil man da ha så høyt som mulig. Disse verdiene er ikke et mål på hvor korrekt en sammenstilling er, men de sier noe om hvor mange gap som er innført og hvor mange like tegn som er sammenstilt.

7.1 Ulike sammenstillingsprogrammer og -algoritmer

Noen programmer og algoritmer for multippel sammenstilling presenteres i oppgaven. Det finnes mange flere enn disse, men de som er presentert her, er hovedtypene, Notredame (2001). Artikkelen Thompson, Plewniak & Poch (1999*b*) undersøker 10 multippel sammenstilling-prgrammer. De bruker aminosyresekvenser fra BAliBASE som testsekvenser og også BAliScore som sammenlikningsverdier. Artikkelen konkluderer med at det ikke finnes en metode eller program som er det beste for multippel sammenstilling. Hvilken frem-

gangsmåte som er den beste er avhengig av sekvensene man skal sammenstille. Artikkelen finner at det er fire programmer som er de som generellt sett gir de beste resultatene. Tre av disse bruker en iterativ fremgangsmåte, det fjerde programmet er Clustal. Testene viser at iterative fremgangsmåter har mange fordeler, men de viser og at fremgangsmåten kan være ustabil. De tre iterative programmene er Prrp, Saga og DiAlign, Thompson, Plewniak & Poch (1999*b*). Programmene bruker ulike strategier for å forbedre sammenstillinger, noe som viser at det er flere måter å gjøre ting iterativt på som fungerer. Artikkelen sier i tillegg at det kan være en fordel å bruke flere programmer basert på ulike teknikker når man skal lage en multippel sammenstilling, for så å sammenlikne sammenstillingene.

Ulike algoritmer for å danne et tre

I forbindelse med algoritmene i Clustal ses det og på ulike måter for å danne et tre. I Clustal og Ittero dannes guidetreet som benyttes for å lage den multiple sammenstillingen ved distansemetoden Neighbour Joining. De andre metodene som presenteres her er Maximum Likelihood og Maximum Parsimony. Distansemetoder benyttes selv om begge de andre metodene vurderer alle trærne og finner det beste, mens distansemetoder ikke vurderer alle trær, men danner et tre. Distansemetoder brukes fordi de er mye raskere metoder.

7.2 Generellt om resultatene

Ittero tar foreløpig ikke hensyn til faktorene Clustal bruker for å forbedre resultatet, slik som sekvensvekting og posisjonsspesifikke gapstraffer, Thompson, Higgins & Gibson (1994*a*). Resultatene fra Ittero er likevel bra i forhold til resultatene fra Clustal, og det ser dermed ut som om det er positivt å bruke modeller for evolusjonær forandring i utregning av substitusjonsmatrise. Samtidig er det vanskelig å si om det er modellene i Ittero som fører til det gode resultatet eller om det skyldes like mye at Ittero bruker kodonet som enheten i motsetning til nukleotidet hos Clustal. Det at den initielle modellen i Ittero også gjør det bra i forhold til Clustal kan tyde på dette. Det hadde dermed vært interessant å innføre de ekstra forbedringene Clustal bruker i Ittero for å se hva som ville skjedd med resultatene da.

For å avgjøre om ulikhetene mellom modellene og Clustal er signifikante kan man bruke Friedman test som brukes i artikkelen Edgar (2004). Dette er derimot ikke mulig her ettersom det ikke brukes mange nok datasett, noe som skyldes at det har vært svært vanskelig å skaffe datasett på kodende nukleinsyrenivå der det finnes en referansesammenstilling. Dette er et generellt problem ved sammenlikning av programmer og metoder.

> The lack of a standard set of preference alignments has meant that existing programs could not be benchmarked and the increase in performance by the new iterative alignment methods could not be accurately measured.
>
> — *Thompson, Plewniak & Poch (1999b)*

7.2.1 De to ulike tellemåtene for parvis likhet

Ved å bruke kodonet som enhet finnes det to tellemåter for likhet mellom sammenstillingen som er funnet og referansesammenstillingen:

- Likhet ved like kodoner i de to sekvensene
- Likhet ved at de kodonene i de to sekvensene koder for samme aminosyre

Datamaterialet som brukes gir ikke noe klart svar på hvilken av de to tellemåtene som er den beste. For datasettene *mcyABC* og Anhydrase gir begge måtene likt resultat. For datasettet βglobin gir aminosyrelikhet best resultat. For gapstraffer på 10.0 er det ingen forskjell da ingen av måtene innfører gap i sekvensene. Ved de andre gapstraffene er det derimot forskjell. For gapstraff 7.5 gir aminosyrelikhet en høyere TC-verdi enn for kodonlikhet, og for gapstraffer på 5.0 og 2.5 er både SPS-verdien og TC-verdien høyere for aminosyrelikhet enn kodonlikhet. For dette datasettet kan det dermed lønne seg å bruke aminosyrelikhet ved lave gapstraffer da dette gir en bedre tilnærming til referansesammenstillingen. Kodonlikhet brukes i oppgaven selv om det for gapstraffer 5.0 og 2.5 for datasettet β-globin gir høyere verdier med aminosyrelikhet. Ved å bruke kodonlikhet mister man ikke den informasjonen som er i det at man har ulike kodoner for samme aminosyre, og kodonet er enhet for programmet Ittero, dermed velges kodonlikhet.

7.2.2 Valg av gapstraffer

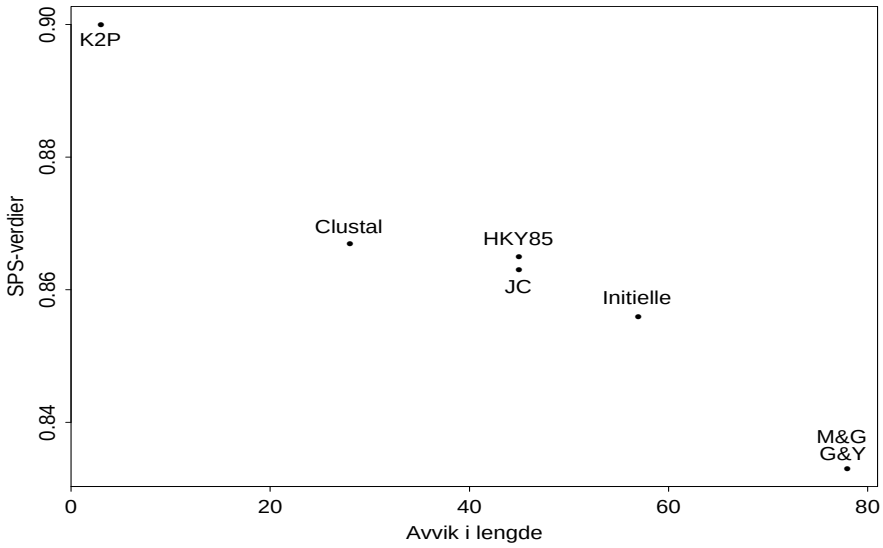
Gapstraffer velges for to av datasettene ut fra SPS-verdi ved bruk av BAliScore og en referansesammenstilling. Som oftest har man ikke en referansesammenstilling å forholde seg til når man skal lage en sammenstilling, og denne metoden kan altså ikke benyttes. Det er heller ikke lett å si noe her om resultatene fra denne måten å gjøre det på. For datasettet β-globin øker SPS-verdi med økende gapstraff, men det er ikke overraskende da referansesammenstillingen ikke inneholder gap, og jo høyere gapstraff, jo mindre sannsynlig er det at gap introduseres. For å kunne si noe generellt om valg av gapstraff måtte man her og hatt flere datasett. Gapstraffer har ikke vært en sentral del av oppgaven, og er dermed ikke tillagt stor vekt i resultatdelen.

