KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

PROBABILISTIC METHODS TO SEARCH FOR REGULATORY ELEMENTS IN SETS OF COREGULATED GENES

Jury:
Prof. dr. ir. J. Vandewalle, voorzitter
Prof. dr. ir. B. De Moor, promotor
Prof. dr. ir. S. Van Huffel
Prof. dr. P. Rouzé (INRA, VIB, U.Gent)
Prof. dr. ir. J. Vanderleyden
Dr. ir. K. Marchal
Dr. ir. Y. Moreau
Dr. ir. J. van Helden (ULB)
Prof. dr. ir. D. Roose

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Gert THIJS

U.D.C. 681.3*F2:575.113

Juni 2003

©Katholieke Universiteit Leuven – Faculteit Toegepaste Wetenschappen

Arenbergkasteel, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2003/7515/24

ISBN 90-5682-414-7

Voorwoord

Wanneer ik in september 1998 begon aan mijn doctoraatsonderzoek, had ik geen idee waar ik aan begonnen was. Niet alleen had ik eigenlijk nog nooit gehoord van bioinformatica, ik durf zelfs zeggen dat ik geen enkele basis had in de moleculaire biologie. Maar de wereld van DNA leek me zo fascinerend dat ik mij er met veel plezier aan gewaagd heb. Dat ik nu deze tekst kan presenteren is dan ook voor een groot stuk te danken aan de hulp van verschillende mensen. Vanaf het begin van mijn onderzoek heb ik genoegen gehad om te kunnen samenwerken met een schare onderzoekers uit zeer diverse disciplines. Graag wil ik hen bij deze dan ook bedanken.

Vooreerst wil ik mijn promotor Prof. Bart De Moor bedanken dat hij het aangedurfd heeft om mij als pas afgestudeerde ingenieur te laten beginnen op een volledig nieuw, interdisciplinair onderwerp waar ik dus nog helemaal niets van afwist. Ik heb mij altijd geïnspireerd gevoeld door Barts 'go for it'-mentaliteit die de bioinformatica-groep binnen ESAT-SCD heeft laten uitgroeien tot de groep die het nu is. Ik voel me vereerd dat hij me de kans gegeven heeft dit avontuur vanaf de start te kunnen meemaken.

Voorts wil ik ook de leden van mijn begeleidings- en leescomité bedanken voor het advies dat ze mij gegeven hebben tijdens mijn doctoraat en dat ze mijn proefschrift hebben doorgenomen. Ik dank hierbij Prof. Jos Vanderleyden dat hij als bioloog openstond voor de bioinformatica, wat resulteerde in een aantal vruchtbare samenwerkingen waarbij de mensen in zijn groep onze software hebben kunnen gebruiken. Ik dank ook Prof. Sabine Van Huffel dat zij tijd heeft vrijgemaakt om mijn proefschrift kritisch te lezen en te becommentariëren.
Finally, Prof. Pierre Rouzé has been a great help for me throughout my research. His enthousiasm and knowledge about bioinformatics and all the relevant literature has been very inspiring for me.

Mijn onderzoek zou ook helemaal niet zo vlot verlopen zijn als nu zonder de hulp van Dr. Yves Moreau en Dr. Kathleen Marchal. Beiden hebben vanuit hun eigen achtergrond mij gedurende mijn onderzoek bijgestaan met raad en daad. Ik ben dan ook zeer blij dat ze deel willen uitmaken van de jury van mijn doctoraatsproefschrift. Ik hoop dat we in de toekomst nog veel projecten samen kunnen verwezenlijken.

Verder wil ik Prof. Joos Vandewalle, als voorzitter, en Prof. Dirk Roose bedanken dat zij deel willen uitmaken van de jury van mijn doctoraatsverdediging.

I also thank Dr. Jacques van Helden of the Université Libre de Bruxelles. I feel a lot of respect for his work and I am therefore very pleased that he has accepted to be a member of the jury.

Binnen het kader van ons gemeenschappelijk STWW-project wil ik ook de collega's van de Bioinformatica groep van de vakgroep Moleculaire Genetica aan de Universiteit Gent bedanken. Ik denk hierbij vooral aan Stephane Rombouts en Magali Lescot met wie het goed samenwerken was. Onze samenkomsten zijn altijd een goede aanleiding geweest om nieuwe stappen te ondernemen in mijn onderzoek.
I would also like to acknowledge the other researcher with whom I have had interesting conversations during my stay at international conferences and workshops. I especially thank Uwe Ohler, Chris Workman and Laurent Marsan who gave me valuable comments on my research. I should also thank Prof Metin Akay for inviting me to the EMBS Satellite Symposium on Bioinformatics in Istanbul where I could present the work on INCLUSive. It goes without saying that I also have to thank all the users of our software who have given me feedback and notified me about problems.

Natuurlijk mag ik mijn collega's binnen de bioi-groep, Bart, Bert, Cynthia, Frank, Frizo, Geert, Janick, Joke, Kristof, Mik, Nathalie, Patrick, Peter, Pieter, Ruth, Steffen, Stein en Tijl, niet vergeten te bedanken. Niet alleen was het goed samenwerken met hen, maar zij zorgden er steeds voor dat het aangenaam toeven was op ESAT en dat het aantal baaldagen beperkt bleef. Vermits de boog ook niet altijd gespannen kan blijven staan, is er ook nood aan ontspanning. Om mijn geest vrij te maken kon ik altijd beroep doen op de "kaarters". Respect.

Dit onderzoek werd mede mogelijk gemaakt door de specialisatiebeurs van het Instituut voor de aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen (IWT-Vlaanderen).

Een speciaal "dank u" gaat uit naar mijn ouders die altijd in mij geloofd hebben en die mij alle kansen geboden die naar dit werk geleid hebben. Ik hoop dat ik hun vertrouwen in mij niet beschaamd heb en dat ik in de toekomst hen nog veel kan teruggeven.
Ik kan natuurlijk niet eindigen zonder Sofie te bedanken. Zij is steeds mijn steun en toeverlaat geweest en heeft er voor gezorgd dat ik zonder zorgen aan mijn doctoraat kon werken.

Abstract

In this dissertation we have developed a suite of algorithms to search for potential transcription factor binding sites based on a probabilistic sequence model. We have approached the problem from two different angles: supervised and unsupervised. To tackle the unsupervised problem, we have extended the original Gibbs sampling algorithm for motif finding. First we have introduced higher-order background models to better discriminate between true motifs and background noise. Second, we have used the probabilistic framework to estimate the number of instances of a motif in a sequence. This has resulted in **MotifSampler**. A thorough analysis of the influence of the different parameters on the performance of the algorithm has allowed us to present a motif finding procedure that can be applied in real biological examples. In this dissertation we discuss in detail the full scale analysis of four different data sets. The great diversity of these examples nicely illustrates the capabilities and limitations of our methodology. The most important result is that a well designed species-specific background model significantly improves the performance of the motif finding algorithm especially when a high level of noise is present in the data set. Based on the probabilistic sequence model, we have also implemented **MotifScanner** to screen for instances of known motifs in DNA sequences. This is the supervised approach. Again we study in detail the influence of the parameters on the number of instances retrieved. The proposed methodology turns out to be more robust to parameter changes than a classical position-weight matrix scoring scheme. If a set of matrices is available, it is also possible to screen a set of promoter sequences and assess the statistical significance of the number of instances found. The examples in yeast show that this approach is applicable but that it is limited by the quality of the motif models present in the database. Finally, we discuss the implementation of an integrated web-based platform for the analysis of microarray data, called **INCLUSive**. With Adaptive Quality-Based Clustering of gene expression measurements, several clusters of genes that have a similar expression profile can be found. The next step is to look at the promoter region of the genes in such a cluster. Therefore, we have implemented an upstream sequence retrieval system to locate the intergenic region on the genomic DNA. The selected sequences can be entered at the web interfaces of both the MotifSampler and the MotifScanner. To illustrate the applicability of our methods, we refer to the work of other researchers who have found specific motifs with MotifSampler in their sets of coregulated genes.

Samenvatting

In dit proefschrift hebben we een set van algoritmen ontworpen om te zoeken naar potentiële bindingsplaatsen voor transcriptiefactoren vertrekkend van een probabilistisch sequentiemodel. We hebben het probleem benaderd vanuit twee gezichtspunten: gesuperviseerd en niet-gesuperviseerd. Om het niet-gesuperviseerd probleem aan te pakken, hebben we het originele *Gibbs sampling* algoritme om motieven te zoeken aangepast. Vooreerst hebben we een hoger-orde achtergrondmodel geïntroduceerd om beter het onderscheid te kunnen maken tussen echte motieven en achtergrondruis. Ten tweede hebben we het probabilistisch raamwerk gebruikt om het aantal instanties van een motief te schatten in een sequentie. Deze uitbreidingen hebben geleid tot de implementatie van **MotifSampler**. Een doorgedreven studie van de invloed van de parameters op de performantie heeft ons dan toegelaten om een uitgekiende strategie voor te stellen om motieven te zoeken in biologische voorbeelden. In deze thesis gebruiken we vier grote voorbeelden voor een gedetailleerde studie. De grote verscheidenheid van deze voorbeelden illustreert duidelijk de mogelijkheden en beperkingen van ons algoritme. Het belangrijkste resultaat is dat een goed ontworpen organisme-specifiek achtergrondmodel de performantie singificant verbetert vooral wanneer een grote hoeveelheid ruis aanwezig is in de dataset. Vertrekkend van het probabilistisch sequentiemodel hebben we ook een gesuperviseerde methode geïmplementeerd om instanties van gekende motieven te detecteren: **MotifScanner**. Een gedetailleerde analyse van de invloed van de parameters op de performantie toont aan dat onze methode robuuster is dan een klassiek schema om met een gewichtsmatrix te scoren. Als een set van bekende matrices voorhanden is, kunnen we een set van cogereguleerde genen screenen en de statistische significantie berekenen van het aantal gevonden instanties. Voorbeelden in gist tonen aan dat deze methode toepasbaar is maar dat de grootste beperking de kwaliteit van de matrices is. Tenslotte bespreken we de implementatie van een geïntegreerd web-gebaseerd platform voor de analyse van microroostergegevens, **INCLUSive**. Met Adaptive Quality-Based Clustering vinden we een aantal clusters van genen met een gelijkaardig expressieprofiel. In de volgende stap willen we de promotersequenties van deze genen bekijken. Daartoe hebben we een systeem ontworpen dat de promoters probeert te lokaliseren in de genoomsequentie. De geselecteerde sequenties kunnen dan verwerkt worden door MotifSampler en/of MotifScanner. Om de toepasbaarheid te illustreren, kunnen we verwijzen naar het werk van anderen die onze algoritmen gebruikt hebben om specifieke motieven te detecteren binnen hun projecten.

Nederlandse samenvatting

Probabilistische methoden om te zoeken naar regulerende elementen in sets van co-gereguleerde genen

Inleiding en situering

Wanneer tijdens het einde van de jaren negentig van vorige eeuw de doorgedreven automatisering zijn intrede doet in de moleculaire biologie, betekent dit een belangrijke doorbraak in het genoom-onderzoek. De belangrijkste exponent van deze evolutie is wel de vervollediging van de menselijke genoomsequentie. Waar men in 1998 nog voorspelde dat het menselijk genoom volledig beschikbaar zou zijn rond 2005, werd de vervollediging al bekend gemaakt op 12 Februari 2001 tijdens een gezamenlijke persconferentie van het publiek gefinancierde *International Human Genome Initiative* en het private *Celera*. In de laatste jaren worden steeds meer genomen volledig beschikbaar gemaakt en dit heeft een grote invloed op de evolutie van het genetisch onderzoek, vooral dan op de manier hoe dat onderzoek nu benaderd wordt.

Een andere belangrijke doorbraak is de beschikbaarheid van metingen van het expressieniveau van een grote set van genen in één experiment. Dit wordt gerealiseerd met microroosters. Microroostertechnologie is gebaseerd op de hybridisatie-eigenschappen van complementaire DNA-sequenties en bestaat in twee vormen: cDNA-microroosters [91] en DNA-chips [59]. cDNA-microroosters worden gefabriceerd door dubbelstrengse cDNA's aan te brengen op een glasplaatje. DNA-chips worden geproduceerd volgens het principe van CMOS microchips door laag na laag sondes op te bouwen bestaande uit nucleotiden (zie ook Figuur 2.10). In beide gevallen wordt een staal van gemerkt mRNA gehybridiseerd op de chip en het niveau van hybridisatie wordt uitgelezen door laserexcitatie. In het geval van cDNA-microroosters bestaat het staal uit een mengsel van zowel mRNA uit een referentiestaal als mRNA van de teststaal. Beide mRNA-stalen worden verschillend gelabeld (typisch groen en rood). In dit geval wordt dus de differentiële expressie gemeten tussen test- en referentiestaal. In Figuur 2.9 geven we een overzicht van de procedure om een meeting te bekomen uit een microroosterexperiment. Microroosters worden tegenwoordig veelvoudig toegepast voor diverse doeleinden. Vooreerst zijn er de meer theoretische toepassingen binnen de moleculaire biologie waar de activiteit van genen in een modelorganisme en zijn mutaties bestudeerd wordt. Het doel is hierbij om genetische netwerken, in de brede zin van het woord, te ontrafelen en te reconstrueren. Maar ook meer praktische toepassingen van microroosters steken de kop op. Hierbij denken we vooral aan klinische toepassingen waarbij de expressie van bepaalde genen gelinkt kan worden aan ziektepatronen. Dit is bijvoorbeeld al succesvol toegepast bij verschillende vormen van kanker [3, 12, 36, 84]. Ons onderzoek kunnen we plaatsen binnen het theoretische raamwerk van de systeembiologie en meer specifiek de ontcijfering van het regulatorische mechanisme op basis van de promotersequenties.

Om voldoende inzicht te verwerven in de problematiek bespreken we in Hoofdstuk 2 ook de biologische concepten van genexpressie. Twee dingen zijn daarbij van belang voor ons onderzoek: het proces van gen tot proteïne en de structuur van een gen. Een gen wordt gedefinieerd als *de volledige DNA-sequenties nodig voor de synthese van een functioneel polypeptide of RNA-molecule*. Voor de beschrijving van de omzetting van de gecodeerde informatie verborgen in een gen tot een proteïne baseren we ons voornamelijk op de excellente review van Orphanides en Reinberg [81]. Figuur 2.5 geeft een overzicht van de verschillende stappen in de vorming van een proteïne uit een gen. Het proces start als een geactiveerde transcriptiefactor de nucleus binnenkomt en kan binden op het DNA. Door de aanwezigheid van een gebonden transcriptiefactor zal het mogelijk zijn om het RNA-polymerase aan te trekken en de transcriptie te starten. Het RNA-polymerase leest de DNA-sequentie en vormt een enkelstrengs RNA-molecule dat complementair is aan het gelezen gen. Aan dit RNA-molecule wordt een kap aan het 5’ einde en een polyadenylatiestaart aan het 3’ einde bevestigd om het molecule te stabiliseren. In de volgende stap wordt het mRNA gevormd door de intronen uit te splitsen. Het mRNA wordt dan samengepakt en getransporteerd naar het cytoplasme alwaar het uiteindelijke proteïne gevormd wordt.

Het belangrijkste aspect van dit proces voor ons onderzoek is dat de transcriptie gestart wordt door het binden van geactiveerde transcriptiefactor met het DNA. Berg en von Hippel [10] hebben aangetoond dat er een zekere complementariteit moet bestaan tussen de actieve site van het proteïne en het DNA vooraleer de transcriptiefactor kan binden met het DNA. Dit betekent ook dat als we verschillende bindingsplaatsen van een bepaalde transcriptiefactor vergelijken dat die bindingsplaatsen bepaalde kenmerken op sequentieniveau gemeenschappelijk hebben. De bedoeling is nu om algoritmen te ontwikkelen die het mogelijk maken die bindingsplaatsen te detecteren. Een belangrijk hulpmiddel hierbij vormen de expressiemetingen uit microroosterexperimenten omdat daar juist het niveau van transcriptie gemeten wordt. De hypothese waarvan vertrokken wordt is dat genen die een gelijkaardige expressie vertonen over het experiment mogelijk gereguleerd worden door dezelfde transcriptiefactor(en) en dus ook een zekere gelijkenis vertonen op sequentieniveau. Dit gegeven vormt de basis voor de door ons voorgestelde algoritmen.

Zoeken naar regulerende elementen in DNA

Voor we de door ons ontworpen algoritmen beschrijven, geven we eerst in Hoofdstuk 3 een overzicht van de bestaande methoden om naar regulerende elementen te zoeken in DNA-sequenties. Om naar een collectie van bindingsplaatsen te refereren introduceren we de term motief. Een specifieke bindingsplaats is dan een instantie van het motief. Een motief stellen we voor aan de hand van een motief model, dit kan zowel een string, een reguliere uitdrukking als een matrix model zijn. Het is duidelijk dat het type van representatie zijn invloed zal hebben op de gebruikte algoritmen. Anderzijds is er ook een verschil in de manier waarop instanties gezocht kunnen worden. Enerzijds is er de gesuperviseerde aanpak waarbij we naar instanties zoeken van een gekend motiefmodel. De andere aanpak is de niet-gesuperviseerde waarbij een motiefmodel geleerd wordt uit een set van sequenties waarvan vermoed wordt dat ze een gemeenschappelijk motief delen.

Representatie van een motief

Laten we eerst eens kijken naar de verschillende manieren om een motiefmodel voor te stellen. De eerste voorstelling is string-gebaseerd. Iedere bindingsplaats kan voorgesteld worden door het DNA-segment waar de transcriptiefactor bindt. Als we deze voorbeelden groeperen dan kunnen we daaruit een zogenaamde consensus te distilleren. Deze consensus wordt gevormd door op elke positie de meest waarschijnlijke nucleotide(n) te selecteren en voor te stellen volgens een bepaald alfabet (zie Notations). Het is ook mogelijk om het motief voor te stellen als reguliere expressie. De andere aanpak bestaat er in om het motief voor te stellen met een matrixmodel, waarbij iedere positie in de matrix een maat is voor het aantal keren dat een bepaald nucleotide gevonden wordt op die specifieke positie. In de thesis gebruiken we een positie-specifieke frequentie matrix als representatie voor een motief. Zo'n PSFM ziet er als volgt uit

$$\boldsymbol{\Theta} = \begin{pmatrix} q_{\mathtt{A},1} & q_{\mathtt{A},2} & \cdots & q_{\mathtt{A},W} \\ q_{\mathtt{C},1} & q_{\mathtt{C},2} & \cdots & q_{\mathtt{C},W} \\ q_{\mathtt{G},1} & q_{\mathtt{G},2} & \cdots & q_{\mathtt{G},W} \\ q_{\mathtt{T},1} & q_{\mathtt{T},2} & \cdots & q_{\mathtt{T},W} \end{pmatrix},$$

waar $q_{b,j}$ de waarschijnlijkheid is om de base b te vinden op positie j in de bindingsplaats. Deze matrix kan ook gebruikt worden om een visuele representatie of logo te maken van een motief model [93].

Zoeken naar gekende motiefinstanties

Zoals hierboven al aangehaald bestaan er twee mogelijke manier om het probleem aan te pakken: string-gebaseerd of matrix-gebaseerd. De meest directe en ook de meest stringente aanpak is om alle segmenten in een DNA-sequentie te vinden die overeenkomen met een bepaalde string. Vermits er variaties voorkomen binnen de collectie van bindingsplaatsen moeten we ook de nodige variatie toelaten binnen de overeenstemmende strings in de DNA-sequenties. Dit is mogelijk als we werken met reguliere uitdrukkingen waarvoor er verschillende efficiënte algoritmen voorhanden zijn.

De klassieke methode om met een matrix te scoren vertrekt van de veronderstelling dat de score van een segment gelijk is aan de score van de individuele basen in het segment. Dit leidt tot volgende algemene formule

$$W(\mathbf{x}) = \sum_{j=1}^{W} W_j(b_j),$$

De waarde van de individuele basen wordt gestockeerd in een positie gewichtsmatrix (position weight matrix, PWM). Bucher [15] stelt de PWM samen als

$$W_j(b) = \log\big(\frac{n_{bj}}{e_{bj}} + \frac{\epsilon}{100}\big) + c_j,$$

met n_{bj} aantal keren b voorkomt op positie j, e_{bj} is het verwacht aantal keren dat b voorkomt op positie j, e_{bj}. De term c_j wordt zodanig gekozen dat het maximum in een kolom j gelijk is aan 0. Schneider et al. [94] kiezen voor de

$$W_j(b) = \log_2 \frac{q_{bj}}{e_b},$$

met q_{bj} frequentie van b op positie j en e_b de frequentie van b in het genoom. Als we vertrekken van de PSFM, dan kunnen we de waarschijnlijkheid dat een segment in de DNA-sequentie gegenereerd is door dit model schrijven als

$$P(\mathbf{x}|\mathbf{\Theta}) = \prod_{j=1}^{W} q_{b_j,j},$$

met b_j de nucleotide op positie j in segment $\mathbf{x}$. Hier is er ook duidelijk een verband met de additieve methode, waar meestal met de logaritme gewerkt wordt van de probabiliteiten. In onze implementatie, *MotifScanner*, gebruiken we rechtstreeks de probabiliteiten om het aantal instanties te schatten.

Overgerepresenteerde motieven in sets van co-gereguleerde genen

Zoals al aangehaald in de inleiding vormen microroosters een handig instrument om het regulatiesysteem van een organisme tot op een zeker niveau te ontrafelen. De premisse waarvan vertrokken wordt, is dat een groep van genen die een gelijkaardig expressieprofiel vertonen doorheen een experiment mogelijk door dezelfde transcriptiefactor(en) gereguleerd worden. Verwacht wordt dan ook dat binnen zo'n set van genen bepaalde regulerende elementen meer frequent voorkomen dan in een random dataset. Het moet dus mogelijk zijn deze overgerepresenteerde motieven te detecteren in een set van sequenties. Ook binnen deze niet-gesuperviseerde aanpak kunnen we het onderscheid maken tussen string- en matrix-gebaseerde methoden.

String-gebaseerde methoden

In de meest eenvoudige vorm kan een zoekmethode herleid worden tot een algoritme om het aantal instanties van een bepaalde string te tellen in de dataset en dit aantal te vergelijken met het verwacht aantal instanties in die dataset. van Helden et al. [111] hebben deze methode volledig uitgewerkt en toegepast in diverse sets van co-gereguleerde genen in gist. De grootste beperking van deze methode is het feit dat er geen variaties toegelaten worden in de gezochte strings. Als deze wel toegelaten worden dan wordt de berekening van het verwacht aantal instanties complexer. Tompa [110] en later ook Nicodème [77] stellen een exacte methode voor om dit probleem op te lossen. Andere methodes steunen op gekende computationele algoritmen zoals suffixbomen [71, 89, 114] en grafentheorie [50, 85].

Matrix-gebaseerde methoden

Als we de startpositie a_k van het motief kennen in sequentie S_k dan kunnen we de waarschijnlijkheid dat de sequentie gegenereerd wordt door het achtergrondmodel $\mathcal{B}_0$ en het motiefmodel $\boldsymbol{\Theta}$ schrijven als

$$P(S_k|a_k,\boldsymbol{\Theta},\mathcal{B}_0) = \underbrace{\prod_{l=1}^{a_k-1} \mathcal{B}_0(b_{kl})}_{\text{achtergrond}} \underbrace{\prod_{l=a_k}^{a_k+W-1} \boldsymbol{\Theta}(b_{kl}, l-a_k+1)}_{\text{motief}} \underbrace{\prod_{l=a_k+W}^{L_k} \mathcal{B}_0(b_{kl})}_{\text{achtergrond}} . \tag{1}$$

Het is dit model dat de basis vormt voor de meeste matrix-gebaseerde methoden. De eerste implementatie van een matrix-gebaseerde methode staat op naam van Hertz et al. [41] met een "*gulzig*" zoekalgoritme *CONSENSUS*. Een andere methode die veel gebruikt wordt binnen het raamwerk van maximale waarschijnlijkheid is het zogenaamde EM algoritme (expectation-maximization). EM is een iteratieve procedure waarbij in de eerste stap (expectation) de verwachte waarde van de gegevens en de ontbrekende data berekend wordt. In de tweede stap (maximization) worden de parameters gezocht die de waarschijnlijkheid van de data maximaliseren. Lawrence en Reilly [53] hebben een eerste versie van EM geïntroduceerd om motieven te zoeken die precies een keer voorkomen in elke sequentie in de dataset. Dit EM algorime werd later uitgebreid door Bailey en Elkan [7] tot MEME dat in staat is meer realistische problemen op te lossen door het toelaten van geen, een of meerdere instanties van een motief in een sequentie. Een andere verbetering van MEME is een initialisatieprocedure om lokale minima te vermijden.

Een andere methode om een probleem met ontbrekende data aan te pakken is de Gibbs sampler. Dit algoritme werd eerst voorgesteld door Lawrence et al. [54] maar een meer technische uiteenzetting werd later uitgewerkt door Liu et al. [63]. Het is op deze methode dat we onze eigen implementatie, MotifSampler gebaseerd hebben. In Hoofdstuk 4 gaan we dan ook dieper in op de funderingen van de Gibbs sampler. De populariteit van deze methode valt ook af te leiden uit het aantal algoritmen dat hierop gebaseerd is. Het originele algoritme was in eerste instantie om motieven te detecteren in proteïne sequenties. Een aangepaste versie specifiek voor DNA is *AlignACE* [88, 47]. Een andere methode is *BioProspector* [64], waarin hoger-orde achtergrondmodellen gebruikt worden. Andere uitbreidingen zijn de mogelijkheid om speciale motiefmodellen, zoals een palindromisch[1] of een motief met een variabele tussenruimte, te kiezen. *Ann_spec* gebruikt de basisidee van de Gibbs sampler maar verschilt in de manier waarop het motief voorgesteld wordt. Workman en Stormo [119] gebruiken een eenlaags neuraal netwerk als motiefmodel.

Gibbs sampling voor het detecteren van motieven

Gegeven een set van sequenties willen we de parameters van het motiefmodel en de startpositie van de motiefinstanties leren op basis van de sequentie data. De gekozen oplossingsmethode, Gibbs sampling, past binnen een breder probabilistisch raamwerk waarin we ons onderzoek kunnen plaatsen.

Algoritmen voor sampling

Monte Carlo technieken werden de laatste jaren geïntroduceerd onder verschillende vormen in diverse probleemgebieden waar samples getrokken moeten worden van complexe distributie [61]. De fundamentele stap in Monte Carlo methoden is het genereren van pseudo-random getallen die een specifieke distributie $\pi(\mathbf{x})$ volgen. Meestal is het onmogelijk om rechtstreeks te samplen van $\pi(\mathbf{x})$ daarom worden alternatieve schema's gebruikt. Een van die schema's is het Metropolis algoritme waarin samples gegenereerd worden van een Markov-keten die $\pi(\mathbf{x})$ als evenwichtstoestand heeft. Het Metropolis algoritme start van een initiële toestand $\mathbf{x}^{(0)}$ en itereert dan over de volgende twee stappen:

1. Perturbeer $\mathbf{x}^{(t)}$ tot $\mathbf{x}'$ en bereken $\Delta h = h(\mathbf{x}') - h(\mathbf{x}^{(t)})$. $\mathbf{x}'$ wordt gegenereerd vanuit een symmetrische transitieprobabiliteitsfunctie T, waarbij

---

[1] Een palindromisch motief is symmetrisch in de zin dat het omgekeerde complement van het motief het motief zelf is. Dit type van motieven wordt regelmatig gevonden omdat de bindingsplaats op de negatieve en positieve streng gelijk is.

$T(\mathbf{x}^{(t)}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x}^{(t)})$.

2. Genereer een random getal $u \sim \text{uniform}(0, 1)$. Als $u \leq \exp(-\Delta h)$ dan kiezen we $\mathbf{x}^{(t+1)} = \mathbf{x}'$ anders kiezen we $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$.

Een specifieke uitbreiding van het originele schema wordt geïntroduceerd door Geman and Geman [35] als de *Gibbs sampler*. Het basisidee is om de onderliggende Markov-keten te ontbinden in een serie van conditionele probabiliteiten volgens een set van geprefereerde richtingen. Veronderstel dat we willen samplen van de distributie $\pi(\mathbf{x})$ en dat we $\mathbf{x}$ kunnen splitsen in drie componenten $\mathbf{x} = (x_1, x_2, x_3)$, dan geldt $\pi(\mathbf{x}) = \pi(x_1, x_2, x_3)$. Startend van $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$, een *systematische scan Gibbs sampler* kan dan geïmplementeerd door te itereren over de volgende stappen:

1. Trek $x_1^{(t+1)}$ volgens $\pi(x_1 | x_2^{(t)}, x_3^{(t)})$.
2. Trek $x_2^{(t+1)}$ volgens $\pi(x_2 | x_1^{(t+1)}, x_3^{(t)})$.
3. Trek $x_3^{(t+1)}$ volgens $\pi(x_3 | x_1^{(t+1)}, x_2^{(t+1)})$.

Het is mogelijk te bewijzen dat de Gibbs sampler geometrisch convergeert naar de evenwichtsdistributie $\pi(\mathbf{x})$ waarbij de snelheid van convergentie afhangt van de correlatie tussen variabelen [60]. Het is mogelijk dit schema verder te vereenvoudigen als het mogelijk is om bepaalde variabelen te groeperen (*gegroepeerde Gibbs sampler*) of ze te reduceren (*gereduceerde Gibbs sampler*) tot een nieuwe variabele. Tanner en Wong [101] hebben ook aangetoond dat de Gibbs sampler toegepast kan worden in een probleem waar waarden of variabelen ontbreken in de gegevens.

Originele Gibbs sampling algoritme

Liu et al. [63] hebben een *gereduceerde Gibbs sampler* ontworpen om te zoeken naar overgerepresenteerde motieven in een set van promotersequenties waarbij ze veronderstellen dat er exact één instantie van het motief te vinden is in elke set. Veronderstel dat N_s het aantal sequenties is in de set, S_k is de k-de sequentie en a_k is de positie van van het motief in sequentie S_k, dan kan de complete data-waarschijnlijkheid als volgt geschreven worden

$$\begin{aligned}\pi(\mathcal{S}, \mathcal{A} | \boldsymbol{\Theta}, \boldsymbol{\theta}_0) &= \prod_{k=1}^{N_s} \pi(S_k, a_k | \boldsymbol{\Theta}, \boldsymbol{\theta}_0) \\ &\propto \prod_{k=1}^{N_s} \left(\prod_{l=1}^{a_k - 1} \boldsymbol{\theta}_0(b_{kl}) \prod_{l=a_k}^{a_k - W + 1} \boldsymbol{\Theta}(b_{kl}, l - a_k + 1) \prod_{l=a_k+W}^{L_k} \boldsymbol{\theta}_0(b_{kl}) \right). \end{aligned} \quad (2)$$

waarin $\boldsymbol{\theta}_0$ het achtergrondmodel is met als parameters $[q_{\texttt{A}0}, q_{\texttt{C}0}, q_{\texttt{G}0}, q_{\texttt{T}0}]^T$ en $\boldsymbol{\Theta}$ is het motief model met als parameters $q_{b,j}$. Liu et al. [63] stellen voor om een Gibbs sampler te ontwikkelen die enkel focust op de aligneringsvector. De sampling procedure kan dan als volgt samengevat worden:

$$\text{Sample} \quad a_k^{(t+1)} \quad \text{volgens} \quad \pi(a_k|\mathcal{A}_{\bar{k}}, \mathcal{S}). \tag{3}$$

Mits de nodige bewerking kan deze vergelijking tot een zogenaamde predictieve scoringsfunctie van de vorm

$$\pi(a_k = i|\mathcal{A}_{\bar{k}}, \mathcal{S}) \propto \prod_{j=1}^{W} \left(\frac{\hat{\boldsymbol{\theta}}_j^{\,b_{i+j-1}}}{\hat{\boldsymbol{\theta}}_0^{\,b_{i+j-1}}} \right). \tag{4}$$

Deze formule zegt in essentie dat de kans dat het motief start op positie i evenredig is aan de kans dat het segment gegenereerd wordt door het motiefmodel gedeeld door de kans dat hetzelfde segment gegenereerd wordt door het achtergrondmodel.

Deze formule vormt de basis van het volgende Gibbs sampling programma:

1. Initialiseer de aligneringsvector $\mathcal{A}$ door random een startpositie te selecteren in elke sequentie.
2. Herhaal voor elke sequentie $S_z, z = 1 \ldots, N_s$
   (a) Creëer de subsets $\tilde{\mathcal{S}} = \{S_i | i \neq z\}$ en $\tilde{\mathcal{A}} = \{a_i | i \neq z\}$ uit de originele datasets door S_z uit te sluiten.
   (b) Berekenen het motiefmodel $\tilde{\boldsymbol{\Theta}}$ op basis van de segmenten met als startpositie $\tilde{\mathcal{A}}$ in de set $\tilde{\mathcal{S}}$ en bereken ook het achtergrondmodel $\tilde{\boldsymbol{\theta}}_0$ op basis van de compositie van alle niet-segmenten posities in $\tilde{\mathcal{S}}$.
   (c) Ken aan alle mogelijke motiefinstanties $\mathbf{x}_{zl}$ in sequentie S_z, met $l = 1 \ldots L_z - W + 1$, een score $W(\mathbf{x}_{zl})$ toe volgens de predictieve scoringsfunctie van Vergelijking 4.9,

$$W(\mathbf{x}_{zl}) = \prod_{j=1}^{W} \frac{\tilde{\boldsymbol{\theta}}_j^{\,(b_{z,l+j-1})}}{\tilde{\boldsymbol{\theta}}_0^{\,(b_{z,l+j-1})}}$$

   (d) Sample een nieuwe startpositie a_z volgens de genormaliseerde probabiliteitsdistributie $\frac{W(\mathbf{x}_{zl})}{\sum_l^{L_z - W+1} W(\mathbf{x}_{zl})}$
3. Herhaal stap 2 totdat de Markov keten convergeert.

Uitbreiding tot MotifSampler

In het originele algoritme zitten een aantal gebreken die een praktische toepassing in een set van co-gereguleerde genen moeilijk maakt. Eerst en vooral is er de aanwezigheid van ruis in een dataset gevonden door microroostergegevens te clusteren. Ruis bestaat onder vorm van sequenties die niet gereguleerd worden door dezelfde transcriptiefactor en dus geen specifieke bindingsplaats hebben. Anderzijds weten we dat er in eukaryoten meerdere bindingsplaatsen van dezelfde transcriptiefactor kunnen voorkomen om het niveau van transcriptie beter te kunnen controleren. Een tweede opmerking die we kunnen maken is dat het originele algoritme vertrekt van een de enkelvoudige frequenties van de nucleotiden als achtergrondmodel. Als we echter kijken naar geavanceerde algoritmen voor genpredictie dan maken die allemaal gebruik van hoger-orde Markov-modellen om een DNA-sequentie te modelleren. Het lijkt dus een aangewezen stap om dit ook toe te passen in motiefdetectie.

Hoger-orde achtergrondmodel

Een eerste aanpassing van de originele Gibbs sampler is de introductie van een hoger-orde achtergrondmodel. Deze aanpassing is ingegeven door de genpredictie software waar een DNA-sequentie gemodelleerd wordt met behulp van een Markov keten [27, 52, 67, 79]. Gegeven het hoger-orde achtergrondmodel $\mathcal{B}_m$, dan kunnen we de waarschijnlijkheid dat de sequentie gegenereerd wordt door dit model schrijven als

$$P(S|\mathcal{B}_m) = p(b_1, b_2, \ldots, b_m) \prod_{l=m+1}^{L} p(b_l|b_{l-1}, \ldots, b_{l-m}). \tag{5}$$

Deze vergelijking kunnen we nu inpassen in de afleiding van de predictieve scoringsfunctie en dit resulteert in volgende scoringsfunctie

$$W(\mathbf{x}_{kl}) = \frac{P(\mathbf{x}_{kl}|\boldsymbol{\Theta})}{P(\mathbf{x}_{kl}|S_k, \mathcal{B}_m)} = \prod_{j=1}^{W} \frac{\boldsymbol{\theta}_j(b_{l+j-1})}{P(b_{l+j-1}|\mathcal{S}, \mathcal{B}_m)}. \tag{6}$$

In woorden wil dit zeggen dat de score van een segment $\mathbf{x}$ dat start op positie l in sequentie S_k gegeven wordt door de verhouding van de waarschijnlijkheid dat het segment gegenereerd wordt door het motiefmodel ten opzichte van de waarschijnlijkheid dat het segment gegenereerd wordt door het achtergrondmodel.

De parameters van een achtergrondmodel worden gestockeerd in een transitiematrix (zie ook Tabel 5.1). Ieder element in de transitiematrix geeft de probabiliteit $P(b|b_1, b_2, \ldots, b_m)$ dat de base b gevonden wordt volgend op het oligonucleotide $b_1, b_2, \ldots, b_m$. Deze transtiematrix kan geconstrueerd worden door alle oligonucleotiden van lengte $m + 1$ te tellen in een referentie dataset en deze te groeperen volgens de m eerste basen. In het vervolg van deze thesis zullen we verschillende voorbeelden aanhalen waaruit blijkt dat de gebruik van een hoger-orde achtergrondmodel gebaseerd op een onafhankelijke, goed gedefinieerde dataset een positie effect heeft op de robuustheid van ons algoritme.

Schatting van het aantal instanties van het motief

Als we een motiefmodel $\boldsymbol{\Theta}$ kennen dan kunnen we proberen te schatten hoeveel instanties van dat motief er aanwezig zijn in de sequenties. Als we het aantal instanties in de sequentie S_k voorstellen als Q_k, dan is het verwacht aantal instanties te berekenen als

$$E_{(S_k,\boldsymbol{\Theta},\mathcal{B}_m)}(Q_k) = \sum_{c=0}^{\infty} c \times P(Q_k = c|S_k, \boldsymbol{\Theta}, \mathcal{B}_m).$$

Als we dit verder uitwerken met de regel van Bayes dan komen we tot volgende vergelijking

$$P(Q_k = c|S_k, \boldsymbol{\Theta}, \mathcal{B}_m) = \frac{P(S_k|Q_k = c, \boldsymbol{\Theta}, \mathcal{B}_m)P(Q_k = c|\boldsymbol{\Theta}, \mathcal{B}_m)}{P(S_k|\boldsymbol{\Theta}, \mathcal{B}_m)}.$$

Hierin onderscheiden we drie delen. De noemer geldt als normalisatiefactor. Het eerste deel van de teller is de waarschijnlijkheid dat de sequentie gegenereerd wordt door het model als er c instanties aanwezig zijn. Dit kan verder uitgewerkt worden door de som te nemen over alle mogelijke combinaties van c instanties. Het derde deel is de prior $P(Q_k = c|\boldsymbol{\Theta}, \mathcal{B}_m)$ die aangeeft wat de a priori kans is om c instanties te vinden gegeven het motiefmodel en het achtergrondmodel. Het is duidelijk dat we deze prior niet exact kennen en dat we dus een benadering moeten voorstellen. Een mogelijke veronderstelling is dat de kans om $c + 1$ instanties te vinden steeds kleiner is dan de kans om c instanties te vinden. Als we dus een waarde γ_1 voor $c = 1$ opgeven dan kunnen we de prior distributie benaderen door een distributie van de vorm

$$[\frac{1-\gamma_1}{C}, \frac{\gamma_1}{C}, \frac{\kappa\gamma_1}{C}, \frac{\kappa^2\gamma_1}{C}, \ldots, \frac{\kappa^k\gamma_1}{C}],$$

waarbij C een normalisatie constante is, k het maximaal aantal elementen en κ een waarde heeft tussen 0 en 1. In ons geval kiezen we $\kappa = 0.25$. Om een idee te krijgen van hoe de prior distributie er uitziet, geven we een voorbeeldje in Figuur 5.7.

Scores

Om de kwaliteit van een motiefmodel te kunnen beoordelen, gebruiken we een aantal scores.

- Consensusscore $2 - \frac{1}{W}\sum_{j=1}^{W}\sum_b q_{bj}\log_2(q_{bj})$, geeft aan hoe sterk geconserveerd het motief is.
- Informatie-inhoud $\frac{1}{W}\sum_{j=1}^{W}\sum_b q_{bj}\log_2\left(q_{bj}/q_{b0}\right)$, geeft aan hoe sterk geconserveerd het motief is in vergelijking met de frequentie van de nucleotiden in de achtergrond.

- Log-likelihood score $\sum_{k=1}^{N_s} \log(\Gamma_k(0)) + \sum_{c=1}^{C_{\text{max}}} \left(\sum_{i=c}^{C_{\text{max}}} \Gamma_k(i) \right) \log(W(\mathbf{x}_{a_{kc}}))$, houdt rekening met de kwaliteit van het motiefmodel, de verhouding ten opzichte van het achtergrondmodel en ook met het aantal instanties van het motief.

Gerelateerd aan de scores is de mogelijkheid om motiefmodellen met elkaar te vergelijken. Hiervoor definiëren we een symmetrische afstandsmaat op basis van de *Kullback-Leiber*-afstand. De afstand tussen twee motieven wordt dan als volgt berekend

$$d(\mathbf{\Theta}^{(1)}, \mathbf{\Theta}^{(2)}) = \frac{1}{2W} \sum_{j=1}^{W} \sum_{b=\mathtt{A}}^{\mathtt{T}} \mathbf{\Theta}_1(b,j) \log \frac{\mathbf{\Theta}_1(b,j)}{\mathbf{\Theta}_2(b,j)} + \mathbf{\Theta}_2(b,j) \log \frac{\mathbf{\Theta}_2(b,j)}{\mathbf{\Theta}_1(b,j)}.$$

Analyse van de parameters

Nu dat we een eerste versie van het algoritme klaar hebben, kunnen we dit gaan testen. Hierbij zijn we vooral geïnteresseerd hoe de performantie van het algoritme beïnvloed wordt door de gekozen parameters. De belangrijkste parameters die een invloed hebben, zijn de lengte van het motief, de prior γ_1 en ook nog de lengte van de sequenties. De performantie meten we aan de hand van de verschillende motiefscores. Als testdata gebruiken we de G-box dataset en ook 10 sets van co-gereguleerde genen in gist (regulons). In deze sets kennen we het te zoeken motief reeds op voorhand en dit helpt ons om de performantie te evalueren.

Een eerste aspect van de Gibbs sampler dat we willen onderzoeken, is hoe de stochasticiteit van de samplingprocedure het convergentiegedrag beïnvloedt. Als voorbeeld van het convergentiegedrag starten we de Gibbs sampler verschillende keren vanuit dezelfde initiële conditie en meten we de motief scores over 500 iteraties. Door de stochasticiteit is het mogelijk dat we vanuit dezelfde initiële condities convergeren naar sterk verschillende oplossingen. Drie van deze resultaten stellen we voor in Figuur 5.3. Eerst is er een voorbeeld waabij de Gibbs sampler al na twintig iteraties het juiste motief gevonden heeft. In het tweede voorbeeld duurt dat meer dan 300 iteraties. In het laatste voorbeeld convergeert de Gibbs sampler naar een verkeerde oplossing. Het is dus moeilijk om ondubbelzinnig een goed aantal iteraties te definiëren.

De volgende stap is uit te zoeken hoe dat de motiefscores gedistribueerd zijn als de procedure meermaals herhaald wordt. Hier bespreken we twee voorbeelden waarbij we de scores bekomen door de Gibbs sampler toe te passen op deze dataset, vergelijken met de scores bekomen met een gelijkaardige random dataset. In het eerste voorbeeld, Figuur 5.4, is er een duidelijk onderscheid tussen de twee distributie. In de echte dataset zien we dat er twee groepen van motieven gevonden worden, waarbij de piek van de best scorende motieven overeenkomt met het echte G-box motief. Het tweede voorbeeld, Figuur 5.5, toont aan dat het verschil niet altijd zo uitgesproken is en dat de scores van de echte motieven gedeeltelijk overlappen met de scores van de motieven in random datasets. In beide gevallen blijkt ook dat de consensusscore eigenlijk geen goede maat is voor de kwaliteit van het motief omdat hier de motieven in random sequentiessets betere scores halen dan in een echte dataset.

De volgende stap in de analyse is het effect van de keuze van de motieflengte op de resultaten. In Figuur 5.6, geven we aan hoe dat de scores veranderen in functie van de motieflengte. Voor zowel de consensusscore als de informatie-inhoud merken we op dat de scores gemiddeld genomen afnemen als de motieflengte stijgt. In het geval van de log-likelihood score zien we dat de distributies meer uniform worden. Wel is er een duidelijke piek te zien voor de motieflengtes die best overeenkomen met het echte motief. De volgende parameter is de prior. De logische impakt van de prior γ_1 is dat het aantal gevonden instanties stijgt met een stijgende prior. Een bijkomend effect is echter dat het aantal keren dat het echte motief gevonden wordt afneemt. De instelling van de prior berust dus op de afweging van enerzijds het aantal instanties dat motief bevat als anderzijds de graad van conservatie van het motief. In sommige gevallen is het ook mogelijk om de prior te laag te kiezen zodat geen motief gevonden wordt.

Uitwerking van een praktische implementatie

Om tot een werkend algoritme te komen dienen we ook nog een aantal praktische aspecten aan te pakken. Zo is er het probleem van de een mogelijke verschuiving van de aligneringsvector waardoor het motiefmodel vastloopt in een lokaal optimum waaruit het niet kan ontsnappen. De oplossing bestaat erin om na een aantal iteraties van de sampler de aligneringsvector enkele posities naar links en rechts te verschuiven en het motiefmodel te selecteren dat de beste score geeft. Om te verhinderen dat de MotifSampler te snel convergeert naar een motief met een beperkt aantal instanties proberen we eerst in elke sequentie één instantie te vinden. Tijdens de sampling-procedure wordt niet noodzakelijk de best scorende instanties geselecteerd. Het is dus aan te raden aan het eind van de procedure enkel de beste instanties naar buiten te geven. Deze convergentiestap herhalen we een aantal keren totdat motiefmodel en aligneringsvector constant blijven.

De combinatie van deze uitgebreide analyse van de verschillende parameters en de praktische aspecten resulteren in de procedure uitgeschreven in het volgende programma:

1. Selecteer het gewenste achtergrondmodel.
2. Definieer de parameters van MotifSampler:
    (a) De lengte W van het gezochte motief (typisch tussen 6 en 15bp).
    (b) De a priori probabiliteit γ_1 om één instantie te vinden.
    (c) Het aantal verschillende motieven n.
    (d) De maximale overlap tussen verschillende motieven.

   (e) Het aantal herhalingen R (eg. 100).

3. Herhaal de MotifSampler R keren met dezelfde parameters:
   (a) Start met een aantal iteraties waarbij er exact één instantie per sequentie gezocht wordt.
   (b) Herhaal een aantal keren de schuif-stap en de samplerprocedure.
   (c) Beëindig de procedure met de convergentie stap.
   (d) Creëer het finale motiefmodel en de bijhorende aligneringsvector.

4. Verwerking van de resultaten:
   (a) Sorteer alle $n \times R$ motieven volgens een specifieke score (eg. log-likelihood score).
   (b) Tel voor elk motief hoeveel gelijkaardige motieven gevonden zijn.
   (c) Rapporteer de beste motieven, daarbij rekening houdend met de gelijkaardige motieven.

Voorbeelden

In hoofdstuk 6 passen we de ontwikkelde strategie toe op vier specifieke datasets, die elk een welbepaald aspect van het bestudeerde probleem belichamen. In het eerste voorbeeld beschouwen we de promotersequenties van genen in planten die een gereguleerd worden door een G-box. Deze G-box speelt een belangrijke rol in de regulatie van genen onder invloed van licht in planten [30]. De genen zijn geselecteerd op basis van de voorbeelden die te vinden zijn in de databank *PlantCARE*. Deze set gebruiken we om de invloed van ruis te testen op de performantie van het algoritme. In het tweede voorbeeld analyseren we tien sets van co-gereguleerde genen in gist, meer bepaald *Saccharomyces cerevisiae*. Deze tien datasets werden aangemaakt door van Helden et al. [111] om hun algoritme te testen. Zo'n set van stroomopwaartse sequenties wordt ook een regulon genoemd. Het derde voorbeeld is speciaal ontwikkeld om de invloed van hoger-orde achtergrondmodellen te testen in prokaryoten [69]. Meer specifiek, bestuderen we hoe het gebruik van een niet-organisme specifiek achtergrondmodel de resultaten van het algoritme beïnvloedt. In het laatste voorbeeld passen we onze methode toe op een standaard microrooster experiment, de celcyclus in gist [97]. Met *adaptive quality-based clustering* definiëren we eerst clusters van genen die een gelijkaardig expressieprofiel hebben. In vier van deze clusters zoeken we dan naar overgerepresenteerde motieven. Laten we nu elk van deze sets afzonderlijk bekijken.

G-box transcriptiefactor

De eerste set van sequenties waarmee we gewerkt hebben tijdens ons onderzoek, zijn de promotersequenties van 33 genen die gereguleerd worden door de G-box transcriptiefactor in planten. Deze genen en ook de geannoteerde bindingsplaatsen vinden we in de databank PlantCARE [57]. Omdat deze bindingsplaatsen experimenteel geverifieerd zijn en we er de exacte positie van kennen is het een dankbare set om onze algoritmen te evalueren.

In eerste instantie bestuderen we de invloed van het achtergrondmodel op de performantie. De eerste mogelijkheid om een achtergrondmodel te berekenen, is op basis van de sequenties in de dataset zelf. De andere mogelijkheid bestaat erin om een onafhankelijke dataset te creëren en deze als basis voor de berekening te gebruiken. De laatste aanpak lijkt de meest betrouwbare als we in staat zijn een goede dataset te definiëren. Vermits de regulerende elementen voornamelijk stroomopwaarst van de start van het gen liggen, moeten we deze regio gebruiken als referentie om het achtergrondmodel te bepalen. In dit geval selecteren we het intergenische gebied van genen waarvan de start gekend is. Het intergenische gebied is de sequentie tussen de start van het gen en de start/einde van het stroomopwaarts gelegen gen. Op basis van deze sequenties kunnen we een betrouwbaar achtergrondmodel berekenen.

In eerste instantie testen we de verschillende achtergrondmodellen op de G-box dataset. Zoals weergegeven in Tabel 6.1, blijkt dat een onafhankelijk achtergrondmodel een veel betere performantie geeft dan een achtergrondmodel dat berekend is op basis van de sequenties in de dataset zelf. Zeker als we naar derde- en vierde-orde kijken, is dit duidelijk merkbaar. De verklaring hiervoor vinden we in Figuur 6.1 waar we de transitiematrices van de derde-orde achtergrondmodellen vergelijken. Als we de wolk van punten bekijken, zien we dadelijk dat er een duidelijk verschil is tussen de twee achtergrondmodellen. De punten die het verst afwijken van de regressielijn zijn juist die punten die komen van het motief in de dataset. Zo komt bijvoorbeeld een `G` meer dan twee keer zoveel voor na `CGT` in deze dataset dan in de intergenische sequenties. Dit maakt dat de score ven instantie van het motief door het achtergrondmodel in het geval van een onafhankelijk achtergrondmodel veel lager is dan in het andere geval. Dit verklaart dan ook het verschil in performantie.

Vermits we ook weten waar dat de G-box instanties geannoteerd zijn in deze dataset, kunnen we de gevonden posities vergelijken met de reële posities. We doen dit enkel voor de motieven gevonden met het onafhankelijke achtergrondmodel. In Tabel 6.2 zien we dat er niet echt een groot verschil is tussen de verschillende modellen. Enkel als we de enkelvoudige frequentie gebruiken als achtergrondmodel is er een kleine terugval in performantie.

In de tweede test met de G-box dataset kijken we hoe dat ruis de performantie van de MotifSampler beïnvloedt. Onder ruis verstaan we in dit geval sequenties waarin geen instantie van het G-box motief te vinden is. In een aantal opeenvolgende tests voegen we steeds tien ruissequenties toe aan de dataset. We gaan tot zestig toegevoegde sequenties wat wil zeggen dat er tot tweemaal zoveel ruis aanwezig is als sequentie die het motief bevatten. In deze sets zoeken we nu naar motieven. In Tabel 6.3 worden de resultaten samengevat. Voor iedere combinatie van achtergrondmodel en ruisniveau duiden we aan hoeveel keer een motief met gelijkaardige consensus als de G-box wordt gevonden over honderd iteraties en ook wat de positie is van het motief in de rangschikking van best scorende motieven. Het is onmiddellijk duidelijk dat een onafhankelijk achtergrondmodel veel beter is dan een achtergrondmodel gebaseerd op de sequenties in de dataset. Anderzijds blijkt dat MotifSampler met een hoger-orde achtergrondmodel goed bestand is tegen de ruis aanwezig in de dataset. Zelfs als slechts de helft van de sequenties in de dataset het motief bevatten vinden we het motief nog terug als best scorende en in bijna alle iteraties. Dit aantal ligt lager als we werken met de enkelvoudige frequentie als achtergrondmodel.

Regulons in Gist

De volgende evaluatieset bestaat uit tien regulons in gist. Een overzicht van de tien datasets wordt gegeven in Tabel 6.4. Het voordeel van deze sets is dat ze diverse types van motieven bestrijken en dus uiterst geschikt zijn om de performantie van ons algoritme te testen onder zeer verschillende randvoorwaarden. We testen MotifSampler met diverse parameter instellingen. Preliminaire testen hebben aangetoond dat een derde-orde achtergrondmodel op basis van alle gist intergenische gebieden de meest bevredigende resultaten geeft. Vermits we normaal gezien niet weten hoe lang het motief is testen we meerdere lengtes, namelijk 6, 8, 10, 12 en 14bp. Voor elke combinatie van parameters zoeken we drie verschillende motieven en herhalen we de procedure honderd keren. Vervolgens kijken we naar de best scorende motieven en rapporteren deze. Een gedetailleerd overzicht van de resultaten wordt gegeven in Tabellen 6.5 tot 6.14.

Een eerste bemerking die we na de analyse van de resultaten kunnen maken is dat ons algoritme een goed niveau van performantie en betrouwbaarheid haalt. In negen van de tien datasets is MotifSampler in staat het juiste motief te vinden en dit consistent voor meerdere lengtes van motieven. De duidelijkste voorbeelden zijn de sets MET, GCN en INO waar voor de 5 geteste lengtes steeds dezelfde motieven als best scorende teruggevonden worden. Een tweede bemerking is dat als het juiste motief niet gevonden wordt als best scorende dan wordt het juiste motief toch nog gevonden in het meest aantal iteraties. Als voorbeeld hiervan verwijzen we naar de set PDR, waar het echte motief nooit als best scorende maar steeds wel als het meest frequente gevonden wordt. In sommige gevallen, bijvoorbeeld TUP en YAP, wordt het motief wel gevonden maar is niet heel uitgesproken. Als we echter kijken naar een combinatie van de rangschikking, het aantal iteraties waarin de motieven gevonden worden en de consistentie over de verschillende motieflengtes, dan komt het echte motief steeds naar boven. Een speciaal geval is de GAL dataset, waar het echte motief bestaat uit twee korte motieven van 3bp en een variabele tussenruimte van 11bp. De consensus van dit motief is CGG-N$_1$1-CCG. Het is in dit geval niet mogelijk het juiste motief te vinden binnen de gekozen parameter grenzen. Als we echter de lengte van het motief verlengen tot 17bp, wordt het juiste motief wel teruggevonden als sterkste motief.

Hoger-orde achtergrondmodellen in prokaryoten

Een derde voorbeeld behelst de studie van hogere achtergrondmodellen in prokaryoten. Momenteel zijn er al genoomsequenties beschikbaar van meer dan honderd prokaryoten wat betekent dat we een uitgebreide vergelijkende studie kunnen maken. Eerst construeren we voor ieder organisme verschillende hoger-orde achtergrondmodellen en vergelijken we deze onderling. De vergelijking vertrekt van de berekende transitiematrices. In deze thesis belichten we twee voorbeelden die een goed beeld geven van hoe de achtergrondmodellen zich verhouden ten opzichte van elkaar. In een eerste voorbeeld vergelijken we de transitiematrix van zowel *Esscheria coli* en *Salmonella typhimurium* (zie Figuur 6.3). Het is duidelijk dat beide organismen niet alleen volgens de evolutie naaste verwante zijn maar ook volgens de samenstelling van hun sequenties. Een ander beeld krijgen we als we *E.coli* en *Streptomyces coelicolor* vergelijken (zie Figuur 6.4). In dit geval is er een groot verschil tussen de twee transitiematrices. Als we deze twee modellen zouden omwisselen dan zal het resultaat sterk beïnvloed worden. Dit effect illustreren we met een voorbeeld.

Als modelsysteem opteren we voor de σ^{54}-factor die voorkomt in verschillende organismen. De bindingsplaats heeft de -24/-14 vorm en heeft de volgende consensus: `TGGCACG-n4-TTGCWn` [9]. Een eerste set bestaat uit de promotersequenties van 15 genen in *E.coli* waarvan experimenteel bewezen is dat ze gereguleerd worden door de σ^{54}-factor. Een gelijkaardige set wordt ook aangemaakt in *Pseudomonas aeruginosa*. Met elk van deze sets voeren we een aantal tests uit om de invloed van het achtergrondmodel op de performantie van de MotifSampler te evalueren. We zoeken in beide sets voor motieven van zeven en zeventien basenparen. We testen het enkelvoudige frequentie en derde-orde achtergrondmodel van vier verschillende organismen. De resultaten worden weergegeven in Tabel 6.15 en 6.16.

De belangrijkste conclusie die we uit deze resultaten kunnen afleiden is dat een hoger-orde organisme-specifiek achtergrondmodel het beste resultaat geeft en dat het gebruik van een “verkeerd” achtergrondmodel aanleiding kan geven tot zeer afwijkende resultaten. Dit is vooral belangrijk als we promotersequenties van verschillende organismen met elkaar willen vergelijken om zo de geconserveerde motieven te detecteren.

Celcyclus in gist

Als voorbeeld van een uitgebreide analyse van microroostergegevens gebruiken we een dataset die door velen als voorbeeld is gekozen, namelijk de celcyclus in gist. De celcyclus bestaat uit een opeenvolging van vier fasen: de fase G1 waarin de cel groeit, de S-fase waarin de DNA synthese plaats vindt, de overgangsfase G2 en uiteindelijk de mitose in de M-fase. De microroostermetingen gebeurden op 18 tijdsstippen gedurende twee opeenvolgende celcycli. Na het voorbereiden van de data, zoeken we naar clusters van genen met *Adaptive Quality-Based Clustering* (AQBC) wat resulteert in achtendertig clusters. Uit deze achtendertig clusters kiezen we er vier die een specifiek profiel vertonen dat overeenkomt met de fases in de celcyclus. De eerste drie clusters (3, 4 en 28 in Figuur 6.6) vertonen een duidelijk periodiek verloop. De vierde cluster (24 in Figuur 6.6) bevat negentien genen die een hoog expressie niveau hebben bij de start van het experiment en die daarna uitgeschakeld zijn.

Van de genen in elk van deze clusters selecteren we 800bp stroomopwaarts die niet overlappen met een ander gen. In elk van de sequenties zoeken we naar motieven waarbij we de lengte variëren van 5 tot 17bp. De resultaten die we bekomen zijn sterk uiteenlopend voor de verschillende clusters. Als vergelijkingspunt nemen we de resultaten van Tavazoie et al. [102], die AlignACE gebruikten om dezelfde dataset te bestuderen. Het meest uitgesproken motief vinden we in cluster 28. Hier vinden we in alle motieven een gemeenschappelijke consensus `ACGCGT`. Deze consensus stemt overeen met het bekende MCB motief waarvan geweten is dat het een belangrijke rol speelt tijdens de celcyclus. Het andere motief dat we verwachten te vinden is het SCB motief, maar dit motief heeft een consensus die gelijkaardig is aan het MCB motief. In cluster 4 vinden we twee motieven, `TTTsGykT` en `TGTTTsTT`, die niet overeenkomen met een bekend motief. Deze motieven worden ook door AlignACE aangeduid als de meest significante motieven in deze cluster. Als we echter zoeken naar langere motieven vinden we ze niet meer terug. Cluster 3 geeft daarentegen geen bevredigend resultaat. In dit geval vinden we enkel motieven die als consensus een opeenvolging van A’s of T’s zijn en dus niet echt een regulatorische functie hebben. Een test met RSA-tools [112] geeft de volgende twee motieven als meest overgerepresenteerde motieven: `TGAAAAAT` en `AAAATTT`. Het eerste motief vertoont een zekere overeenkomst met het RRPE motief. Het tweede motief komt overeen met het motief dat we vinden bij de lengtes 7 en 8bp. De laatste cluster (24 in Figuur 6.6) geeft voor de korte motieflengtes een gemeenschappelijke consensus `ATGAAAC`. Dit motief vertoont grote gelijkenis met het STE12 motief dat we in SCPD vinden. Verdere analyse van dit resultaat geeft aan dat een van de genen waarvan bewezen is dat het gereguleerd wordt door STE12 ook in de cluster gevonden is. Dit is een indicatie dat het motief dat gevonden is waarschijnlijk ook effectief van belang is voor de genen in deze cluster. Voor de langere motieflengtes vinden we een sterk motief, `ATATATGnnTCAGATA`, dat gevonden wordt in 7 genen.

Zoeken naar gekende regulerende elementen

In Hoofdstuk 5 en 6 bespreken we de detectie van ongekende motieven vertrekkende van een set van sequenties waarin een waarin een vermoedelijk motief verborgen is. Vermits er echter ook bekende motiefmodellen voorhanden zijn, is het ook mogelijk instanties te zoeken van deze gekende motieven. Hiervoor hebben we een algoritme ontwikkeld dat gebaseerd is op de kernmodules die ook aanwezig zijn in MotifSampler. Meer bepaald gebruiken we het probabilistische model om het aantal instanties te schatten in een sequentie.

Anderzijds ontwikkelen we als referentiesysteem een aangepaste versie van het klassieke schema om te scoren met een positie-gewichtsmatrix.

MotifLocator

Om te scoren met een positie-gewichtsmatrix introduceren we *MotifLocator* en hiervoor gebruiken we een score van de vorm

$$W(\mathbf{x}) = \log\left(\frac{P(\mathbf{x}|\boldsymbol{\Theta})}{P(\mathbf{x}|S,\mathcal{B}_m)}\right) = \sum_{j=1}^{W}[\log(\boldsymbol{\Theta}(b_j,j)) - \log(P(b_j|S,\mathcal{B}_m))].$$

Deze score komt overeen met de scores die we berekenen in de predictieve scoringsfunctie van de MotifSampler. Ieder segment in de dataset wordt dus gescoord met zowel het motiefmodel $\boldsymbol{\Theta}$ als met het achtergrondmodel $\mathcal{B}_m$. Alle scores $W(\mathbf{x})$ worden genormaliseerd als $\frac{W(\mathbf{x})-W_{min}}{W_{max}-W_{min}}$. Om een selectie te maken van de instanties wordt een drempelwaarde ingesteld en alle segmenten met een score groter dan die drempelwaarde worden weerhouden. In Figuur 7.2 geven we aan hoe dat die scores verdeeld zijn voor het MCB motief in alle stroomopwaartse sequenties in gist. De grafiek duidt ook aan dat het aantal geselecteerde instanties exponentieel stijgt met een verlaging van de drempelwaarde.

MotifScanner

De andere methode die we voorstellen is *MotifScanner*. Dit algoritme is gebaseerd op de methode om het aantal instanties te schatten in een sequentie op basis van het probabilistiche sequentiemodel. De methode kunnen we als volgt samenvatten:

1. Score elk segment $\mathbf{x}_l$ in S met het motiefmodel $\boldsymbol{\Theta}$.
2. Score elk segment $\mathbf{x}_l$ in S met het achtergrondmodel $\mathcal{B}_m$.
3. Initialiseer de prior distributie $[1-\gamma_1,\gamma_1]$.
4. Bereken $P(Q=0|S,\boldsymbol{\Theta},\mathcal{B}_m)$ en $P(Q=1|S,\boldsymbol{\Theta},\mathcal{B}_m)$
5. Zolang dat $P(Q=i|S,\boldsymbol{\Theta},\mathcal{B}_m) > \epsilon$
    (a) verhoog i
    (b) herreken $P(Q=c|S,\boldsymbol{\Theta},\mathcal{B}_m)$ voor $c = 0\ldots i$.
6. Bereken het verwachte aantal instanties als $E_{(S,\boldsymbol{\Theta},\mathcal{B}_m)}[Q]$.
7. Selecteer de Q posities met de beste score als instanties.

Invloed van de sequentielengte

Analyse van verschillende datasets heeft aangetoond dat de lengte van de sequentie een niet onbelangrijke invloed hebben op het zoeken naar motieven. Omdat we werken met het probabilistic sequentiemodel speelt de context waarin de motieven verborgen zijn een belangrijke rol in MotifScanner. Deze invloed kunnen we als volgt verklaren. Als de sequentie, waarin de motiefinstanties verborgen zijn, langer wordt dan zal de verhouding van signaal ten opzichte van ruis verkleinen en het wordt dus moeilijker om dit signaal op te pikken. Hoe dit in ons model zit wordt verduidelijkt in Figuur 7.5 waar zowel de probabiliteit van een sequentie gegenereerd door het achtergrond model als de probabiliteit dat de sequentie gegenereerd wordt door het model als het aantal instanties c gegeven is. Als illustratie van het effect van de sequentielengte zoeken we naar instanties van SP1, TBP en E2F in 4000 menselijke promotersequenties. Figuur 7.4 toont de resultaten van MotifLocator en MotifScanner. Het is duidelijk dat de drie motieven elk een ander gedrag vertonen. Eerst en vooral zien we dat met MotifLocator het aantal instanties lineair stijgt met de motieflengte. In geval van de MotifScanner zien we een trend waarbij het aantal gevonden instanties niet meer evenredig stijgt met de groeiende lengte. In sommige gevallen worden er zelfs minder instanties gevonden. Ook de onderlinge verhouding tussen de twee methodes verschilt van motief tot motief. Bijvoorbeeld in het E2F voorbeeld worden steeds meer instanties gevonden met MotifScanner dan met MotifLocator terwijl in het TBP voorbeeld dit net andersom is.

Analyse van promotersequenties

De recente beschikbaarheid van volledig in kaart gebrachte genomen laat toe op grote schaal deze genomen te bestuderen. Een aspect dat binnen dit onderzoek een belangrijke plaats inneemt zijn de promotersequenties. Daarom bestuderen we twee datasets: alle stroomopwaartse regio's in gist en een selectie van 4000 stroomopwaartse sequenties van 2000bp uit het menselijke genoom. We screenen alle stroomopwaartse sequenties in gist met de matrixmodellen uit SCPD [124]. In de tweede test zoeken we naar de instanties van een aantal gewervelde motiefmodellen uit Transfac [117].

Vooreerst bestuderen we de nucleotidecompositie in de geselecteerde set van stroomopwaartse sequenties. Daartoe aligneren we de sequenties aan de startpositie en tellen op iedere positie het aantal nucleotiden. Dit resulteert in de samenstelling zoals voorgesteld in Figuur 7.6 voor gist en Figuur 7.10 voor mens. In gist is de samenstelling constant over de volledige stroomopwaartse regio. Enkel in het gebied vlak voor de start van het gen is er een kleine wijziging. In mens zien we daarentegen een sterk wijzigende samenstelling. In het gebied ver stroomopwaarts gelegen is de samenstelling duidelijk AT-rijk, dit verandert naar een GC-rijk gebied in de 300bp vlak voor de start van het gen. De conclusie die we hieruit kunnen trekken is dat als we naar een motiefmodel op zoek zijn, we rekening dienen te houden met de gebied waarin we verwachten dat het motief te vinden zal zijn. Zo is er een duidelijk verschil tussen de proximale promoter in mens, die GC-rijk is, en de distale promoters die een andere samenstelling hebben.

Nu dat we de promotergebieden goed gedefinieerd hebben, kunnen we op zoek gaan naar de instanties van de motieven waarover we beschikken. Een interessant gegeven is om te bekijken waar die instanties gevonden worden. Om dit visualiseren berekenen we voor alle instanties van een motief de positie relatief ten opzichte van de start van het gen. Deze posities worden dan uitgezet in histogram. Voor gist rapporteren we de distributieplots van alle vierentwintig motieven uit SCPD in Figuren 7.7, 7.8 en 7.9. Als we deze plots vergelijken dan merken we dat er een aantal klassen zijn. Vooreerst zijn er twee motieven, ABF1 en REP1, die een duidelijke voorkeur vertonen voor een bepaald gebied. Andere motieven, zoals MCB, ROX1, SCB, TBP en UASPHR, vertonen een lichte stijging in aantal gevonden instanties dichter bij de start van het gen. De meeste motieven vertonen een eerder uniforme distributie over het volledige stroomopwaartse gebied. Tenslotte zijn er nog een aantal motieven, zoals GAL4, MIG1, PDR1/PDR3, RLM1 en SMP1, waarvan er slechts een beperkt aantal instanties gevonden wordt in de meer dan 6400 sequenties. In de menselijke promoters bespreken we twee voorbeelden uit de meer dan 400 motieven, namelijk SP1 en TBP. SP1 vertoont een distributie die een piek vertoont in het gebied -150/-50 relatief ten opzichte van de start van het gen. Voor TBP de distributie is meer uniform met een trend voor meer instanties in het gebied het verst verwijderd van de start van het gen. TBP is dan ook een AT-rijk motief terwijl het gebied vlak voor het gen eerder GC-rijk is.

Statistische significantie

Als we beschikken over een goede set van motiefmodellen, dan kunnen we ook uitzoeken welke motieven significant meer instanties hebben in een bepaalde set van genen dan we zouden verwachten vergeleken met een referentie set. Om de statistische significantie te berekenen gebruiken we de methodologie zoals die ook voorgesteld werd door van Helden et al. [111]. Eerst berekenen we de verwachte frequentie $F_e\{\mathbf{\Theta}\}$ van een motief $\mathbf{\Theta}$ door een referentieset te screenen. Hiermee kunnen we verwacht aantal instanties in de dataset berekenen:

$$E(\mathrm{inst}\{\mathbf{\Theta}\}) = F_e\{\mathbf{\Theta}\} \times 2 \times \sum_{k=1}^{N_s}(L_k - W + 1) = F_e\{\mathbf{\Theta}\}T.$$

Als we een binomiaal model gebruiken dan kunnen we de probabiliteit om n of meer instanties te vinden in de dataset berekenen als

$$P(\mathrm{inst}\{\mathbf{\Theta}\} \geq n) = 1 - \sum_{j=0}^{n-1} \frac{T!}{(T-j)! \times j!} \times (F_e\{\mathbf{\Theta}\})^j \times (1 - F_e\{\mathbf{\Theta}\})^{T-j}.$$

Dit geeft ons een p-waarde waarop we een drempelwaarde kunnen instellen. van Helden et al. [111] stelden voor om deze p-waarde om te vormen tot een significantiecoëfficiciënt $\mathrm{sig} = -\log_{10}(P(\mathrm{inst}\{\boldsymbol{\Theta}\} \geq n) \times D)$, waarbij D het aantal verschillende motieven is. Deze methodologie werd door Aerts et al. [1] in TOUCAN verwerkt.

Als voorbeeld gebruiken we de tien regulons en de clusters uit de celcyclus in gist die we ook al gebruikt hebben om MotifSampler te evalueren. Eerst berekenen we de verwachte frequentie van al de motieven uit SCPD in de stroomopwaartse sequenties in gist. Hierbij moeten we wel rekening houden met de prior γ_1. Zoals aangegeven is in Tabel 7.3 heeft een verandering in prior een sterk effect op de verwachte frequentie. De resultaten van deze analyse hebben we samengevat in Tabellen 7.4 en x7.5. De algemene conclusie is dat deze aanpak goede resultaten oplevert als de juiste matrixmodellen beschikbaar zijn. In de regulons hebben de echte motieven in de dataset duidelijk een veel hogere significantiecoëfficiciënt dan de andere motieven. Als echter juiste motiefmodel niet beschikbaar is dan is het dus onmogelijk de juiste oplossing te vinden.

INCLUSive: geïntegreerde webapplicatie

Eén van de betrachtingen van ons onderzoek was om applicaties te ontwikkelen die de bioloog kunnen assisteren bij de verwerking van experimentele data. In eerste instanties resulteerde dat in een aantal prototypes van algoritmen die al wel gebruikt konden worden maar die nog verdere verfijning nodig hadden. Om deze algoritmen beschikbaar te maken aan de potentiële gebruikers hebben geöpteerd om ze aan te bieden via een webpagina. Deze aanpak laat toe om een direkte interactie op te zetten met de eindgebruiker die de applicatie al kan testen. Dit leverde ons reeds vanaf de opstart een continue stroom van informatie van de gebruikers die ons toelaat om de algoritmes beter aan te passen aan de noden van diezelfde gebruiker. Na verloop van tijd bleek ook dat de belangrijkste vraag vanuit gebruikerskant was om de beschikbare applicaties met elkaar te integreren om zo de analyse stappen verder te vereenvoudigen. Dit leidde tot de implementatie van *INCLUSive* [108].

Om de werking van INCLUSive te illustreren, analyseren we een klein microroosterexperiment in *Arabidopsis*. In dit experiment wordt het effect op de genexpressie van een mechanische verwonding in de bladeren gemeten over acht verschillende tijdspunten [86]. De dataset bestaat uit 150 genen die een rol spelen in het beschermingsmechansime van de plant en ook nog 16 controlegenen. De identificatie van de genen gebeurt aan de hand van hun accessienummer en gennaam.

We starten de analyse met het toepassen van AQBC [26] op de expressieprofielen. Als parameters kiezen we 0.95 voor het kwaliteitscriterium en we verwachten dat er minstens 4 genen in de cluster zijn. Dit resulteert in de zes clusters die afgebeeld worden in Figuur 8.3. Voor elk van de genen in die clusters moeten we nu het stroomopwaartse gebied definiëren. Een methode om dit te doen is het gen lokaliseren op de genoomsequentie en dan op basis van de annotatie de promotersequentie te extraheren. Deze methode is natuurlijk afhankelijk van de kwaliteit van de annotatie. Nu recent steeds meer volledige genomen beschikbaar zijn, kunnen we ook beroep doen op deze specifieke genoomdatabanken.

Eens dat we de set van promotersequenties geselecteerd zijn kunnen we zowel MotifSampler als MotifScanner gebruiken om overgerepresenteerde motieven te zoeken. De resultaten worden weergegeven in de zes tabellen (8.2 tot 8.7). We geven hier telkens de vijf best scorende motieven aan voor de 8, 10 en 12bp. In vijf van de zes clusters vinden we motieven die in voldoende iteraties gevonden worden en ook consistent over de verschillende lengtes. Maar deze resultaten zijn minder uitgesproken dan de resultaten in de voorbeelden van Hoofdstuk 6. Een zelfde resultaat wanneer we motieven zoeken met de MotifScanner. Ook hier vinden we enkele significante motieven, maar ze zijn zeker niet zo significant als in de voorbeelden in Hoofdstuk 7.

Om het hoofdstuk over INCLUSive te eindigen, geven we een overzicht van het gebruik van INCLUSive door andere onderzoekers. Eerst en vooral kunnen we op gebruikersaantallen wijzen. Ongeveer 200 verschillende gebruikers hebben de AQBC getest wat resulteerde in meer 1900 submissies. De website van MotifSampler is al iets langer beschikbaar en hier zitten we dan ook al aan meer dan 500 verschillende gebruikers die ongeveer 10.000 verschillende tests hebben uitgevoerd. Terwijl de meeste van die analyses simpele tests waren van de software, hebben sommige van die analyse ook tot effectieve resultaten geleid. Getuige hiervan zijn zes publicaties waarin resultaten bekomen met MotifSampler beschreven zijn. Er zijn een aantal publicaties waarin een microroosterexperiment in *Arabidopsis* beschreven wordt [21, 51, 72, 87]. Ook in gist is de MotifSampler een bruikbaar instrument gebleken voor de studie van de YRR1 transcriptiefactor [56]. Tenslotte hebben Ohler et al. [80] MotifSampler toegepast in de studie van promotersequenties in *Drosophila melanogaster*.

Conclusies

In dit doctoraatsonderzoek hebben we gewerkt aan een set van algoritmen die de biologen moeten assisteren in de zoektocht naar regulerende elementen in DNA-sequenties. We hebben zowel methodes ontwikkeld die zoeken naar instanties van gekende regulerende elementen, *MotifLocator* en *MotifScanner* als een algoritme, *MotifSampler*, waarmee we zoeken naar statistisch overgerepresenteerde motieven in de promotersequenties van co-gereguleerde genen.