7.2.3 Iterasjon

Tanken bak iterasjon over en progressiv algoritme er å bedre parameterne. Ettersom den første sammentillingen lages med en substitusjonsmatrise som ikke tar hensyn til noen biologiske faktorer er tanken at man må få bedre sammenstillinger og parametre etter iterasjon. Resultatene viser at parameterne forandrer seg etter den initielle iterasjonen. Parameterne konvergerer etter andre iterasjon for alle datasettene og modellene med unntak av datasettet Anhydrase. Her konvergerer M&G nukleinsyre-modellen etter tre iterasjoner og JC-modellen konvergerer ikke. At en modell ikke konvergerer kan ses som et tegn på at denne modellen ikke er god for datasettet og bør dermed ikke brukes.

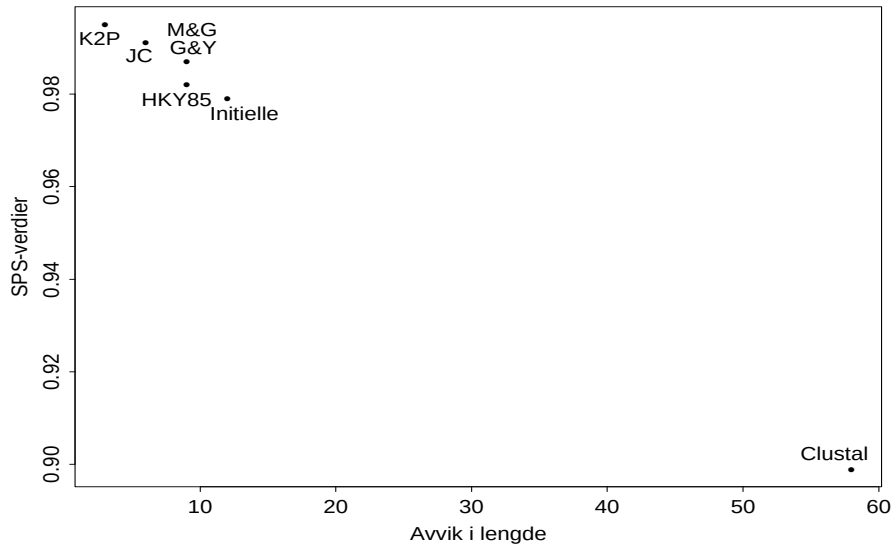
7.2.4 Sammenstillingenes lengde

Lengden av en sammenstilling sier først og fremst noe om hvor mange gap som er introdusert i hver sekvens. En sammenstilling med mange gap og få sammenstilte er generellt sett ikke en bra sammenstilling, Nei & Kumar (2000). Dette må her ses i sammenheng med lengden på referansesammenstillingen. Et spørsmål er om man kan si noe om hvor bra en sammenstilling er i forhold til hvor stort avviket i lengde er i forhold til referansesammenstillingen. For å se på dette plottes avvik i lengde mot SPS-verdiene. Figur 7.1 viser dette for datasettet *mcyABC*. Figuren viser at det er en tilnærmet monotont synken-



Figur 7.1: Figuren viser avviket i lengde plottet mot SPS-verdi for datasettet *mcyABC*.

de sammenheng mellom avvik i lengde og SPS-verdi. Figur 7.2 viser forholdet mellom avvik i lengde og SPS-verdier for datasettet β-globin ved gapstraff 2.5. Også her er det en tilnærmet monotont synkende sammenheng. I figur 7.1 og 7.2 viser M&G til både M&G aminosyre-modellen og M&G nukleinsyre-modellen da disse gir samme resultat. For disse to datasettene er alle sammenstillinge-

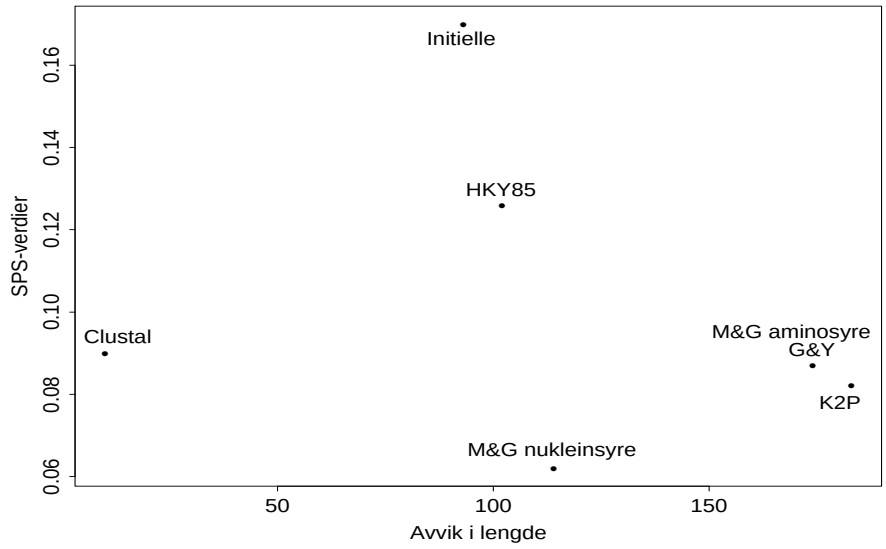


Figur 7.2: Figuren viser avviket i lengde plottet mot SPS-verdi for datasettet β-globin

ne lenger enn referansesammenstillingen. Dette er ikke tilfellet for datasettet Anhydrase, her er alle sammenstillingene kortere enn referansesammenstillingen. For dette datasettet er det ingen klar sammenheng mellom avvik i lengde og SPS-verdi. Dette vises i figur 7.3. For dette datasettet er også avvikene større enn for de to andre datasettene, og dette kan være årsaken til at det ikke er en sammenheng her. Det er også mulig at sammenstillingen ikke er god for datsettet. Sammenstillingen er hentet fra BAliBASE og de tilsvarende nukleinsyresekvensene funnet ved hjelp av SwissProt. Det er mulig at det ikke er optimalt å lage en sammenstilling for nukleinsyrer ut fra dens sammenstilling av aminosyrer da det blir mange ulike nukleotider ved så lav likhet over sekvensene.

7.2.5 BAliScore-verdiene

Ettersom man ikke kjenner den evolusjonære historien til et sett av sekvenser, vet man heller ikke hva som er den rette sammenstillingen for sekvensene. Det at man sjekker en sammenstilling mot en referansesammenstilling krever at



Figur 7.3: Figuren viser avviket i lengde plottet mot SPS-verdi for datasettet Anhydrase.

man går ut fra at referansesammentillingen er en bra sammenstilling for disse sekvensene. Referansesammenstillingen som brukes her er laget av noen som kjenner sekvensene, og dette gjør at vi velger å tro at sammenstillingene de har konkludert med er gode sammenstillinger.