Contents

Voorwoord **i**

Abstract **iii**

Samenvatting **v**

Nederlandse samenvatting **vii**

Contents **xxix**

Notation **xxxv**

Publication List **xxxvii**

1 Introduction **1**

1.1 From expression to regulation: integrated microarray data analysis 1

1.2 Chapter-by-Chapter overview 5

2 Concepts of Gene Regulation **9**

2.1 DNA: the code of life 9

2.1.1 Genes and proteins 10

2.1.2 Cellular structure 12

2.1.3 Organization of genes on the genomic DNA 13

2.2 The process of gene expression 15

2.3 Gene regulation 17

2.4 Regulatory proteins 19

2.5 Measuring gene expression 21

2.5.1 cDNA microarray technology 22
2.5.2 DNA-chip technology 23
2.6 Applications of microarray 24
2.6.1 Some issues on preprocessing 24
2.6.2 Finding groups of coexpressed genes 25
2.6.3 Deciphering the regulatory mechanism 25
2.6.4 Reconstruction of genetic networks 26
2.6.5 Clinical usage 26
2.7 Conclusions 27

3 Current State of Motif Finding Algorithms 29
3.1 Terminology 29
3.2 Modeling transcription factor binding sites 30
3.2.1 Example: G-box binding factor in plants 30
3.2.2 String-based representation 31
3.2.3 Matrix representation 31
3.2.4 Sequence Logo: visual representation 34
3.3 Localizing instances of known motif models 34
3.3.1 String-based searches 35
3.3.2 Matrix-based searches 35
3.4 Searching in sets of upstream sequences 37
3.4.1 String-based approaches 38
3.4.2 Matrix-based methods for motif finding 40
3.4.3 Combined and extended methods 43
3.5 Motif finding on the net 46

4 Gibbs Sampling for Motif Finding 49
4.1 Markov Chain Monte Carlo (MCMC) 49
4.1.1 Metropolis algorithm 50
4.2 Gibbs Sampler 51
4.2.1 Basic principle 51
4.2.2 Grouping and collapsing variables 52
4.3 Missing data problem 52
4.3.1 Basic principles 53
4.3.2 Expectation-Maximization 54
4.3.3 Gibbs sampling and data augmentation 54

4.4 Gibbs sampling for motif finding 55
4.4.1 Basic sequence model 55
4.4.2 Notations 56
4.4.3 Deploying a collapsed Gibbs sampling scheme 57
4.4.4 Predictive update formula 58
4.4.5 Algorithm 59
4.4.6 Main problems 60
4.5 Conclusions 61

5 MotifSampler: implementation and performance analysis[2] 63
5.1 Higher-order background models 64
5.1.1 Construction of the transition matrix 64
5.1.2 Intergenic sequences 66
5.1.3 Extended sequence model and predictive update formula 67
5.2 Estimating the number of copies 68
5.3 Core sampling procedure 71
5.4 Resulting motif model 71
5.4.1 Motif scores 72
5.4.2 Comparison of different motif models 72
5.5 Discussion of the algorithm performance 73
5.5.1 Data sets for performance testing 73
5.5.2 Running time analysis 74
5.5.3 On the convergence of the MotifSampler 75
5.5.4 Distribution of motif scores 77
5.5.5 Effect of the motif length 80
5.5.6 Estimation of number of motif instances 81
5.6 Elaboration of a motif finding strategy 83
5.6.1 Convergence steps 84
5.6.2 Shifting the alignment vector 84
5.6.3 Inclusion of the complementary strand 85
5.6.4 Finding different motifs 85
5.6.5 MotifSampler 86
5.6.6 Analyzing repeated runs 86

6 MotifSampler: case studies 89
6.1 G-box[3] 90

6.1.1 Data set 90
6.1.2 Creation of the background model 90
6.1.3 Influence of the background model 91
6.1.4 Predicted motif positions 92
6.1.5 Robustness in presence of noise 93
6.1.6 Discussion 95
6.2 Yeast regulons: analysis 95
6.2.1 Data sets 95
6.2.2 Experimental setup 96
6.2.3 Results 97
6.2.4 Discussion 104
6.3 Higher-order background models: A case study in prokaryotes[4] 105
6.3.1 Comparison of the background models 105
6.3.2 Influence of the background model on the motif detection 106
6.4 Cell cycle in yeast 111
6.4.1 Clusters of coexpressed genes 111
6.4.2 Motif finding 112
6.5 Conclusion 116

7 Searching for known transcription factor binding sites[5] 119
7.1 PWM-based approach to locate motif instances 119
7.2 Using the probabilistic model to locate motif instances 120
7.2.1 Computational complexity 122
7.3 Comparison of MotifLocator and MotifScanner 122
7.3.1 Threshold vs. prior 122
7.3.2 Influence of the sequence length 123
7.3.3 Discussion 127
7.4 Large scale promoter analysis 127
7.4.1 Yeast upstream sequences 128
7.4.2 Human promoter sequences from Ensembl 129
7.4.3 Discussion 138
7.5 Searching for known motifs in sets of coregulated genes 139
7.5.1 Finding statistical significant instances 139
7.5.2 Expected frequency in yeast genome 140
7.5.3 Yeast regulons 140

7.5.4 Cell cycle clusters 142
7.6 Conclusions 143

8 INCLUSive: Integrated Search for Regulatory Elements[6] 145

8.1 Basics of INCLUSive 145
8.2 Defining groups of coexpressed genes 147
8.2.1 Adaptive quality-based clustering 147
8.2.2 Clustering example: mechanical wounding in plants . . 147
8.3 Upstream sequence retrieval 150
8.3.1 Parsing of annotated sequences 150
8.3.2 Gene indexing and upstream region identification 151
8.3.3 Aligning genes on genomic sequences 153
8.4 MotifSampler 154
8.4.1 MotifSampler web interface 154
8.4.2 Motif finding in plant wounding clusters 156
8.5 MotifScanner 160
8.5.1 Web interface 160
8.5.2 Example: plant wounding clusters 160
8.6 Application of INCLUSive 163
8.7 Conclusions 165

9 Conclusions 167

9.1 Accomplishments 167
9.2 Future research directions 170

A. Glossary 173

Bibliography 177

Notation

List of symbols

b	nucleotide, A,C,G or T
S	DNA sequence
L	length of the DNA sequence
$\mathcal{S}$	set of DNA sequences
N_S	number of sequences in set $\mathcal{S}$
S_k	the k-th sequence in set $\mathcal{S}$
L_k	length of sequence S_k
l	index in sequence, with $1 \leq l \leq L$
b_l	nucleotide b at position l in sequence S
W	motif length
$\mathbf{x}$	oligonucleotide of length W
$\mathbf{\Theta}$	position-specific frequency matrix of length W
θ_i	nucleotide probabilities at position i in PSFM $\mathbf{\Theta}$
$q_{b,i}$	gives the probability of finding b at position i in the motif
$\mathcal{B}_m$	background model of order m

Degenerate consensus symbols

M	A or C
R	A or G
W	A or T
S	C or G
Y	C or T
K	G or T
B	C, G or T
D	A, G or T
H	A, C or T
V	A, C or G
N	A, C, G or T

Acronyms

BLAST	Basic local alignment search tool
cDNA	Complementary DNA
CS	Consensus score
DNA	Deoxyribonucleic acid
GBF	G-box binding factor
HMM	Hidden Markov Model
IC	Information content
LL	Log-likelihood score
mRNA	Messenger RNA
PSFM	Position Specific Frequency Matrix
PWM	Position Weight Matrix
RNA	Ribonucleic acid
RNAP	RNA polymerase
SNF	Single nucleotide frequency
TBP	TATA-binding protein
TF	Transcription factor
TFBS	Transcription factor binding site
TLS	Translation start
TSS	Transcription start

Publication List

Most of the work discussed in this thesis has been published in one of the following publications.

International Journals

- Thijs G., Lescot M., Marchal K., Rombauts S., De Moor B., Rouzé P., Moreau Y., 2001. *A higher-order background model improves the detection by Gibbs sampling of potential promoter regulatory elements*, Bioinformatics, **17**(12),1113-1121.
- Thijs G., Marchal K., Lescot M., Rombauts S., De Moor B., Rouzé P., Moreau Y. 2002. *A Gibbs Sampling method to detect over-represented motifs in upstream regions of coexpressed genes*, Journal of Computational Biology, special issue RECOMB'2001, **9**(2), 447-464.
- Lescot M.. Déhais P., Thijs G., Marchal K., Moreau Y., Van de Peer Y., Rouzé P., Rombauts S., 2002. *PlantCARE, a database of plant cis-acting regulatory elements and a portal to tools for in silico analysis of promoter sequences*, Nucleic Acids Research, **30**(1), 325-327.
- Thijs G., Moreau Y., Smet F. D., Mathys J., Lescot M., Rombauts S., Rouzé P., De Moor B., Marchal K. 2002. *INCLUSive: INtegrated CLustering, Upstream sequence retrieval and motif Sampling*, Bioinformatics, **18**(2), 331-332.
- De Smet F., Mathys J., Marchal K., Thijs G., De Moor B., Moreau Y., 2002. *Adaptive quality-based clustering of gene expression profiles*, Bioinformatics, **18**(5), 735-746.
- Moreau Y., De Smet F., Thijs G., Marchal K., De Moor B. 2002. *Functional bioinformatics of microarray data: from expression to regulation*, Proceedings of the IEEE, **90**(11), 1722-1743.
- Marchal K., Thijs G., De Keersmaecker S., Monsieur P., De Moor B., Vanderleyden J., 2003. *Genome-specific higher-order background models to improve motif detection*, Trends in Microbiology, **11**(2), 61-66.

- Aerts S., Thijs G., Coessens B., Staes M., Moreau Y., De Moor B., 2003. *Toucan: Deciphering the cis-regulatory logic of coregulated genes*, Nucleic Acids Research, **31**(6), 1753-1764.

International Conference

- Thijs G., Moreau Y., Rombauts S., De Moor B., Rouzé P., 1999. *Recognition of gene regulatory sequences by bagging of neural networks*, In Proceedings 9th Intl. Conf. on Artificial Neural Networks, Edinburgh (Scotland), pages 988-993.
- Thijs G., Rombauts S., Lescot M., Marchal K., De Moor B., Moreau Y., Rouzé P., 2000. *Detection of cis-acting regulatory elements in plants: a Gibbs sampling approach*, In Proceedings of 2nd Bioinformatics of Gene Regulation (BGRS'2000), Novosibirsk (Russia), pages 119-123.
- Thijs G., Marchal K., Lescot M., Rombauts S., De Moor B., Rouzé P., Moreau Y., 2001. *A Gibbs Sampling method to detect over-represented motifs in upstream regions of coexpressed genes*, In Proceedings 5th Annual Intl. Conf. on Computational Biology (RECOMB'2001). Montreal (Canada), pages 296-301.
- Lescot M. , Rombauts S., Thijs G., Marchal K., De Moor B., Moreau Y., Rouzé P., 2001. *In silico search of plant cis-acting regulatory elements*, In Proceedings Journées ouvertes biologie informatique et mathematique (JOBIM), Toulouse (France).
- Moreau Y., Thijs G., Marchal K., De Smet K., Mathys J., Lescot M., Rombauts S., Rouzé P., De Moor B., 2002. *Integrating quality-based clustering of microarray data with Gibbs sampling for the discovery of regulatory motifs*, In Proceedings Journées ouvertes biologie informatique et mathematique (JOBIM), St-Malo (France), pages 75-79.

CHAPTER 1

Introduction

1.1 From expression to regulation: integrated microarray data analysis

In 1998, Elsevier published a supplement to their Trends journals entitled '*Trends guide to bioinformatics*'. In this visionary guide, several well respected researchers gave an overview of the state-of-the-art and cutting edge methods in computational biology and bioinformatics. They also tried to make some predictions on the most interesting problems at hand in the coming years. The main focus in this guide was on gene finding and database searches. The main reason for this view was that the complete human genome was not yet available and it would not be completely available to the research community for many more years. In 1998 the forecast was that the human genome would be fully sequenced in 2005 and that of mouse in 2008. However, due to fierce competition between the publicly funded *international human genome initiative* and the private company *Celera*, less than three years later these predictions were already overruled by the announcement of the publications of the first draft of the human genome at a joint press conference on February 12, 2001 [103, 115] . Since then the genome of several higher eukaryotes, among them the model organisms Arabidopsis, fruitfly and mouse, have been fully sequenced.

In the late nineties genomics research was at the dawn of a new area: the introduction of high-throughput experiments in molecular biology. Not only sequencing technology boosted at that time, but also microarray technology was introduced in many laboratories. Microarray technology suddenly allowed researchers to have a glance at the complete transcriptome of an organism in one experiment. Such a wealth of information in one experiment has opened new research directions in molecular biology and genomics. Up to a few years ago there was an adage in molecular biology that said ”*one gene, one PhD*”. This meant that most researchers were focusing on one or, at most, a limited set of genes and they tried to learn all there was to know about this particular gene. Nowadays, given the plentitude of available data and new technologies, researchers can move to the study of systems at a higher level and should try to glue together the knowledge about several specific genes into a more global view. To facilitate the handling of this huge amount of data, there needs to be very specific and advanced tools and that is where computational biology or bioinformatics comes into play.

These technological advances have numberless implications in bioinformatics research. In the beginning of the nineties bioinformatics research was mostly directed towards sequence analysis with an emphasis on gene detection. Trying to identify the genes in newly sequenced DNA was an important problem then since not so much information about genes was available. Since the continuous completion of new genomes, the emphasis in sequence analysis has shifted from gene prediction to cross-species sequence comparison. If we look at the research articles published in a specialized journal as *Bioinformatics* nowadays, most of these articles focus on the computational aspects of microarray analysis. This ranges from probe design for gene-chips, over image analysis to the use of expression data to classify disease patterns and predict gene networks. For each of these tasks the biologists need specialized tools that assist them in the study of various problems.

One of the interesting applications of microarrays is the ability to study the process of transcriptional gene regulation. The basic assumption is that genes that show a similar expression profile in an experiment might be regulated by the same transcriptional regulators. The flowchart of the process is given in Figure 1.1. The procedure is started from several microarray experiments. Typically these are time course experiments in which the expression of the genes is measured at several time points in the process under study. These expression data are then preprocessed to asses the quality of the measurements and to filter the measurements. The remaining data are then clustered to find groups of genes that show a similar expression profile over the experiment. Next, for each gene in a selected cluster the promoter or upstream region is identified. In the next step, statistically overrepresented motifs are retrieved from such a set of sequences as potential transcription factor binding sites. These potential binding sites could give some further information on the function of the genes. To close the loop, the results of the *in-silico* analysis should be coupled back to the biological experiment. From the functional classification of the genes in the clusters and the potential transcription factor binding sites, the biologist should be able to define new hypotheses that can be be verified in a wet lab experiment. It is within this framework of the analysis of expression data linked to the study of transcriptional regulation that our main contributions are situated.

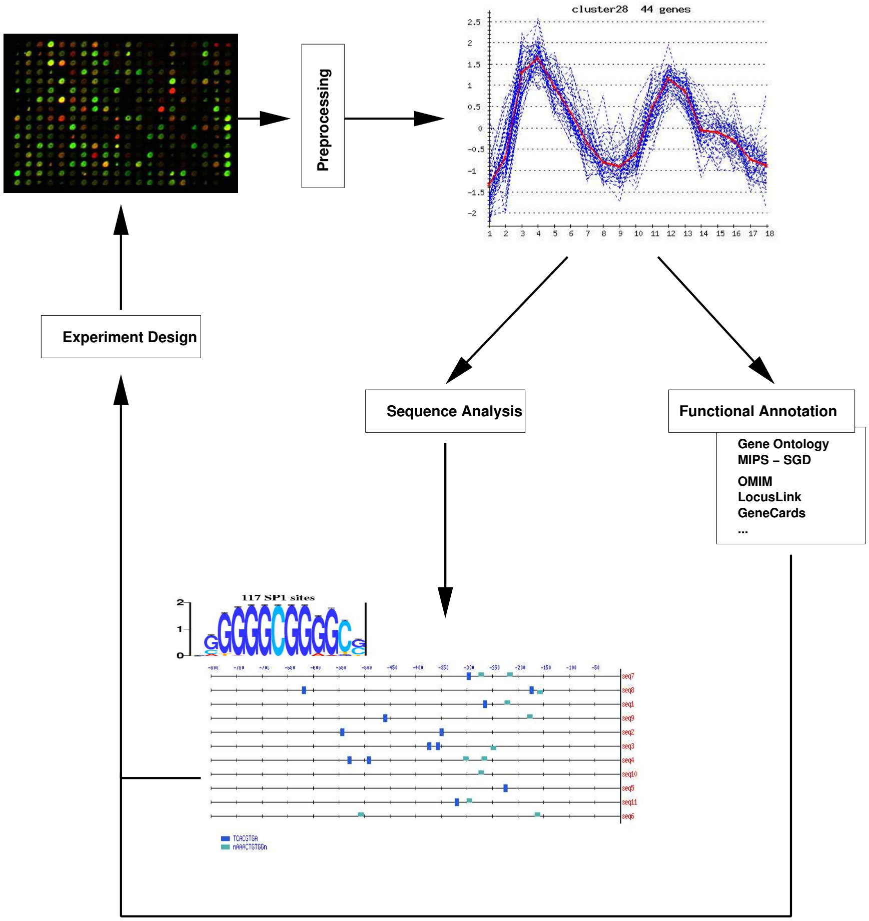


Figure 1.1: Flowchart of the analysis of gene expression data to find transcription factor binding sites specific to a set of genes that share the same expression profile. The process start from several microarray experiments. The data in these experiments are then preprocessed and clustered to find groups of genes that show a similar expression profile over the experiment. The next step is the analysis of the clusters. First, several sources of information are consulted to functionally annotate the genes in the cluster. To study the cluster at a sequence level, the promoter region is identified. Statistically overrepresented motifs, that might be potential transcription factor binding sites, are retrieved from such a set of sequences. These motifs are also checked and functionally annotated. To close the loop, the hypothesis generated *in silico* should be tested in a new wet lab experiment.

Deliverables and accomplishments

As there are many aspects to the analysis of microarray data, as depicted in Figure 1.1, we decided to focus on a few topics. In this thesis, our main focus is on the detection of transcription factor binding sites in the upstream or promoter region of genes that are coregulated. To solve the problem of finding transcription factor binding sites there are two approaches: supervised and unsupervised. In the supervised approach, known examples are used to construct a model of the binding site. With this model new sequences can be scored and potential binding sites can be retrieved. In the unsupervised approach, the motif model is learned from a set of sequences in which there might be a common binding site. In this work we have explored the possibilities and limitations of both approaches.

In the very first phase of our research, we tried to construct a neural network model to detect specific transcription factor binding sites [107]. However, this approach did not work as expected mainly because of the lack of sufficient data to train the models. Therefore, we have decided to move to the unsupervised setting where we start from available microarray experiments. In this case, we search for overrepresented motifs in the upstream region of such a set of coexpressed genes. The rationale behind this search is that genes that share a similar expression profile over an experiment might be controlled by the same transcriptional regulators and thus share similar binding sites. Our approach to solve this type of problem is embedded in a probabilistic framework. We have extended the original Gibbs sampling method for motif finding with an higher-order background model and included a probabilistic method to estimate the number of motif instances. We did not only spend much time on the theoretical and algorithmic aspects of the sampling procedure but we also elaborated on the practical implementation of the algorithm. This results in an implementation called MotifSampler [69, 104, 106]. After the detection of potential binding sites it might be useful to locate instances in other sequence. This brings us back to the supervised approach. From the components in MotifSampler two algorithms emerged that can be used to screen sequences with known motif models MotifLocator and MotifScanner. MotifLocator is a method based on a classical position weight matrix scoring scheme and MotifScanner uses the probabilistic sequence model to estimate the number of instances [1].

To facilitate the use of the algorithms designed during our research, we have set up a web based system where interested users can submit their data and perform part of the work flow as outlined in Figure 1.1. At first, we have built user friendly interfaces to the different algorithms like adaptive quality-based clustering [26] and MotifSampler and MotifScanner. In a next step, we have tried to provide the link between the clustering and the sequence analysis by implementing an upstream sequence retrieval system. This suite of web tools is called INCLUSive [108]. In a later phase, when our algorithms have grown beyond the prototypical designs, also stand alone version of the motif finding tools have been made available to the interested users. The suite of tools part of INCLUSive is accessible from our web site:

`http://www.esat.kuleuven.ac.be/~dna/BioI/Software.html`

That our methods are applicable in real biological examples and can produce meaningful results is illustrated in Chapter 8. There we will refer to the work of others who have been using these tools for the analysis of their specific data sets [21, 51, 56, 72, 80, 87].

1.2 Chapter-by-Chapter overview

Let us now look at how this text of this thesis is structured. This thesis starts with the concepts of gene regulation and the current status of motif finding methods. Next, we introduce the basics of sampling methods and the Gibbs sampler method for motif finding on which our algorithm is based. In the next chapters we move to the specificities of our implementation **MotifSampler** and illustrate the implementation with four case studies. Building on the probabilistic sequence model we then present two methods to find instances of known motif models in sequences. Finally, we end the dissertation with the description of our integrated online system to analyze microarray data, **INCLUSive**.

Chapter 2. Concepts of gene regulation

We start this dissertation with a short introduction to the most important biological concepts needed to understand the problem at hand. There are two main parts in this chapter. First we introduce the different components involved in the process of gene expression and regulation. Starting from DNA, we describe the cellular structure in prokaryotes and eukaryotes. Next, the process of gene expression is introduced. In the next section we focus more specifically on the transcriptional regulators that control the gene expression. As an exmple of gene regulation we describe the well known *lac* operon in *E. coli*. Finally, several examples are given of how transcription factors can interact with the DNA to regulate the transcription process.

In the second part of this chapter, we discuss the technology of microarrays that has been used to measure gene expression on a genomic scale. There exists two basic technologies: cDNA microarrays and DNA chips. We give the an introduction to both technologies and pinpoint their advantages and limitations. These expression measurements have many possible applications both on the theoretical as on the more practical level. We give a quick overview of potential applications. Our work is situated in the study of the transcriptional mechanism based on expression measurements.

Chapter 3. Motif finding in sets of coregulated genes

Once we have introduced some of the basic concepts of biology, we move on to the *in silico* analysis of *cis*-acting regulatory elements. To this end, we first define the different representations of transcription factor binding sites which can be divided in string-based and matrix-based representations. To search binding sites in DNA sequences there exists two approaches: (1) supervised, where we search for a known model and (2) unsupervised methods in which motifs are learned from specific sequence sets. We present the current status of methods designed to solve both types of problems. The emphasis is on the particularities of each method with respect to the other methods. First we give an overview of the methods to locate known transcription factor binding sites in a DNA sequence. Next, we introduce the methods that are designed to find statistically overrepresented motifs in sets of promoter sequences of coregulated genes. Once we have introduced the methods, we move to the extended and combined methods.

Chapter 4. Gibbs sampling for motif finding

One particular method to find over-represented motifs in sets of coregulated genes is Gibbs sampling. In this chapter we discuss the basic principles of Gibbs sampling and how they can be applied to motif finding. We start with the basic principles of sampling methods like Markov chain Monte Carlo methods and how sampling can be used in a general missing value setting. The Gibbs sampler is introduced as a special sampling scheme which is well suited for the missing data problem. From this basic theory, we give the basic ideas of using a Gibbs sampler in a motif finding algorithm as they were introduced by Lawrence et al. [54]. After outlining the algorithm itself, we also point out some of the weaknesses of this initial algorithm for which we will present potential solutions in the remainder of the text.

Chapter 5. MotifSampler: implementation and performance analysis

After the introduction of the basic algorithm for motif finding we introduce several extensions to this basic Gibbs sampler. The first extension is the implementation of higher-order background models in the probabilistic sequence model. Second, we use the probabilistic framework to estimate the number of instances of a motif in a sequence. This leads to a first revised version of the core sampling step. Next, we do an in-depth analysis of this core sampling procedure with some real biological data sets. We analyze for instance the convergence, the distribution of motif scores, the influence of the different parameters on the performance. Once, we have gained insight in the behavior of our implementation, we incorporate the core sampling step in a more practical algorithm. The practical implementations include such steps as an extra convergence step, inclusion of both strands, searching for different motifs. Finally, we propose an hands-on strategy that can be applied in a real biological motif finding problem.

Chapter 6. MotifSampler: case studies

To illustrate the applicability of our motif finding algorithm in real biological problems, we show the results on four major data sets. The first data set is the G-box data set, which contains 33 upstream sequences of genes that are known to be regulated by the G-box transcription factor involved in light regulation in plants. The second example is the analysis of ten regulons in yeast. These regulons were introduced by van Helden et al. [111] and cover a wide range of typical transcription factor binding sites. These regulons are therefore very useful as a benchmark data set. We apply our MotifSampler to those ten data sets and analyze the behavior in function of the different types of motif models. This analysis shows that we have implemented a well performing algorithm that can handle different types of motifs. Next, we look at the influence of higher-order background models in prokaryotes. The continuous growth of the number of available bacterial genomes allows us to build reliable background models for all these sequence bacteria. In this study, we use the well-known σ^{54} transcription factor which is present in many bacteria as a model system. We construct two comparable data sets in *Escherichia coli* and *Pseudomonas aeruginosa*. The results show that using a wrong background model has a profound impact on the performance of our algorithm. This analysis also gives us some insight on how we should apply the MotifSampler for phylogenetic footprinting. Finally, to illustrate the applicability of our method in a real microarray data set, we work with the well-known cell cycle experiment in yeast of Spellman et al. [97]. Adaptive Quality-Based Clustering [26] is used to find sets of tightly coexpressed genes. From the set of clusters retrieved, four specific clusters are further analyzed in detail. In two of these clusters we find specific motifs that can be related to known transcription factors. In a third cluster two motifs are found that seems to be specific for this cluster but that have no known counterparts. In the fourth cluster no motif is found that shows any characteristics of regulatory elements. This example shows the potentials and limitations of our motif finding approach.

Chapter 7. Searching for known transcription factor binding sites

In Chapter 3 we explained that motif finding could be either supervised or unsupervised. The Gibbs sampling method described in the previous chapters was designed to solve the unsupervised problem. At first we discuss an adapted version of the classical position weight matrices methods to work with higher-order background models. Next, the probabilistic sequence model is used to estimate the number of motif instances in a sequence of a known motif model. The influence of the parameters of both methods are evaluated and the results are compared. As test set we use large sets of promoter sequences in yeast and human. The search for known motif models can also be applied on a set of coregulated genes. Therefore it is necessary to asses the statistical significance of the number of motif instances with respect to the expected number of instances based on frequency of occurrence in a reference data set. To compute the statistical significance we use the binomial model used by van Helden et al. [111] and also implemented in *Toucan* [1]. To illustrate the applicability of this methodology we use the yeast regulons and the cell cycle clusters. The test show that the statistically significant motifs in these sets corresponds the expected ones. The main drawback of this methodology is the dependence on the motif models stored in the different databases.

Chapter 8. Integrated search for regulatory elements

From the start of our research, it was the goal to design an algorithm that was easy to use and also applicable in real life examples. Therefore, we have implemented from the very start a web interface to our algorithm, where non-expert users could try it on there specific data sets. This allowed us to interact directly with the end-users and to gain valuable feedback on the applicability of our approach. In this chapter we discuss the development of *INCLUSive*, an online system to analyze microarray data from clustering to motif finding. To guide us through the process, we use a small data set from a microarray experiments in *Arabidopsis thaliana* [86]. We start with defining clusters of coexpressed genes with adaptive quality-based clustering. For the genes in those clusters, the upstream regions delineated on the genomic sequence. Once the sequence sets are prepared MotifSampler is applied to retrieve potential transcription factor binding sites. We test several parameter settings to cover a wide range of different types of motifs. We also apply MotifScanner to this data set to find significant known motifs. Finally, we show some results of other researchers that have used our tool in their work [21, 51, 56, 72, 80, 87]

CHAPTER 2

Concepts of Gene Regulation

To start this dissertation, we will point out some of the basic concepts of molecular biology. This overview is not exhaustive but only defines the most important notions needed to understand the remainder of the text. We start with the very basis of DNA, the cellular library. Our main focus is on transcriptional regulation, therefore we discuss transcription factors more in detail. After the introduction of the basic biological concepts, we will discuss cDNA microarrays and genechips. Microarrays is one of the major technological advances which have been driving, together with the growing availability of completed genomes, research in molecular biology and genomics in the recent years. Microarrays allow measuring the expression level of several thousands of genes in one experiment. Here we discuss briefly the technology, the importance and the limitations of microarrays. To end this chapter we give an outline of the possible applications of microarray technology and where our research should be situated within this field of research.

2.1 DNA: the code of life

In 1953, Watson and Crick [116] achieved one of the most important scientific breakthroughs of the twentieth century when they revealed the structure of DNA. DNA or *Deoxyribonucleic acid* carries the code of life and is in its native form a linear polymer. DNA forms a double helix of two anti-parallel chains with complementarity nucleotide sequences. The basic building block of the DNA sequence is formed by the nucleotides: *adenine* (`A`), *cytosine* (`C`), *guanine* (`G`) and *thymine* (`T`). The complementary of DNA stems from the binding of `A` to `T` and `C` to `G`. We will now introduce some important biological concepts that are needed to facilitate the reading of the remainder of the text. Examples and definitions are extracted from the books by Alberts et al. [2] and Lodish et al. [66] and are also reconstructed from fruitful discussions with biologists.

2.1.1 Genes and proteins

DNA is the cellular library that contains the necessary information to create, develop and maintain a living cell. While DNA contains the code of life, the proteins or polypeptides are the real functional elements within a cell. It is the code of these building blocks of a cell that is contained in the genes hidden in the DNA sequence. Since the definition of a gene can differ depending on the source, we opt to use the following definition for a gene in this text:

A gene is the entire nucleic acid sequence needed for the synthesis of a functional polypeptide or RNA molecule.

Polypeptides are, like DNA, linear polymers built from only 20 different amino acids. To form an amino acid sequence, a gene is first transcribed into a RNA sequence. RNA or *ribonucleic acid* is a single stranded molecule related to DNA. In most cases, RNA is the intermediate between the gene and the protein, but it can also have function similar to proteins. For instance, small RNA molecules are involved in the transportation of basic building blocks within a cell. The transcription of a gene means that a RNA sequence complementary to one of the DNA strands is formed. This RNA sequence is further processed to form the template, *messenger RNA* (mRNA), of the protein sequence. mRNA is translated into an amino acid sequence using a specific genetic code. Each three nucleotides of the mRNA sequence are translated into one particular amino acid. Such a triplet of nucleotides is called a *codon*. Table 2.1 gives all the possible translations. For instance, the triplet `AUC` is translated into the amino acid *isoleucine*. Since there are 64 possible codons and only 20 amino acids it is clear that there is some redundancy in the genetic code. For instance, the codons `UCA`, `UCC`, `UCG`, `UCU`, `AGU` and `AGC` all code for the amino acid serine. There are also two special signals that indicate the boundary of the protein sequence. The *start codon*, `AUG` encodes for methionine. The *stop codons*, `UAA`, `UAG` and `UGA`, do not specify an amino acid but constitute a signal to terminate translation.

The function of a protein is determined by its three-dimensional structure which is in turn derived from the spatial folding of the amino acid sequence. The structure of the protein defines for instance the distribution of charge over its surface, which in turns defines the ability to bind other compounds. In Section 2.4 we give some examples of regulatory proteins that interact with the DNA sequence. These examples are a nice illustration of how a protein derives its function from its three-dimensional structure.

Table 2.1: Genetic code

1st[a]	2nd[a]				3rd[a]
	U	C	A	G	
U	Phenylalanine (F)	Serine (S)	Tyrosine (Y)	Cysteine (C)	U
U	Phenylalanine (F)	Serine (S)	Tyrosine (Y)	Cysteine (C)	C
U	Leucine (L)	Serine (S)	**Stop** ()	**Stop** ()	A
U	Leucine (L)	Serine (S)	**Stop** ()	Tryptophan (W)	G
C	Leucine (L)	Proline (P)	Histidine (H)	Arginine (R)	U
C	Leucine (L)	Proline (P)	Histidine (H)	Arginine (R)	C
C	Leucine (L)	Proline (P)	Glutamine (Q)	Arginine (R)	A
C	Leucine (L)	Proline (P)	Glutamine (Q)	Arginine (R)	G
A	Isoleucine (I)	Threonine (T)	Aspargine (N)	Serine (S)	U
A	Isoleucine (I)	Threonine (T)	Aspargine (N)	Serine (S)	C
A	Isoleucine (I)	Threonine (T)	Lysine (K)	Arginine (R)	A
A	**Start** (M)	Threonine (T)	Lysine (K)	Arginine (R)	G
G	Valine (V)	Alanine (A)	Aspargic acid (D)	Glycine (G)	U
G	Valine (V)	Alanine (A)	Aspargic acid (D)	Glycine (G)	C
G	Valine (V)	Alanine (A)	Glutamic acid (E)	Glycine (G)	A
G	Valine (V)	Alanine (A)	Glutamic acid (E)	Glycine (G)	G

[a] Letter at specific position in the triplet that forms the codon.
Eg. triplet ACU is translated into a Threonine.

2.1.2 Cellular structure

Any living organism is built of cells of which the number can range from one to several millions. A cell is a limited space in which many actions and reactions take place. It has an outer border formed by a membrane made of fatty acids (lipids) through which no polar molecules can flow. In the 1950s scientists discovered that there exists two major classes of cells in the biological world: prokaryotic or anuclear cells and eukaryotic cells. Prokaryotes include all bacteria both *eubacteria* and *archaebacteria*. Eukaryotes come in great variety and include all organisms from protist and fungi, over plants to animals and humans.

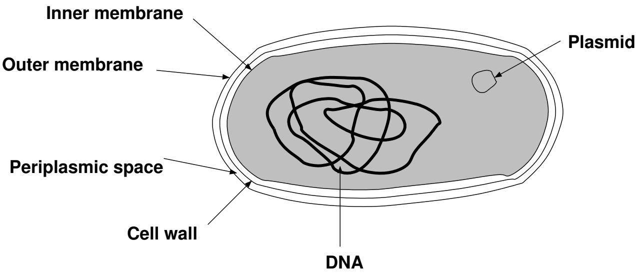


Figure 2.1: Prokaryotic cell. The cell is surrounded by several membranes that regulate the inlet and outlet of the cell. The chromosomal DNA and plasmid float around inside the cell.

Let us first look at prokaryotic cells. A picture of such a prokaryotic cell is given in Figure 2.1. The chromosomal DNA floats around inside the cell and is not surrounded by a separate membrane as is the case in eukaryotic cells. In some cases a plasmid, a linear or circular extrachromosomal DNA molecule, is also present in the cell. The cellular membranes regulate through channels and transporters the entry of substances needed for the cells life cycle and the exit of the substances produced within the cell. The DNA itself forms in most bacteria a single circular molecule.

A eukaryotic cell as shown in Figure 2.2, displays a more complex structure than the prokaryotic example. The eukaryotic cell is surrounded by a plasma membrane and is equiped with several intracellullar compartments, called organelles. The plasma membrane regulates the cell inlet and outlet and also maintains the proper ionic composition and osmotic pressure of the cytosol. The nuclear membrane surrounds the DNA, thereby separating the DNA from the cytoplasm. The nucleus is the largest organelle but cells contain many others like the endoplasmic reticulum, the lysosomes and the mitochondria. The mitochondria are the primary source of energy for cellular growth and metabolism. Inside the nucleus the genes are transcribed into mRNA which is then transported through the endoplasmic reticulum into the cytoplasm where it is translated into a protein.

The eukaryotic DNA resides inside the nucleus and is divided among several

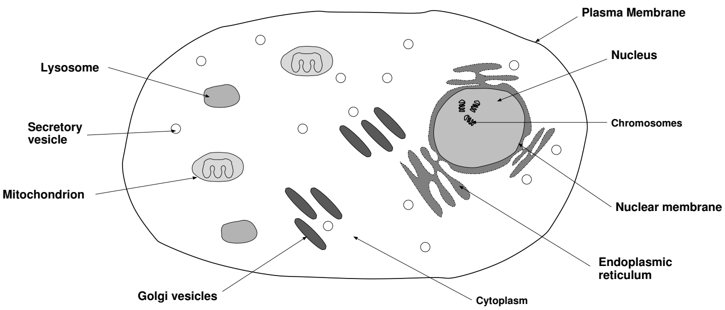


Figure 2.2: Eukaryotic cell. The cell is equiped with several organelles, larger intracellullar compartments. The DNA, compacted in several chromosomes, resides in the nucleus surrounded by the nuclear membrane and the endoplasmic reticulum. The outside of the nucleus is the cytoplasm in which several organelles can be found. For instance, the mitochondrion is the energy-generating organelle.

chromosomes. To allow the cell to contain the DNA, the DNA needs to be highly compacted in to small structures called *chromatin*. The most abundant proteins associated with the packaging of DNA are the *histones*, proteins that are remarkably well conserved among all eukaryotes. To compact the DNA string a disk shaped protein complex of histones is formed around which a stretch of 146 nucleotides is wrapped. This complex is called *nucleosome*. To further compact the DNA, six nucleosomes are packed into a spiral by binding to another histone. Further coiling of the chromatin forms the highly packed chromosome.

2.1.3 Organization of genes on the genomic DNA

Recent completion of several genomes has learned us that a very large fraction of the genome is non-coding in all complex higher organisms. So far, there has been no function assigned to a large part of this so-called ”junk DNA”. One type of junk DNA are the many repetitious regions that have been found along the DNA of multicellular organisms. Surprisingly, these regions could be found at different positions in the DNA of individuals of the same species. Therefore, these regions are assumed to have a strong influence on the evolution of an organism but they tend to have no role in the life cycle itself of the organism. Part of the DNA that effectively plays a role in the life cycle of an organism are the coding sequences that are hidden in a plentitude of non-coding, probably non-functional, regions.

Eukaryotic genes are physically separated from each other in the genomic DNA. Figure 2.3 shows the conformation of a eukaryotic gene, including the proximal and the distal promoter region. Following the definition from the beginning of this chapter the gene is everything that is needed to form the protein. In eukaryotes, genes consist of a fraction that is actually translated into amino

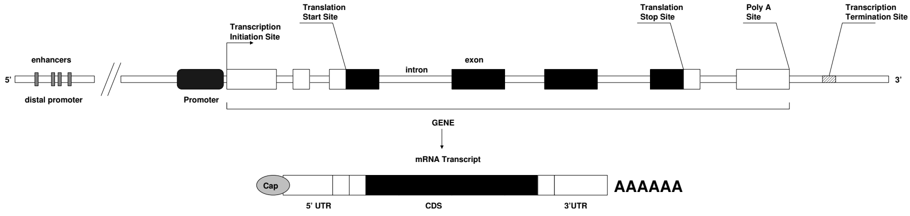


Figure 2.3: Anatomy of an eukaryotic gene. The gene starts at the transcription start sites with the 5'UTR and ends with the 3'UTR at the poly A site. Next there are several exons, indicated in black, that form the coding sequence. Just upstream of the gene is the proximal promoter. Further upstream the distal promoter region is located where transcription factors can bind to enchancers to regulate the transcription.

acids, *exons*, the non-coding part is spliced out and is referred to as *introns*. The primary transcript contains both the exons and the introns. Those introns might very long with respect to the actual coding sequence (CDS) that is translated into the protein sequence. Comparison of several genomes teaches us that evolutionary pressure selects for maintenance of relatively similar sequences in exons while great variability is observed among introns. Figure 2.3 also shows the mRNA sequence formed from the primary transcript after splicing out the introns. The mRNA sequence consists of the CDS, surrounded with the 5' untranslated region (UTR) and the 3'UTR[1].

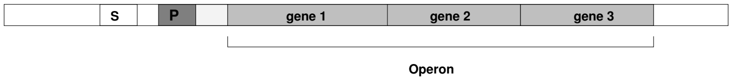


Figure 2.4: Prokaryotic gene. The operon consists of three genes that are transcribed in tandem. Just upstream of the operon resides the promoter (P) where the RNA-polymerase can bind. The level of transcription can be influenced by transcription factors that can bind to the binding site (S).

Sequencing of bacterial genomes revealed that they contain large stretches of uninterrupted protein-coding regions. Bacteria lack introns, although they are found in a few exceptions. Genes encoding for proteins involved in related functions are found to cluster together on the genome. The genes in such a cluster comprise one single *transcriptional unit* and are referred to as an operon. The structure of an operon is illustrated in Figure 2.4. In this example the operon consists of three genes that are transcribed in tandem. The transcription is controlled by the RNA-polymerase that can bind to the promoter (P) just upstream of the first gene of the operon. In the neighborhood of the promoter, we find also a binding site of a transcription factor that can regulate the level of transcription of the operon.

---

[1] The terms 5' and 3' are used to indicate respectively the start and end of a DNA sequence.

2.2 The process of gene expression

The term *gene expression* refers to the process whereby the information encoded in the genes is decoded into a particular protein. One of the basic principles of molecular cell biology is that the characteristics and behavior of a cell are determined by the set of proteins contained in the cell. This set of proteins is determined by the stability of these proteins, by the set of mRNAs that are translated and by the translation rate of these mRNAs. The level of mRNA is, in turn, determined by the genes that are transcribed and also the rate of transcription. For a recent update on the latest insights in the process of eukaryotic gene expression, we refer the interested reader to the excellent review by Orphanides and Reinberg [81].

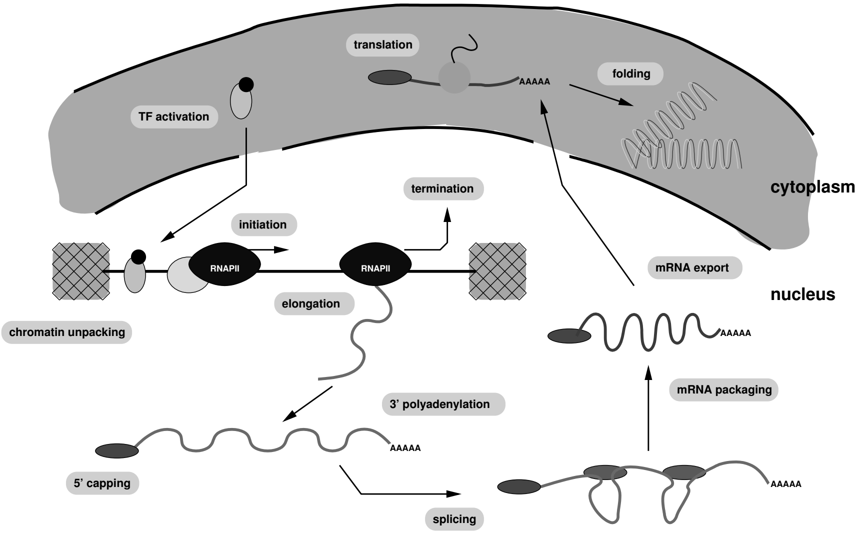


Figure 2.5: An iterative view on the process of eukaryotic gene expression [81]. When an activated transcription factor enters the nucleus of the cell, it can bind to the DNA accessible in the decompacted chromatin. This will attract the RNA-polymerase and the transcription of the gene is initiated. The RNAP reads the gene sequence and creates the RNA sequence until the termination signal is encountered. The newly formed RNA is then capped at its 5’ end and a polyadenylation tail is added at the 3’ end. In the next step the intron are spliced out of the RNA sequence to form the messenger RNA. This mRNA is packed and exported to the cytoplasm where it is translated into an amino acid sequence that folds into a protein.

In eukaryotes, the synthesis of a functional protein from a gene encoded in DNA is a more complex task that involves many biological processes. A schematic representation of the process is shown in Figure 2.5. At the initiation phase, one or more transcriptional regulators are activated and enter the nucleus to start the gene expression. To be able to transcribe a particular gene, the gene should be made accessible through decompacting the chromatin structure. If the DNA sequence is accessible, the transcription factors could bind to the *cis*-regulatory[2] regions in the neighborhood of the gene. If bound, they form complexes to attract, together with other components, the RNA polymerase (RNAP) to the start site of the gene. The RNAP is the basic machinery of the transcription process. The RNAP opens the double stranded DNA, reads the gene sequence and transcribes it into a single stranded RNA molecule. As soon as the RNA molecule is being formed a cap site is added to the 5' end of the newly formed RNA. The cap sites protects the RNA from degradation and mediates early transcriptional events like elongation of the RNA. When the RNAP reaches the end of the gene sequence, the transcription is terminated, the newly synthesized RNA is cleaved and a polyA tail is added to the 3' end of the RNA molecule. The transcribed RNA contains the sequence that will be translated to form the protein but also untranslated regions at the beginning and the end. The part of the RNA molecule that is translated into amino acids is the coding sequence and is formed by the exons. The region between the exons, the introns, are removed by pre-mRNA splicing. The resulting messenger RNA (mRNA) is folded and ready for transport from the nucleus into the cytoplasm. The processes of transcription and translation are in eukaryotes physically separated in the cell. Transcription occurs in the nucleus of the cell, the synthesized mRNA is transported to the cytoplasm. The translation of mRNA into protein takes place on large ribosomes. mRNA is translated into amino acids by selecting a triplet of nucleotides, a codon. Each codon is translated into its respective amino acid. To form the complete protein, first the start codon is located by translation initiation factors. Next, each codon is read and translated until the the stop codon is encountered. Finally the protein folds to a stable conformation. However, the protein undergoes many post-translational modifications to become the final functional component.

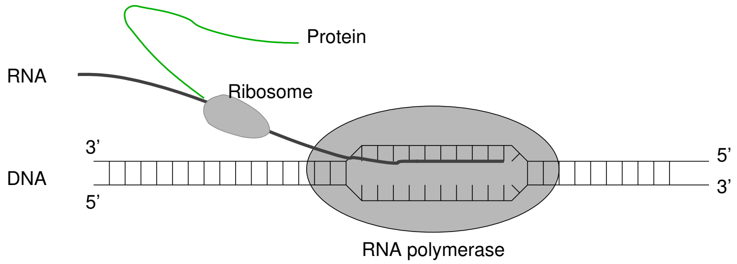


Figure 2.6: Prokaryotic gene expression. The RNAP opens the double stranded DNA and reads one strand of the DNA to form the RNA transcript. While the transcript is being formed the translation can start. Therefore, the ribosome first binds to the RNA and then the protein synthesis starts at the initiation codon.

The previous paragraphs described the process gene expression in eukaryotes, let us now move to prokaryotes. The pathway from DNA to RNA to

---

[2] The term *cis* refers to the fact that the transcription factor binds to the same DNA as the one on which the gene resides which it expression it controls.

protein is less complex in prokaryotes than it is in eukaryotes. The transcription of a prokaryotic gene is started when the RNAP is bound to the -35 box[3] of the core promoter. The RNAP opens the DNA, reads the DNA sequence and the RNA is synthesized. In prokaryotes, the mRNAs do not undergo any significant forms of modification to prepare the mRNA for translation. In most cases, the translation of the primary transcript is already started even before the transcript is completed. This process of transcription and translation is illustrated in Figure 2.6. If part of the RNA transcript is formed the ribosome binds to the RNA at the Shine-Dalgarno sequence[4] and the protein synthesis is started at the initiation codon. The translation is terminated when the stop codon is encountered.

2.3 Gene regulation

Gene regulation is the ensemble of processes by which diverse factors influence the differential control of gene action at the level of transcription or translation. In eukaryotes, gene expression can be regulated at many steps in the pathway from DNA to protein. The first and most important step is the transcriptional regulation, since none of the other step is necessary/possible if no primary RNA transcript is formed. The next step is the RNA processing control which involves the splicing of exon and introns. Since the mRNA is transported from the nucleus to the cell, a mechanism that controls this transport is present in the cell. Once the mRNA arrives in the cytoplasm, the mRNA can be either degraded or translated. The final step is the control of the protein activity after the protein is formed. Since in prokaryotes the process of gene expression involves less steps from gene to protein, some of the control step in eukaryotes are not necessary in prokaryotes. For instance, since there is no physical separation in the cell between transcription and translation there is no need for an mRNA transport mechanism.

To illustrate the process of transcriptional regulation we use the well known example of the *Lac* operon. Figure 2.7 gives the structure of the *lac* operon in *Escherichia coli*, which was first studied in the 1960s. The *lac* operon consists of three genes: *lacZ*, which encodes β-galactosidase; *lacY*, which encodes lactose permease; and *lacA*, which encodes thiogalactoside transacetylase. *lacZ* and *lacY* are required for the metabolism of lactose. The physiological function of *lacA* is still not so well understood. Just upstream of the first gene is the transcription control region. Further upstream we find *lacI* that encodes the *lac* repressor. The repressor can bind to the operator (O) in the region just upstream of the start site of the operon and will block the RNA-polymerase from transcribing the gene.

There exists three operating modes for the transcription of the *lac* operon. *E. coli* prefers metabolizing glucose above all other sugars, but when the level of glucose drops, the bacteria switches to the consumption of lactose, if available.

---

[3] Position relative to the start site of the gene.

[4] Short target site, `AGGAGGU`, 3-10bp upstream of the start codon.

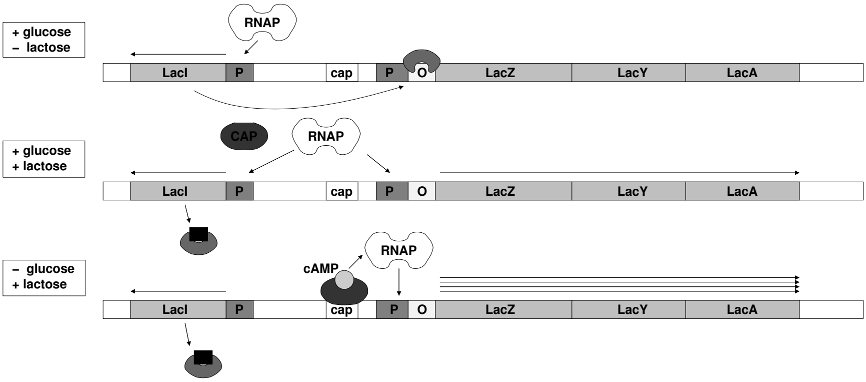


Figure 2.7: Three operation modes of the *lac* operon in *E.coli*. In the first mode glucose is present and lactose absent in the medium. The repressor is bound to the operator region (O) and the *lac* operon is not transcribed. In the second mode, when both glucose and lactose are present, lactose binds to the repressor and the operon is transcribed at a low rate. In the third mode of operation no glucose is present. Due to the absence of glucose an alerting signal cAMP is produced that forms a complex with the CAP factor. The cAMP-CAP complex will bind to the cap site near the promoter (P) and increase the transcription level of the *lac* operon.

This process involves several regulatory proteins and three modes of operation can be distinguished as shown in Figure 2.7. When glucose is present and no lactose is present in the medium, the *lac* repressor is expressed and binds to the operator of the *lac* operon thereby blocking the expression of the enzymes encoded in the operon. If glucose and lactose are present in the medium, lactose binds to the *lac* repressor and the repressor no longer binds to the operator and the three genes are coordinately expressed. The level of expression now depends on the level of glucose present in the medium. When the level of glucose decreases, an alerting signal is produced under the form of cAMP. The level of the cAMP in the cell is sensed by a protein called *catabolite activator protein* (CAP). The complex of cAMP-CAP can bind to DNA upstream of the operator and regulates the level of expression. In the second mode of operation glucose is still present, therefore the level of cAMP is low and the cAMP-CAP complex is not formed. As an effect the RNA polymerase will not bind effectively to the DNA and the level of transcription of the operon is low. When only lactose is present in the medium, the level of cAMP will be high and the cAMP-CAP complex binds to the CAP site thereby inducing maximal transcription of the *lac* operon.

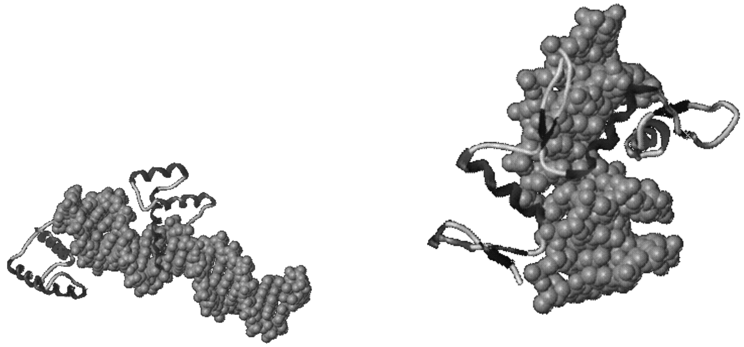
This simple example gives an idea on how an organism can adopt itself to the changing environmental conditions. These processes will of course be more complex if we look at multicellular eukaryotic organisms like flowering plants and vertebrates.

2.4 Regulatory proteins

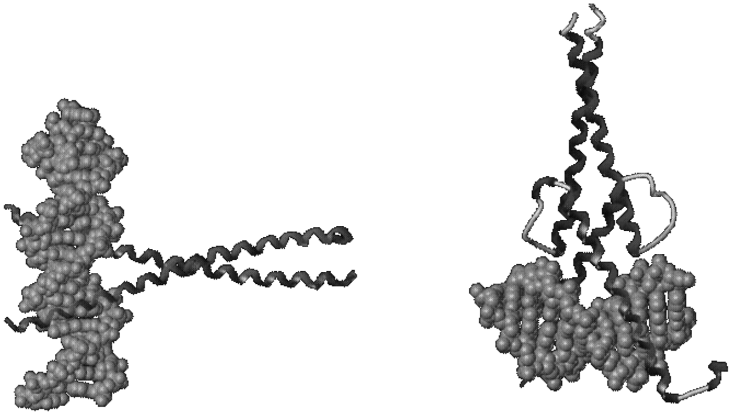
So far we have looked at the gene expression process. Let us know look in more detail at transcriptional regulators. In the 1950s the first evidence was given of the existence of regulatory proteins that could switch on or off the transcription of sets of genes. Later it was shown that combinations of different regulators alters the expression of a gene in different cell types (for a review we refer to Ogata et al. [78]). Recently, with the completion of the human genome, it turned out that the number of genes (±30000, depending on the source) was actually much smaller than expected at the beginning –and even just before completion– of the human genome project (an estimate of 100000 genes in 1997). This much higher number of genes was expected by comparing humans with less complex organisms like *Caenhoribitis elegans* which has already 20000 different genes and extrapolating these numbers. After the completion of the human genome sequence much lower estimates are brought forward. It turns out that the great diversity in cell specialization is merely achieved through the difference in complexity of the control mechanism than through the actual number of different genes. Since the goal of our research is to develop algorithms that detect transcription factor binding sites, it is important to have a good idea how this interaction is accomplished. Therefore we need to look at the three dimensional structure of the transcription factors.

Transcriptional regulators come in a variety structures that interact with specific DNA sequences. For instance, α-helices in the DNA-binding domain of a protein are oriented in such a way that they make specific hydrogen bonds and van der Waals interactions in the major groove of the DNA. The major groove can also be read by a set of β-sheets. The transcription factors are classified according their DNA-binding domain. The active domains of the factors in a class have a characteristic amino acid composition. In Figure 2.8 we give some examples of these structural classes. These examples are adapted versions of the models found at `http://www.rtc.riken.go.jp/jouhou/image/gallery.html`.

The simplest and most common DNA-binding motif is the helix-turn-helix (HTH) motif. A special class of HTH is termed the *homeodomain* proteins. Homeodomain proteins are named after a class of genes in *Drosophila melanogaster* in which a specific conservation of the amino acid composition was first discovered. These genes play an important role in the control of the cell type. This motif was later also found in vertebrate proteins and in yeast proteins that regulate the mating type. Figure 2.8(A) shows the engrailed box bound to DNA from two different angles. Zinc finger proteins got their name from the fact that they are folded around a central Zn^{2+} ion. Within the zinc fingers we can distinguish multiple classes. The most important class is the C_2H_2 zinc finger. Figure 2.8(B) gives an example of such a C_2H_2 zinc finger protein. The protein has three α helices that interact with the DNA sequence. The figure nicely illustrates how these three helices fit within the curvature of the DNA double helix. Another class of transcription factors form the *Leucine*



A. Engrailed homeodomain **B**. Zinc finger protein



C. GCN4 Leucine zipper **D**. Homodimer of 2 helix-loop-helix

Figure 2.8: Examples of different transcription factors bound to the DNA sequence. (A) Two viewpoints of an engrailed homeodomain bound to DNA. (B) The three helices of a zinc finger protein are aligned to the DNA. The two examples in (C) and (D) nicely show how the helices of the the transcription factors fit within the grooves of the DNA sequence. (figures adapted with kind permission from Prof. A. Sarai of the Riken Tsukuba Institute, Japan).

zippers that can come in dimeric formation of two α helices. Typically, there is a leucine residue found at every seventh position in the dimerization region. Figure 2.8(C) shows the *Gcn4* transcription factor in yeast. The transcription factor consists of two α helices that bind to the DNA in two adjacent major groves. The last class of transcription factors have also the dimeric conformation as the basic zipper classes, but they have the helix-loop-helix (HLH) motif. An HLH motif consists of a short α-helix connected through a loop with a longer α-helix. Figure 2.8(D) displays two monomers held together in a four helix bundle to form a dimer. Each monomer contributes part to the DNA binding.

2.5 Measuring gene expression

Advances in technology have introduced high-throughput analysis into molecular biology and genomics research. The completion of the human genome is one of these milestones driven by high-throughput sequencing technologies [103]. Another important technological breakthrough was the ability to miniaturize mRNA abundance measurements onto a single chip. Such a chip is called a *microarray*. The basic concept of microarray technology lies in the hybridization properties of complementary DNA sequences. This property could be exploited on a large scale through the immobilization of target nucleic acids on a glass surface and the detection of these targets with radioactive complementary probes. It was the ability to spot several thousands of targets on one small glass slide that have boosted the applicability of microarrays.

The availability of complete genomes have also allowed studying the expression of genes on a genomic scale. DeRisi et al. [29] introduced the first microarray with virtually all genes from *Saccharomyces cerevisiae* to study the metabolic and genetic control of their expression. In such an experiment, it is thus possible to get a view on the complete transcriptome[5] under specific conditions. One of the first experiments, in which this methodology has been successfully applied, is the study of the cell cycle in yeast [22, 97]. This data set has become a benchmark data set to evaluate different types of algorithms that work with microarray data[6]

In this section we will discuss more in detail the concepts behind microarray technology. Currently, two basic array technologies are available to produce microarrays: *cDNA microarrays* and *DNA-chips*. Although the basic ideas and the goal of both technologies are the same, there are some very distinct differences. These differences have their impact on how to analyze the results obtained with either method. Especially the methods to preprocess the measurement data are heavily influenced by the used methodology.

---

[5] Transcriptome is one of those new ’omic’ terms and refers to the study of transcribed genes.

[6] In Chapter 6, we use the same data set to illustrate the applicability of our motif finding method.

2.5.1 cDNA microarray technology

cDNA microarrays are small glass slides on which double-stranded DNA is spotted using high-speed robotics [91]. The cDNAs are selected from a library of fully sequenced clones, collections of expressed sequence tags (ESTs), or any other source. The selected cDNAs are spotted and chemically linked to the glass surface, and are finally denatured. Figure 2.9 shows the setup of an cDNA microarray experiment. In an experiment, the total RNA pool or the mRNA is extracted from the test sample under study and fluorescently labelled using a single round of reverse transcription. The mRNA isolated from a reference sample is also fluorescently labelled using a different label. The quality and the purity of the labelled RNA is essential in assessing the quality of the microarray experiment. Once both samples are labelled, they are mixed together and simultaneously hybridized to the prepared cDNA microarray. After hybridization the slide is read using a laser excited at the different wave lengths corresponding to the fluorescent labels. As a result we obtain two images, one in the green channel and one in the red channel. These images are further analyzed using digital image processing techniques to obtain the expression measurements. This analysis involves the localization of the spots by applying a grid to the image and measuring background and spot intensities. The fact that one has to use image processing steps to retrieve the measurements is

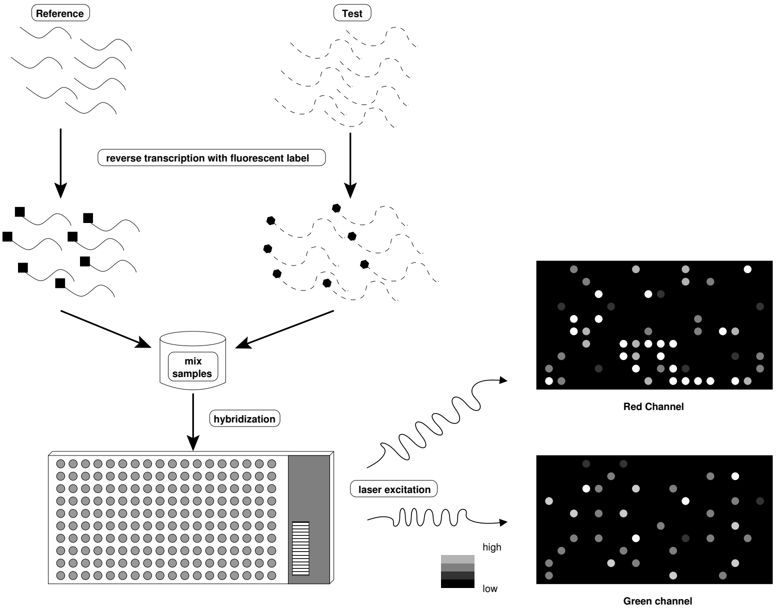


Figure 2.9: Principle of a cDNA microarray. In the first step, mRNA is extracted from the test sample and a reference sample. Both RNA samples are fluorescently labeled with a different tag. Next both samples are mixed and hybridized onto a prepared cDNA microarray. Finally the level of hybridized RNA is measured by laser excitation.

one of the main sources of noise in microarray analysis.

2.5.2 DNA-chip technology

The other technology are the so-called DNA-chips developed by Affymetrix, Inc [59]. DNA-chips are high density synthetic oligonucleotide arrays synthesized on a silicon waver. The two enabling technologies are photolithography and solid-phase DNA synthesis. The production process resembles the production of CMOS chips and is illustrated in Figure 2.10. To start, synthetic linkers with photochemically removable protecting groups are attached to a glass substrate. In the first step of the production, a photolythographic mask is applied and specific areas are illuminated. These sites become deprotected by removing the attached protecting group and are activated. Next, a new set of protected nucleotides are deposited and they couple to the activated sites. In the succession of the process, a new mask is applied for each type of nucleotide to build up the full oligonucleotide array. On a typical array the oligonucleotides are of length 20 to 25 nucleotides. In an experiment, a fluorescently tagged mRNA sample is hybridized to the chip. A fluorescence image is created by measuring the emissions by laser excitation. In this setup it is thus not necessary to mix test and reference sample. Typically, one image is created from a reference sample which is used for comparison.

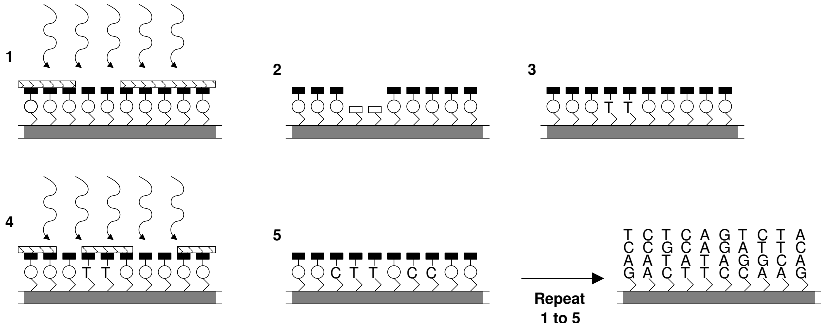


Figure 2.10: Principles of a DNA-chip. In step (1) a mask is applied to the chip and the illuminated regions become deprotected (2). Next (3) protected nucleotides couple with the activated sites. To deposit other nucleotides, a next mask is applied and new sites become deprotected (4). The next batch of protected nucleotides couples with the activated sites (5). By repeating the steps a full DNA-chip is built.

Since oligonucleotide arrays are built from sequence information alone, they do not need any intermediates as in the cDNA technology. It is thus very important to design for each reference sequence a highly specific probe as a dense collection of complementary short probes. The main points of attention to improve the quality are that probes are complementary to selected genes and unique to related genes. Also the absence of near-complementary RNAs in the sample is important. Another aspect in assuring the quality of the chips is the presence of redundancy and mismatch control on the chip. The main disadvantage of gene-chips is that this technology is proprietary and the design of very specific chips is not always feasible within smaller projects.

2.6 Applications of microarray

The availability of genome-wide expression profiles has allowed researchers to explore new directions in functional genomics [4, 123]. This type of data provide the biologist with valuable information on when and where a gene is transcribed, how its expression changes over time or between different cell lines and possibly how its expression is influenced by other events.

2.6.1 Some issues on preprocessing

Before we dive into the application themselves, it is really necessary to point to the issue of preprocessing microarray data. Given the experimental setup of a microarray, the expression data from such a microarray experiment are very noisy in nature. In each step of the preparation of the test and reference sample and at each stage in the process errors could be introduced. Another issue is the retrieval of the expression data, which is done with several image processing steps and this certainly leads to noisy measurements. This is also exposed by researcher who have difficulties to reproduce the results obtained with microarrays. Therefore, it is very important to assess the quality of the expression data. Here we point out the most important steps involved in preprocessing microarray data (for a review see Marchal et al. [68]).

1. *Normalization*. Normalization of the hybridization intensities within a single array is needed to compensate for the bias introduced by several sources of experimental noise.

2. *Nonlinear transformations*. Passing the expression data through a log-transformation or any other non-linear transformation is common practice when working with expression ratios since the distribution of the data is not symmetric. It is known that measuring light intensities does not scale linearly.

3. *Replacing missing values*. In the simplest case, missing values are replaced by zero or an average expression value. However this might easily lead to erroneous results. There exists more advanced techniques like *k-nearest neighbors* and *singular value decomposition*, that exploit the correlation present in microarray data.

4. *Filtering*. A important fraction of the genes on a microarray do not contribute to the biological process under study and show little variation in their expression profile. Thresholding the standard deviation is common

practice to leave out the constitutive genes, that have a constant expression profile over the experiment, from the analysis.

5. *Standardization*. To compare expression profiles over different experiments, the relative change is more important than the absolute values. Therefore, expression profiles are rescaled such that they have zero mean and unit standard deviation.

2.6.2 Finding groups of coexpressed genes

In functional genomics, people often rely on the *guilt-by-association* principle. For instance, homology searches in databases have been the primary source of information when working with newly sequenced genes. If two genes or proteins have a similar sequence or structure, they might exhibit the same functionality. The same holds for expression data. This principle is also widely adopted in the microarray community: genes having a similar expression profile are expected to be functionally related.

The primary tool to find groups of coexpressed genes is cluster analysis. The first studies used classical algorithms like hierarchical clustering [31], *k-means* [102] and self-organizing maps [100] to identified clusters of genes. Later, more advanced methods have been designed to address one or more problems specific to the process of clustering gene expression profiles. These algorithms are the self-organizing tree algorithm [40], model-based clustering [121], quality-based clustering [26, 43] and many others. In Chapter 8 we present the integration of *adaptive quality-based clustering* [26] with our motif finding algorithm to automate the process of finding specific transcription factor binding sites in sets of coregulated genes.

Given the plentitude of clustering algorithms, it is an important question how to assess the quality of the clusters retrieved with each method. Intuitively, a good clustering algorithm should ensure that genes clustered together are similar. To assess this similarity there exists several metrics (for a review see Moreau et al. [74]). In the end, the application in which the clusters will be used, determines the metrics used to assess the quality of retrieved clusters.

2.6.3 Deciphering the regulatory mechanism

From the early days of microarrays it was clear that these expression measurements could provide useful information on transcriptional regulation. Shortly after the first microarray experiments were published, the potential of microarrays to decipher the regulatory mechanism became widespread [16, 123]. Given a set of genes that share a similar expression profile, a logical step in the analysis is to search for the mechanism that explains this coexpression. One possible explanation is that those genes might be controlled by the same transcriptional regulator. A logical step is then to search in the control region of these genes for common features that might be transcription factor binding sites.

The first extensive studies to find common *cis*-regulatory elements in sets of coexpressed genes were performed in yeast. Roth et al. [88] characterized the yeast transcriptome under different regulatory conditions and presented AlignACE to search for regulatory motifs. Later, Tavazoie et al. [102] used AlignACE to explore the clusters found in the expression data generated by Cho et al. [22]. We have presented one of the first studies in plants to find regulatory elements based on a small microarray experiment in *Arabidopsis thaliana* [105] (see also the example in Chapter 8). In Chapter 3 we discuss in more detail the current status of the algorithms to detect *cis*-acting regulatory elements. This work is situated in this area and focuses on the analysis of sets regulatory sequences found by expression profiling experiments.

2.6.4 Reconstruction of genetic networks

Taking the analysis of expression data one step further is the reverse engineering of the interaction network that can explain the observed behavior. In first instance, genetic network reconstruction was limited to looking at overrepresentation of functional classes in gene clusters [102]. Although this reveals useful information, this approach relies too much on functional annotation that is still incomplete to a large extend. It is however possible to extract more in-depth information from expression data. Recent studies show the applicability of diverse modeling techniques to this problem. Boolean models of regulatory pathways have been designed to infer relations between genes from expression data [58]. Currently, the focus has shifted to the construction of Bayesian networks to infer relations between genes from expression data [83, 95]. Using a Bayesian framework to study expression data allows solving more complex problems. One could incorporate prior biological knowledge, find probabilistically supported gene interactions, retrieve subnetworks and give leads to design new experiments to validate networks.

2.6.5 Clinical usage

Moving away from the theoretical setting of systems biology, microarrays can also be applied in a more applied setting like a clinical environment. One of the potential applications of microarray is in diagnostics. It would be interesting to be able to classify patients according to their genetic profile. Their have been a few studies in which genetic profiles of patients with different types of cancer have been created with microarray technology [3, 12, 36, 84]. In such a study it is important to know which genes best explain the class differences. The knowledge of genetic preposition of the patient would also help the medical doctor in designing a patient-specific treatment. The pharmaceutical industry has already fully adopted microarray technologies. The high-throughput facilities of these companies have been equipped with microarrays and are screening the effects of new drug targets on daily basis.

2.7 Conclusions

In this chapter we introduced the basic concepts of molecular cell biology and genomics. Our main focus was on the process of gene expression and the mechanisms involved in the regulation of the gene expression. In this respect we introduced the concept of transcription factor and gave several examples of the different classes. An enabling technology to study transcriptional regulation are microarrays. Microarrays allow researchers to study the complete transcriptome of an organism. We described the two technologies currently available: cDNA microarrays and gene-chips. Given the importance of this technology we discussed it in detail and gave an overview of the possible applications. One of these application forms the main topic of our research. In this dissertation we will work towards the detection of common *cis*-regulatory elements in sets of coexpressed genes.

CHAPTER 3

Current State of Motif Finding Algorithms

In this chapter we give an overview of the most important methods to detect cis-acting regulatory elements. To start, we define some basic conceptual terms which will be used throughout the text, like motif, regulatory element, instance and motif model. Next, the two types of representation of transcription factor binding sites will be discussed. These two types are the string-based and the matrix-based representation. We then give an overview of the algorithms to detect binding sites in one or more DNA sequences. There are basically two approaches to this problem: (1) search for known motifs (supervised) and (2) search for unknown motifs in sets of sequences that should share the same motif (unsupervised). First we discuss the methods to detect instances of a known motif in a given DNA sequence. More interesting for us are the unsupervised algorithms. In this approach one tries to identify statistically overrepresented motifs in sets of coexpressed genes. We try to give a detailed overview of the current state of motif finding algorithms. Next we also give some insights in advanced and related methods. To end the chapter we give an overview of the available methods and where to find them.

3.1 Terminology

To start this chapter we first need to define the terminology with respect to the detection of transcription factor binding sites (TFBSs). First of all, a binding site of a specific transcription factor is also referred to as a *cis*-acting **regulatory element**. The term *cis*-acting refers to the fact that the binding sites is located on the same DNA as the gene it is controlling. To model the binding site of a specific transcription factor, we will look at several distinct examples.

There exists several terms to describe a set of binding sites related to the same transcription factor. Probably the most important concept for us is that of a **motif**. Another term that is frequently used to indicate the binding sites is **box**. In the remainder of this dissertation, we use **motif** as an abstract, higher-level concept to define the binding site(s) for a specific transcription factor. An annotated binding site or a predicted binding site is in this respect referred to as an **instance** of the motif. The representation or description of the **motif** is defined by the term **motif model**. We will see that there exists several ways to represent a motif and that this is also related to the different algorithms involved in the analysis of motifs.

3.2 Modeling transcription factor binding sites

The basic model to describe transcription factor binding sites finds its origin in the biochemical process of protein-DNA interaction. Berg and von Hippel [10] show that the selection of binding sites by a transcription factor arises from the specific interaction between the active site in the protein and the content of DNA sequence at the binding site. Protein-DNA interaction is primarily based on the sequence-dependent hydrogen bond patterns exposed in the major groove of the double helix (for examples see also Figure 2.8). These patterns must be more or less complementary to the pattern in the active site of the transcription factor. This leads to the fact that binding sites of a specific transcription factor share some common features that almost always appear at the same position in the recognition sites. Although there is a core of the recognition site that is conserved over the different examples, there are difference between the different recognized sites. Berg and von Hippel give three reasons to explain the variability in the binding sites: (1) certain sites may require a different binding affinity according to their function, (2) certain positions do not influence the binding affinity and (3) some base-pairs might be required for regulation, like the binding of inhibitors. Within a cell, a transcription factor is able to detect these short binding sites hidden in a DNA sequence. Summarized we can state:

> *Hidden in the regulatory region upstream of a gene, there exist short segments of DNA that act as binding site for a specific transcription factor*.

This observation has formed the basis of almost all computational methods to detect potential binding sites.

3.2.1 Example: G-box binding factor in plants

To illustrate the different representations, we choose the G-box binding factor (GBF) in plants as an example [30]. GBF is a well studied and well documented transcription factor that is involved in the response to light in plants. Table 3.1 lists the sequences recognized by GBF as they are described in the literature and annotated in PlantCARE [57]. The binding sites shown in Table 3.1 are selected from different plants. In this table the annotated binding sites are aligned around the core `CACGTG` of the motif. In capital letters, the recognized binding site is shown as it is described in the literature. To complete the sequence we have added some surrounding base-pairs as they are found in the genomic sequence. These selected flanking base-pairs are shown in lower case. The size of the reported sites varies from six to thirteen base-pairs. Looking at these examples shows clearly that there exists a core, `CACGTG`, of six base-pairs that is well-conserved with only a very few number of substitutions over all examples. The surrounding base-pairs show on the other hand a large level of variability.

3.2.2 String-based representation

The first way to represent a motif is by using a string representation. The most used form is the consensus sequence [99]. A consensus sequence is the representation of the known binding sites of a specific TF by one single string using a degenerated alphabet (see Notations). Typically, the consensus sequence is derived from the aligned binding sites by choosing at each position the most prominent base or combination of bases. The way the letters from the degenerated alphabet are chosen depends on the implementation. There exist no exact rules to construct a consensus. We followed the rules of thumb as suggested by [32]. They represent a position with one of the four nucleotides if this nucleotide occurs in at least 60% of the binding sites at that position. If there are two dominant nucleotides, for instance both nucleotides occurring in at least 35% of the binding sites, a symbol of the degenerated alphabet for two bases is selected. These rules can also be extended to a three letter substitute. A related approach is the representation of a motif as a regular expression.

As an example we build a consensus from the G-box binding sites in Table 3.1. After aligning all sites, it is clear that there exists a short core of six base-pairs, starting at position five, which in most cases is like `CACGTG`. The other positions do not seem to be very conserved. Only at the fourth position, the two bases (`A` and `C`), which both have a frequency larger the 35%, are preferred over the others. This position can be represented with the symbol `M` (see also Notations). At all the other positions there is no dominant two-letter combination, so we use the symbol `n` there. This gives us the following consensus: `nnnMCACGTGnnnnnn`. We use upper and lower case letters to better accentuate the difference between core of the binding site and the flanking bases.

3.2.3 Matrix representation

Although the consensus sequence gives an easy to interpret impression of the binding site, it merely serves as an average binding site and is rather limited in capturing the true specificities of the binding sites. A better way to represent a

Table 3.1: Annotated G-box binding sites in PlantCARE

Organism	ACCN[a]	Start[b]	End	TSS[c]	Site
A.thaliana	M12196	-276	-267	-58	atGCCACGTGGAcgaa
A.thaliana	X51370	-124	-114	N/A	acGCCACGTGGTAgat
A.thaliana	X13611	-261	-249	N/A	ctTCCACGTGGCATTa
B.juncea	X67833	-66	-61	0	attaCACGTGatcgcc
B.napus	X61937	-155	-147	-69	ttaACACGTGGCgtag
B.napus	X75334	-221	-215	N/A	ctgcCACGTGgcctta
B.oleracea	X98521	-224	-215	0	tTAACACGTAGgtcaa
B.rapa	Y13108	-65	-60	0	attaCACGTGatgcat
C.annum	AJ005588	-411	-406	0	atccCACGTGttatgt
C.roseus	Y10182	-108	-103	0	actcCACGTGgtacat
D.carota	D16255	-72	-67	0	tggtCACGTGctgctt
D.carota	D16255	-298	-293	0	aataTACGTGaaggag
G.max	M16889	-209	-197	0	ccTCCACGTGTCACTt
G.hirsutum	X54091	-193	-182	0	cTTCCACGTGGCAggt
H.annuus	AJ224116	-139	-134	N/A	atacCACGTGccaaca
H.annuus	AJ224116	-191	-186	N/A	aggaCACGTGtatctc
H.annuus	Y00431	-207	-194	0	tTGACACGTGGCTCTc
L.esculentum	X05984	-258	-247	0	cTGACACGTGGCAccc
L.esculentum	X66069	-157	-146	N/A	cttACACGTGTCACCt
L.esculentum	S44160	-256	-245	0	ctgACACGTGGCACCc
N.plumbaginifolia	X12512	-244	-234	0	atCAGACGTGGCAaat
P.sativum	D88260	-69	-64	0	aaaaCACGTGatgttc
P.sativum	D88261	-69	-64	0	aaacCACGTGatgttc
P.sativum	D88262	-69	-64	0	aaacCACGTTatgttc
P.sativum	D10661	-86	-81	0	gaagCACGTGaagtaa
P.sativum	X14207	-72	-62	0	atGCCACCTGGCAgat
P.sativum	X05979	-111	-104	0	aagcCACGTGAAAgaa
P.sativum	K02067	-162	-152	0	gtCACACATGGAAgag
P.sativum	AF060237	255	250	N/A	aaacCACGTGatgttc
S.tuberosum	X04753	-107	-96	0	gTGCCACGTGTCAact
S.tuberosum	Z13987	-276	-266	N/A	aTCACACGTGGCaact
S.tuberosum	Z35160	-232	-226	N/A	cagcCACATGGcaaaa
S.oleracea	S45033	-144	-134	0	acGCCACGTGGCAgta
				Consensus:	nnnMCACGTGnnnnnn

[a] GenBank accession number of the sequence in which binding site is annotated.
[b] start of the binding site relative to start of annotated gene.
[c] transcription start site relative to start of annotated gene.

motif is a matrix model. The basic type of matrix is referred to as counts and just gives the exact number of times each nucleotide is found at every position. As an example we show in Figure 3.1 the counts of the aligned GBF binding sites from Table 3.1.

$$\begin{array}{c} \texttt{A} \\ \texttt{C} \\ \texttt{G} \\ \texttt{T} \end{array} \begin{pmatrix} 18 & 8 & 8 & 14 & 0 & 33 & 0 & 2 & 0 & 1 & 9 & 6 & 17 & 5 & 13 & 9 \\ 8 & 4 & 4 & 17 & 31 & 0 & 33 & 1 & 0 & 0 & 2 & 16 & 3 & 10 & 7 & 9 \\ 3 & 2 & 13 & 1 & 1 & 0 & 0 & 30 & 0 & 31 & 17 & 1 & 9 & 9 & 2 & 3 \\ 4 & 19 & 8 & 1 & 1 & 0 & 0 & 0 & 33 & 1 & 5 & 10 & 4 & 9 & 11 & 12 \end{pmatrix}$$

Figure 3.1: Matrix of nucleotide counts constructed from the aligned sites in Table 3.1.

From this matrix of nucleotide counts we can compute the fraction or frequency of each base at a position in the binding site. We will refer to such a matrix as a position-specific frequency matrix (PSFM). The PSFM computed from the aligned G-box binding sites reported in Table 3.1 is represented in Figure 3.2. In the remainder of this dissertation we will use $\mathbf{\Theta}$ as mathematical symbol for a PSFM of length W. Each entry in this matrix gives the probability $q_{b,j}$ of

$$\begin{array}{c} \texttt{A} \\ \texttt{C} \\ \texttt{G} \\ \texttt{T} \end{array} \left(\begin{array}{cccccccccc} 0.51 & 0.24 & 0.24 & 0.41 & 0.04 & 0.91 & 0.03 & 0.08 & 0.03 & \ldots \\ 0.24 & 0.14 & 0.14 & 0.49 & 0.86 & 0.03 & 0.91 & 0.05 & 0.03 & \ldots \\ 0.11 & 0.08 & 0.38 & 0.05 & 0.05 & 0.03 & 0.03 & 0.84 & 0.03 & \ldots \\ 0.14 & 0.54 & 0.24 & 0.05 & 0.05 & 0.03 & 0.03 & 0.03 & 0.91 & \ldots \end{array} \right.$$
$$\left. \begin{array}{ccccccc} 0.05 & 0.27 & 0.19 & 0.49 & 0.16 & 0.38 & 0.27 \\ 0.03 & 0.08 & 0.46 & 0.11 & 0.30 & 0.22 & 0.27 \\ 0.87 & 0.49 & 0.05 & 0.27 & 0.27 & 0.08 & 0.11 \\ 0.05 & 0.16 & 0.30 & 0.13 & 0.27 & 0.32 & 0.35 \end{array} \right)$$

Figure 3.2: Position-specific frequency matrix constructed from the aligned sites in Table 3.1.

finding a given base b at position j in the binding site, where j varies from 1 to W, the length of the motif. For instance at the sixth the nucleotide `A` appears in approximately 91% of the examples while `C`, `G` and `T` occur in only 3% of the cases.