SPS-verdien er den verdien det er lagt mest vekt på fordi den angir hvor lik resultatsammenstillingen er referansesammenstillingen med hensyn på hvor mange av sekvensene som er riktig sammenstilt. TC-verdien vil ikke være spesielt informativ i enkelte tilfeller, for eksempel hvis en av sekvensene er helt feil sammenstilt i forhold til referansesammenstillingen. I dette tilfellet vil TC-verdien være null selv om resten av sammenstillingen kan være bra, Lassmann & Sonnhammer (2002). SPS-verdien påvirkes ikke på denne måten da de riktig sammenstilte sekvensene gir uttelling uavhengig av en feilsammenstilt sekvens.

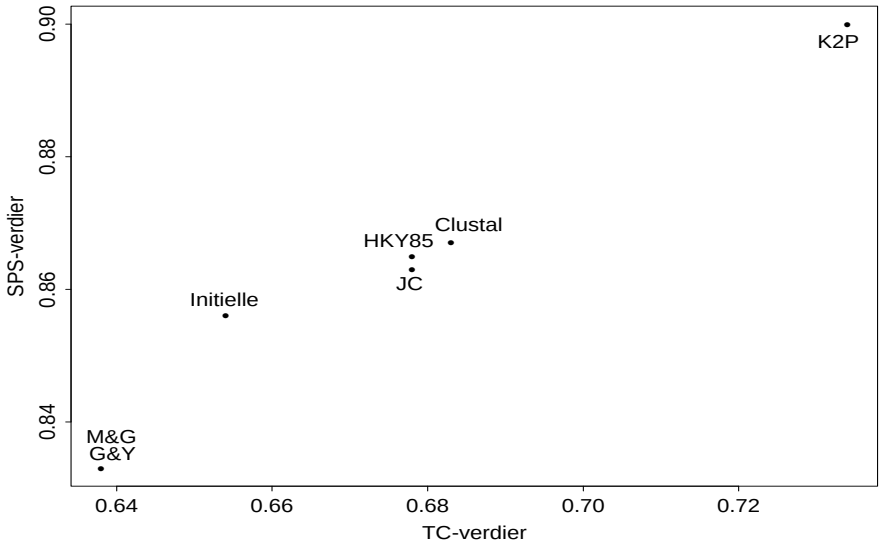
For datasettet *mcyABC* viser resultatene at nukleinsyremodellene gjør det bedre for SPS-verdi enn kodonmodellene. Resultatet fra Clustal ligger midt i mellom disse. En av modellene skiller seg litt ut, K2P. K2P-modellen tar hensyn til faktorene rate for gjennomsnittlig forandring μ, og transisjons- transversjonraten κ. Det ser ut som om akkurat disse faktorene spiller en viktig rolle for dette datasettet ettersom resultatene er så bra for K2P.

For datasettet β-globin har alle modellene i Ittero høyere SPS- og TC-verdier enn Clustal. Dette skyldes at Clustal innfører gap for alle gapstraffer, og flere gap enn modellene i Ittero. Som for datasettet *mcyABC* er det modellen K2P som har gir den høyeste SPS-verdien og underbygger det at faktorene i denne modellen spiller en positiv rolle ved sammenstilling. Det vil være interessant å gå videre med dette for å se på hvorfor det er slik. Dette vil være et forslag til videre arbeid.

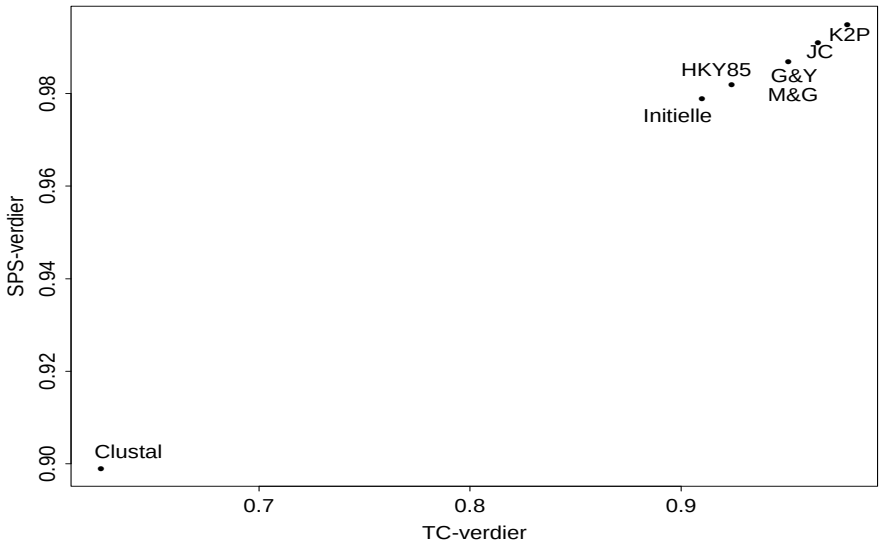
Resultatene til Anhydrase datasettet er forholdsvis lave for alle modellene i Ittero, og også for Clustal. Det kan se ut som om denne fremgangsmåten ikke egner seg for dette datasettet eller at sekvensene er for ulike til å sammenstilles på nukleinsyrenivå.

Forholdet mellom SPS-verdi og TC-verdi

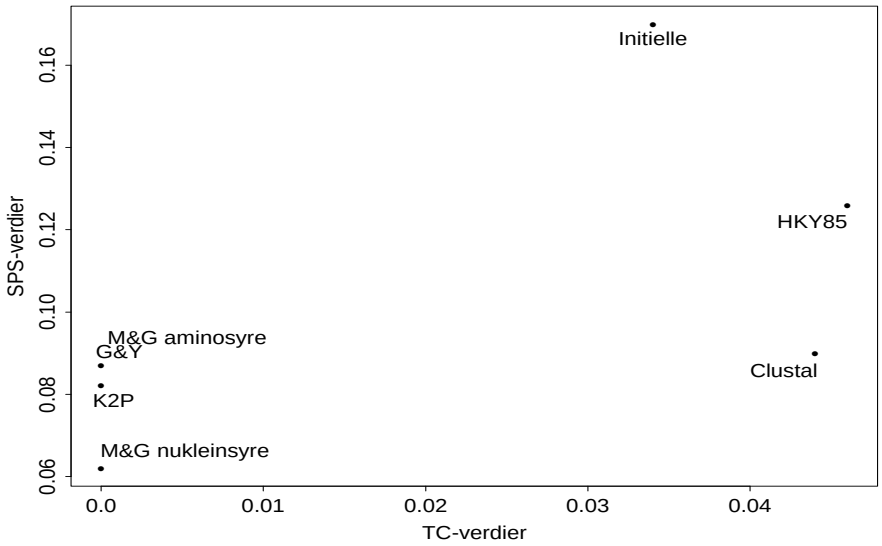
Ser man på forholdet mellom SPS-verdi og TC-verdi kan man se at for datasettene *mcyABC* og β-globin er det en klar sammenheng mellom de to verdiene. Ved høy SPS-verdi har modellene og en høy TC-verdi. Dette gjelder ikke for datasettet Anhydrase der det ikke er noen sammenheng mellom verdiene. For dette datsettet er begge verdiene forholdsvis lave, og som sagt ser det ikke ut som om disse sekvensene er egnet for sammenstilling på nukleinsyrenivå. Sammenhengen mellom SPS-verdi og TC-verdi er vist i figurene 7.4, 7.5 og 7.6.