To get true probabilities in the matrix, we need to compensate for the zero occurrences in the counts. Therefore a small pseudocount is added at each position in the matrix of counts before computing the frequencies. From a Bayesian perspective, it means that if we have not seen an example yet, it does not mean that it does not exist. In this respect, the pseudocounts are mostly chosen proportional to the prior probability of finding the nucleotide in the genome and the contribution of the pseudocounts will be smaller if the number of examples to build the matrix increases. The more examples are present the higher the probability is that the constructed motif model will resemble the true model.

3.2.4 Sequence Logo: visual representation

Finally, there exists also a visual way to represent a motif model which is very useful to interpret the collection of binding sites. Schneider and Stephens [93] used information theory to visualize the set of aligned binding sites. A sequence logo is made by stacking at each position the bases in such a fashion that the height of the stack corresponds to the amount of conserved information $R(j)$ at position j. Given the frequency matrix, the amount of information is calculated as

$$R(j) = 2 - (-\sum_{b=A}^{T} q_{b,j} \log_2 q_{b,j}) \qquad \text{bits.} \tag{3.1}$$

The height of each base in the stack is proportional to the amount of information and the relative frequency of the base and is given by $q_{b,j}R(j)$. As an example we can compute the amount of information at the sixth position in our G-box example as

$$R(6) = 2 + 0.91 \log_2 0.91 + 3 \times 0.03 \log_2 0.03 = 1.82. \qquad \text{bits.}$$

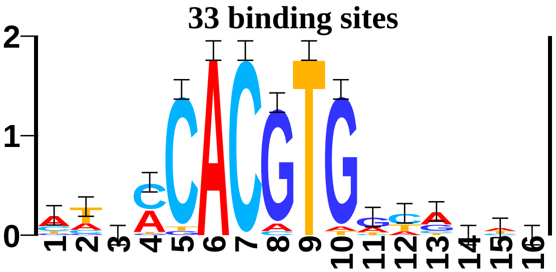


Figure 3.3: Illustration of the sequence logo for the G-box binding sites. The logo immediately draws the attention to which positions are important within the set of binding sites and shows how the core looks like.

Figure 3.3 gives an example of the motif logo compiled from the aligned G-box sites. Such a logo immediately shows which position are most important in the binding and how well-conserved the binding site is. The positions that correspond to the core of the motif contain the largest amount of information. From Figure 3.3 we also learn that the tail of the binding site, positions 14, 15 and 16, that is annotated in many examples as being part of the binding site (see Table 3.1), does not carry any information when compared to a larger set of examples.

3.3 Localizing instances of known motif models

As stated in the introduction to this chapter, there are two ways to search for motifs: supervised and unsupervised. Let us start with an overview of the supervised methods to find known transcription factor binding sites. The identification of potential binding sites starts with matching the chosen motif model against the input sequence(s). The matching can be either string-based or matrix-based. Once the potential sites are identified, some statistic has to be applied to identify the significant instances since not all identified instances will be true binding sites.

3.3.1 String-based searches

The simplest technique to find a potential binding site is to apply a straightforward string search with an identified binding site. Instances of a motif are identified as the subsequences within the DNA sequence that exactly match the given binding site. Allowing only exact matches might be to stringent since there might be some variation in the base composition of the binding site (as was stated in the introduction of this chapter). Scores are calculated by counting the number of mismatches between the site and the query string. Therefore one can look at the full string or make a distinction between the core of the motif and the flanking nucleotides. A threshold on the score is used to classify the selected instances.

If a set of binding sites is accessible, the string-based methods can be extended to search with the consensus sequence of the motif. Since the consensus is based on a degenerated alphabet it can be transformed to a regular expression. For example, the symbol `W` can be replaced by $\{\texttt{A,T}\}$. There exists many programs that can be used to efficiently screen texts with regular expressions (e.g. *fuzznuc* and *dreg* in the EMBOSS toolbox, `http://www.emboss.org/`). Again, a score can be computed based on the number of matches and mismatches and a threshold can be applied to select the best instances.

3.3.2 Matrix-based searches

A reasonable assumption to start from is that each nucleotide in the motif is independently recognized by the active site of the transcription factor. Given this approximation we can compute a score $W(\mathbf{x})$ for a given oligonucleotide $\mathbf{x}$ as the aggregate of the contributions of the individual nucleotides in $\mathbf{x}$. This leads for a motif of length W to a score of the form

$$W(\mathbf{x}) = \sum_{j=1}^{W} W_j(b), \tag{3.2}$$

where $W_j(b)$ is the contribution of the nucleotide b at position j in $\mathbf{x}$. Since this score is additive, it depends on the length of the motif model and it is therefore difficult to compare scores over different motifs. To compensate for the differences, these scores are then normalized with respect to the minimal

W_{min} and maximal W_{max} possible score

$$W_{\text{min}} = \sum_{j=1}^{W} \min_b W_j(b) \qquad \text{and} \qquad W_{\text{max}} = \sum_{j=1}^{W} \max_b W_j(b).$$

The normalized score for a given segment is given by

$$\tilde{W}(x) = \frac{W(\mathbf{x}) - W_{\text{min}}}{W_{\text{max}} - W_{\text{min}}}. \tag{3.3}$$

The assignment of a value to the weights w_{bi} depends on the selected algorithm. The weights are mostly computed directly from the count matrix and are stored in a position weight matrix (PWM). Berg and von Hippel [10] have shown that binding activity is proportional to the logarithm of the observed frequency of the nucleotides, which is also applied in most methods. To select instances a threshold is chosen.

Bucher [15] has applied this methodology to the detection of four eukaryotic regulatory elements: TATA-box, cap-signal, CCAAT-box and GC-box. Bucher computed the weights w_{bi} as

$$W_j(b) = \log\big(\frac{n_{bj}}{e_{bj}} + \frac{\epsilon}{100}\big) + c_j,$$

where n_{bj} is the number of times nucleotide b is counted at position j, e_{bj} is the expected frequency of b at position j, ϵ is a smoothing constant to compensate for zero occurrences of b and finally c_j is an additional term to render column maxima to be zero. The expected frequencies e_{bj} are calculated from either mononucleotide or dinucleotide composition in the genome. Bucher [15] implemented an algorithm that learns the optimal width and threshold for each of the four matrices from promoter sequences in the *Eukaryotic Promoter Database* (EPD, `http://www.epd.isb-sib.ch/`).

Schneider et al. [94] derive the construction of a PWM from information theory (the same methodology that was also applied to create sequence logos). This leads to the following choice for the weights

$$W_j(b) = \log_2 \frac{q_{bj}}{e_b},$$

where q_{bj} is the frequency of b in the binding site at position j and e_b is the genomic frequency of b. Stormo [99] could also relate the derivation of the weights to likelihood statistics and thermodynamics.

The program *Match*, that is incorporated in the TRANSFAC database [117], uses a similar strategy to find motif instances. Each matrix in TRANSFAC is built from binding sites found in the literature and is described by its core and the context. Match defines a threshold for both the core and the complete motif. Predefined thresholds are constructed by comparing the scores of the motif in upstream region and in the second exon. The assumption is that there should be more binding sites present in the upstream region than in the second exon.

Within the framework of our dissertation, we have also developed two programs, *MotifScanner* and *MotifLocator*, to find known binding sites in a given sequence. Both programs have at a higher level the same purpose but the operation and implementation is intrinsically different. MotifScanner starts from the probabilistic sequence model based on the assumption that motifs are hidden in a noisy background sequence. MotifLocator follows the rationale outlined in this section and assigns a score to each segment in the sequence. We will discuss these methods in full detail in Chapter 7.

3.4 Searching in sets of upstream sequences

In the previous section we started from known motifs and tried to localize them within a DNA sequence. Another approach is to search for common features in the regulatory mechanism of genes that behave similarly under certain conditions. Genes that are controlled by the same regulatory mechanism are coregulated. If we are able to define the promoter or upstream regulatory regions of such a set of genes, we should be able to learn the common features of this sequence set. In Figure 3.4, we show an example of such a set of upstream regions of coregulated genes where the known binding sites of two transcription factors are annotated. The sequences are aligned in such a way that the translation start site of the genes is located on the right hand side. These genes are regulated by two factors who bind respectively to `TCACGTG` and `AAACTGTGG`. The goal is to identify from the sequence data these motifs that are specific for this set of sequences.

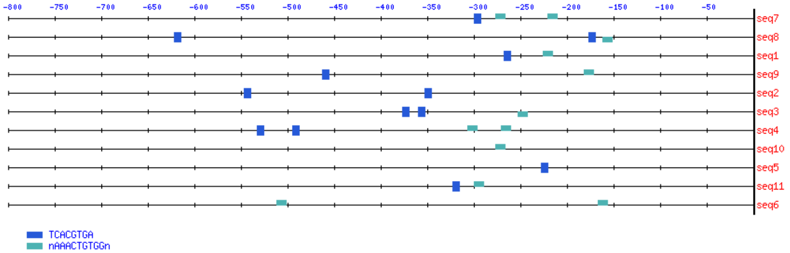


Figure 3.4: Localization of the instances of the two known motifs `TCACGTG` and `AAACTGTGG` in a set of upstream sequences of coregulated genes.

In this section we discuss the different methods that are available to search for motifs that are overrepresented in the upstream region of a set of genes. These methods can be divided in two large classes: string-based methods and methods based on matrix models. Although the division is clear for most algorithms, there exists methods that try to combine both methodologies.

3.4.1 String-based approaches

The first set of algorithms starts from the representation of a motif as a string. These methods start from basic counting algorithms, where the frequency of the motif in a given sequence set is compared to the expected number of occurrences. To account for more advanced models, other techniques like suffix trees and graphs are used to represent sequences and to improve the performance.

Oligonucleotide frequency analysis

The most intuitive approach is a string-based approach to search for overrepresented strings in a set of coregulated sequences. Using such an approach, overrepresentation is typically measured by exhaustive enumeration of all oligonucleotides of a specific length. The observed number of occurrences of a given motif is compared to the expected number of occurrences. The expected number of occurrences and the statistical significance of a motif can be estimated in many ways. In this section we give an overview of the different methods.

A basic version of the enumeration methods was implemented by van Helden et al. [111]. They presented a simple and fast method for the identification of DNA binding sites in the upstream regions from families of coregulated genes in *S. cerevisiae*. Their method searches for short motifs of five, six, or seven base pairs long. First, for each oligonucleotide of a given length, the expected frequency is computed from all the non-coding, upstream regions in the genome of interest. Based on this frequency table, the expected number of occurrences of a given oligonucleotide in a specific set of sequences can be estimated. Given the expected number of occurrences, the probability of observing at least the counted number of occurrences in the data set can be computed using a binomial statistic. Next, a significance coefficient is computed taking into account the distinct number of all possible oligonucleotides. Finally, the retrieved oligos are grouped together to extend the motifs.

A similar methodology was proposed by Bussemaker et al. [17], who used a probabilistic segmentation model to represent upstream regulatory regions. A DNA sequence is decomposed into words and concurrently a *dictionary of words* is composed. The *dictionary* is the set of all words and their associated probabilities. This dictionary can be used to compute the expected frequency of the given oligonucleotide.

Spaced-dyads

van Helden et al. [113] extended their method to find spaced dyads. Spaced dyads are motifs consisting of two short conserved boxes separated by a region of fixed size and variable content, called spacer. According to the biological model, the length of the conserved words is fixed to three nucleotides but the length of the spacer is different for each motifs. Therefore different lengths of the spacer have to be systematically examined. van Helden et al. choose to check all lengths between 0 and 16. There are two ways to compute the significance of this type of motif. First the frequency of each conserved word is counted and then these frequencies are combined to compute the expected frequency of the dyad. Second, the expected frequency can be based on the estimated complete dyad frequency from a background data set.

Exact methods to estimate oligo frequencies

The greatest shortcoming of this implementation of oligonucleotide analysis is that there are no variations allowed within an oligonucleotide. Tompa [110] addressed this problem when he proposed an exact method to find short motifs in DNA sequences. Tompa used a different measure than van Helden et al. to calculate the statistical significance of motif occurrences. First, for each *k*-mer s with an allowed number of substitutions, the number of sequences in which s is found is calculated. Next, the probability p_s of finding at least one occurrence of s in a sequence drawn from a random distribution is estimated. Finally, the associated *z*-score is computed as

$$z_s = \frac{N_s - Np_s}{\sqrt{Np_s(1-p_s)}}.$$

z_s gives a measure of how unlikely it is to have N_s occurrences of s given the expected expected number of occurrences Np_s. Tompa proposed an efficient algorithm to estimate the expected frequency p_s of a word from a set of background sequences based on a Markov chain.

Nicodème [77] presented another exact algorithm to compute the statistics of number of occurrences of a regular expression in random text. His approach is based on the rationality of generating functions, analytic combinatorics and computer algebra for determining and analyzing generating functions. Such a generating function encodes exactly all the information relative to the frequency of occurrence of a pattern in random text of arbitrarily large sizes.

Suffix trees

Another interesting string-based approach is based on the representation of a set of sequences with a suffix tree [89, 114]. Vanet et al. [114] have used a suffix tree to search for single motifs in whole bacterial genomes. The representation of upstream sequences as suffix trees resulted in an efficient implementation despite the large number of possible combinations. This method was extended by Marsan and Sagot [71] to search for combinations of motifs. The proposed configuration of a structured motif is a set of p motifs separated by a spacer that might be variable. The variability is limited to ± 2bp around an average gap length. They also allow for a small variability within the binding sites by setting a maximal number of mismatches between the selected site and the consensus sequence.

Graph-based methods

Within the class of string-based methods, graph-based approaches are very popular among computer scientists. Pevzner and Sze [85] introduced the so-called *subtle* motifs as a computational challenging problem. These motifs are built from a consensus of a certain length in which a specific number of substitutions is allowed and there is no restriction on the position where the substitution occurs. Pevzner and Sze [85] presented two methods WINNOWER and SP-STAR to solve this problem. Later the method was refined to a new algorithm MULTIPROFILER [50]. These motifs have been proven to be computational efficient.

3.4.2 Matrix-based methods for motif finding

Instead of the string based approaches, the problem of motif finding can also be tackled by trying to learn a matrix model that describes the binding sites. In this section we focus on the two main implementations:
Expectation-Maximization and *Gibbs sampling*.

Basic sequence model

The matrix-based methods mostly start from a probabilistic sequence model. This model ranges from simple and rather stringent models to more complex models, which better represent the biological problem at hand. The basic model assumes that there is a common transcription factor and there is exactly one instance of this motif hidden in every sequence in the data set. Except for the motif instances, the sequences are considered as a noisy sequence where each base is generated according to a specific **background model**. In this example, each base is drawn independently from a single discrete distribution $\mathcal{B}_0 = [q_{\mathtt{A}0}, q_{\mathtt{C}0}, q_{\mathtt{G}0}, q_{\mathtt{T}0}]^T$. The motif model itself is represented with a position-specific frequency matrix $\mathbf{\Theta}$ of length W as seen in Section 3.2.

$$\mathbf{\Theta} = \begin{pmatrix} q_{\mathtt{A},1} & q_{\mathtt{A},2} & \cdots & q_{\mathtt{A},W} \\ q_{\mathtt{C},1} & q_{\mathtt{C},2} & \cdots & q_{\mathtt{C},W} \\ q_{\mathtt{G},1} & q_{\mathtt{G},2} & \cdots & q_{\mathtt{G},W} \\ q_{\mathtt{T},1} & q_{\mathtt{T},2} & \cdots & q_{\mathtt{T},W} \end{pmatrix}, \tag{3.4}$$

where $q_{b,j}$ is the probability of finding nucleotide b at position j in the binding site. Each column i in the PSFM corresponds to the parameters of a discrete distribution represented as $\boldsymbol{\theta}_j = [q_{\mathtt{A},j}, q_{\mathtt{C},j}, q_{\mathtt{G},j}, q_{\mathtt{T},j}]^T$.

If we know the start location a_k of the motif instance in sequence S_k, the probability of this sequence given the motif position, the motif model $\mathbf{\Theta}$, and the background model $\mathcal{B}_0$ is

$$P(S_k|a_k, \boldsymbol{\Theta}, \mathcal{B}_0) = \underbrace{\prod_{l=1}^{a_k-1} \mathcal{B}_0(b_{kl})}_{\text{background}} \underbrace{\prod_{j=1}^{W} \boldsymbol{\theta}_j(b_{k,a_k+j-1})}_{\text{motif}} \underbrace{\prod_{l=a_k+W}^{L_k} \mathcal{B}_0(b_{kl})}_{\text{background}}, \qquad (3.5)$$

where b_{kl} is the nucleotide at position l in sequence S_k and $\mathcal{B}_0(b_{kl})$ the probability of finding the nucleotide b_{kl} according to the background model.

Equation 3.5 forms the basic foundation of our research. To expand this model to a whole sequence set, we first need to define a sequence set as $\mathcal{S} = \{S_k|k = 1, \ldots, N_S\}$ and the *alignment* as the set of all the start positions $\mathcal{A} = \{a_k|k = 1, \ldots, N_S\}$. Given the assumption that all sequences are independent, then the probability of the whole sequence set $\mathcal{S}$, given the alignment $\mathcal{A}$, the motif model $\boldsymbol{\Theta}$ and the background model $\mathcal{B}_0$ is

$$P(\mathcal{S}|\mathcal{A}, \boldsymbol{\Theta}, \mathcal{B}_0) = \prod_{k=1}^{N_S} P(S_k|a_k, \boldsymbol{\Theta}, \mathcal{B}_0). \qquad (3.6)$$

Greedy search

One of the first implementations to learn a matrix representation from a set of sequences was a greedy search algorithm to find the matrix with the highest information content [41]. Their algorithm was capable of identifying a common motif that is present once in every sequence. Over the recent years, this algorithm has been substantially improved [42]. With *CONSENSUS*, Hertz and Stormo have provided a framework to estimate the statistical significance of a given information content score based on large deviation statistics.

Expectation Maximization

Within the maximum likelihood estimation framework, Expectation-Maximization (EM) is the first choice of optimization algorithm. EM is a two-step iterative procedure for obtaining the maximum likelihood parameter estimates for a model of observed data and missing values. The algorithm is started with a set of initial model parameters. In the *expectation step*, the expectation of the data and missing values is computed given the current set of model parameters. In the *maximization step*, the model parameters that maximize the likelihood are computed. EM iterates over the two steps until the parameters have converged. Since EM is a gradient ascent method, it is guaranteed to converge, but EM strongly depends on the initial conditions. Poor initial parameters may lead EM to converge to a local minimum.

EM for motif finding was introduced by Lawrence and Reilly [53] and was an extension of the greedy algorithm of Hertz et al. [41]. Although it was primarily intended for searching motifs in related proteins, the described method could also be applied to DNA sequences. Their proposed model is conform the assumptions outlined above. Each sequence contains exactly one instance of the motif, which might be reasonable in proteins but is too strict in DNA. The starting position of each motif instance is unknown and is considered as being a missing value from the data. If the motif positions are known then the observed frequencies of the nucleotides at each position in the motif are the maximum likelihood estimates of model parameters. To find the starting positions, each subsequence is scored with the current estimate of the motif model. These updated probabilities are used to re-estimate the motif model. This procedures is repeated until convergence. While performing well for extended protein motifs, EM often suffers badly from local minima for short DNA motifs. Shortly after the first implementation of EM for motif finding, [19] proposed an EM algorithm to search for gapped motifs.

Since assuming there is exactly one copy of the motif per sequence does not hold for binding sites in DNA sequences, Bailey and Elkan proposed an advanced EM implementation for motif finding called MEME [7, 8]. To overcome the problem of initialization and getting stuck in local minima, MEME proposes to initialize the algorithm with a motif model based on a contiguous subsequence that gives the highest likelihood score. Therefore, each substring in the sequence set is used as a starting point for a one-step iteration of EM. Then the computed motif models are ordered in decreasing order of likelihood. The best motif is retained and used for further optimization steps. After convergence the corresponding motif positions are masked and the procedure is restarted with the next motif model in the list.

Gibbs Sampling

The applicability of Gibbs sampling to solve a missing value problem [101], has lead to the implementation of a Gibbs sampler for motif finding. The derivation of the exact algorithm was presented by Lawrence et al. [54] and a more detailed and technical presentation of the method was described by Liu et al. [63]. In Chapter 4 we will discuss in detail the foundations of the Gibbs Sampling method and we will give the basic algorithm. Several groups also proposed methods to fine-tune the Gibbs sampling algorithm for motif finding in DNA sequences and sets of coregulated genes. Here we give a short description of these methods. Gibbs sampling for motif finding forms the major topic of our research and this dissertation. Algorithmic aspects and results of our implementation, MotifSampler [104, 106], are given in Chapter 5.

A version of the Gibbs sampling algorithm that was especially tuned towards finding motif in DNA sequences is *AlignACE* [47, 88]. This Gibbs sampling algorithm was the first reported to be used for the analysis of promoter sequence of gene clusters. Several modifications were made in AlignACE with respect to the original Gibbs sampling algorithm. First, one motif at the time was retrieved and the positions were masked instead of simultaneous multiple motif searching. Second, AlignACE was implemented with a fixed single nucleotide background model based on base frequency in the sequence set.

Also, both strands were included in the search. Finally, in the latest version, the *maximum a posteriori* likelihood score was used to judge the quality of different motifs.

Liu et al. [64] implemented *BioProspector* which extends the Gibbs sampling strategy to search for common motifs in the regulatory region of coexpressed genes. First, BioProspector uses zero to third-order Markov background models to describe the DNA sequences. The predictive update formula (see Chapter 4) is changed in such a way that the probability of the instance being generated by the background model is given by this higher-order background model. Second, the core sampling step was replaced by a *threshold sampler*. This threshold sampling step was incorporated to estimate the number of copies of a motif in a sequence. The program defines two threshold T_L and T_H. Instances with a score higher than T_{H} will be automatically selected while there will be one motif sampled from those motifs that have a score between T_{L} and T_{H}. T_{H} is set proportional to the product of the average length of the input sequences and the motif width. T_{L} is initialized at 0 and linearly increase till it reaches the value of $T_{\mathrm{H}}/8$. This threshold sampling step ensures faster convergence. BioProspector also proposes two possible alternative motif models. The first possibility is to search for palindromic motifs. The second possibility is to search for a gapped version of the motif model, where the motif consists of two blocks separated by a gap of variable length. The gapped version searches for two motifs at the same time that occur within a given gap range.

Ann_spec [119] has its origin in the Gibbs sampling framework but approaches the representation of the motif model rather differently. The motif model is represented with a sparsely encoded perceptron with one processing unit. The weights of the perceptron resemble the position weight matrix. This model is based on the approximation of the total protein binding energy by the sum of partial binding energies at the individual nucleotides in the binding sites. The use of a perceptron is also justified by the fact that it can be used to approximate posterior probability distributions. A gradient descent training method is used to find the parameters of the perceptron. To train the perceptron, a training set of positive examples is selected using a Gibbs sampling procedure. Negative examples can be either constructed from random sequences or from genomic data. To improve the specificity of the motif model, a background model based on an independent data set is preferred.

3.4.3 Combined and extended methods

So far, we discussed the basic algorithms to find motifs in sets of coregulated genes. Typically these algorithms work on a two-step basis: first defining a set of regulatory regions and next motif finding. However, the availability of new sources of large scale measurements have increased the demand for specialized methods. Especially, the presence of noise in clusters forces researchers to apply new methods. Also, the continuous completion of new genomes allows researchers to compare related species and to infer functional elements by phylogenetic footprinting. Another extension is the search for cooperating transcription factors. In higher eukaryotes this type of regulation is imminent and hence there is growing demand for algorithms that can search for multiple motifs at the same time.

Correlating expression data and motif finding

The two-step approach of clustering the expression profiles and looking for common motifs is one way to analyze microarray experiments. However, several groups have proposed to directly correlate expression data and motifs, since clustering is rather sensitive to noise. Bussemaker et al. [18] proposed a model where instances of a motif additively contribute to the log-expression of the level of a gene. For each oligomer of a specified length (up to 7bp) the correlation with the expression is measured by fitting a single-motif model to the data for all genes. Next, an iterative procedure is used to select the significant motifs.

Holmes and Bruno [45] suggested to map the two-step approach into an integrated likelihood framework. The two stages can be viewed as operating on the marginal distributions of a joint probabilistic model for sequence and expression data. They reformulated the problem as follows: find clusters of genes that have both a similar expression profiles and similar promoters. Their implementation is *kimono*, k-means integrated models for oligonucleotide arrays. A specific scoring function is designed to compare each sequence-expression pair to alternative sequence-expression models estimated dynamically from the data. Each model corresponds to a cluster in the k-means formulation. The scoring function is a weighted sum of a sequence score (a log-odds score derived from an alignment estimated by Gibbs sampling) and an expression score (distance from the mean expression profile for the cluster). The weighting factor determines the principal factor in the analysis. If most weight is put on the expression data the model resembles the classical two-step approach.

Another type of expression data, that is exploited with *MDscan* [65], is the combination of chromatin immunopreciptation followed by cDNA microarrays hybridization (*CHIP-array*). Chip-arrays have become a useful tool to study genome-wide protein-DNA interactions. The best ranked sequences of these experiments have the highest probability to have multiple protein-DNA interaction sites. MDscan uses this ranking information to speed up and improve the motif detection procedure. MDscan starts with a word-counting strategy on the top scoring sequences to look for sets of abundant oligonucleotides. A matrix is built from each set and only the top motifs are retained for refinement. In the updating steps, each candidate motif is used to score all the remaining sequences. A oligonucleotide is added to the motif model if the score of the matrix is increased. This refinement procedure is repeated several times until the motif stabilizes.

Motif finding for phylogenetic footprinting

Due to the imminent increase of fully sequenced genomes, many researcher are working on the comparison of related species. Phylogenetic footprinting allows biologist to delineate the regions within DNA which are conserved between species. Since these regions are conserved by selective pressure throughout evolution, they are supposed to play an important role in the development of the organism. Analysis of the conserved regions upstream of a gene can reveal useful insights in the function and expression of the gene. Blanchette et al. [13] have developed a word-counting method to identify common binding sites in orthologous sequences. They start from the phylogenetic tree of N_S related species and use this information in the optimization procedure when computing subsequence scores. The rationale behind their approach is that closely related species have a much higher level of homology and this will bias the results. To compensate for this bias, it is necessary to weight the sequences such that the common motif in all sequences is favored. McCue et al. [73] have used a Gibbs motif finding algorithm for phylogenetic footprinting.

Searching for combinations of motifs

Currently, the focus in higher eukaryotes is shifting from single motif detection to finding modules or combinations of different motifs. As stated in Chapter 2, comparison of completed genomes illustrates that number of genes does not correlate directly to the complexity of the organism [70]. The great diversity in specialization of human cells –all of them with the same 30,000 genes– can be best explained by the variance in the *cis*-regulatory control mechanism [81]. Combining variable *cis*-regulatory elements results in different gene expression dependent on the environmental or cellular conditions. From these observations it is clear that algorithms that search for combinations of binding sites of cooperating transcription factors gain importance in bioinformatics research. We give some examples of such algorithms here.

Ann_spec was recently extended to search for cooperatively acting transcription binding factors by GuhaThakurta and Stormo [37]. *Co-Bind* searches for two motifs simultaneously by combining the weights that optimize the objective functions of the two individual perceptrons. In simulated data and examples in yeast and *E.coli*, the identification of two motifs simultaneously improved significantly the detection of the true motifs compared to the classical methods searching for one motif at the time.

More complex examples are tackled in human, mouse and fly data sets. Several groups have worked with modules of known motifs [11, 33, 38]. They typically start by screening sequences with a collection of PWMs generated from known binding sites of related factors. Berman et al. [11] searched for regions with high concentration of Bcd, Cad, Hb, Kr and Kni binding sites in regions of the *Drosophila melanogaster* genome. Halfon et al. [38] also used *Drosophila* as a test case of their method. Frith et al. [33] identify clusters of motifs by fitting an hidden Markov model (HMM) to the regulatory sequences. They have developed a method to compute the statistical significance of a cluster of motifs given the motif models and the parameters of the HMM.

3.5 Motif finding on the net

To end this chapter on the current status of motif finding algorithms, we summarize the results in two tables. Table 3.2 gives an overview of the most important database in which transcription factors, experimental verified binding sites and motif models are stored. The largest database is TRANSFAC in which transcription factors can be found for all organisms. There exists also more specialized databases like PlantCARE and SCPD. For a more complete review of all molecular biology database available online, we can refer the interested reader to the annual database issue of *Nucleic Acids Research* (`http://nar.oupjournals.org/`).

In Table 3.2 an overview of the methods described in this chapter for which we could find a reference webpage is summarized. These methods can either be directly used online or a software package is available for download and local installation. One problem we would like to mention here is that currently there is no common standard to represent the results of a motif finder. The plentitude of different formats makes it hard to compare results of large scale analysis.

Table 3.2: Availability of motif finding algorithms

Package	URL
TRANSFAC	`http://transfac.gbf.de/` (all species)
SCPD	`http://sigma.cshl.org/` (yeast)
PlantCARE	`http://psb.rug.ac.be/` (plants)
RegulonDB	`http://www.cifn.unam.mx/Comptutational_Genomics/regulondb/`
PLACE	`http://www.dna.affrc.go.jp/htdocs/PLACE` (plants)
Match	`http://transfac.gbf.de/`
EMBOSS	`http://www.emboss.org/`
RSA tools	`http://www.ucmb.ulb.ac.be/bioinformatics/rsa-tools/`
YMF	`http://abstract.cs.washington.edu/~blanchem/cgi-bin/YMF.pl`
Consensus	`http://ural.wustl.edu/softwares.html`
MEME	`http://meme.sdsc.edu/meme/website/`
Gibbs Motif Sampler	`http://bayesweb.wadsworth.org/gibbs/gibbs.html`
AlignACE	`http://atlas.med.harvard.edu/`
INCLUSive	`http://www.esat.kuleuven.ac.be/~dna/BioI/Software.html`
BioProspector	`http://bioprospector.stanford.edu/`
MDscan	`http://bioprospector.stanford.edu/MDscan/`
kimono	`http://www.okchicken.com/~yam/kimono`
COMET	`http://zlab.bu.edu/~mfrith/comet`

CHAPTER 4

Gibbs Sampling for Motif Finding

In this chapter we outline the underlying methodologies on which we build our motif finding algorithm. In the first section we restate the Metropolis algorithm to draw samples from complex distributions. In the next section Gibbs sampling is introduced as special sampling scheme. Another important issue we address, is the missing data problem and the basic models to find the parameters of such a model. Then the link is made between the missing data problem and Gibbs sampling. Based on these theoretical foundations, the basic algorithm of the Gibbs sampler for motif finding [54] is derived and a high level implementation is described. To end the chapter, we also discuss some of the problems of this basic version that we will address in the next chapters.

4.1 Markov Chain Monte Carlo (MCMC)

As pointed out by Liu [61], Markov chain Monte Carlo methods have become increasingly important for scientific computing and many applications have been developed in recent years that use Markov chain Monte Carlo or related methods. Advances in Monte Carlo methods include, among others, clustering algorithms, data augmentation, parameter expansion and many different related sampling schemes. These advances are explained by the trend of applying Markov Chain Monte Carlo methods (MCMC) to problems where samples are generated from very complex distributions. The interested reader we can refer to the book by Liu [61] in which full details are given about sampling methods in scientific computing.

4.1.1 Metropolis algorithm

In many problems one needs to compute an integral of the form

$$I = \int_D f(\mathbf{x}) d\mathbf{x},$$

where D is some region in a possibly high-dimensional space and $f(\mathbf{x})$ is the target function. An estimation of I can be computed as

$$\hat{I} = \frac{1}{N}\big(f(\mathbf{x}_1) + f(\mathbf{x}_2) + \cdots + f(\mathbf{x}_N)\big),$$

if one is able to draw N independent and identically distributed random samples from D. Generating these samples that follow a specific distribution is thus an important step and this is where Monte Carlo methods appear.

A fundamental step in all Monte Carlo methods is to generate pseudo-random numbers that follow a specific probability distribution $\pi(\mathbf{x})$. Typically, $\mathbf{x}$ resides in a higher dimensional space $\mathbb{R}^k$ and directly generating independent samples from $\pi(\mathbf{x})$ is in most practical cases unfeasible. Potential problems might be that the generated samples are dependent or that the distribution to generate the samples differs from $\pi(\mathbf{x})$. There exist several methods to generate dependent or independent sample from a *trial distribution* $p(\mathbf{x})$, which differs from, but should be similar to $\pi(\mathbf{x})$. Most well-known schemes are *rejection sampling*, *importance sampling* and *sampling-importance-resampling*. Another scheme is presented in the Metropolis algorithm which serves as the basic building block of MCMC. In the Metropolis algorithm dependent samples are generated from an evolving Markov chain with $\pi(\mathbf{x})$ as its equilibrium state.

Let us now briefly restate the Metropolis algorithm that forms the basis of many sampling schemes. Assume that the target probability distribution $\pi(\mathbf{x})$ can be written in the form $\pi(\mathbf{x}) = c\exp(-h(\mathbf{x}))$, with c a normalizing constant. Metropolis introduced the fundamental idea of evolving a Markov chain to achieve the sampling of $\pi(\mathbf{x})$. The Markov chain is such that the equilibrium distribution of this chain is the target distribution $\pi(\mathbf{x})$.

The Metropolis algorithm starts from an initial state $\mathbf{x}^{(0)}$ and iterates the following two steps:

1. Randomly perturb the current state from $\mathbf{x}^{(t)}$ to $\mathbf{x}'$ and calculate $\Delta h = h(\mathbf{x}') - h(\mathbf{x}^{(t)})$. $\mathbf{x}'$ can be generated from a *symmetric* probability transition function T, with $T(\mathbf{x}^{(t)}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x}^{(t)})$.
2. Generate a random number $u \sim$ uniform$(0, 1)$. Let $\mathbf{x}^{(t+1)} = \mathbf{x}'$ if $u \leq \exp(-\Delta h)$ and $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$ otherwise.

The Metropolis algorithm is based on a *trial-and-error* strategy. At each iteration, the current state of $\mathbf{x}^{(t)}$ is randomly perturbed and the gain in the objective function from this perturbation $\mathbf{x}^{(t+1)}$ is computed. Next, the new state $\mathbf{x}^{(t+1)}$ is accepted if the gain is larger than $\log(u)$ with u a random number drawn uniformly between 0 and 1. If this gain is smaller than $\log(u)$ the new configuration is rejected. The perturbation rule is in Metropolis' algorithm restricted to a symmetric probability transition function $T(\mathbf{x}, \mathbf{x}')$. This means that the probability of obtaining $\mathbf{x}'$ from $\mathbf{x}$ is the same as the chance of obtaining $\mathbf{x}$ by perturbing $\mathbf{x}'$ or written mathematically

$$T(\mathbf{x}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x}).$$

Later Hastings extended the Metropolis algorithm to the case where T is not necessarily symmetric. He only imposed the restriction that $T(\mathbf{x}, \mathbf{x}') > 0$ if and only if $T(\mathbf{x}', \mathbf{x}) > 0$.

Although it is possible, in theory, to use MCMC methods to generate random samples from virtually any type of target distribution, a potential problem of these methods is that the resulting samples might be highly correlated. Therefore, the estimates resulting from these sampling have greater variance then those resulting from independent samples.

4.2 Gibbs Sampler

The Gibbs sampler was introduced by Geman and Geman [35] and is a special case of the original MCMC scheme. The basic idea of the Gibbs sampler is constructing the underlying Markov chain by composing a sequence of conditional distributions along a set of selected directions. When using the classical Metropolis algorithm, the transition function $T(\mathbf{x}, \mathbf{x}')$ is often chosen to be a locally uniform move. Although such a proposal is operationally simple, the performance is often observed as being inefficient. A possible solution to this problem is the use of conditional transitions as introduced in the Gibbs sampler. Tanner and Wong [101] linked the Gibbs sampler to the missing data problem in their data augmentation algorithm. Gelfand and Smith [34] linked the conditionals in the Gibbs sampler to those commonly available in Bayesian and likelihood computations.

4.2.1 Basic principle

Before we talk about the application of Gibbs sampling for motif finding, let us first explain the basic principles of the Gibbs sampler with a simple example. Suppose we like to take samples from a target distribution $\pi(\mathbf{x})$ and that we can decompose the variable $\mathbf{x}$ into three components $\mathbf{x} = (x_1, x_2, x_3)$. The target distribution becomes then the joint over these three variables $\pi(x_1, x_2, x_3)$. Further suppose that it is too difficult to sample directly from this joint distribution, but that we are able to construct the conditional distributions $\pi(x_1|x_2, x_3)$, $\pi(x_2|x_1, x_3)$ and $\pi(x_3|x_1, x_2)$ an that we are also able to sample from them. Starting from the initial conditions $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$, a *systematic scan Gibbs sampler* can be implemented by iterating over the following steps:

1. Draw $x_1^{(t+1)}$ from $\pi(x_1|x_2^{(t)}, x_3^{(t)})$
2. Draw $x_2^{(t+1)}$ from $\pi(x_2|x_1^{(t+1)}, x_3^{(t)})$
3. Draw $x_3^{(t+1)}$ from $\pi(x_3|x_1^{(t+1)}, x_2^{(t+1)})$

It can be shown that each individual conditional update leaves the target distribution $\pi(\mathbf{x})$ invariant. Under regularity conditions the Gibbs sampler chain converges geometrically and the convergence rate is related to the correlation between variables (for the proof see [60]). Instead of *systematic scan Gibbs sampler* also a *random scan Gibbs sampler* can deployed. In that case we first select one of the variables at random and draw it conditional on the current state of the other variables. Since conditional moves seems to be more effective then random perturbations, the use of conditional distributions in each iteration explains in a sense its popularity.

4.2.2 Grouping and collapsing variables

In some cases it might be advantageous to group together some components to design a reduced form of the Gibbs sampler. Assume that in our three component example we are able to group together x_2 and x_3 as x_g. This allows creating a sampler on the reduced variable $\mathbf{x}^\star = (x_1, x_g)$. In this scheme we can draw x_2 and x_3 simultaneously conditional on x_1, which results in the following scheme.

1. Draw $x_1^{(t+1)}$ from $\pi(x_1|x_g^{(t)})$.
2. Draw $x_g^{(t+1)}$ from $\pi(x_g|x_1^{(t+1)})$.

One step further is the collapsed Gibbs sampler. Assume x_3 can be integrated out such that the marginal density $\pi(\mathbf{x}^-)$ can be computed as $\pi(\mathbf{x}^-) = \int \pi(\mathbf{x})dx_3$. It is then possible to construct a sampling scheme on the reduced variable $\mathbf{x}^- = (x_1, x_2)$.

Liu [60] has shown that under mild conditions the collapsed sampler converges faster than the grouped sampler. The grouped sampler in turn converges faster than the basic Gibbs sampler.

4.3 Missing data problem

Missing data form an important problem in scientific computations. The general framework of the missing data formulation originated from the need to analyze survey data where parts of the data were missing. The question one can ask is how we can conduct computations if part of the data is missing. But not only surveys contain missing data, also other problems can be formulated as a missing data problem. In this case, some variables might be thought of as being missing but ”missing”should not really be seen in a literal sense. Using the missing data framework helps in making some model assumptions explicit and it also provides a framework for inference. As we will show in the next sections, the missing data formulation also helps in introducing meaningful variables that might be high-dimensional into the model. As an example of such a missing data problem we can refer to a classification problem that is formulated as probabilistic model. In such a case, the class labels are missing from the observations. By adding these labels to the model an augmented system might appear that is easier to solve.

4.3.1 Basic principles

Let us first formulate the missing data problem mathematically. If we represent the parameters by θ and the complete-data as $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ are the observed variables and the $\mathbf{y}$ missing observations. The *complete-data likelihood* is then

$$L(\theta; \mathbf{x}, \mathbf{y}) = \pi(\mathbf{x}, \mathbf{y}|\theta).$$

The goal is to find those parameters that best describe the data. There exists two possible ways to solve the missing data problem: maximum-likelihood estimation (MLE) and Bayesian inference.

In a *maximum-likelihood estimation* method the unknown parameters θ are estimated by finding the value $\hat{\theta}$ that maximizes the complete-data likelihood $\pi(\mathbf{z}|\theta)$. The maximum-likelihood estimate can be found as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \quad L(\theta; \mathbf{x}).$$

To estimate θ we have to evaluate at least the likelihood function of the observed data. This likelihood function can be found by marginalizing the missing data from the complete-data likelihood:

$$L(\theta; \mathbf{x}) = \int \pi(\mathbf{x}, \mathbf{y}|\theta) d\mathbf{y}.$$

In the *Bayesian inference* methods, the posterior distribution of θ, $p(\theta|\mathbf{z})$ is needed to make inferential statements. In this setting we impose a prior distribution $f(\theta)$ on the unknown parameters θ. The joint distribution of the data and the parameters is written as

$$f(\theta, \mathbf{x}, \mathbf{y}) = \pi(\mathbf{x}, \mathbf{y}|\theta) f(\theta).$$

To compute the posterior distribution of θ, $f(\theta|\mathbf{x})$ we need to carry out the computation of the integral

$$f(\theta|\mathbf{x}) = \int \pi(\theta|\mathbf{x}, \mathbf{y}) f(\mathbf{y}|\mathbf{x}) d\mathbf{y}.$$

In both cases, one needs to solve an integral over a complex function which is very hard to compute analytically. To solve such a problem sampling procedures can be deployed to estimate the integral with a sum over the samples. Hence, to estimate these integrals one can rely again on sampling methods.

4.3.2 Expectation-Maximization

To compute the maximum-likelihood estimates, Dempster et al. [28] introduced the Expectation-Maximization (EM) algorithm. EM is a two-step iterative procedure to find the parameters $\hat{\theta}$ that maximize the complete-data log-likelihood function $L(\theta; \mathbf{z})$. EM starts with an initial guess $\theta^{(0)}$ of the parameters.

1. **Expectation step**

$$g(\theta|\theta^{(t)}) = E[(\log L(\theta; \mathbf{z})|\mathbf{x}, \theta^{(t)}] = \int \log\big(L(\theta; \mathbf{z})\big)\pi(\mathbf{y}|\mathbf{x}\theta^{(t)})d\mathbf{z}.$$

2. **Maximization step**

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}} \quad g(\theta|\theta^{(t)}).$$

In the expectation step, the expected value of the complete-data log-likelihood is computed based on the current set of parameters and the observed data. In the maximization step the value of the parameter that maximizes the expectation of the complete-data log-likelihood is computed. In other words, in the expectation step a probabilistic value is filled in for the missing data. These values can then be used to compute the parameters that best describe the data. It can be proven that the likelihood will always increase with this iteration scheme. This means also that the EM algorithm is deterministic in nature. Therefore, the EM can only guarantee to convergence to a local optimum $\hat{\theta}$ of the observed data likelihood. There is no principled way to escape from such a local mode. This implies that the solution found by EM strongly depends on the initial conditions. It is thus of great importance to find the right initial conditions to start the EM procedure.

4.3.3 Gibbs sampling and data augmentation

While EM gives the maximum likelihood estimates, the missing data problem could also be solved as a Bayesian inference problem. The particular structure of the Gibbs sampler makes it well suited in such a Bayesian inference problem. The goal is design a sampling scheme where one is able to sample from the equilibrium distribution $\pi(\mathbf{y}, \theta|\mathbf{x})$ of the augmented system. After initialization of the prior distribution $p_0(\theta)$, the following iteration scheme is applied:

1. Draw $\theta^{(t+1)}$ from $\pi(\theta|\mathbf{x}, \mathbf{y}^{(t)})$
2. Draw $\mathbf{y}^{(t+1)}$ from $\pi(\mathbf{y}|\mathbf{x}, \theta^{(t+1)})$

The Gibbs iterations correspond to drawing the parameter values θ conditional on currently imputed missing data $\mathbf{y}$ and then imputing the missing data $\mathbf{y}$ conditional on the current parameters θ. The implementation of this sampling scheme might be straightforward if the prior distribution $p_0()$ is one of the standard ones (eg. Dirichlet or Beta distributions).

4.4 Gibbs sampling for motif finding

If we look again at the motif finding problem, the problem is two-fold: (1) what are the parameters of the motif model and (2) where are the motif instances located in the sequence set. It is clear that this problem can be formulated as a missing data problem. The starting positions of the motif instances are the missing values and the parameters come from the motif model. Lawrence and Reilly [53] introduced an EM implementation for the detection of transcription factor binding sites in a set of DNA sequences. Based on the relationship between Gibbs sampling and EM, explained in the previous sections, a Gibbs sampler can be used to solve this problem. In this section we show how the principles of the Gibbs sampler can be combined into a workable algorithm as it was first described by Lawrence et al. [54]. The algorithm is in essence a collapsed version and the full details are given by Liu et al. [63].

4.4.1 Basic sequence model

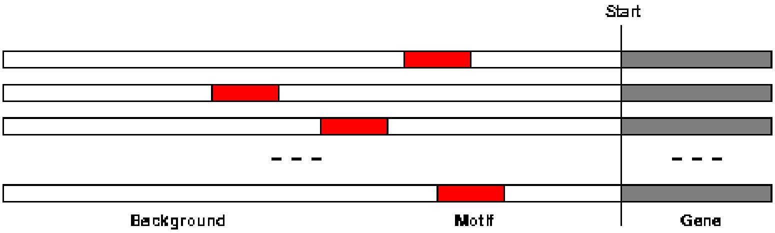


Figure 4.1: Schematic representation of an upstream sequence set. Each sequence is expected to contain exactly one copy of the given motif model.

We start from the basic model in which we assume that there is exactly one motif instance in each sequence S_k as is depicted in Figure 4.1. Given a set of sequences $\mathcal{S} = \{S_k | k = 1 \dots N_s\}$, the goal of the algorithm is twofold: (1) identify the starting position of the instance of the motif in each DNA sequence to which the common factor might bind and (2) estimate the parameters of the motif model. The motif model is represented with position-specific probability matrix $\boldsymbol{\Theta}$. This model $\boldsymbol{\Theta}$ corresponds to a product multinomial model and can be written as

$$\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_W],$$

with $\boldsymbol{\theta}_j = [q_{\mathtt{A},j}, q_{\mathtt{C},j}, q_{\mathtt{G},j}, q_{\mathtt{T},j}]^T$. The instances themselves are represented as $\mathbf{x}_{a_k}$, with the start position a_k of the selected instance in sequence S_k. The nucleotides not belonging to a motif instance are assumed to be drawn independently from a background model $\boldsymbol{\theta}_0$, where $\boldsymbol{\theta}_0$ is a multinomial model with parameters $[q_{\mathtt{A}0}, q_{\mathtt{C}0}, q_{\mathtt{G}0}, q_{\mathtt{T}0}]^T$.

Treating the starting positions $\mathcal{A} = \{a_k | k = 1 \dots N_s\}$ as missing data and assuming independence of sequences in the data set, the complete data likelihood can be written as

$$\begin{aligned}\pi(\mathcal{S},\mathcal{A}|\boldsymbol{\Theta},\boldsymbol{\theta}_0) &= \prod_{k=1}^{N_s}\pi(S_k,a_k|\boldsymbol{\Theta},\boldsymbol{\theta}_0)\\ &\propto \prod_{k=1}^{N_s}\left(\prod_{l=1}^{a_k-1}\boldsymbol{\theta}_0(b_{kl})\prod_{j=1}^{W}\boldsymbol{\theta}_j(b_{k,a_k+j-1})\prod_{l=a_k+W}^{L_k}\boldsymbol{\theta}_0(b_{kl})\right).\end{aligned} \tag{4.1}$$

b_{kl} is the nucleotide found at position l in sequence S_k in the data set. As already shown in Equation 3.5, the likelihood of the sequence consists of two parts corresponding to the background and motif region within the sequence. We use the term alignment vector to refer to the set of starting positions $\mathcal{A}$.

4.4.2 Notations

To facilitate the description of the algorithm we introduce a number of alternative notations. Let us first denote the nucleotides `A`, `C`, `G` and `T` themselves as binary vectors

$$\begin{aligned}\mathtt{A} &= [1000]^T & \mathtt{C} &= [0100]^T\\ \mathtt{G} &= [0010]^T & \mathtt{T} &= [0001]^T.\end{aligned}$$

This representation is a sparse encoding scheme and assures that the Hamming distance between the nucleotides is equal. This encoding has also been used in applications where neural networks have been trained to detect features in DNA sequences [107, 109, 120].

Further, we need some definitions to represent vector manipulations. Given two vectors $\mathbf{v} = [v_1, v_2, v_3, v_4]^T$ and $\mathbf{w} = [w_1, w_2, w_3, w_4]^T$, we define that

$$\begin{aligned}|\mathbf{v}| &= |v_1| + |v_2| + |v_3| + |v_4|\\ \mathbf{v}+\mathbf{w} &= [v_1+w_1, v_2+w_2, v_3+w_3, v_4+w_4]^T\\ \mathbf{v}^{\mathbf{w}} &= v_1^{w_1}v_2^{w_2}v_3^{w_3}v_4^{w_4}\\ \mathbf{v}/\mathbf{w} &= [v_1/w_1, v_2/w_2, v_3/w_3, v_4/w_4]^T.\end{aligned}$$

Given this vector representation of the nucleotides we can rewrite the term $\boldsymbol{\theta}_j(b_l)$ as $\boldsymbol{\theta}_j^{b_l}$.

Next, we define a counting function $\mathbf{h}(.)$. This function will sum over all elements in a set and return a vector $[c_{\mathtt{A}}, c_{\mathtt{C}}, c_{\mathtt{G}}, c_{\mathtt{T}}]^T$ with $c_{\mathtt{A}}$ the total number of occurrences of `A` in the set. As an example of this counting function, $\mathbf{h}(\mathcal{S})$ gives the total number of `A`, `C`, `G` and `T` in the sequence set $\mathcal{S}$. Combining the vector notation and counting functions leads to the following expression for the likelihood of the sequence being generated by the background model $\boldsymbol{\theta}_0 = [q_{\mathtt{A},0}, q_{\mathtt{C},0}, q_{\mathtt{G},0}, q_{\mathtt{T},0}]^T$

$$\pi(\mathcal{S}|\boldsymbol{\theta}_0) = \prod_{k=1}^{N_s}\prod_{l=1}^{L_k}\boldsymbol{\theta}_0^{b_l} = \prod_{k=1}^{N_s}\boldsymbol{\theta}_0^{\mathbf{h}(S_k)} = \boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S})}.$$

To further extend this notation, let us indicate the set of the nucleotides at position j in the motif instances as $\mathcal{S}_{\{\mathcal{A}(j)\}}$. The set of nucleotides not belonging to a motif instance is defined as $\mathcal{S}_{\{\bar{\mathcal{A}}\}}$. Applying the counting function $\mathbf{h}$ over these sets, we can reformulate the complete data likelihood as defined in Equation 4.1.

$$\begin{aligned}\pi(\mathcal{S}, \mathcal{A}|\mathbf{\Theta}, \boldsymbol{\theta}_0) &\propto \prod_{k=1}^{N_s}\left(\prod_{l=1}^{a_k-1} \boldsymbol{\theta}_0(b_l) \prod_{j=1}^{W} \boldsymbol{\theta}_j(b_{a_k+j-1}) \prod_{l=a_k+W}^{L_k} \boldsymbol{\theta}_0(b_l)\right) \\ &= \boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}})} \prod_{j=1}^{W} \boldsymbol{\theta}_j^{\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{\theta}_j$ is the vector representing column j in the motif model $\mathbf{\Theta}$. Since $\mathbf{h}(\mathcal{S}) = \mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}}) + \sum_{j=1}^{W} \mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})$, another interesting representation of the complete data likelihood that can be derived is that

$$\pi(\mathcal{S}, \mathcal{A}|\mathbf{\Theta}, \boldsymbol{\theta}_0) \propto \boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S})} \prod_{j=1}^{W} \frac{\boldsymbol{\theta}_j^{\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})}}{\boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})}}.$$

4.4.3 Deploying a collapsed Gibbs sampling scheme

So far, we have only given the complete data likelihood, let us now explain how we can use it to implement a Gibbs sampler. The idea is to evolve a Markov chain that has the joint distribution $\pi(\mathcal{A}, \mathbf{\Theta}|\mathcal{S})$ as its equilibrium state. Liu et al. [63] have shown how a collapsed Gibbs sampling scheme that focuses solely on the alignment vector $\mathcal{A}$ can be derived to solve this problem. The idea is to collapse the motif parameters $\mathbf{\Theta}$ from the joint distribution to retrieve a simpler distribution to sample from.

As stated above, the goal of the Gibbs sampler in the missing data problem is to be able to sample from the joint distribution of missing values and parameters,

$$\pi(\mathbf{\Theta}, \boldsymbol{\theta}_0, \mathcal{A}|S) = \pi(\mathbf{\Theta}, \boldsymbol{\theta}_0|\mathcal{A}, \mathcal{S})\pi(A|\mathcal{S}). \tag{4.3}$$

To solve this problem, we should implement the following data augmentation scheme:

1. Given the current parameter values of $\mathbf{\Theta}$ and $\boldsymbol{\theta}_0$, impute the alignment vector $\mathcal{A}$.
2. Given the current imputation of the aligned instances, sample the parameter values of $\mathbf{\Theta}$ and $\boldsymbol{\theta}_0$.

Using this scheme directly seems not feasible. However, Liu et al. have shown that it is possible to implement a collapsed Gibbs Sampler that has the distribution $\pi(\mathcal{A}|\mathcal{S})$ as it is equilibrium state, where it is not necessary to sample from the complex distribution $\pi(\mathbf{\Theta}, \boldsymbol{\theta}_0|\mathcal{A}, \mathcal{S})$. The scheme of a systematic scan Gibbs sampler for this problem can be implemented as follows:

1. Sample $a_1^{(t+1)}$ from $\pi(a_1|a_2^{(t)},\ldots,a_N^{(t)},\mathcal{S})$.
2. Sample $a_2^{(t+1)}$ from $\pi(a_2|a_1^{(t+1)},a_3^{(t)},\ldots,a_N^{(t)},\mathcal{S})$.
3. $\cdots$
4. Sample $a_N^{(t+1)}$ from $\pi(a_N|a_1^{(t+1)},\ldots,a_{N-1}^{(t+1)},\mathcal{S})$.

If we introduce $\mathcal{A}_{\bar{k}}$ as the set of alignment positions excluding a_k we can rewrite the basic conditional sampling step as

$$\text{Sample} \quad a_k^{(t+1)} \quad \text{from} \quad \pi(a_k|\mathcal{A}_{\bar{k}},\mathcal{S}). \tag{4.4}$$

The next step is to find a way to represent this distribution and sample from it.

4.4.4 Predictive update formula

Knowing the relation between posterior and joint distribution

$$\pi(\mathcal{A}|\mathcal{S}) \propto \pi(\mathcal{A},\mathcal{S}), \tag{4.5}$$

the predictive distribution for $\mathcal{A}$ is derived to implement the conditional sampling steps.

The usage of a Dirichlet prior $f(\boldsymbol{\theta}_0)$ for the background model $\boldsymbol{\theta}_0$ and a product Dirichlet distributions prior $g(\boldsymbol{\Theta})$ for the model $\boldsymbol{\Theta}$ make it possible that the parameters $\boldsymbol{\theta}_0$ and $\boldsymbol{\Theta}$ can be integrated out analytically from the joint distribution. The joint distribution is written as

$$\pi(\mathcal{A},\mathcal{S}) = \int\int \pi(\mathcal{A},\mathcal{S}|\boldsymbol{\theta}_0,\boldsymbol{\Theta})f(\boldsymbol{\theta}_0)g(\boldsymbol{\Theta})d\boldsymbol{\theta}_0 d\boldsymbol{\Theta}. \tag{4.6}$$

Substituting equation 4.2 in the integral of equation 4.6 gives rise to following equation

$$\begin{aligned}
\pi(\mathcal{A},\mathcal{S}) &\propto \int\int \boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}})} \prod_{j=1}^{W} \boldsymbol{\theta}_j^{\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})} f(\boldsymbol{\theta}_0)g(\boldsymbol{\Theta})d\boldsymbol{\theta}_0 d\boldsymbol{\Theta} \\
&= \int \prod_{j=1}^{W} \boldsymbol{\theta}_j^{\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})} g(\boldsymbol{\Theta})d\boldsymbol{\Theta} \int \boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}})} f(\boldsymbol{\theta}_0)d\boldsymbol{\theta}_0. \\
&= \prod_{j=1}^{W} \left(\int \boldsymbol{\theta}_j^{\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}})} g(\boldsymbol{\theta}_j)d\boldsymbol{\theta}_j \right) \int \boldsymbol{\theta}_0^{\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}})} f(\boldsymbol{\theta}_0)d\boldsymbol{\theta}_0.
\end{aligned}$$

$\boldsymbol{\theta}_j$ refers to the vector representing the jth column in the motif model and $g(\boldsymbol{\theta}_j)$ is the Dirichlet prior for $\boldsymbol{\theta}_j$. Given the definition of the Dirichlet priors

$$\begin{aligned}
Dir(\boldsymbol{\theta}_0|\alpha) &= \frac{\Gamma(\sum \alpha_b)}{\prod(\Gamma(\alpha_b))} \prod_b \boldsymbol{\theta}_0(b)^{\alpha_b - 1} \\
Dir(\boldsymbol{\theta}_j|\beta_j) &= \frac{\Gamma(\sum \beta_b)}{\prod(\Gamma(\beta_b))} \prod_b \boldsymbol{\theta}_j(b)^{\beta_b - 1}
\end{aligned}$$

we derive the following predictive distribution for $\mathcal{A}$:

$$\pi(\mathcal{A}|\mathcal{S}) \propto \Gamma(\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}}) + \alpha) \prod_{j=1}^{W} \Gamma(\mathbf{h}(\mathcal{S}_{\{\mathcal{A}(j)\}}) + \beta_j). \tag{4.7}$$

Given $\mathcal{A}_{\bar{k}}$ as the set of starting positions excluding a_k and Equation 4.7, the following expression for the predictive distribution of a_k is obtained:

$$\pi(a_k|\mathcal{A}_{\bar{k}}, \mathcal{S}) = \frac{\pi(a_k, \mathcal{A}_{\bar{k}}|\mathcal{S})}{\pi(\mathcal{A}_{\bar{k}}|\mathcal{S})} = \frac{\pi(\mathcal{A}|\mathcal{S})}{\pi(\mathcal{A}_{\bar{k}}|\mathcal{S})}. \tag{4.8}$$

It is possible to use this formula directly, but Liu et al. [63] proposed an approximation which leads to a more intuitive formula. If the set R_1 is much larger than the set R_2 and the composition of R_2 is relatively diverse then

$$\frac{\Gamma(\mathbf{h}(R_1 \oplus R_2))}{\Gamma(\mathbf{h}(R_1))} \approx \mathbf{h}(R_1)^{\mathbf{h}(R_2)}.$$

This approximation is surely true in the case of a motif instance of length W in comparison with the set of all non-site nucleotides. Therefore the first factor in Equation 4.8 can be approximated as

$$\frac{\Gamma(\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}_{\bar{k}}\}}) + \alpha)}{\Gamma(\mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}_{\bar{k}}\}}) + \alpha - \mathbf{h}(\mathcal{S}_{\{a_k\}}))} \approx \mathbf{h}(\mathcal{S}_{\{\bar{\mathcal{A}}\}})^{\mathbf{h}(\mathcal{S}_{\{a_k\}})}.$$

This approximation leads to the following expression for the predictive update

$$\pi(a_k = i|\mathcal{A}_{\bar{k}}, \mathcal{S}) \propto \prod_{j=1}^{W} \left(\frac{\hat{\theta j}}{\hat{\boldsymbol{\theta}}_0}\right)^{b_{i+j-1}} \tag{4.9}$$

In words, this equation tells that the probability of the segment at position i in sequence S_k being a motif instance is proportional to the likelihood ratio of the segment being generated by the motif model $\hat{\boldsymbol{\Theta}}$ and the site being generated by the background model $\hat{\boldsymbol{\theta}}_0$. This formula forms the basis of the Gibbs sampling algorithm for motif finding.

4.4.5 Algorithm

Given the predictive update formula of the Gibbs sampler we can construct the basic algorithm for finding a motif which has exactly one instance in each sequence in the data set.

Program 1: Basic Gibbs sampling algorithm for motif finding

1. Initialize the alignment vector $\mathcal{A}$ by selecting at random a starting position in each sequence.
2. For each sequence $S_z, z = 1 \ldots, N_s$
    (a) Create subsets $\tilde{\mathcal{S}} = \{S_i | i \neq z\}$ and $\tilde{\mathcal{A}} = \{a_i | i \neq z\}$ from the original data sets by excluding the selected sequence S_z.
    (b) Compute the motif model $\tilde{\mathbf{\Theta}}$ from the segments indicated by $\tilde{\mathcal{A}}$ in the set $\tilde{\mathcal{S}}$ and the background model $\tilde{\boldsymbol{\theta}}_0$ from all non-site positions in $\tilde{\mathcal{S}}$.
    (c) Assign to each possible motif instance $\mathbf{x}_{zl}$ in sequence S_z, with $l = 1 \ldots L_z - W + 1$, a score $W(\mathbf{x}_{zl})$ given by the predictive update formula of Equation 4.9,

    $$W(\mathbf{x}_{zl}) = \prod_{j=1}^{W} \frac{\tilde{\boldsymbol{\theta}}_j^{(b_{z,l+j-1})}}{\tilde{\boldsymbol{\theta}}_0^{(b_{z,l+j-1})}}$$

    (d) Draw the new alignment position a_z according to the normalized probability distribution $\frac{W(\mathbf{x}_{zl})}{\sum_l^{L_z - W + 1} W(\mathbf{x}_{zl})}$
3. Repeat from Step 3 until the Markov chain converges.

The algorithm as it is conceived here needs only one parameter to be defined by the user namely the motif length W.

4.4.6 Main problems

Although the algorithm can work as it is shown here there are some shortcomings that we would like to address in the following chapters.

Exactly one instance of a motif in a sequence might be a reasonable model assumption when aligning protein sequences but is not exactly true for DNA sequences. When considering higher eukaryotes a more reasonable model would allow for multiple instances of the same motif model within one sequence. Moreover, if the sequence sets are generated from clustered expression data, it is very likely that there will be sequences in this data set that do not carry any motif instance. Microarray data are noisy in nature, the resulting clusters are thus also noisy. Therefore, it is necessary to design a motif finding algorithm that is robust to this noise.

In the basic algorithm of [54], which was designed for protein sequences, the parameters of the background distribution $\boldsymbol{\theta}_0$ were recalculated in each iteration. However, in the case of DNA sequences we can assume that the background frequency based on the non-site positions does not differ significantly from the background frequencies calculated from the full data set. Therefore it is possible to compute the parameters $\boldsymbol{\theta}_0$ at the start of the algorithm and keep them fixed during the execution of the algorithm. Another problem is that the background model only uses the single nucleotide frequency, which might be not sufficient enough to capture the specific content of the upstream region.

Although it is proposed as a solution to avoid local optima, the stochasticity of the sampling procedure has it own drawbacks. When started from the same initial alignment vector, the Gibbs sampler can converge to different solutions. It is thus important to find a good way to handle this stochasticity and the filter the true motifs from the local optima. Related to this problem is the definition of convergence of the Gibbs sampler. Since the sampler does not converge to a point estimate as was the case in EM, it is harder to define a closed convergence criterion.

4.5 Conclusions

In this chapter we have given an outline of the fundamentals of sampling methods: Metropolis algorithm and Gibbs sampling. The principles of the missing data problem were restated and the link was made between EM and Gibbs sampling. Finally, we have shown how the original Gibbs sampling algorithm for motif finding was constructed. In the next chapter we will further build on the foundations outlined in this chapter to construct our adapted version of the Gibbs sampler. Our version will be particularly tailored to find motifs in sets of coexpressed genes.

CHAPTER 5

MotifSampler: implementation and performance analysis[1]

After the introduction of the basic theory of sampling and MCMC methods in the previous chapter, we move to the main topic of this dissertation which is the implementation of our Gibbs sampling algorithm for motif finding, ***MotifSampler****. This description is split in three parts: (1) theoretical aspects; (2) evaluation of the algorithm; and (3) practical implementation aspects. In the first section we introduce higher-order background models within the Gibbs sampler framework. In the second section we discuss the methodology to estimate the number of motif instances within the probabilistic framework of the Gibbs sampling. These two adjustments are combined to create the core sampling procedure of the Gibbs sampler. The result of this sampling procedure is a position-specific frequency matrix. The different types of scores that are related to the motif model are discussed. To gain insight in the sampling procedure, we conduct several tests in which we quantify the sensitivity of the algorithm when the parameters are changed. Finally, the implications of the performance analysis lead us to the practical implementation of the MotifSampler. Building on the expertise gained from the multitude of conducted tests, we are able to propose an elaborated strategy for motif finding.*

---

[1] The first results of this work have been published in the proceedings of RECOMB'01 [105]. Later, an extended version of the conference paper was published in Journal of Computational Biology [106]. A more detailed study of the creation and effect of higher-order background models has been described in Bioinformatics [104].

5.1 Higher-order background models

One of the first shortcomings pointed out to us when we started working on motif detection was that the first generation of motif detection algorithms like MEME [8], Gibbs Sampler [54] and AlignACE [88], all use a simple single nucleotide frequency model to compute the background probabilities. However, two examples indicate that this might not be the best approach. First, if we look at most of the state-of-the-art gene prediction models we noticed that these methods use higher-order Hidden Markov Models (HMM) to model coding and non-coding regions in DNA. In this respect, we could refer here to *Glimmer* [27], *HMMGene* Krogh [52] and *GeneMark.hmm* [67]. Markov models have also been introduced to predict promoter regions in higher eukaryotes [6]. Recently this method was refined to use interpolated Markov chains to increase the sensitivity and specificity of the promoter prediction [79]. Second, oligonucleotide analysis has shown that estimating the expected frequency of a given oligomer from a single nucleotide model gives bad estimates [112, 96]. Given these two examples, it seems reasonable to chose also an higher-order background model within the Gibbs sampling framework for motif detection. Similar approaches have also been proposed in the last years by Workman and Stormo [119] and Liu et al. [64]. In the latest version of MEME there is now also the possibility to use higher-order background models. Another related approach, is the use a position-specific background model estimated with a Bayesian segmentation model [62]. This model accounts for the varying composition of the DNA upstream of a gene [73].

5.1.1 Construction of the transition matrix

When referring to high-order background models, we start from the basic assumption that a DNA sequence can be generated with a Markov model of order m. This means that the probability of observing a certain nucleotide in a sequence depends on the m previous nucleotides in the sequence. The likelihood of a sequence being generated with an higher-order background model of order m can then be written as

$$P(S|\mathcal{B}_m) = p(b_1, b_2, \ldots, b_m) \prod_{l=m+1}^{L} p(b_l|b_{l-1}, \ldots, b_{l-m}), \tag{5.1}$$

where $\mathcal{B}_m$ represents the parameters of the higher-order background model. These parameters are $p(b_1, b_2, \ldots, b_m)$, the probability of finding a specific m-mer, and $p(b_l|b_{l-1}, \ldots, b_{l-m})$, the probability of finding the base b_l given the m previous bases in the sequence. The latter is stored in the transition matrix of the Markov model.

To construct a transition matrix for a background model of order m, we count all oligonucleotides of length $m+1$ in the reference data set. We rearrange the counts in a matrix of dimension $m \times 4$, such that each row has the same first m bases while each column corresponds to the last base in the oligonucleotides.

Next, a pseudocount is added and each row is normalized to one so that they represent probabilities.

Table 5.1: Second-order background model from *A.thaliana* intergenic regions.

	$P(bb)^a$	$P(b\|bb)^b$			
		`A`	`C`	`G`	`T`
`AA`	0.135	0.4272	0.1513	0.1459	0.2756
`AC`	0.052	0.3904	0.1776	0.1329	0.2991
`AG`	0.051	0.4029	0.1450	0.1461	0.3060
`AT`	0.107	0.3185	0.1621	0.1782	0.3412
`CA`	0.060	0.3963	0.1729	0.1144	0.3165
`CC`	0.027	0.4229	0.1673	0.1303	0.2795
`CG`	0.020	0.3592	0.1319	0.1638	0.3451
`CT`	0.052	0.2750	0.2072	0.1288	0.3890
`GA`	0.057	0.3679	0.1431	0.1886	0.3004
`GC`	0.022	0.3650	0.1838	0.1171	0.3341
`GG`	0.026	0.3490	0.1468	0.1660	0.3382
`GT`	0.052	0.2657	0.1634	0.1899	0.3810
`TA`	0.094	0.3498	0.1436	0.1488	0.3578
`TC`	0.057	0.3449	0.1601	0.1234	0.3716
`TG`	0.058	0.3355	0.1325	0.1869	0.3451
`TT`	0.132	0.2451	0.1586	0.1748	0.4215
	SNF :	0.3449	0.1581	0.1556	0.3414

[a] probability of finding dimer in the intergenic sequences.
[b] representation of second-order transition matrix.

Table 5.1 gives an example of a transition matrix constructed from *A.thaliana* intergenic regions. Each entry in the matrix represents the probability of finding the respective nucleotide given the two preceding nucleotides in the sequence. For comparison, we also include the single nucleotide frequency (SNF) in this table. Some rows, like `GG`, are rather similar to the SNF while others are significantly different. This has a profound impact on the computed probabilities especially if the sequences become longer.

As an example, we look at the probability of the sequences `AAAAAAA` and `CGCGCGC` being generated by this second-order background model using Equation 5.1.

$$\begin{aligned} P(\mathtt{AAAAAAA}|\mathcal{B}_m) &= P(\mathtt{AA})P(\mathtt{A}|\mathtt{AA})P(\mathtt{A}|\mathtt{AA})P(\mathtt{A}|\mathtt{AA})P(\mathtt{A}|\mathtt{AA})P(\mathtt{A}|\mathtt{AA}) \\ &= 0.135 \times 0.427 \times 0.427 \times 0.427 \times 0.427 \times 0.427 \\ &= 1.92e-3 \qquad (\text{SNF} = 5.81e-4) \end{aligned}$$

$$\begin{aligned} P(\mathtt{CGCGCGC}|\mathcal{B}_m) &= P(\mathtt{CG})P(\mathtt{C}|\mathtt{CG})P(\mathtt{G}|\mathtt{GC})P(\mathtt{C}|\mathtt{CG})P(\mathtt{G}|\mathtt{GC})P(\mathtt{C}|\mathtt{CG}) \\ &= 0.020 \times 0.132 \times 0.117 \times 0.132 \times 0.117 \times 0.132 \\ &= 2.30e-6 \qquad (\text{SNF} = 2.41e-6) \end{aligned}$$

At first, this example illustrates that there are differences between the scores from the higher-order background model and the single nucleotide frequency model. For these short examples the effect is rather small, but it will become much larger if the sequences become larger. Furthermore, if we compare both sequences, it is about 300 times more likely to find the sequence `AAAAAAA` than the sequence `CGCGCGC`.

To reliably construct this matrix we need a sufficient amount of sequence data. The number of values in the transition matrix is equal to $4^m \times 4$. In a sequence set of N_s sequences of L basepairs, there are $N_s \times (L - m)$ oligonucleotides of length $m + 1$. To fill in the matrix with reliable values, the number of oligonucleotides should be sufficiently greater than $4^m \times 4$.

5.1.2 Intergenic sequences

An important aspect of the construction of a background model is the selection of the right sequence data. It is known that the nucleotide composition changes over the different regions in the DNA sequence. For instance, the composition of a coding sequence significantly differs from the non-coding regions which is exploited by all gene recognition methods. Building and using the right background model has a major impact on the performance of the motif finding procedure. For instance, in Section 6.3 we illustrate how in prokaryotes the choice of a background model influences the motif detection process, especially when the wrong background model is used. In Section 7.4 we illustrate how in human the sequence composition changes over the upstream region.

What region we should select, is actually defined by the location of the transcription factor binding sites that are responsible for the particular gene expression. Since TFBSs are mostly located in the non-coding region upstream of a gene, that region is the most interesting to be used. The intergenic region is the non-coding region between two genes. Figure 5.1 shows the three possible configurations of the intergenic regions in a genome. The configuration is defined by the direction in which the gene is transcribed. The first configuration consists of two genes coding in tandem on the same strand. In this case the intergenic region is expected to contain only one core promoter. The second configuration is the case where both genes are pointing away from each other. In their intergenic region divergent promoters are expected to control transcription. In the last case the transcription of the two genes is convergent. No promoter regulatory element is expected to occur in the intergenic region. To build a background model we use only those regions in which a core promoter is present.

Although the selection of the intergenic region seems reasonable, we should be cautious when delineating the subregion to be used in motif finding. First of all, there are significant differences between the intergenic regions in different organisms, not only in nucleotide composition but also in length. The size of the intergenic regions depends on the compactness of the genome. For in-

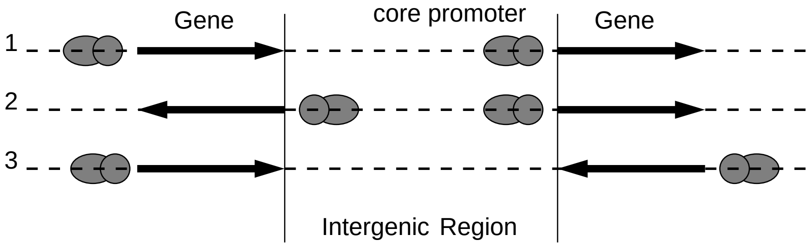


Figure 5.1: The three possible configurations of the intergenic region between to consecutive genes in the genome. The thick arrow displays the direction in which the gene is transcribed. The core promoter is located just upstream of the start site of the gene. (1) When the two consecutive genes code in tandem on the same strand, only one core promoter is found in the intergenic region. (2) The two consecutive genes are transcribed away from each other which means that the intergenic region contains two core promoters. (3) When the two consecutive genes are transcribed towards each other, no promoter is found in the intergenic region.

stance, the intergenic regions in *A.thaliana* are much shorter than the regions from human or *D.melanogaster*. Even within one species, the length of different intergenic regions can vary significantly. Pavy et al. [82] have shown that within *A.thaliana* the intergenic regions vary by at least two orders of magnitude from less than 100bp to regions over 10kb. If we look at the human genome, there the intergenic regions can be much longer than 100kb. Using such a long region within a motif finding algorithm will not give reliable results, since there will be too much noise present in the data set. In such a case one needs to define more accurately the region in which the potential binding sites are located.

5.1.3 Extended sequence model and predictive update formula

Having introduced the higher-order Markov model, let us look back at the sequence model introduced in Chapter 3.

$$P(S_k|a_k, \boldsymbol{\Theta}, \mathcal{B}_m) = \underbrace{\prod_{l=1}^{a_k-1} \mathcal{B}_m(b_{kl}, S_k)}_{\text{background}} \underbrace{\prod_{j=1}^{W} \boldsymbol{\theta}_j(b_{k,a_k+j-1})}_{\text{motif}} \underbrace{\prod_{l=a_k+W}^{L_k} \mathcal{B}_m(b_{kl}, S_k)}_{\text{background}} . \tag{5.2}$$

$\mathcal{B}_m(b_l, S_k)$ is the probability of finding the nucleotide b_l in the context of the sequence S_k according to the background model $\mathcal{B}_m$ and is found as the corresponding entry in the transition matrix.

Given the extended sequence model we can now recalculate the predictive update formula of the Gibbs sampler as defined in Chapter 3. Substituting Eq. 5.2 in Eq. 4.6 and leaving out the parameters of the background model, since they are fixed now, results in the following adjusted predictive update

$$\pi(a_k = i|\mathcal{A}_{\bar{k}}, \mathcal{S}) \propto \prod_{j=1}^{W} \frac{\boldsymbol{\theta}_j(b_{i+j-1})}{P(b_{i+j-1}|\mathcal{S}, \mathcal{B}_m)}. \tag{5.3}$$

This equation is the likelihood ratio of the probability that the segment starting at position i is generated by the motif model $\hat{\boldsymbol{\Theta}}$ and the probability that the segment is generated by the background model $\mathcal{B}_m$. If we define $\mathbf{x}_{kl} = [b_l, b_{l+1}, \ldots, b_{l+W-1}]$ as the segment of length W starting at position l in sequence S_k, we can then also define the *segment score* as

$$W(\mathbf{x}_{kl}) = \frac{P(\mathbf{x}_{kl}|\boldsymbol{\Theta})}{P(\mathbf{x}_{kl}|S_k, \mathcal{B}_m)} = \prod_{j=1}^{W} \frac{\boldsymbol{\theta}_j(b_{l+j-1})}{P(b_{l+j-1}|\mathcal{S}, \mathcal{B}_m)}, \tag{5.4}$$

where $P(\mathbf{x}_{kl}|\boldsymbol{\Theta})$ is the probability that the segment $\mathbf{x}_{kl}$ is generated by the motif model $\boldsymbol{\Theta}$ and $P(\mathbf{x}_{kl}|S_k, \mathcal{B}_m)$ is the probability that the segment $\mathbf{x}_{kj}$ is generated by the background model $\mathcal{B}_m$. Using this segment score $W(\mathbf{x})$ we can rewrite equation 5.2 as

$$\begin{aligned} P(S_k|a_k, \boldsymbol{\Theta}, \mathcal{B}_m) &\propto \prod_{l=1}^{L} P(b_l|\mathcal{S}, \mathcal{B}_m) \prod_{j=1}^{W} \frac{\boldsymbol{\theta}_j(b_{k,a_k+j-1})}{P(b_{a_k+j-1}|\mathcal{S}, \mathcal{B}_m)} \\ &= P(S_k|\mathcal{B}_m) W(\mathbf{x}_{k,a_k}). \end{aligned} \tag{5.5}$$

In a practical implementation we do not use this equation directly. For numerical reasons, it is better to take the logarithm of this equation:

$$\log(P(S_k|a_k, \boldsymbol{\Theta}, \mathcal{B}_m)) \propto \log(P(S_k|\mathcal{B}_m)) + \log(W(\mathbf{x}_{k,a_k})). \tag{5.6}$$

This formula is easily adapted if there are multiple instances of the same motif in a sequence. We only need to add more terms of the form $\log(W(\mathbf{x}_{k,a}))$ to Equation 5.6. The fact that we can write full sequence likelihood in this compact form allows to efficiently implement the computation steps of our algorithm.

5.2 Estimating the number of copies

In the previous chapter we already stated that the model, where we search for exactly one motif in a sequence, is not fully compliant with real biological examples. When searching for motifs in sequence sets generated from expression data, it is important to have an algorithm that can handle the noise present in such a data set. One aspect of the noise is that there might be sequences in the data set that do not contain the common motif and as such they contaminate the data set. Second, in eukaryotes there might be several binding sites of the same transcription factor present in the upstream region of the gene. This has implications on the motif finding algorithm. In this section we present a method to estimate the number of copies of a motif in a sequence which fits nicely within the probabilistic sequence framework.