Figur 7.4: Figuren viser forholdet mellom SPS-verdi og TC-verdi for datasettet *mcyABC*.



Figur 7.5: Figuren viser forholdet mellom SPS-verdi og TC-verdi for datasettet β-globin.



Figur 7.6: Figuren viser forholdet mellom SPS-verdi og TC-verdi for datasettet Anhydrase.

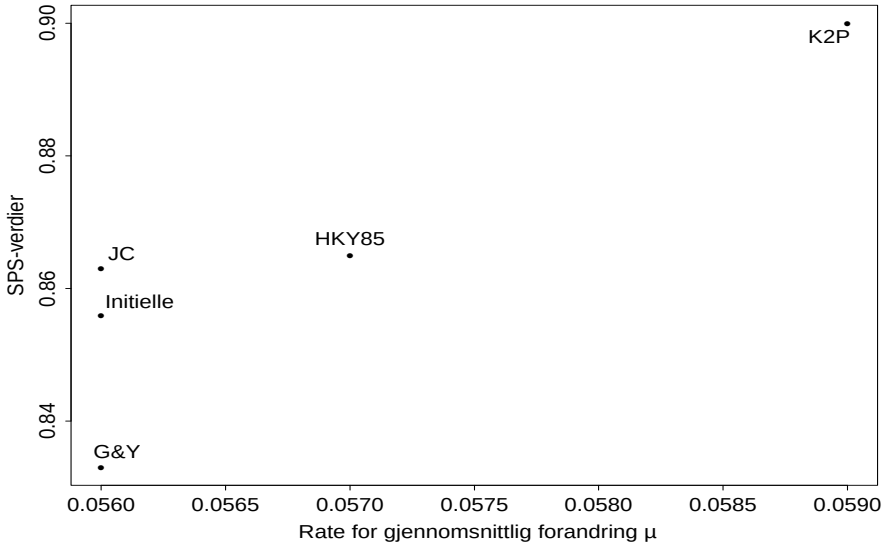
7.2.6 Guidetrær

Siden guidetrærne bestemmer rekkefølgen sekvensene skal sammenstilles i, vil rekkefølgen i guidetreet direkte påvirke den endelige sammenstillingen. Gap som introduseres tidlig i den multiple sammenstillingen vurderes aldri på nytt. Hvis de første sammenstillingene innholder mange gap, vil den endelige sammenstillingen inneholde mange gap. På tross av dette viser ikke resultatene at guidetrærne påvirker resultatene. De fleste danner forholdsvis like guidetrær med unntak av datasettet Anhydrase. Men heller ikke for dette er det noen klar sammenheng mellom guidetreet og resultat. Spesielt kan vi legge merke til at de to modellene som har høyest og lavest SPS-verdi gir samme guidetre.

7.2.7 Parametre

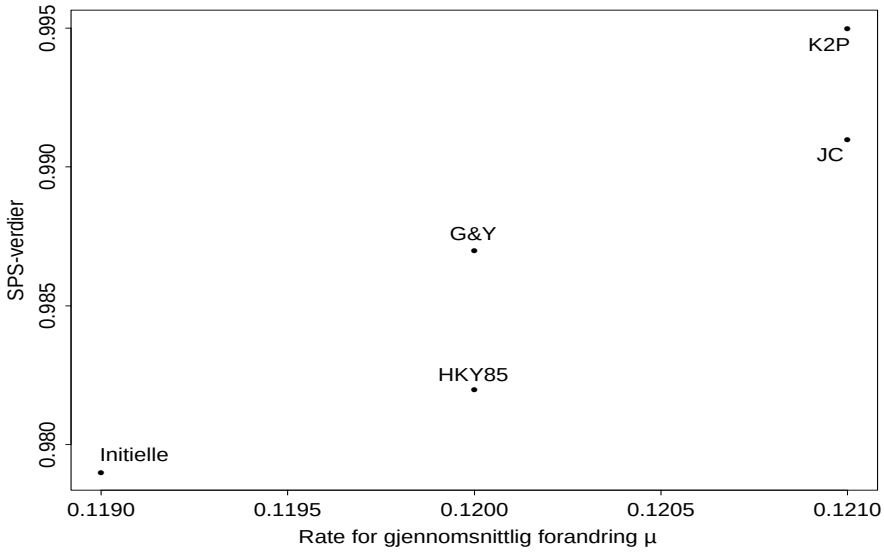
De ulike modellene i Ittero benytter ulike parametre, og spørsmålet er hvordan disse kan påvirke resultatene. Parameterne i seg selv sier mer om sekvensene enn de gjør om sammenstillingen. Transisjons- transversjonsraten og parameterne for synonym og ikke-synonym substitusjonsrate forteller hvor mange ulike forandringer som har funnet sted i datamaterialet, de sier ikke noe om en sammenstilling er god eller ikke. Så lenge man ikke vet noe om hvordan disse parameterne bør være for dette datasettet kan man heller ikke si noe

om resultatet i forhold til de. Parameteren μ angir gjennomsnittlig forandring i datasettet. Parameteren er gitt som antall forandringer delt på antall forandringer pluss antall som ikke er forandret. Hvis denne parameteren øker, sier det at det har blitt flere forandringer. Hvis den synker har det blitt færre forandringer. En bra sammenstilling har så mange som mulig like sammenstilte, og ut fra dette argumentet bør μ være lavest mulig. Fra resultatene kan man se at μ øker eller er uforandret etter den initielle sammenstillingen. For å se om det er noen sammenheng mellom μ og SPS-verdien for de ulike modellene som parameterne inngår i ble disse plottet. Figur 7.7, 7.8 og 7.9 viser resultatene. Fra figurene kan man se at for datasettene *mcyABC* og β-globin er det en

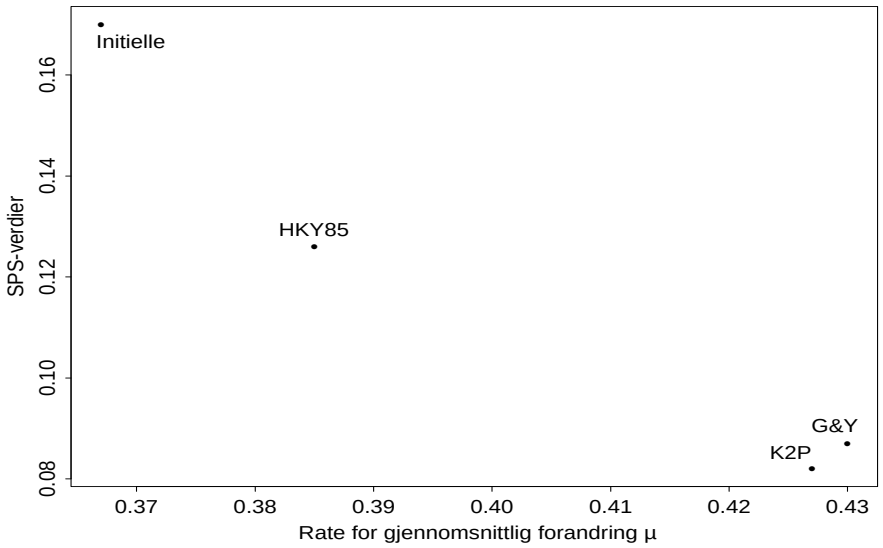


Figur 7.7: Figuren viser μ plottet mot SPS-verdier datasettet for *mcyABC*

tendens til at SPS-verdi øker med økt μ, mens for datasettet Anhydrase skjer det motsatte. For disse datasettene kan man altså ikke si at μ bør være lav. En grunn til at man får en sammenheng mellom høy μ og høy SPS-verdi kan være at en lav μ gir en "dårlig" substitusjonsmatrise, sammen med de andre parameterne. Dette har ikke oppgaven gått noe inn på, men det vil være en del av videre arbeid å se på hvordan substitusjonsmatrisene påvirker resultatet.