To solve this problem we introduce a new variable Q_k, that is the number of instances of the motif $\mathbf{\Theta}$ in the sequence S_k. It is clear that Q_k is missing from our observations so we need to estimate it. In our approach, we propose to compute the expected number of instances given the sequence, motif model and background model as

$$E_{(S_k,\mathbf{\Theta},\mathcal{B}_m)}(Q_k) = \sum_{c=0}^{\infty} c \times P(Q_k = c|S_k, \mathbf{\Theta}, \mathcal{B}_m). \tag{5.7}$$

To compute this expectancy we need to compute $P(Q_k = c|S_k, \mathbf{\Theta}, \mathcal{B}_m)$, the probability of finding c instances of the motif $\mathbf{\Theta}$ in the sequence S_k. These probabilities are the parameters of a discrete probability distribution Γ_k. If we could find the parameters of this distribution, we would be able to find the number of motif instances in the sequence. Applying Bayes' theorem leads to the following expansion

$$\begin{aligned} \Gamma_k(c) &= P(Q_k = c|S_k, \mathbf{\Theta}, \mathcal{B}_m) \\ &= \frac{P(S_k|Q_k = c, \mathbf{\Theta}, \mathcal{B}_m)P(Q_k = c|\mathbf{\Theta}, \mathcal{B}_m)}{P(S_k|\mathbf{\Theta}, \mathcal{B}_m)}. \end{aligned} \tag{5.8}$$

In this equation we distinguish three different parts. Let us look at each part separately.

The denominator is the probability $P(S_k|\mathbf{\Theta}, B_m)$ which serves as a normalizing factor. This probability can be calculated by taking the sum over all possible numbers of copies

$$P(S_k|\mathbf{\Theta}, \mathcal{B}_m) = \sum_{c=0}^{\infty} P(S_k|Q_k = c, \mathbf{\Theta}, \mathcal{B}_m)P(Q_k = c|\mathbf{\Theta}, \mathcal{B}_m).$$

The first factor in the numerator, $P(S_k|Q_k = c, \mathbf{\Theta}, B_m)$, is the probability of the sequence given the motif model $\mathbf{\Theta}$, the background model B_m, and the number of instances c. This probability can be calculated by summing over all possible combinations of c motifs,

$$\begin{aligned} & P(S_k|Q_k = c, \mathbf{\Theta}, \mathcal{B}_m) = \\ & \qquad \sum_{a_1} \cdots \sum_{a_c} P(S_k|\mathcal{A}_k^c, Q_k = c, \mathbf{\Theta}, \mathcal{B}_m)P(\mathcal{A}_k^c|Q_k = c, \mathbf{\Theta}, \mathcal{B}_m), \end{aligned} \tag{5.9}$$

where $\mathcal{A}_k^c = \{a_{ki}|i = 1, \ldots, c\}$ is the set of the start positions of the c different motif instances. The probability $P(S_k|\mathcal{A}_k^c, Q_k = c, \mathbf{\Theta}, \mathcal{B}_m)$ can be easily calculated with Equation 5.5 as is shown in this example with $c = 2$,

$$\begin{aligned} P(S_k|Q_k = 2, \mathbf{\Theta}, \mathcal{B}_m) &= \sum_{a_{k1}=1}^{L-2W+1} \sum_{a_{k2}=W+a_{k1}}^{L-W+1} \left[P(S_k|\mathcal{B}_m)W(\mathbf{x}_{a_{k1}})W(\mathbf{x}_{a_{k2}})\right] \\ &= P(S_k|\mathcal{B}_m) \sum_{a_{k1}=1}^{L-2W+1} W(\mathbf{x}_{a_{k1}}) \sum_{a_{k2}=a_{k1}+W}^{L-W+1} W(\mathbf{x}_{a_{k2}}). \end{aligned}$$

Note that, although the complexity depends on the number of copies c (all possible combinations of c motifs in the sequence have to be taken into account), it is possible to efficiently calculate Eq. 5.9 in linear time using Eq. 5.6. The prior $P(\mathcal{A}_k^c|Q_k = c, \mathbf{\Theta}, \mathcal{B}_m)$ in Eq. 5.9 is assumed to be independent of the motif model and the background model. We assume that each position in the sequence is equally probable a priori to be the start site of the motif. The prior is therefore inversely proportional to the total number of possible combinations of c motifs of length W in a sequence of length L_k, which is $\frac{(L-nW+n)!}{(L-nW)!n!}$.

The second factor in the numerator in Equation 5.8 is the prior $P(Q = c|\mathbf{\Theta}, \mathcal{B}_m)$, the probability of finding c copies given the motif model and the background model. This prior probability of finding c motif instances depends on the relation between the motif model and the given background model. For instance, the a priori probability of finding a strong GC-rich motif in a AT-rich background is much lower than finding a weak AT-rich motif in the same background. Since the distribution is unknown we need to propose a reasonable model. To construct the prior probability distribution, we introduce here a new parameter γ_1. Initially, we create the prior distribution Γ_0 with parameters $[1-\gamma_1, \gamma_1]$. Next the distribution is further refined, intuitive it seems reasonable to assume that the prior probability of finding $c+1$ motif instances is always smaller than finding c instances. $\Gamma_0(c+1)$ is chosen equal to $\kappa\Gamma_0(c)$ with κ between 0 and 1 such that the added value in the prior is always smaller than last one. The parameters of the distribution can be written as

$$[\frac{1-\gamma_1}{C}, \frac{\gamma_1}{C}, \frac{\kappa\gamma_1}{C}, \frac{\kappa^2\gamma_1}{C}, \ldots, \frac{\kappa^k\gamma_1}{C}],$$

with C a normalizing constant. The rationale is that the chance of finding $c+1$ instance is always smaller than finding c instances, since if you find $c+1$ instances there are many combinations in which you will find c instances. In our implementation, we have chosen to set $\kappa = 0.25$. In Section 5.5 we will discuss in detail the influence of this parameter γ_1 on the performance of our algorithm.

From a more practical point of view, Equation 5.7 is not workable as such. Taking the sum over all possible number of instances from zero to infinity is impractical. Therefore we suggest two possible ways to solve this summation. In the first version of the algorithm, we introduced a parameter C_{max} to set the maximal number of instances to be expected in each sequence. The sum in Equation 5.7 is substituted with a sum going from 0 to C_{max} thereby assuming that $\Gamma_k(c)$ is equal to zero for all c greater then C_{max}. If we have computed $\Gamma_k(c)$, for $c = 1, \ldots, C_{\text{max}}$, this probability can be used to estimate the expected number of copies Q_k of the motif in the given sequence S_k. The second approach is to compute Equation 5.7 as long as $\Gamma_k(c)$ is larger than a predefined small value ϵ. If for a certain number of instances c', $\Gamma_k(c')$ becomes smaller than ϵ the contribution of $\Gamma_k(c)$ can be ignored by setting it to zero. The same holds for all number of copies greater than c'. In this text we focus on the last form, which is also implemented in the stand alone version of our program.

5.3 Core sampling procedure

Combining the extensions, described in the previous sections, results in a first implementation of an adapted Gibbs sampler. In Table 2 the sampling procedure is summarized. This procedure is very similar to the procedure introduced in Section 4.4.5 except for the extras introduced in this chapter. Since the background model is fixed, the score $P(\mathbf{x}_{kl}|S_k, \mathcal{B}_m)$ of the segments $\mathbf{x}_{kl}$ being generated by the background model $\mathcal{B}_m$ can be computed beforehand for $k = 1 \dots Ns$ and $l = 1 \dots L_k$.

Program 2: Core sampling procedure of the MotifSampler.

For each sequence $S_z, z = 1 \dots, N_s$

1. Create subsets $\tilde{\mathcal{S}} = \{S_i | i \neq z\}$ and $\tilde{\mathcal{A}} = \{a_i | i \neq z\}$ from the original data sets by excluding the selected sequence S_z.
2. Compute the motif model $\tilde{\mathbf{\Theta}}$ from the segments indicated by $\tilde{\mathcal{A}}$ in the set $\tilde{\mathcal{S}}$.
3. Assign to each segment $\mathbf{x}_{zl}$ in sequence S_z, with $l = 1 \dots L_z - W + 1$, a weight $W(\mathbf{x}_{zl})$ as the ratio of the probability that the corresponding segment is generated by the motif model $\tilde{\mathbf{\Theta}}$ and the probability that segment is generated by the background model $\mathcal{B}_m$:
$$W(\mathbf{x}_{zl}) = \frac{P(\mathbf{x}_{zl}|\tilde{\mathbf{\Theta}})}{P(\mathbf{x}_{zl}|S_z, \mathcal{B}_m)}.$$
4. Update Γ_z, for $c = 0, \dots, C_{\text{max}}$, where C_{max} is such that $\Gamma_z(C_{\text{max}})$ is smaller than ϵ.
5. Compute the expected number of motif instances as $E[Q_z] = \sum_{c=0}^{C_{\text{max}}} c\Gamma_z(c)$.
6. Draw a new alignment vector $\mathcal{A}_z^c = \{a_{zc} | c = 1 \dots Q_z\}$ from the normalized probability distribution $W(\mathbf{x}_{zl})$.

5.4 Resulting motif model

So far, we have mainly discussed the basics of the algorithm, let us now look at the output. The final result of the MotifSampler consists of three main components:

1. the position-specific probability matrix $\mathbf{\Theta}$
2. the alignment vector $\mathcal{A}$

3. the distributions Γ_k to estimate the number of motif instances.

Naturally, there exists a clear relation between these three components. From these three types of information, different scores with their own characteristics can be calculated: *consensus score*, *information content* and *log-likelihood*. In the remainder of the text we will focus on these scores if we discuss the performance of the algorithm.

5.4.1 Motif scores

The consensus score is a measure for the conservation of the motif. A perfectly conserved motif has a score equal to 2 while a motif with a uniform distribution has a score equal to 0.

$$\text{Consensus Score} = 2 - \frac{1}{W}\sum_{j=1}^{W}\sum_{b\in\{\text{A,C,G,T}\}}\boldsymbol{\theta}_j(b)\log_2(\boldsymbol{\theta}_j(b)) \tag{5.10}$$

The information content or Kullback-Leiber distance between the motif and the single nucleotide frequency takes into account how well the motif is conserved but also how much the motif differs from the single nucleotide distribution. This score is maximal if the motif is well conserved and differs considerably from the background distribution.

$$\text{Information Content} = \frac{1}{W}\sum_{j=1}^{W}\sum_{b\in\{\text{A,C,G,T}\}}\boldsymbol{\theta}_j(b)\log_2\left(\frac{\boldsymbol{\theta}_j(b)}{\boldsymbol{\theta}_0(b)}\right). \tag{5.11}$$

As a final score we consider the log-likelihood score which is derived from the complete data-likelihood, $\log P(S, \mathcal{A}|\boldsymbol{\Theta}, \mathcal{B}_m)$. The motif and alignment positions are the results of maximum likelihood estimation and therefore the log-likelihood is a good measure for the quality of the motif. In this particular case, we are especially interested in the positive contribution of the motif instances to the complete data-likelihood. The log-likelihood score is calculated as

$$\text{Log-Likelihood Score} = \sum_{k=1}^{N_s}\log(\Gamma_k(0)) + \sum_{c=1}^{C_{\text{max}}}\Big(\sum_{i=c}^{C_{\text{max}}}\Gamma_k(i)\Big)W(\mathbf{x}_{ka_c}) \tag{5.12}$$

where $\mathbf{x}_{ka_c}$ is the segment starting at the position indicated by the c-th element in the alignment vector A_k of sequence S_k. The log-likelihood score depends on the strength of the motif and also on the total number of instances of the motif. Each of these scores accounts for a specific aspect of the motif.

5.4.2 Comparison of different motif models

Assume that multiple motifs are retrieved and we would like to know which of these motifs are the same or we would like to know if the retrieved motifs are present in a list of known motifs. In both cases we need to compute some measure of similarity to compare two motifs. A natural measure of similarity between two probabilistic models is the **Kullback-Leiber** distance between the two models. However, the Kullback-Leiber distance is a non-symmetric measurement, which means that $d(\mathbf{\Theta}_1, \mathbf{\Theta}_2) \neq d(\mathbf{\Theta}_2, \mathbf{\Theta}_1)$. Hence, to make the distance measure symmetric we decide to take the average of $d(\mathbf{\Theta}_1, \mathbf{\Theta}_2)$ and $d(\mathbf{\Theta}_2, \mathbf{\Theta}_1)$. Given two motif models $\mathbf{\Theta}^{(1)}$ and $\mathbf{\Theta}^{(2)}$, this results in the following formula for the distance measure

$$d(\mathbf{\Theta}_1, \mathbf{\Theta}_2) = \frac{1}{2W} \sum_{j=1}^{W} \sum_{b=\mathtt{A}}^{\mathtt{T}} \mathbf{\Theta}_1(b,j) \log \frac{\mathbf{\Theta}_1(b,j)}{\mathbf{\Theta}_2(b,j)} + \mathbf{\Theta}_2(b,j) \log \frac{\mathbf{\Theta}_2(b,j)}{\mathbf{\Theta}_1(b,j)}. \quad (5.13)$$

This score will be 0 if two motifs are perfectly equal. We will refer to $\mathbf{\Theta}^{(1)}$ as the query motif and $\mathbf{\Theta}_2$ the subject. To compensate for the difference in length between query and subject and also for a possible shift, we slide the query motif along the subject and compute the mutual information of the overlapping part. The minimum of these scores is compared to a given threshold (e.g. 0.65). If the score is below the threshold the query motif and subject motif are classified as being similar.

5.5 Discussion of the algorithm performance

In the previous sections we have given a description of our algorithm. In this section we will explain how the MotifSampler depends on the different parameters. We look at execution time, convergence criteria and the influence of the stochasticity of the Gibbs sampler on the retrieved motif models. Next, we discuss the effect of changing the parameters on the scores of the motifs. From the study of the influence of parameters we build up the necessary experience to apply our algorithm, dependent on the type of data, in conditions that are as optimal as possible. At the end of this section we propose a strategy to deploy when using the MotifSampler.

5.5.1 Data sets for performance testing

For this a first evaluation of our algorithm we use three data sets selected in yeast and plants. These data sets contain all well studied transcription factor binding sites and are well suited to test the algorithm since we already know what type of motif we are looking for.

G-box binding factor in Plants

As a first test case, we use the GBF data set extracted from PlantCARE [57]. This data set was already defined in Chapter 3 where we used the G-box to illustrate the different motif representation. This set contains 33 sequences containing 500bp upstream of the start of the gene. In these sequences, one or more instances of the GBF are experimentally annotated. The consensus sequence of the G-box binding site is `CACGTG`. The annotated binding sites in this data set can be found Table 3.1.

Yeast regulons

In the following evaluation of the algorithm we also use two sets of promoter sequences in yeast. These sets were introduced the first time by van Helden et al. [111] to their oligo analysis algorithm. The first set, MET, contains eleven promoter sequences of genes repressed by methionine. This repression is regulated through two different factors, a complex Cbfl-Met4p-Met28p and either Met31p or Met32p. The Cbfl-Met4p-Met28p complex recognizes sites similar to `TCACGTG` while Met31p or Met32p recognize `AAAACTGTGG`. In Figure 3.4 we have already shown the position of the transcription factor binding sites. The second set, NIT, contains the promoter sequences of seven genes that are expressed when nitrogen sources like glutamine, glutamate and ammonia are present in the media. The common recognition site in these sequences is `GATAAG`, which is bound by factors like Gln3p, Nillp, Gzf3p and Uga43p. These upstream regions of 800bp are selected using RSA-tools [112].

5.5.2 Running time analysis

To have an idea of the execution time of our algorithm we should focus on the core of the algorithm. In each iteration we iterate over N_s sequences where we need to build a motif model, score a sequence of length L_k and estimate the number of motif instances. To build the motif model we need to iterate over $N_s - 1$ sequences and select the correct instances in these sequences. The execution time of this step depends on the size of the data set. Scoring a sequence with a given motif model is done in linear time. Also the estimation of the number of motif instances is done in time proportional to the sequence length. Based on this crude analysis we estimate that the time of one full iteration is of order $N_s(N_s - 1 + 2L_k)$.

Figure 5.2 displays the execution time of the basic algorithm in function of both the sequence length and the number of sequences. Each point in this grid corresponds to average time to find one motif in a particular data set of N_s sequences of length L. The number of sequences N_s varies from 10 to 90 and the sequence length L increases from 200bp to 2000bp in steps of 200bp. The surface illustrates the linear dependency of the MotifSampler on the sequence length and also the quadratic scaling with the number of sequences.

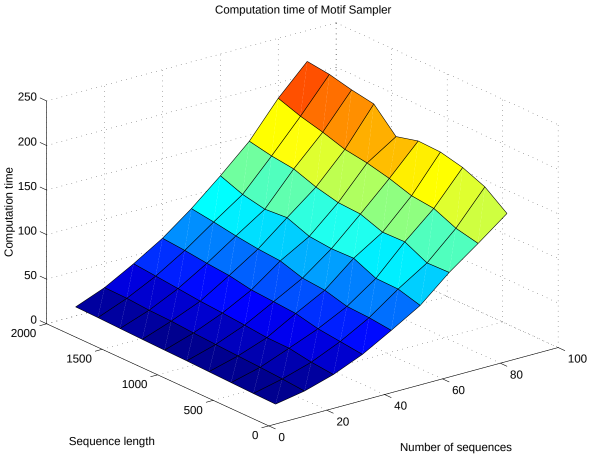


Figure 5.2: The execution time of the MotifSampler scales quadratically with the number of sequences and linearly with the sequence length.

5.5.3 On the convergence of the MotifSampler

Due to the stochastic nature of the sampling process it is difficult to define a reliable stop criterion based on the difference between consecutive motif models (cfr. EM). In our implementations we use a two step approach. In the description of the algorithm we have already stated that the algorithm is stopped after a fixed number of iterations. The definition of this number of iterations of the core sampling procedure is more based on heuristics than on sound theoretical criteria.

To illustrate the behavior of the Gibbs sampler we monitor in Figure 5.3 the evolution over 500 iterations of the information content (Eq. 5.11), log-likelihood (Eq. 5.12) and the number of motif instances (Eq. 5.7). For this example we selected the MET data set. One iteration is seen as the update of the whole data set. In each of the plots we start the sampling steps from the same initial conditions but the final motif differs. The three selected motif examples are `nCACGTGA` and `AACTGTGG` (the true motifs) and `ATATATAT`. The left plot in Figure 5.3 shows the behavior if the true motif is found. After approximately 20 iterations the motif is found and the MotifSampler stays in this state for the remainder of the procedure. The changes by the sampling of motif instances have only a minor impact on the constructed motif model. This means that the scores remain rather constant. In the second example, the MotifSampler first finds a more degenerate motif that has only seven instances. Due to the combination of the degree of degeneracy and limited number of instances, the slightest change in configuration can have a profound effect on the motif model. If we look at the center plots of Figure 5.3 we see that the scores are rather low. The ever changing consensus sequences (data not shown) also indicate that the motif model is very unstable. However, after an unpredictable number of iterations (here around iteration 320) the algorithm escapes from

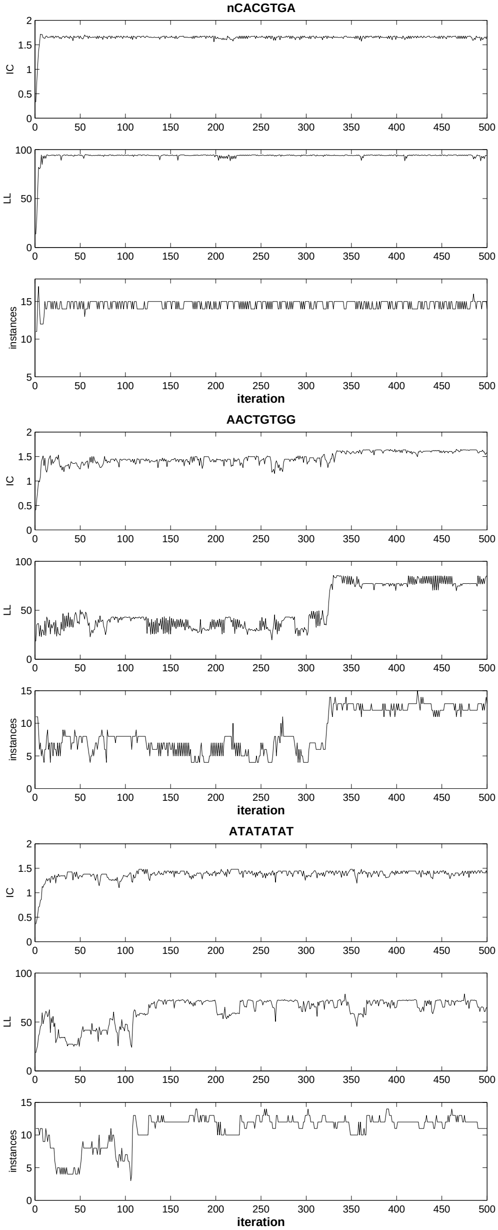


Figure 5.3: Evolution over 500 iterations of the information content, log-likelihood and number of motif instances for three different motif models found in the MET data set. The consensus sequence of the retrieved motifs are `nCACGTGA`, `AACTGTGG` and `ATATATAT`.

this unstable state and picks up one of the true motifs. In the last case we notice an even more chaotic behavior. A motif similar to the final motif model is here picked up after more than 100 iterations. This motif model has as `ATATATAT` and, given the `AT`-rich background, there are more instances present in the data set that resemble this consensus. The result is that in consecutive sampling steps different instances will be selected and the motif scores do not stay fixed.

Here we presented three examples to illustrate the evolution of the different scores in function of the number of iterations. The goal of this analysis is to draw conclusions on what criteria we should use to stop the iterations. Although the same initial conditions were chosen, different motifs are found after 500 iterations. While in the first case one of the true motif is already found after 10 iterations, was found after 500 iterations in the last example a false motif. It is thus hard to unambiguously define the optimal number of iterations. However, by repeating this analysis multiple times we notice that the true motifs are mostly found after a limited number of iterations. This is positive, since the order of our algorithm is quadratic in the number of sequences and we would like to keep the number of iterations as low as possible. Another remark we can make is that if we could prevent the algorithm initially to evolve to a local optimum with only a limited number of instances, this would probably augment the chance of picking up the true motif. As a solution we suggest to start the iterations with the assumption that there is exactly one motif instances per sequence. After a few iterations in this mode, the core procedure can be started.

5.5.4 Distribution of motif scores

Since the Gibbs sampler is a stochastic algorithm it can generate different results from the same initial conditions (as in Figure 5.3). This means that if we start the algorithm multiple times from the same initial condition, we will retrieve different motifs after a fixed number of iterations. It is thus important to have an idea how the scores of these motifs are distributed over the different runs. Moreover, in statistics one often compares results obtained with a given data set with the results obtained with random data. By comparing both distributions we should be able to find the significant differences.

Let us analyze the G-box data set in this way. For this purpose we generate 100 simulated data sets that have the same properties as the G-box data set. Thus, each set consists of 33 sequences of 500bp generated from the fourth-order *A.thaliana* background model[2]. Another technique that has been used to generate random sequences is *shuffling*. The input sequences are shuffled in such a way that the m-mer composition is conserved. For the analysis we run the MotifSampler 100 times on each data set with the same set of parameters. We search in each run for three different motifs of 8bp with a prior γ_1 set to 0.5. The results of this analysis are given in Figure 5.4 where the distributions of the consensus score (Eq. 5.10), information content (Eq. 5.11), log-likelihood

---

[2] data available at our website `http://www.esat.kuleuven.ac.be/~dna/BioI/`

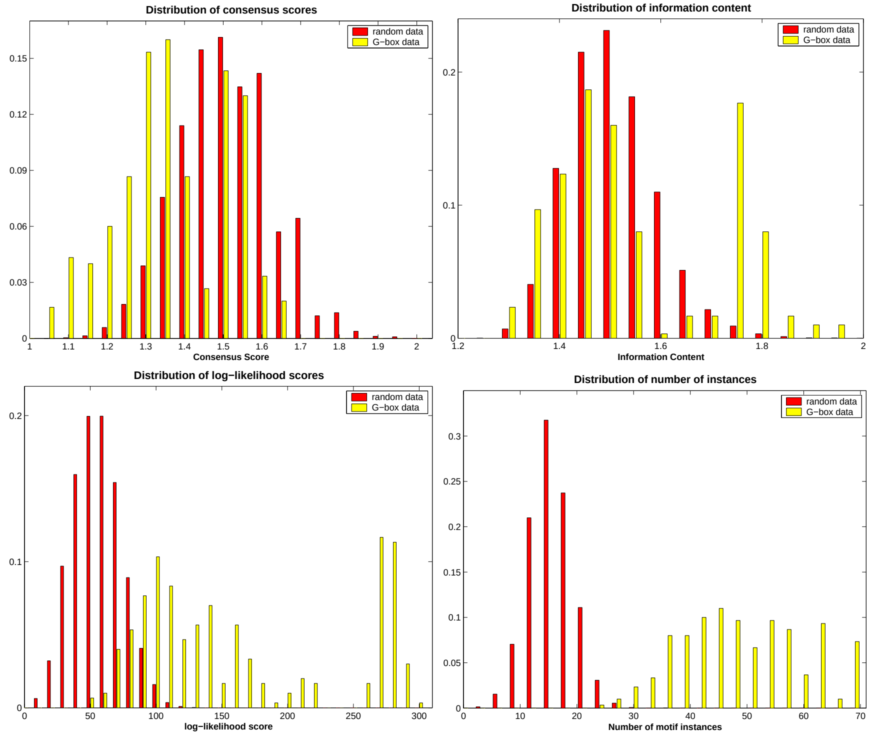


Figure 5.4: Comparison of the distribution of the scores of the motifs found in the G-box data set and in 100 random data sets with the same configuration as the G-box data set. The displayed scores are consensus score (top left), information content (top right), log-likelihood (bottom left) and the number of instance found (bottom right). The scores in the random data sets show a normal distribution in each case. The scores of the motifs found in the G-box data set show two distinct peaks. The peak of the highest scores corresponds to the true motifs.

score (Eq. 5.12) and the number of instances (Eq. 5.7) are plotted for both data sets.

The first plot of Figure 5.4 gives the distribution of the consensus scores. The consensus scores gives an indication of the level of conservation of the motif model but does not take into account the number of motif instances. Motifs built from only two or three instances might have a larger consensus score than a motif built from a more extensive set of instances. The distribution of the scores in the G-box data sets forms two peaks. However, the highest scores completely overlap the distribution of the consensus scores in the simulated data. On average, the consensus score of a motif found in the random data set is better than the consensus score of a motif found in the real data set. From this we learn that the consensus score is not suited to distinguish between motifs found in the random data set or in the real data set. The second plot of Figure 5.4 shows a different picture. Here the histogram of the information content is plotted. The distribution of scores from the G-box data has again two parts and the scores from the simulated data are normally distributed. But here the peak of the highest scoring motifs does no longer overlap with the scores from the simulated data. The explanation is that in the simulated there is no motif inserted thus the motifs found in such a set are more likely to look like the background. Therefore the information content will be lower. The distribution of the log-likelihood in the third plot of Figure 5.4 gives the most convincing result. In this case there is clear distinction between the two distributions. The log-likelihood of the motifs found in the G-box data set is significantly higher than the log-likelihood of those found in the random data. Finally, the distribution of the number of motif instances is given. In the random data set motifs have on average fifteen instances, which is approximately in half of the sequences in the data set. In the G-box data set this number lies much higher. With an average of 48 instances, there is more than one instance per sequence found. This also explains the difference in consensus score. A motif built from a limited set of sequences is typically more conserved than a motif with more instances.

Since the G-box is a rather strong motif that differs significantly from the background model we also repeat the analysis on another data set. We select the NIT data set from the yeast regulons. This set contains only 7 sequences of 800bp and has as known binding site `GATAAG`. This motif is harder to find in this data set than the GBF (see results in Section 6.2). Again for comparison, 100 sequence sets are created by randomly selecting 7 sequences of 800bp from a set of 1600 upstream regions in yeast.

In Figure 5.5 we observe similar behavior as in the G-box data set although it is less explicit. The distribution of consensus scores is again the same in both data sets. The distributions of the information content and log-likelihood scores of the motifs in the NIT data set are slightly higher than those from the random data. The most important difference between the G-box and NIT example is that there is no distinct peak of higher scoring motifs in the NIT data set. This implies that the motif hidden in this data set is not as explicitly present as the motif in the G-box example. It will thus be harder to pick up the

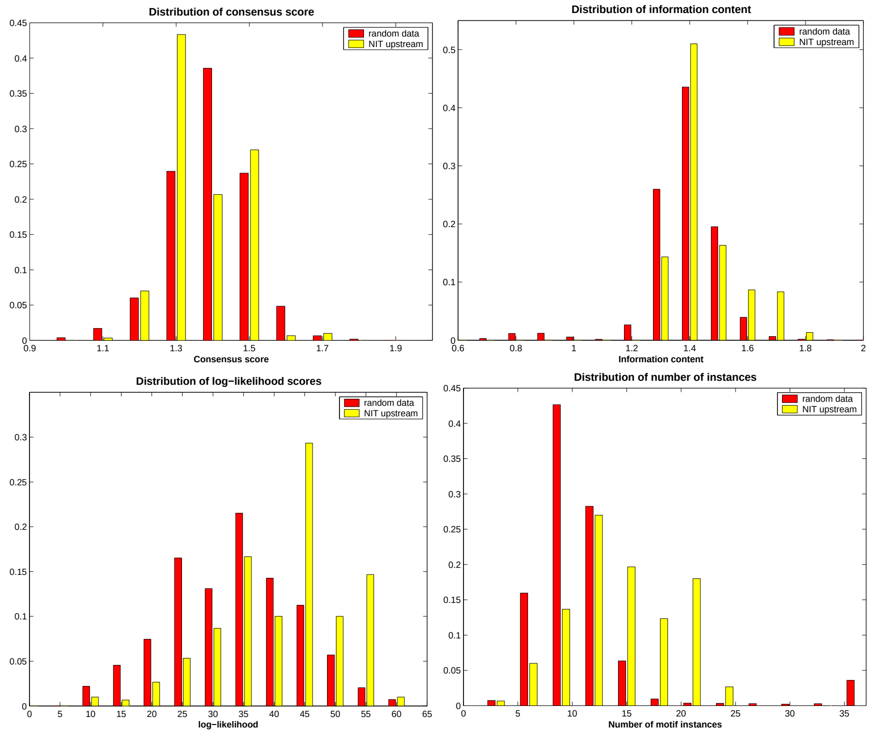


Figure 5.5: Comparison of the distribution of the motif scores in the upstream region of the NIT genes and 100 random selected sets from all yeast upstream regions.

true motif.

From this analysis we could conclude that it is necessary to do multiple runs to generate a distribution of scores. In the optimal setting we would compare the motifs found in the data set with the motifs found in a similar random set. Screening a large collection of random data sets might be too time consuming. However, we can already learn something from the distribution of the motifs found in the true data set. If the distribution of the log-likelihood score shows two distinct peaks chances are high that the true motif is found in the peak with the highest scores. A possible approach would be to look at the highest scoring motifs and also take into account the number of times similar motifs are found. The more similar motifs are found with a high score the more outspoken the distribution will be.

5.5.5 Effect of the motif length

Another parameter of our algorithm is the length of the motif model. In our current implementation this length is fixed and needs to be defined by the user. Therefore it is interesting to know what the effect is of this parameter on the retrieved motif model. As an example we use the MET data set in which two conserved motifs are found, namely `TCACGTG` and `AAAACTGTGG`. We fix all parameters except the motif length which is varied from 5 to 14bp. Each test is repeated 100 times. We compute the distribution of the scores over the set of motifs found. In Figure 5.6 the distribution of the consensus score, the information content and log-likelihood score are plotted as a function of the motif length. The distributions of the consensus score and the information content show a similar trend. As the motif length increases the scores move towards lower values. This can be explained rather easily. Both the consensus score (Eq. 5.10) and the information content (Eq. 5.11) are computed as the average over the full motif. If the motif length increases more, probably uninformative, positions are added to the model. These positions will of course lower the average score of the motif. In both cases the distributions also show a peak at the higher scores for some of the motif lengths. This indicates that for that length there is a distinct, strong motif present in the data set.

A different trend is found in the distributions of the log-likelihood scores for the different motif lengths (bottom Figure 5.6). Here, the distributions are more smeared out over a longer range of scores if the motif length increases. However, there is still a distinct peak found at higher scores for some of the motif lengths. It is this peak that tells us that the length of the true motifs corresponds to this length.

5.5.6 Estimation of number of motif instances

So far we have not discussed how to estimate the number of different motif instances. Let us first have a look at the prior distribution that we have introduced in Section 5.2. The parameter that defines this distribution is $P(c = 1|\mathbf{\Theta}, \mathcal{B}_m)$. As an example we plot two resulting prior distributions constructed from two different priors in Figure 5.7. The plot on the left has prior set to 0.3 and shows a declining distribution. The plot on the right has prior set to 0.8. and is maximal at $c = 1$. Since we need to renormalize the parameters of the distribution such that their sum is equal to one, the value of $\Gamma_0(1)$ is lower than the initial value of γ_1.

Next, we study the effect of the prior on the number of motif instances found in a sequence set. We expect that the number of instances will grow with increasing prior. To illustrate this, we used the MET and NIT data set from the yeast regulons. In each test we increase the prior and search for one motif of 8bp and this is repeated 1000 times. Each test results in a list of motifs in which we search for the respective consensus sequences `TCACGTG` and `GATAAG`. Finally we compute the average number of instances found in a test.

Table 5.2 shows the evolution of the number of motif instances found in function of an increasing prior. A zero means that the correct motif was not found for this prior. As expected, the number of motif instances increases with an increasing prior. For low priors only segments that very closely match the motif model are selected, while at higher priors the motif model becomes more

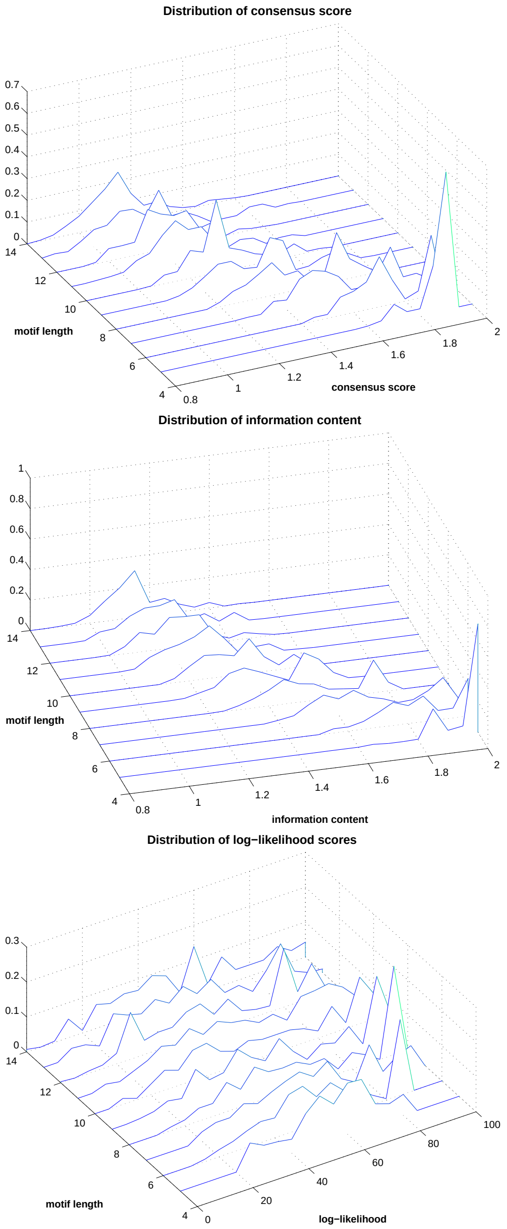


Figure 5.6: Distribution of consensus score, information content and the log-likelihood in function of the changing motif length. The average value of the consensus score and the information content tends to decrease as the length of the motif increases. The distribution of the log-likelihood score becomes more flat if the motif length increases. In all three cases, there is a peak at the higher scores where the motif length corresponds to the true motif.

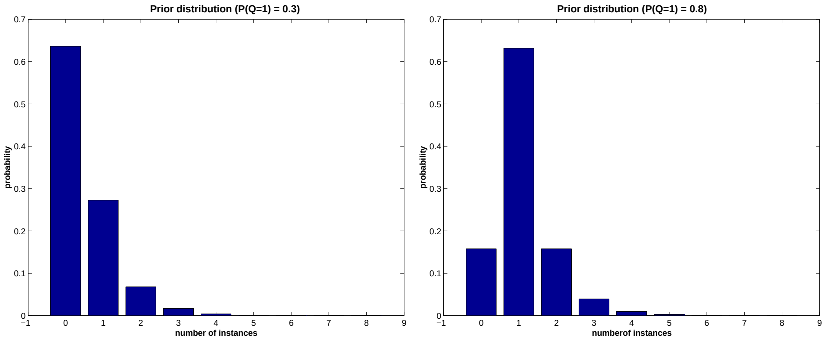


Figure 5.7: Distribution of the prior probability of finding c copies built from different starting values. (left) prior γ_1 is 0.3 and (right) prior γ_1 is 0.8.

Table 5.2: Number of instances in function of prior.

$\gamma_1 =$	.005	.01	.05	.1	.2	.3	.4	.5	.6	.7	.8	.9	.95	.99
MET	9	11	16	19	20	20	21	23	23	25	26	27	28	30
NIT	0	6	9	11	15	18	19	20	21	22	23	24	25	25

degenerate and more instances are selected. A secondary effect is that the number of times the motif is found in the set of 1000 motifs drops. For instance, the percentage of runs in which the true element `TCACGTG` is found in the MET data sets starts at a level of 92% with a prior of 0.005 and drops to and stabilizes round 55% for priors of 0.5 and larger. This means that when the prior is increased the chance of finding other motifs increases. One suggestion might be to always set the prior to low values. However, if the prior is set too low then no stable motif might be found or there is only a limited number of instances is found. This is exemplified in the NIT data set when the prior is set to 0.005 and no motif is found.

5.6 Elaboration of a motif finding strategy

In the previous section we have described the core of our Gibbs sampling algorithm and did several tests to evaluate the behavior under diverse conditions. The question we should ask ourselves is now how we can apply the findings from the performance analysis to our algorithm? The purpose of this analysis was to deploy a strategy to tackle the motif finding problem within the capabilities of our implementation.

There are two parts in this motif finding strategy. First, there is the MotifSampler algorithm itself where the implications are in the convergence behavior. Although a good implementation of the sampling procedure is essential to the performance of the algorithm it is not sufficient to have an actual working algorithm. Therefore some extras are to be added to this sampling procedure. We discuss some practical problems and propose solutions that can be combined into the final algorithm. The second part concerns the analysis of the motifs found in repeated runs of the MotifSampler and in with different parameters.

5.6.1 Convergence steps

Because of the stochastic nature of the Gibbs sampler, it does not converge to a fixed point estimate as EM does. Therefore it is not possible to stop the iterations if there is no difference anymore between two consecutive found motif models or alignment vectors. To overcome the convergence problem, we decide to add an extra convergence step to our Gibbs sampler implementation, in which we do no longer sample the new alignment vector but rather select the best scoring instances. This convergence step is stopped when the alignment vector remains unchanged, which is typically after four or five iterations. This procedure will normally converge, but sometimes it might get stuck in an oscillating mode. Such a mode is for example the case where in the first step one instance is added to the alignment vector. This results in a slightly modified motif model such that in next step the just added instance is again excluded. If this happens this convergence step will oscillate between the two modes.

Program 3: Convergence procedure

1. Build the motif model from the current alignment vector.
2. Score each sequence with the new model.
3. Compute for each sequence the number of motif instances.
4. Update the alignment vector by selecting the best scoring instances in each sequence.
5. Iterate from 1 until the alignment vector stays the same.