Figur 7.8: Figuren viser μ plottet mot SPS-verdier for datasettet β-globin med gapstraff 2.5



Figur 7.9: Figuren viser μ plottet mot SPS-verdier for datasettet Anhydrase

7.3 Konklusjon

Resultatene fra datasettet Anhydrase viser ingen av sammenhengene de to andre datasettene gjør, og BAliScore-verdiene er også forholdsvis lave. Det ser altså ikke ut som om datasettet egner seg for å bruke her. Dette kan skyldes at sekvensene er for ulike, eller at det ikke blir en bra sammenstilling av å oversette en aminosyresammenstilling til nukleinsyresammenstilling for disse sekvensene. Ut fra resultatene fra datasettene *mcyABC* og β-globin kan det sies at det er en fordel å ta hensyn til biologiske faktorer i de evolusjonære modellene ved multippel sammenstilling, og det å benytte kodonet som enhet ved sammenstilling av kodende nukleinsyresekvenser. Det vil være spennende å gå videre med dette og undersøke nærmere for begge punktene. Noen forslag til dette gis under videre arbeid.

7.4 Videre arbeid

Det er mange aspekter ved multippel sammenstilling, og det er ikke mulig å se på alle disse i en oppgave. Denne oppgaven har sett litt på multippel sammenstilling generellt, og spesielt om det er mulig å bruke evolusjonære modeller og kodonet som enhet til å danne multiple sammenstillinger for kodende nukleinsyresekvenser. Ved å forandre frekvensparameteren π i evolusjonære modeller til å representere begge de to nukleotidene man ser på, ikke bare frekvensen i foreldresekvensen som vanlig i fylogentisk analyse, kan denne brukes. Nukleotidene fra begge sekvensene må brukes da vi ikke antar noe om hvem som har evolvert fra hvem. De andre parameterne kan man bruke slik de var oppgitt i modellene. Resultatene viser at det er fullt mulig å bruke modellene, og at det kan gi positive resultater. Noen apsekter man kunne ha gått videre med foreslås her.

- Det vil være naturlig å utvide Ittero med de ulike metodene som brukes i Clustal for å forbedre en sammenstilling. Dette er blant annet sekvensvekting, posisjonsspesifikke gapstraffer og ulike valg av substitusjonsmatriser for ulike deler av sammenstillingen, Thompson, Higgins & Gibson (1994*a*). I tillegg til å se om dette vil bedre resultatet, vil det kunne gi svar på om noen av modellene er spesielt egnet, og om de gode resultetene kan skyldes bruk av kodonet som enhet og ikke nukleotidet.

- Utvide med flere datasett for å kunne se på statistisk signifikans. Ved å se på statistisk signifikans kan man si mer om modellene gir forbedringer i multiple sammenstillinger. Dette er ikke gjort da det var tidkrevende å finne gode datasett med referansesammenstillinger.

- Se på hvordan substitusjonsmatrisene er i forhold til resultatene. For å kunne sammenlikne substitusjonsmatrisene for alle modellene må man se på substitusjonsmatrise for de 61 sensekodonene. Kan det være slik at noen av modellene ikke egner seg, da substitusjonsmatrisene ikke blir gode? Dette kan for eksempel være at for kodonmodellene vil det være null for alle kodoner som har mer en en ulikhet. Dette kan kanskje føre til at det blir lavere verdier for en sammenstilling med kodonmodeller ved den dynamiske programmeringen, og det blir enklere innført gap. Under dette punktet kan man og se på hva det er som gjør at modellen K2P gir bra resultater, om substitusjonsmatrisen for K2P skiller seg fra de andre.

Bibliografi

Barton, G. J. (2001), 'Bioinformatics: A practical guide to the analysis of genes and proteines,- kap.9 creation and analysis of protein multiple sequence alignments', *Wiley Interscience* .

Baumbusch, L. O., Thorstensen, T., Krauss, V., Fischer, A., Naumann, K., Assalkhou, R., Schulz, I., Reuter, G. & Aalen, R. B. (2001), 'The *Arabidopsis thaliana* genome contains at least 29 active genes encoding set domain proteins that can be assigned to four evolutionary conserved classes', *Nucleic Acid Research* **29**(21), 4319–4333.

Becker, W. M., Reece, J. B. & Poenie, M. F. (1996), 'The world of the cell', *The Benjamin/Cummings Publishing Company, Inc.* .

Cavalli-Sforza, L. L. & Edwards, A. W. F. (1967), 'Phylogenetic analysis: Models and estimation procedures', *Evolution* **21**, 550–570.

Dictionary (1996), 'Dictionary of biology', *Oxford University Press, Third Edition* .

Dutta, S. & Goodsell, D. S. (Juni 2004), 'Carbonic anhydrase', *http://www.rcsb.org/pdb/molecules/pdb49_1.html* .

Edgar, R. C. (2004), 'Muscle: multiple sequence alignment with high accuracy and high throughput', *Nucleic Acids Reasearch* **32**(5), 1792–1797.

Edgar, R. C. & Sjølander, K. (2003), 'Satchmo: sequence alignment and tree construction using hidden markov models', *Bioinformatics* **19**(11), 1404–1411.

Ewens, W. J. & Grant, G. R. (2001), 'Statistical methods in bioinformatics, an introduction. kapittel 6', *Springer Verlag* .

Giribet, G. & Wheeler, W. C. (1999), 'On gaps', *Molecular Phylogenetics and Evolution* **13**(1), 132–143.

Giribet, G., Wheeler, W. C. & Muona, J. (2002), 'Dna multiple sequence alignments', *Molecular Systematics and Evolution: Theory and Practice. Birkhauser Verlag* pp. 107–114.

Goldman, N. & Yang, Z. (1994*a*), 'A codon-based model of nucleotide substitution for protein-coding dna sequences', *Molecular Biology and Evolution* **11**(5), 725–736.

Goldman, N. & Yang, Z. (1994*b*), 'Models of dna substitution and the discrimination of evolutionary parameters', *Proceedings of the 17th International Biometric conference* **1**.

Gotoh, O. (1996), 'Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments', *Journal of Molecular Biology* **264**, 823–838.

Gupta, S. K., Kececioglu, J. D. & Schaffer, A. A. (1995), 'Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment', *Journal of Computational Biology* **2**, 459–472.

Hall, B. G. (2001), 'Phylogenetic trees made easy', *Sinauer Associates, Inc* .

Hickson, R. E., Simon, C. & Perrey, S. W. (2000), 'The performance of several multiple-sequence alignment programs in relation to secondary-structure features for an r-rna sequence', *Molecular Biology and Evolution* **17**(4), 530–539.

Jakobsen, K. S., Mikalsen, B., Boison, G., Skulberg, O. M., Fastner, J., Davies, W., Gabrielsen, T. M. & Rudi, K. (2003), 'Natural variation in the microcystin synthetase operon *mcyABC* and impact on microcystin production in *Microcystis* strains', *Journal of Bacteriology* **185**(9), 2774–2785.

Jones, D. T. (1997), 'Progress in protein structure prediction', *Current Opinion in Structural Biology* **7**, 377–387.

Kilpelainen, P. (Juli 2004), 'Computing multiple string alignments', *http://www.cs.uku.fi/ kilpelai/BSA04/lectures/slides12.pdf* .

Lassmann, T. & Sonnhammer, E. L. L. (2002), 'Quality assesment of multiple alignment programs', *FEBS Letter* **529**, 126–130.

Lecompte, O., Thompson, J. D., Plewniak, F. & Poch, T. J. (2001), 'Multiple alignment of complete sequences (macs) in the post-genomic era', *Gene* **270**, 17–30.

Lewis, C. T. (Februar 2003), 'Types of multiple alignment', *http://homepage.usask.ca/čtl271/857/paper1_overview.shtml* .

Lewis, P. O. (2001), 'Phylogenetic systematics turns over a new leaf', *Trends in Ecology and Evolution* **16**(1).

Maddison, D. & Maddison, W. (Juni 2004), 'Mcclade', *http://macclade.org/index.html* .

MAPA (Oktober 2002), *Multiple Alignment and Phylogenetic Analysis*, http://cmgm.stanford.edu/classes/pdf/phylogenetic.pdf.

Morgenstern, B., Frech, K., Dress, A. & Werner, T. (1998), 'Dialign: Finding local similarities by multiple sequence alignments', *Bioinformatics* **14, no.3**, 290–294.

Mount, D. W. (2001), 'Bioinformatics, sequence and genome analysis', *Cold Spring Harbor Laboratory Press* .

MSA (September 2002), 'Multiple sequence alignments', *http://www.bioinf.org/molsys/ data/Alignments.ppt* .

Muse, S. & Gaute, B. S. (1994), 'A likelihood approach for comparing synonymous and nonsynonymous rates, with applications to the chloroplast genome', *Molecular Biology and Evolution* **11**(5), 715–724.

Navarro, G. (2001), 'A guided tour to approximate string matching', *ACM Computing Surveys* **33 no.1**, 31–88.

Nei, M. & Kumar, S. (2000), 'Molecular evolution and phylogenetics', *Oxford University Press* .

Nielsen, R. & Yang, Z. (1998), 'Likelihood models for detecting positively selected amino acid sites and applications to the hiv-1 envelope gene', *Genetics* **148**, 929–936.

Notredame, C. (1998), 'Utilisation des algorithmes genetiques pour l'analyse de sequences biologiques', *Universite Paul Sabatier, France* .

Notredame, C. (2001), 'Recent progress in multiple sequence alignments: a survey', *Pharmacogenomics* **3**, 131–144.

Notredame, C., Higgins, D. G. & Heringa, J. (2000), 'T-coffee: A novel method for fast and accurate multiple sequence alignment', *Journal of Molecular Biology* **302**, 205–217.

Ouyang, M. (Mai 2004), 'Pairwise alignment and dynamic programming', *http://www.ccl.rutgers.edu/ ouyang/ 5020/PairwiseAlignmentAndDynamicProgramming.ppt* .

Schadt, E. E., Sinsheimer, J. S. & Lange, K. (2002), 'Applications of codon and rate variation models in molecular phylogeny', *Molecular Biology and Evolution* **19**(9), 1550–1562.

Swofford, D. L., Olsen, G. J., Waddel, P. J. & Hillis, D. M. (1996), 'Molecular systematics,- kap.11, phylogenetic inference', *Sinauer Associates, Inc* .

Thompson, J. D., Higgins, D. G. & Gibson, T. J. (1994*a*), 'Clustalw: improving the sensitivity of progressive multiple sequnece alignments through sequence weighting, position spesific gap penalties and weight matrix choice', *Nucleic Acid Research* **22**, 4673–4680.

Thompson, J. D., Higgins, D. G. & Gibson, T. J. (1994*b*), 'Improved sensitivity of profile searches through the use of sequence weights and gap excision', *CABIOS* **10**(1), 19–29.

Thompson, J. D., Jeanmougin, F. & Gibson, T. J. (1998), 'Muliple sequence alignment with clustalx', *Trend in Biochemical Sciences* **23, no10**, 403–405.

Thompson, J. D., Plewniak, F. & Poch, O. (1999*a*), 'Balibase: a benchmark alignment database for the evaluation of multiple alignment programs', *Bioinformatics* **15**(1), 87–88.

Thompson, J. D., Plewniak, F. & Poch, O. (1999*b*), 'A comprehensive comparison of multiple sequence alignment programs', *Nucleic Acids Research* **27**(13), 2682–2690.

Turner, P. C., McLennan, A. G., Bates, A. D. & White, M. R. H. (2000), 'Instant notes, molecular biology', *BIOS Scientific Publishers Limited* .

UML (Juli 2004), 'Uml', *http://www.omg.org* .

Vingron, M. & Waterman, M. S. (1994), 'Sequence alignment and penalty choice, review of concepts, case studies and implications', *Journal of Molecular Biology* **235**, 1–12.

Whelan, S., Li, P. & Goldman, N. (2001), 'Molecular phylogenetics state-of-the-art methods for looking into the past', *Trends in Genetics* **17**(5), 262–272.

Winter, P. C., Hickey, G. I. & Fletcher, H. L. (2002), 'Instant notes, genetics,- second edition', *BIOS Scientific Publishers Limited* .

Yang, Z. (Mai 2004), 'Codon substitution models and phylogenetic analysis of protein coding genes', *http://www.sanbi.ac.za/tdrcourse/material/d10_Yang2001WoodsHole.pdf* .

Yang, Z. & Nielsen, R. (1998), 'Synonymous and nonsynonymous rate variation in nuclear genes of mammals', *Journal of Molecular Evolution* **46**, 409–418.

Yang, Z., Nielsen, R., Goldman, N. & Pedersen, A. K. (2000), 'Codon-substitution models for heterogeneous selection pressure at amino acid sites', *Genetics* **155**, 431–499.

Tillegg A

Ordliste

Beskrivelsene er hentet fra bøkene Winter, Hickey & Fletcher (2002), Turner *et al.* (2000) og Dictionary (1996).

Aminosyre Aminosyrer er enhetene i proteiner. Aminosyrer har den generelle formen $RCH(NH_2)COOH$ hvor R er sidekjeden som er spesiell for hver aminosyre. Vi har 20 forskjellige aminosyrer.

Base En base er enten en purin eller en pyrimidin, som igjen er en av byggesteinene i DNA. Det er fire ulike baser i DNA og en femte i RNA.

DNA Deoxyribonucleotic acid. DNA er bærer av den genetiske informasjonen og finnes i alle levende celler. Et DNA-molekyl er bygget opp av to komplementære kjeder. Hver kjede er bygget opp av nukleotider.

Dynamisk programmering Ved dynamisk programmring løses delproblemer av det problemet man ser på. Løsningene av disse delproblemene brukes til å finne den overordnete løsningen.

Fylogeni Fylogeni er den evolusjonære historien og forandringer for nedstamming fra et felles opphav for en art eller en høyere taksonomisk gruppe. Ordet brukes også om et fylogenetisk tre, se Tre.

Fylogenetisk analyse Fylogenetisk analyse er å oppdage evolusjonære relasjoner mellom ulike arter. Det brukes og til å analysere genfamilier og å finne den evolusjonære historien til spesifikke gener.

Genetisk kode Den genetiske koden er hvordan de ulike kodonene koder for ulike aminosyrer. Dette vises i tabell B.1.

Guidetre Et guidetre er et tre som sier i hviken rekkefølge sekvenser skal sammenstilles i en multippel sammenstilling. Et guidetre er det andre steget i en progressiv sammenstillingsalgoritme. Her lages treet ut fra hvor like sekvensene man ser på er.

Homologi Sekvenser sies å være homologe hvis de har et felles opphav. Innen fylogeni brukes homologi om spesielle egenskaper hos ulike individer som genetisk stammer fra samme opphav. Innen molekylærbiologi brukes homologi ofte om likhet uten at det trenger å være en genetisk relasjon.

Kodon Kodoner er tripletter av nukleinsyrer som koder for enten en aminosyre eller start, stopp. Se tabell B.1.

Leseramme En leseramme bestemmer startpunktet for hvor i en DNA-sekvens avlesingen starter. For den ene strengen i DNA har man tre mulige leserammer, se figur A.1.

```
a  a  c  t  g  g  t  a  a  c
      ↑  ↑  ↑
      1  2  3
```

Figur A.1: Figuren viser de tre mulig leserammene for en sekvens

Markovmodell En markovmodell er en type tilstandsmaskin der de ulike tilstandene kan gå over til andre tilstander med en viss sannsynlighet.

Mutasjon Mutasjoner er spontane forandringer i DNA som ikke repareres og som fører til mangfold i arvematerialet. Noen ulike typer mutasjoner:

- Basepar substitusjoner:

  Missense mutation En mutasjon der man får en forandring i aminosyren kodonet koder for.

  Nonsense mutation En nonsene mutasjon er en mutasjon der et kodon forandres til å bli et stopp-kodon. Hvis for eksempel UGC, Cystein, muterer i tredje posisjon, oppstår et stopp-kodon. Se B.1.

  Stille (silent) mutation En stille mutasjon er en mutasjon der man får en forandring i kodonet, men at det nye kodonet koder for samme aminosyre. En stille mutasjon er det samme som en synonym substitusjon.

- Basepar innsettelse eller fjerning

  Leseramme mutasjon (Frameshift mutation) En mutasjon som fører til en forandring i leserammen. Disse mutasjonene overlever sjelden da de kan føre til store forandringer i proteinet.

Nukleinsyrer Nukleinsyrer er polymerer som er bygget opp av nukleotider. Nukleinsyrene er de to molekylene DNA og RNA.

Nukleotider Nukleotider er byggesteinene i DNA og RNA. En nukleotide består av en base, et sukkermolekyl og en fosfatgruppe.

Nukleært DNA DNA som er i cellekjernen, og ikke for eksempel i mitokondriene.

Protein Et protein består av en kjede av aminosyrer. Hvert protein har sin bestemte sammensetning av aminosyrer. Proteiner er essensielle for alle levende organismer, som for eksempel enzymer og kontroll av genekspresjon.

Puriner Puriner er nukleotidene A og G

Pyrimidiner Pyrimidiner er nukleotidene T, U og C

RNA Rubonucleic acid. Nukleinsyre som består av en kjede av nukleotider. RNA er involvert i transkripsjonen av genetisk inormasjon.

Sekvenslikhet Se homologi.

Substitusjonsmatrise En substitusjonsmatrise er en tabell over hvilke verdier man skal sette for å bytte ut et tegn med et annet. Eksempel på en substitusjonsmatrise for DNA er gitt i figur A.2. Dette er IUB-verdiene som brukes i Clustal og for den initielle modellen i Ittero.

	A	G	T	C
A	1.9			
G	0.0	1.9		
T	0.0	0.0	1.9	
C	0.0	0.0	0.0	1.9

Figur A.2: Eksempel på en substitusjonsmatrise for DNA

Synonym substitusjon En synonym substitusjon er en substitusjon av en nukleotide der det ikke bli forandring i aminosyren. En synonym substitusjon er det samme som en stille mutasjon.

Transisjon En nukleotide substitusjon fra purin til purin, f.eks A$\longleftrightarrow$G, eller fra pyrimidin til pyrimidin

Transversjon En nukleotide substitusjon fra en purin til en pyrimidin eller motsatt. Eksempel A$\longleftrightarrow$T.

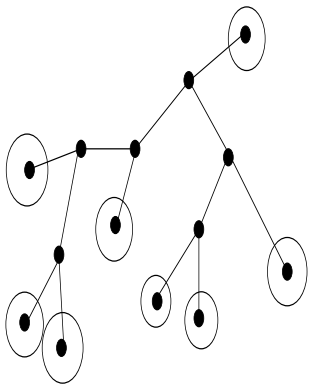
Tre Ordene tre og fylogeni brukes om hverandre. Et tre representerer den evolusjonære historien til en art eller en høyere taksonomisk gruppe. Et tre kan være rotet eller ikke-rotet.

- Om et tre:
  - Hvis en intern node har tre kanter, representerer noden en bifurkasjon eller dichotomy
  - Hvis det er mer enn tre kanter:
    multifurkasjon eller polytomy
  - Binært tre: alle interne noder er bifurkale
- Et ikke-rotet tre er et tre der det tidligste tidspunktet for en mutasjon eller et felles opphav ikke er definert, Swofford *et al.* (1996).
  Et ikke-rotet fullstendig binært tre har T terminale noder og T-2 interne noder. Treet har 2T-3 kanter hvor T-3 er interne og T er perifere. Det totale antall av ulike ikke-rotete fullstendig binærtre for T taxa er:

$$B(T) = \prod_{i=3}^{T} (2 * i - 5)$$

  Ved å legge til en rot legges det til en intern node og to interne kanter. Ettersom roten kan plasseres blant enhver av de 2T-3 kantene, antall mulige rotete trær økes med en faktor på 2T-3

  Eksempel på et Ikke-rotet fullstendig binær-tre vises i figur A.3



Figur A.3: Binært tre. For å vise forhold mellom interne og eksterne noder er de eksterne avmerket med ring rundt.

Tillegg B

Genetisk kode

Her vises tabellen over kodonene og hvilke aminosyrer de koder for. Det henvises til denne tabellen flere steder. Tabellen er hentet fra Becker, Reece & Poenie (1996).

Første posisjon	Andre posisjon: t	Andre posisjon: c	Andre posisjon: a	Andre posisjon: g	Tredje posisjon
t	ttt phe ttc phe tta leu ttg leu	tct ser tcc ser tca ser tcg ser	tat tyr tac tyr taa stopp tag stopp	tgt cys tgc cys tga stopp tgg trp	t c a g
c	ctt leu ctc leu cta leu ctg leu	cct pro ccc pro cca pro ccg pro	cat his cac his caa gln cag gln	cgt arg cgc arg cga arg cgg arg	t c a g
a	att ile atc ile ata ile atg met/start	act thr acc thr aca thr acg thr	aat asn aac asn aaa lys aag lys	agt ser agc ser aga arg agg arg	t c a g
g	gtt val gtc val gta val gtg val	gct ala gcc ala gca ala gcg ala	gat asp gac asp gaa glu gag glu	ggt gly ggc gly gga gly ggg gly	t c a g

Tabell B.1: Tabell over hvilke kodoner som koder for de ulike aminosyrene.

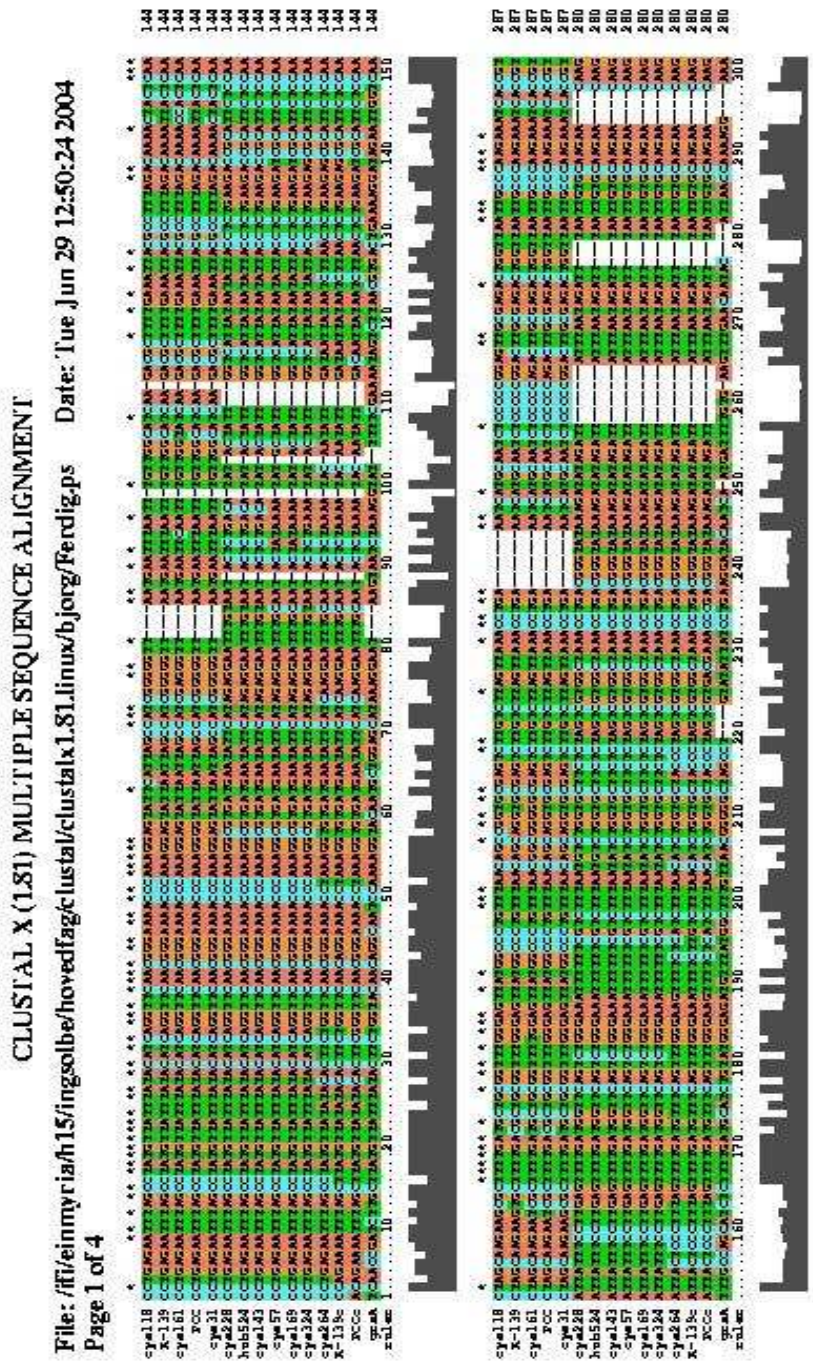
Tillegg C

Resultateksempler

Her vises eksempler på de multiple sammenstillingene. Disse er altfor omfattende til å vise i sin helthet, og det er derfor gitt eksempler på hvordan de ser ut.

C.1 Clustal

Et lite utsnitt av et eksempel på resultat fra Clustal. Eksemplet er fra datasettet *mcyABC*.



Figur C.1: Figuren viser et utsnitt av resultatfil fra Clustal

C.2 Ittero

Utsnitt fra et eksempel på multippel sammenstilling med Ittero. Datasettet er *mcyABC* og den initielle modellen.

```
gapOpen: -10.0 gapExtend: -0.2
Multiple: gapOpen: -10.0 gapExtend: -0.1
ID cya57
ID cya143
ID cya228
ID hub524
ID cya169
ID cya324
ID PCCc
ID cya264
ID K-139c
ID grsA
ID cya31
ID cya161
ID PCC
ID cya118
ID K-139

1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggc gtg
aca gaa aat tta gct tat gtt ata tac act tcg ggt tca acg gga aaa cct aaa ggt gtg
cct gag aat tta gcc tat gtt ata tac act tct ggt tca acg gga aaa cct aaa ggt gtg
cca gaa aat tta gct tat gtt ata tac act tct ggt tca aca gga aaa cct aaa ggt gtg
tca acc gat ctt gct tat gtt att tat act tct ggt aca aca ggc aat cca aaa ggt aca
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa ggg gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta
cct gag aat tta gcc tat gtt att tat aca tca ggt tca acg gga aaa ccc aaa gga gta


21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
atg aat att cat aga gga --- att tgc aat act ata aaa tat gct att --- ggt cat tat
atg aat att cat aga gga --- att tgt aat act ctg aca tat act att --- ggt cat tat
atg aat att cat aga gga --- att tgt aat act ctg aca tat act att --- ggt cat tat
atg aat att cat aga gga --- att tgt aat act ctg aca tat act att --- ggt cat tat
atg aat att cat aga gga --- att tgc aat act ata aaa tat act att --- ggt cat tat
atg aat att cat aga gga --- att tgt aat act ctg aaa tat act att --- ggt cat tat
atg aat att cat caa gga --- att tgc aat act cta aaa tac aat att --- gac aat tat
atg aat att cat caa gga --- att tgc aat act cta aaa tac aat att --- gat aat tat
atg aat att cat caa gga --- att tgc aat act cta aaa tac aat att --- gat aat tat
atg ctg gag cat aaa gga ata agt --- aat --- cta aag gta ttt ttc gaa aat agt ctt
tta att agc cat cgg ggg tta atg --- aat --- tta att tgt tgg cat caa gac gct ttt
tta att agc cat cga ggg tta atg --- aat --- tca att tgt tgg tat caa gac gct ttt
tta att agc cat cgg ggg tta atg --- aat --- tta att tgt tgg cat caa gac gct ttt
ttc att agc cat cgg ggg tta atg --- aat --- tta att tgt tgg cat caa gac gct ttt
tta att agc cat cgg ggg tta atg --- aat --- tta att tgt tgg cat caa gac gct ttt
```

Figur C.2: Figuren viser et utsnitt ev en resultatfil fra Ittero med datsett *mcyABC* og den initielle modellen.