5.6.2 Shifting the alignment vector

Another problem of the basic Gibbs sampler, as it is outlined here, is that the sampler might get stuck in a shifted version of the motif model. Figure 5.8 gives an example of such a shifted version of the motif model. In this small example we can clearly see the conserved part as `CACGTGy`. The Gibbs sampler could get stuck in a local minimum that is a shifted version of the true motif. In this example the shifted motif is given by the consensus `CGTGynn`. As we can see this consensus still contains a part of the true consensus, but it also covers some non-informative positions. Because of the link between the motif model and alignment vector, it is impossible for the sampler to escape from

	`TACAACCACGTGGCCTCAGACT`
	`ACTACTCACGTGCTCATCATCA`
	`TTTGATCACGTGGAATACCGGA`
	`GTATTCCACGTGGCAATTATCA`
	`CATGTGCACGTGCTGTCAGATG`
	`TATCCACACGTGGCGATGATGA`
true:	`CACGTGy`
shifted:	`CGTGynn`

Figure 5.8: Example of the consensus of a shifted version of the retrieved motif and the consensus of the true motif.

such a local optimum. Therefore, to escape from these local optima, a shifting procedure is inserted at certain points during the core sampling iterations. The aim is to find a better motif by shifting the alignment vector a few positions to the left and to the right. For each shifted alignment vectors the corresponding motif model is constructed and the corresponding motif scores are computed. The alignment vector that corresponds to the best motif is then selected.

5.6.3 Inclusion of the complementary strand

Since transcription factors can bind to the double stranded DNA, it is also useful to include the complementary strand into the analysis procedure. The straight forward way to tackle this problem is to double the size of the data set by including the reverse complement of each individual sequence. However, with this approach the noise present in the data set will also be doubled. Therefore we suggest a more careful approach. In the algorithm, the predictive update distribution of both the sequence S_z and its reverse complement is computed. Next, the distribution Γ_z is calculated and updated for both the strands. Finally, the alignment positions are sampled from S_z and then the positions on the reverse complement of the sequence are sampled. It is also possible to apply a mask on one strand that masks the positions found on the other strand.

5.6.4 Finding different motifs

So far, we have only discussed the issue of finding one motif that can have multiple copies but we would also like to find multiple motifs. To find more than one motif at the time we need to run the MotifSampler several times and in each run we will mask the positions of the motifs previously found. By masking the positions it will be impossible to find the same motif twice. Masking a certain position in the sequence can be achieved by forcing the weights $W(\mathbf{x}_{zj})$ to be 0 for all segments $\mathbf{x}$ that overlap with the previous motifs. The allowed overlap is a parameter that the user of the algorithm has to define but it has only a minor impact on the performance.

5.6.5 MotifSampler

The proposed extensions should allow to design a practical implementation with parameters that are easy to understand and to choose. The analysis of the convergence behavior teaches us that the sampling procedure can evolve to a rather unstable motif that has only a limited number of instance. After an unpredictable number of iterations this motif jumps to a more reliable motif model that has more instances. Since we want to limit the number of iterations we like to prevent the MotifSampler to get stuck too fast in such an unstable motif, we start with a few extra iterations in which we sample one motif instance in each sequence. We have implemented this algorithm first in Matlab and later an improved version in C++. The resulting motif models are reported in a specific matrix file. The instances are reported in a GFF file. To preserve the relation between instances and motif model has each entry in the GFF file has an ID that corresponds with the respective ID in the matrix file.

Program 4: Description of the MotifSampler algorithm

- Initialization: Compute the background model probability $P(\mathbf{x}_{kl}|S_k, \mathcal{B}_m)$ of all segments $\mathbf{x}_{kl}$ of length W with $k = 1 \ldots Ns$ and $l = 1 \ldots L_k - W + 1$.
- Repeat for each requested motif:
    1. Sample the alignment vector $\mathcal{A}$ by selecting at random one starting position in each sequence.
    2. Run the core sampling procedure thereby sampling exactly one instance per sequence for a fixed number of iterations.
    3. Do for a number repetitions:
        (a) Shift alignment vector.
        (b) Run the core sampling step.
    4. Run the convergence step.
    5. Create final motif model and alignment vector.
    6. Apply mask to sequences based on retrieved alignment vector.

5.6.6 Analyzing repeated runs

As discussed in the previous section: to be optimal, we should create random data sets similar to the input data set and perform an extended analysis of the distribution of motif scores to find the best motifs. However, this might take a long time given the complexity of the algorithm especially if the data set becomes large. The results above learned us that if we perform multiple runs with the same parameters and we select those motifs with the largest log-likelihood scores we will find in most cases the true motifs. The plots in Figure 5.4 also show a distinct peak in the distribution of the motif scores. This means that if we find the same high scoring motifs in a significant part of the runs, it is a good indication that this motif is specific to this set. Therefore, we should also compute for each of the topscoring motifs how many times a similar motif is found in the data set.

Combining these findings results in the following procedure that can be applied to use the MotifSampler.

Program 5: Procedure to apply MotifSampler in real life example.

1. Select the background model.
2. Define the parameters of the MotifSampler:
   (a) Motif length W (typical in the range from 6 to 15bp.).
   (b) Number of motif instances: we can opt for setting C_{max} or the defining the prior.
   (c) Number of different motifs n.
   (d) Maximum allowed overlap between different motifs.
   (e) Number of repetitions R (eg. 100).
3. Run the MotifSampler for R times with the defined parameters.
4. Post-processing of the results:
   (a) Rank all $n \times R$ motifs according to a specific score (eg. log-likelihood score).
   (b) Count for each motif how many similar motifs are found in the total list.
   (c) Report the best motifs, thereby removing the similar motifs from the list.

In the next chapter we will show that this approach can be used to find the true hidden binding sites in diverse sequence sets.

CHAPTER 6

MotifSampler: case studies

Time has come to test the MotifSampler on some real life biological examples. In this chapter we present the results obtained with four different data sets. The first set of sequences are the upstream regions of genes in plants that are regulated by the G-box binding factor. At first we show how the different types of background model influence the motif detection process. Next, we introduce noise in the data set to test the robustness of our algorithm. The result show that the higher-order background model based on a set of carefully selected intergenic sequence gives the best results. The second test consists of ten regulons in yeast constructed from sets of genes regulated by the same transcription factor. This set is used to show how the algorithm performs under various conditions. In the third example we examine the usage of higher-order background models in prokaryotes. The test case is the well known σ^{54} *factor in E.coli and P.aeruginosa. The tests show that if the wrong background model is used, the results will be seriously contaminated. Finally, we study the application of the MotifSampler in a real microarray experiment. As a test case, we use the well known cell cycle experiment of Spellman et al. [97]. We analyze in detail four clusters of coexpressed genes found with Adaptive Quality-Based Clustering. In three out of four clusters we find specific motifs, while in one cluster nothing specific is found. In two clusters we find motifs that are known to be involved in the cell cycle.*

6.1 G-box[1]

6.1.1 Data set

The first experiments are conducted with the G-box data set that we have used in many examples since the beginning of our research [104, 105, 106, 107]. The sequence set itself was already introduced as an example in Chapter 3. Here we just repeat that the set consists of 33 sequences of 500bp upstream of genes experimentally verified to be regulated by GBF. The consensus of the binding site is described as `CACGTG`. To evaluate the robustness of our motif finding algorithm to noise we also use a set of random selected genes in *A.thaliana*. The random set consists of 87 sequences of 500bp upstream of translation start in genes from *A.thaliana*. The upstream region of the genes in this set are supposed to contain no G-box binding site. This set is used to introduce noise into the test sets and to evaluate the performance of the Motif Sampler under noisy conditions.

6.1.2 Creation of the background model

To have the best possible representation of promoter sequences or intergenic sequences, a data set consisting of carefully selected intergenic sequences is constructed [104]. We follow a previously described rationale that was developed to build Araset [82]. To define clean intergenic sequences, all complete cDNAs of *A.thaliana* were downloaded and aligned on BAC sequences. The aligned genes were checked manually by an expert. Each time the cDNAs matched two consecutive genes on the BAC, the intergenic sequence was extracted. The sequences with a length below 10 kb were then extensively checked for any potential coding sequences that was not annotated, using BLAST [5] for homology searches and prediction software as *EuGène* [92]. 78 intergenic sequences were retained, representing a total of 156.087 bp. These sequences were added to the 94 intergenic sequences in Araset, resulting in a data set with 341.248 bp. Figure 5.1 showed the three different configurations in which neighboring genes can occur. 105 sequences have the first configuration (1) consisting of two genes coding in tandem on the same strand. 38 sequences have the second configuration (2) where both genes are pointing away from each other. In the last case (containing 29 sequences) the transcription of the two genes is convergent. The transition matrix was only built from the intergenic sequences of the classes (1) and (2), which likely contain either one or two core promoters.

A thorough study of this set of intergenic sequences shows that when all hexamers are counted, there are more than 340.000 examples in this data set. Although this number is much higher than the 4^6 possible hexamers, the results show that we have to handle this data with care. The hexamer with highest number of occurrences was `AAAAAA` with 2018 instances. On the other

---

[1] These results have been partially described in an article in Bioinformatics [104].

hand, there was no instance at all found of the hexamer `GCGGGC`. The consequence of this observation is that the fifth-order background model is not reliable, since the entry $P(\texttt{C}|\texttt{GCGGG})$ in the transition matrix would only consist of a pseudocount.

6.1.3 Influence of the background model

In the first set of experiments we look at the influence of the type of sequence data on the composition of the background model. Let us first look how the algorithm behaves if the background model is computed from the sequences in the G-box data set. In Figure 6.1, we compare the transition matrices of the third-order background models generated from respectively the intergenic data set and the G-box data set. This scatterplot shows that there is a significant difference between the two models. Zooming in on the individual points we single out one particular entry in the transition matrix, namely $P(\texttt{G}|\texttt{CGT})$. This probability of finding a `G` after `CGT` is almost 2.5 times higher in the G-box sequence than in the intergenic sequences. This comes as no surprise since `CGTG` is part of the true motif, `CACGTG`, that is over-represented in the G-box data set.

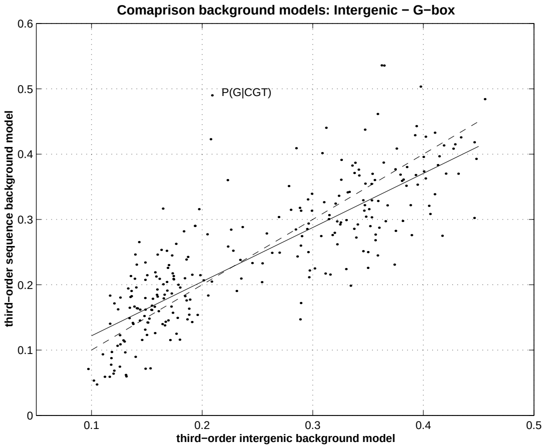


Figure 6.1: Comparison of third-order transition matrices computed from intergenic sequences and the G-box data set. The full line represents the fitted linear regression, while the dashed line corresponds to $x = y$. We indicate one of the outliers $P(\texttt{G}|\texttt{CGT})$ which corresponds to part of the G-box consensus `CACGTG`.

It is clear that using the background model constructed from the input data in a motif finding procedure will give rather skewed results. This is illustrated in Table 6.1, where we have run the G-box sequence set through the MotifSampler testing 10 different background models. At first we use the background models of order 0, 1, 2, 3 and 4 from the intergenic sequences. In this case each time the best motif found corresponds to the G-box. The G-box consensus is also found in more than 90% of the runs. For the second test, we construct the background models from the input sequences. In this case the numbers are far from good. Only for the orders 0, 1 and 2 the G-box motif is the best motif found. However the number of times it is found is much lower than with the intergenic background model. In the case of the third- and fourth-order background model the G-box motif is not found at all. The fact that it is also only found once with the second-order model indicates that in this case it is rather an exception than a true motif.

Table 6.1: Best scoring motif found in G-box data set.

	intergenic		**data set**	
order[a]	**#sim**[b]	**best**[c]	**#sim**	**best**
0	94	nmCACGTG	17	mCACGTGk
1	100	mCACGTGn	52	mCACGTGn
2	90	nmCACGTG	1	nCACGTGn
3	91	nnCACGTG	0	GTnnGATk
4	90	nmCACGTG	0	CCTTATCn

[a] order of the background model
[b] number of motifs similar to the best scoring motif
[c] consensus of the best scoring motif

This test shows that the use of a background model that is built from the sequences in the data set is not a good practice, especially for higher-order background models. The background model built from the data set itself is skewed and will give higher probabilities to the motif instance than when the same instances are scored with the independent background model. It is thus advisable to always use such an independent background model. In the next sections we will also explain that not every independent background model is usable, it should correspond to the type of region in which the motifs are hidden.

6.1.4 Predicted motif positions

To further analyze the results we check the retrieved motif instances. Since we have built the G-box data set from known regulatory elements stored in PlantCARE, the position of the experimentally verified G-box binding sites is known. It is thus possible to compare the retrieved motif instances with those known instances. Since the G-box is palindromic we count each instance on both strands. This means we have a total of 70 instances in 33 sequences. As potential G-box motifs, we select all motifs that have `CACGTG` in their consensus sequence. The instances of these motifs are compared with the true motif positions. Now, we can distinguish three cases. First, a retrieved instance overlaps with the true instance and is thus correctly predicted. If the motif does not overlap with a true motif it is classified as ’wrong’. If there is no overlapping instance found in the set of retrieved motif instances, then this instance is classified as 'missed'.

In this example, we look at the motifs obtained with the intergenic background models (see Table 6.1). We do not look at the results with the background models from the sequence data, since in this set the G-box motif was not always found. For each of the tests, we show in Table 6.2 the average and standard deviation of the following values: (1) the total number of instances found, (2) the percentage of correctly predicted instances, (2) the percentage of missed instances and (4) the percentage of wrongly positioned instances.

Table 6.2: Classification of predicted motif positions

order	**#inst**^{a}	**#correct**^{b}	**#missed**^{c}	**#wrong**^{d}
0	66.09±5.89	75.0±7.4%	29.2±6.9%	25.0±7.6%
1	60.80±5.76	85.6±6.9%	25.7±6.0%	14.4±8.9%
2	54.36±6.61	90.4±12.7%	29.8±9.9%	9.6±8.5%
3	57.72±6.01	83.2±10.3%	31.4±8.5%	16.8±8.5%
4	54.56±7.13	84.4±12.5%	34.2±9.7%	15.6±9.1%

[a] number of instances found in G-box data set.
[b] percentage of instances that overlaps with true motif instances.
[c] percentage of true motif instances that are not found.
[d] percentage of instances that do not overlaps with true motif instances.

Analysis of Table 6.2 shows that there is not a great difference between the different background models. Although, we notice that with the single nucleotide background model (order 0) the most instances are found, but that 25 percent of these instances are wrong predictions. The number of wrongly predicted instances lays round 10 percent lower for the other background models. In this example we see that the second-order background model is slightly better than the other models and that the single nucleotide model has the worst performance.

6.1.5 Robustness in presence of noise

An important aspect in the analysis of our approach is the evaluation of our algorithm when noise is present in the data set. Noise in motif finding is the presence of sequences in which no motif instance is hidden. Especially when one is working with microarray data noise can be present in the data set. It is thus important to find out how the algorithm behaves under such noisy conditions. To mimic noisy conditions, we construct several data sets starting from the G-box data set to which we add an increasing number of random selected sequences in which no G-box binding site is present. In each step we add ten extra sequences from the random selection of sequence. This results in seven data sets where the number of sequences ranges from 33 to 93. Sequentially adding these noisy sequences to the G-box data set, will make it harder to retrieve the true G-box instances.

In this set of experiments we run MotifSampler with the motif length set to 8bp and the prior γ_1 set to 0.5. In each run we search for three different motifs.

In Table 6.3 we summarize the results of this study. In this table we show the number of times the G-box consensus is found among the list of 300 motifs together with the rank of the best scoring G-box motif. The top part of the table gives the results with the intergenic background model. The results in the bottom part are generated with a background model built from the input sequences.

We see immediately in the bottom part of Table 6.3 that using a background model built from the input data is not advisable. The G-box is only found as a stronger motif with the first-order model from the sequences. In the other cases the G-box might be found as the best scoring one but it is only found in a very limited number of runs. When using the intergenic background models the results look much better. Up to the level of 30 noisy sequences, the G-box motif is found as the best scoring motif if we use the higher-order background models. When the level of noise increases above the level of 50 percent in the data set, the G-box motif is no longer found as the top scoring motif but it is still found among the best scoring ones. Considering the number of times the G-box motif is found, there is a trend to find the G-box motif in a decreasing number of repeats. Again, the fastest decline is reported when using the single nucleotide model. For instance, when 50 noisy sequences are added to the G-box data set the G-box motif is found as the fourth motif and it is found in 33 runs. When using the higher-order models the G-box motif is found as the second best motif in the data and it is also found in more than twice as many runs as with the single nucleotide model.

Table 6.3: Number of times the motif `CACGTG` is found in an increasing noisy data sets.

noise[a]:	**0**	**10**	**20**	**30**	**40**	**50**	**60**
Intergenic background model							
0[b]	85(1)[c]	76(1)	67(1)	69(2)	37(4)	33(4)	6(4)
1	100(1)	98(1)	94(1)	95(1)	86(2)	86(2)	58(3)
2	92(1)	90(1)	88(1)	92(1)	59(2)	73(2)	37(2)
3	92(1)	84(1)	87(1)	81(1)	64(2)	67(2)	41(2)
4	88(1)	76(1)	81(1)	77(1)	53(1)	65(2)	34(4)
Sequence set background model							
0	17(1)	7(1)	9(1)	0(1)	0(1)	1(10)	0(0)
1	52(1)	56(1)	32(1)	36(1)	6(1)	6(2)	2(-)
2	1(1)	4(1)	8(1)	6(1)	1(1)	1(-)	1(-)
3	0(0)	1(1)	3(1)	1(1)	0(1)	2(-)	0(0)
4	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)

[a] Number of added noisy sequence to the G-box data set.
[b] Order of the background model.
[c] Number of times the G-box motif is found (rank of G-box motif).

6.1.6 Discussion

In this first case study, we investigated three aspects of our algorithm. First, we test how the choice of the background model influences the performance of the motif detection algorithm. Therefore, we compare two different approaches to create the background model. We create different higher-order background models from an independent data set of carefully selected intergenic sequences and also from the sequences in the data set. The results show that using an independent background model is much more reliable than the background model built from the input sequence data. The second aspect, we discussed, is the accuracy of our approach. To quantify this we compared the retrieved instances with the instances annotated in the database. This analysis shows that the single nucleotide background model performs a bit worse than the higher-order background model. The third aspect is the robustness of the algorithm when noise is present in the data set. To test the influence of noise we added gradually more random sequences to the G-box data set and run our sampling procedure on this data set. These results show that our algorithm is capable of finding the true motif as the best scoring motif even if the number of noisy sequences in the data set equals that of G-box sequences. For increasing level of noise G-box motif is still found, but the position at which the G-box motif occurs in the list of best scoring motifs drops. However, for the higher-order background models the number of times a G-box motif is found in 100 runs stays high.

6.2 Yeast regulons: analysis

A good evaluation of the elaborated procedure for the MotifSampler is a full scale analysis of the ten yeast regulons introduced by van Helden et al. [111]. The primary goal of this analysis is to demonstrate that we are able to find different types of common binding sites in various data sets. Such a large scale analysis also allows to pinpoint the weaknesses of our approach and to have a better understanding of the parameters.

6.2.1 Data sets

van Helden et al. [111] published an extensive analysis of ten sequence sets in yeast to illustrate the applicability of their oligonucleotide frequency methodology to detect common transcription factor binding sites in sets of coregulated genes. They referred to such a set of coregulated genes as a *regulon*. Since yeast is a widely studied model organism it provides numerous examples of coregulated genes. Knowledge about the transcription factor involved in the common expression of these genes makes such a regulon well suited to test motif finding methods.

In Table 6.4 we give an overview of the regulons and the consensus of the

Table 6.4: Known motifs in the yeast regulons.

Family	$N_s{}^a$	Consensus	Transcription Factors
GAL	6	`CGGnnnnnwnnnnnCCG`	Gal4p
GCN	38	`rrTGACTCTTT`	Gcn4p
HAP	8	`CCAAy`	Hap(2,3,4,5)p
INO	11	`CATGTGAAwT`	Ino2p/Opi1p
MET	11	`TCACGTG`	Cbflp-Met24p-Met28p
		`AAAACTGTGG`	Met(31,32)p
NIT	7	`GATAAG`	Gln3p, Nillp, Gzf3p, Uga43p
PDR	7	`TCCGCGGA`	Pdrlp, Pdr3p
PHO	5	`GCACGTGGG`	Pho4p
		`GCACGTTTT`	Pho4p
TUP	25	`kAnwwwwATsyGGGGw`	Mig1p
YAP	16	`TTACTAA`	Yap1p

^{a} number of sequences in the data set.

binding site that has been associated with a specific transcription factor. Two sets, TUP and YAP, are selected from the first microarray experiment on a genomic scale in yeast [29]. The regulons cover a wide range of different motif types. There are some well-conserved motifs that differ significantly from the `AT`-rich background like `GCACGTGGG` in the PHO regulon and `TCCGCGGA` in the PDR regulon. The motif present in the HAP regulon is a rather `CCAAy`, which will be harder to detect. The other extreme of the spectrum is the spaced dyad, `CGGnnnnnwnnnnnCCG`, bound by Gal4p. Two-third of this motif is not conserved and does not carry any information, but the motif has some similarity since the two conserved parts are palindromic.

6.2.2 Experimental setup

We run the stand-alone version of the MotifSampler on the ten regulons. For this analysis we try the following parameter settings:

- Motif length: 6, 8, 10, 12, 14
- Prior probability of finding one instance: 0.5
- Background model: third-order yeast intergenics
- Number of different motifs: 3
- Number of repetitions: 100

We will only modify the motif length in this analysis. The other parameters are kept fixed. A prior γ_1 of 0.5 is chosen to allow sufficient variability in the retreived motif model. As the background model, we use the third-order model built from all the upstream regions in yeast. These sequences were created using RSA-tools [112], where 800bp upstream of all genes in yeast was selected and the parts overlapping with the upstream ORF were removed. The choice of parameters implies that we end up with five times 300 motifs for each of the ten different data sets. To evaluate these motifs we first compute for each motif how many similar motifs are found within the list. Similarity is computed using the Kullback-Leiber distance defined in Equation 5.13. Next, we rank all motifs according to their log-likelihood score, thereby removing all the similar motifs with a lower score from the list. Finally, we try to locate by visual inspection within this list the motif that resembles best the consensus of the known motif in the regulon.

6.2.3 Results

The most important results are summarized in Tables 6.5 to 6.14. In this full scale analysis, we only report the best scoring motif for each motif length in the respective regulons. If this motif does not resemble the true motif, we also report the first motif(s) in the list that resemble the consensus of the true motif(s), if it is present. For each selected motif we report in the respective tables the consensus of the motif, the number of sequences in which the motif is found, the number of motif instances, the consensus score (Eq. 5.10), the information content (Eq. 5.11), the log-likelihood score (Eq. 5.12), the rank of the motif in the list of top scoring motifs and, finally, the number of times a similar motif is found.

We now discuss the results of each regulon individually and point out strengths and the weaknesses of our algorithm. We start with those regulons in which the true are more easily found and we end with the special case of the GAL regulon, where are rather degenerated motif needs to be retrieved.

MET regulon

The first regulon we analyze is the MET regulon, which we have already used in other tests in the previous chapter to illustrate the behavior of the algorithm. The genes in this set are regulated by two transcription factors in response to methionine. This set contains thus two common motifs namely `TCACGTG` and `AAAACTGTGG`. If we look at Table 6.5, we see that these two motifs are in each of the tests found as the two best scoring motifs and this for all of the motif lengths. The position in the ranking might change over the different motif lengths. Let us elaborate on those results in more detail. First, we can see that the log-likelihood score is of the same order for both motifs, which explains the possible switch at the top. However the number of similar motifs is much higher for the `TCACGTG` motif than for the `AAAACTGTGG` motif. The results nicely show that both motifs are very pronounced within this specific data set.

Table 6.5: Best scoring motifs in the MET regulon

consensus	#seq.[a]	#inst.[b]	cs	ic	ll	rank	#sim.[c]
`CACGTG`	9	26	1.91	2.11	84.75	1	74
`AAACTG`	11	19	1.79	1.73	78.75	2	4
`TCACGTGA`	9	20	1.70	1.80	96.90	1	86
`AACTGTGG`	11	13	1.67	1.69	87.25	2	22
`AAAAnTGTGG`	11	14	1.59	1.55	99.48	1	14
`TCACGTGAnn`	10	22	1.37	1.45	98.75	2	85
`rAAAACTGTGGn`	11	15	1.33	1.32	101.91	1	12
`nnTCACGTGAnn`	9	28	1.07	1.15	97.17	2	31
`nnnnnCACGTGAnn`	9	22	1.04	1.06	101.82	1	23
`mAAACTGTGGnkny`	9	11	1.26	1.26	98.74	2	4

[a] number of sequences in which motif is found
[b] number of instances found
[c] number of similar motifs found in the list of all motifs found

GCN regulon

As is shown in Table 6.6, the topscoring motif found in this regulon is consistent over all motif length. The same core is found as best motif for each motif length. The number of motifs similar to the topscoring motif ranges from 100 to 37, which indicates the stability of the motif. The retrieved consensus sequences are also fairly similar to the consensus, `rrTGACTCTTT`, of the known TFBS. However, there are some minor differences. When searching for a longer motif, the core of the binding site is denoted as `TGAsTCA`. In this example a `A` is selected at the end instead of a `T`, and also a `G` or a `C` is selected in the center. One possible reason to explain this is that the consensus `TGAsTCA` is palindromic and is therefore preferred by our algorithm over the less symmetric `TGACTCT` by our algorithm. We also notice that the tail of `T`’s is not picked up in these best scoring motifs. Only the palindromic core contains information while the surrounding position are uninformative.

Table 6.6: Summary results: GCN

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`TGACTC`	30	55	1.85	1.91	225.65	1	100
`nTGACTCn`	32	52	1.52	1.49	244.95	1	65
`nTGAsTCAnn`	32	66	1.22	1.19	243.33	1	51
`nnnTGAGTCAnn`	34	66	1.04	1.01	270.64	1	45
`nnnnnTGAsTCAnn`	32	70	0.90	0.87	277.58	1	37

for a legend of the table, see Table 6.5

INO regulon

The consensus of the known binding site in the INO regulon has as its consensus sequence `CATGTGAAwT`. The reverse complement of this consensus is `AwTTCACATG`. It is this reverse complement that is found as the topscoring motif for each of the motif lengths. Compared to the previous two regulons, we notice that the number of similar motifs is slightly lower than in the MET and the GCN regulon.

Table 6.7: Summary results: INO

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`CACATG`	7	12	1.81	1.86	60.19	1	41
`TTCACATG`	7	11	1.66	1.58	67.47	1	20
`TTCACATGCm`	7	10	1.43	1.44	75.62	1	23
`TTCACATGsnnC`	6	9	1.36	1.43	79.97	1	28
`nTTCACATGCmnCm`	7	10	1.20	1.22	92.47	1	28

for a legend of the table, see Table 6.5

PHO regulon

The Pho4p *helix-loop-helix* can bind to with high affinity to `GCACGTGGG` and with medium affinity to `GCACGTTTT`. Since both motif models are partly overlapping (they both share the part `GCACGT`), it will always be harder to find both at the same time. If a motif is found that contains this core, it can partially mask the instances that corresponds one of the two possibilities. This effect will be more pronounced for the shorter motifs. Table 6.8 shows that in the first three cases (6, 8 and 10bp), we only find the high affinity site. When we look at the longer sites it is also possible to find the medium affinity site, but here the reverse complement of the binding site with medium affinity is selected. Analysis of the instances that make up the different motifs shows that instances might be

Table 6.8: Summary results: PHO

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`CGTGsG`	5	11	1.60	1.93	45.74	1	84
`CACGTG`	5	12	1.77	1.96	45.25	2	100
`CACGTGnn`	5	13	1.39	1.60	52.70	1	97
`snCACGTGsG`	5	10	1.14	1.35	55.97	1	75
`CACGTGGnrnkA`	5	8	1.16	1.30	62.74	1	21
`AAACGTGnGAwT`	5	9	1.22	1.20	60.01	2	24
`nnrsCACGTGsn`	5	14	1.01	1.20	57.88	3	43
`ysCACGTGnrnnnr`	5	11	0.99	1.13	62.85	1	44
`AACGTGCGwnTwnm`	5	10	1.06	1.06	62.25	2	15

for a legend of the table, see Table 6.5

part of both forms. This implies that the both forms will be not found in the same run, which is also reflected in the number of similar motifs found. For the short motifs the top scoring motif is found in almost all runs. For the motifs of length 6bp we also included the second motif in the ranking. This motif overlaps with the best scoring and is found in all runs.

NIT

Looking at Table 6.9 it seems that the common element in the NIT family is harder to find with our algorithm than in the previous regulons. Except for the case where W is 8bp, this consensus is not found as the best motif and only. This is somewhat contrasting with the results of van Helden et al. [111] where the hexamer `GATAAG` is found as the most significantly over-represented hexamer in this data set. In the particular case of the motif length set to 8bp, the MotifSampler prefers the reverse complement of the motif `CTTATC` over the described consensus `GATAAG`. For the longer motif lengths the true consensus is not found in the top five. However, if we look at the number of similar motifs, we see that the motifs with the true consensus are still found in 20 runs. If we look back at the distribution of the log-likelihood in Figure 5.5, then we see that there is no distinct peak of high scoring motifs present. This means that the motif is not very distinct from random motifs found in this data set and thus much harder to retrieve.

Table 6.9: Summary results: NIT

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`CCATGG`	7	15	1.47	1.68	51.79	1	4
`CTTATC`	7	18	1.87	1.78	50.89	4	46
`sCTTATCw`	7	18	1.43	1.39	61.80	1	54
`CGyGCCGCwC`	6	11	1.10	1.39	69.46	1	4
`wGATAAGrwn`	7	21	1.22	1.15	62.40	4	21
`kkCAGynyGrrT`	7	12	1.09	1.21	78.59	1	23
`rswGATAAGnwG`	7	15	1.11	1.09	60.64	11	22
`GCAGnryGrrTwCC`	7	13	1.02	1.16	84.63	1	24
`TnnCTTATCTnnkC`	7	10	1.06	1.08	60.16	10	20

for a legend of the table, see Table 6.5

HAP regulon

The consensus sequence where the Hap TF binds is reported as the rather short consensus `CCAAy`. Having a core of only four nucleotides means that it will be hard to find in this data set. This is illustrated by van Helden et al. [111] who also do not find any significant hexamer that resembles this consensus. Indeed, if we look for a motif of length 6bp, we also do not find any motif of six nucleotides that resembles the true consensus. Since the known motif is only 5bp long, we also try this motif length. Again, we do not even find the true motif in the list of all motifs found. However, if we look at the results in Table 6.10, we see that there are motifs found that contain the same core as the described binding site. This motif is not always the best, it is found within the top five, but it is only found in a very limited number of repetitions. For comparison we checked the number of occurrences of `CCAA` in this sequence set. It occurs 52 times without any mismatch on a total of 6376 possible sites in this data set. That the true motif is not found as statistically overrepresented is also showed in the results by van Helden et al. [111]. They find as possible motifs: `AGAGAGA` and `ATGGGGC`.

Table 6.10: Summary results: HAP

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`GCCTG`	8	14	1.69	2.03	56.82	1	7
`CTGCCs`	8	13	1.53	1.81	59.78	1	17
`nGTGGrCA`	7	16	1.27	1.42	59.62	1	8
`CCAAnnAs`	7	13	1.41	1.43	58.63	5	7
`ACCAATmAnA`	7	10	1.36	1.26	68.45	1	6
`ACmrkwrCmrCC`	7	8	1.10	1.20	77.34	1	1
`sACCAATmAnAm`	7	10	1.27	1.24	70.25	4	5
`TnsACCAATmAnAm`	8	9	1.16	1.13	75.27	1	4

for a legend of the table, see Table 6.5

PDR regulon

In the PDR family we expect to find a motif with a palindromic consensus sequence `TCCGCGGA`. Although this motif is found in all cases, it is never found as the best scoring motifs but it is present in the top three. The main indication that the lower ranked motifs are the true motifs are the fact that those motifs are found in almost all runs. However, there is something special about the best scoring motifs of length 12 and 14bp. These motif seem to partially overlap with the consensus of the true motif. This is confirmed if we look at the sites that are chosen as motif instances within the sequences. A minority of these sites do contain the consensus sequence of the known binding site. This means that the best scoring site is probably a false positive that partly overlaps and masks the true motif instances.

TUP regulon

The binding sites of the Mig1p zinc finger form the rather degenerated consensus sequence `kAnwwwwATsyGGGGw`. Since the consensus of the true motif is already degenerated it will be harder for the MotifSampler to pick up this signal and it is also hard to decide which motifs do resemble this consensus. In Table 6.12 we focus on the part `GGGGw` and its reverse complement `wCCCC`. For

Table 6.11: Summary results: PDR

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`GGCwCC`	7	12	1.52	1.81	53.66	1	28
`CGyGGA`	7	19	1.70	1.98	43.88	4	100
`nGnkCCrC`	7	17	1.20	1.46	59.89	1	6
`TCCGCGGA`	7	27	1.45	1.67	55.67	3	90
`CrnGCATwTA`	7	13	1.16	1.17	63.39	1	5
`nnTCCGCGGA`	7	28	1.25	1.42	61.53	2	84
`nynCGnrGATns`	7	16	0.97	1.13	70.01	1	2
`nTCCGCGGAnns`	7	22	1.13	1.28	63.77	3	55
`CnTnnmCrCmGAGA`	5	6	1.18	1.31	73.55	1	21
`nknTCCGCGGAnnn`	7	27	0.93	1.08	65.93	3	53

for a legend of the table, see Table 6.5

the consensus sequences of length 6 and 8bp do share the same core `CCCCrC` which is the combination and the reverse complement of the overrepresented hexamers `GTGGGG` and `GCGGGG`. For the other motif lengths it is not clear how reliable the results are since the number of similar motifs found drops below 20.

Table 6.12: Summary results: TUP

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`CCCCrC`	23	45	1.65	1.99	157.76	1	100
`mCCCCrCm`	21	37	1.34	1.59	153.52	1	63
`nnTnCCCCkC`	22	38	1.08	1.27	162.03	1	2
`nnCCCCrCnn`	22	35	1.13	1.29	156.03	3	15
`CsrnmysCnmnn`	22	33	0.87	1.06	176.28	1	2
`nGnGGGskrnky`	20	34	0.95	1.14	163.30	3	12
`nyCnTGnrnGnnnn`	22	37	0.83	0.99	172.54	1	4
`nGyGGGGnnnnykn`	19	42	0.75	0.95	170.13	2	5

for a legend of the table, see Table 6.5

YAP regulon

For the YAP regulon we get some mixed results as shown in Table 6.13. In none of the cases is the true motif found as the best scoring one. In the case of motif length 6bp we find part of the reverse complement at rank seven. In the other cases the consensus that closes matches the known consensus is `TTAnTAA` is found within in the top three and in significantly more runs than the best scoring motif. This consensus is palindromic and this explains why the nucleotide in the center does not tribute to the motif model compared to the palindromic part.

Table 6.13: Summary results: YAP

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`wTTGCG`	15	26	1.62	1.68	96.48	1	12
`mTTAGT`	14	27	1.77	1.62	86.58	7	24
`TGmGCrTG`	13	20	1.28	1.50	101.44	1	5
`GmTTAnTA`	12	27	1.52	1.40	99.38	2	23
`wkrnCyTGCA`	14	18	1.11	1.16	100.06	1	2
`nnTTAnTAAT`	12	40	1.24	1.09	99.78	2	47
`AGTnwGCAnATT`	11	18	1.17	1.12	110.01	1	3
`ATTAnTAAksAr`	14	35	1.11	0.99	107.65	3	30
`CnATnCnrnTmrTT`	13	20	0.96	0.92	111.40	1	1
`nTnATTAnTAATnn`	12	24	1.11	0.98	105.54	3	19

for a legend of the table, see Table 6.5

GAL regulon

A special case is present in the GAL regulon. The consensus of the binding site of Gal4p transcription factor is $\mathtt{CGGn_5Wn_5CCG}$ and is hard to detect since only seven out of seventeen positions in the consensus actually comprise the binding site. This is also reflected in the results summarized in Table 6.14. The important part of the binding site is formed by the two short palindromic sequences `CGG` and `CCG`. In most case neither of these short sequences is part of the topscoring motif, except in the case of motif length is 10bp. The motifs in which we find this short consensus are situated lower in the ranking. Although the MotifSampler retrieves motifs in which the true consensus is found it is hard to judge these results especially since we lost connection between the two sides with the short motif length. To have an idea on the co-occurrence of the two short sites we need to compare the alignment vectors.

Table 6.14: Summary results: GAL

consensus	#seq.	#inst.	cs	ic	ll	rank	#sim.
`CsGnGC`	6	12	1.40	1.80	47.34	1	18
`sCGnnCGs`	6	13	1.21	1.60	55.37	1	21
`CCGAAynG`	6	10	1.20	1.29	47.84	4	12
`nsCGCnCGGm`	6	13	1.01	1.28	55.40	1	9
`TrmkCCGnws`	6	11	1.06	1.14	55.33	2	2
`nCrCnCGkryGA`	5	11	1.07	1.27	57.71	1	19
`CnnTsnTCCGwr`	5	9	1.10	1.24	57.49	2	14
`CGGmnsnCTkTC`	6	8	1.12	1.28	57.36	3	11
`nGGmnnACTnTysn`	6	9	1.06	1.20	76.86	1	13
`wCGGmGsnCTnTys`	6	8	1.07	1.21	76.84	-	-
`CwsTsnTCCGwrnG`	6	10	0.99	1.15	65.99	2	32
`CGGnnnnswnnnnnCCG`	6	23	0.83	1.04	72.37	1	44

for a legend of the table, see Table 6.5

Since we know that the true consensus has a length of seventeen nucleotides, we also run the same data set again with the motif length set to 17. This gives us a much more satisfactory result. As can be seen in Table 6.14 the best motif found corresponds exactly to the known motif.

The reliability of this result can also be seen if we compare the histograms in Figure 6.2 of the log-likelihood scores for all the different motif lengths. For $W = 6$ and $W = 8$ the scores are more or less normally distributed. If we look at the distribution of the scores for motifs of length 17bp, this distribution shows a peak around low scores but, more importantly, there is a distinct peak a the higher scale of the scores and this peak nicely coincides with the scores of the motifs that match the true motif. The fact that there is no peak in distribution for smaller motifs indicates that for these lengths the set behaves more or less like a random data set.

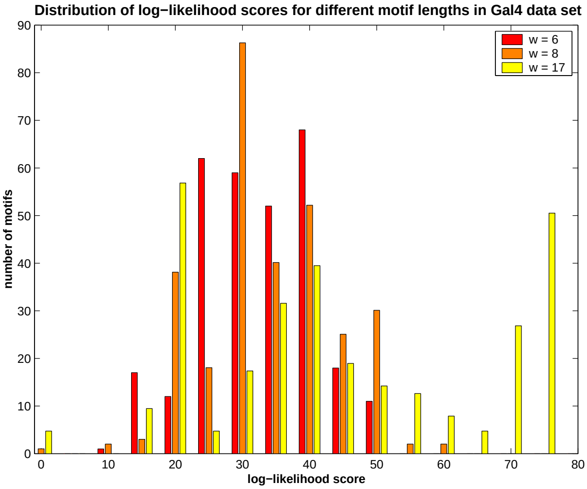


Figure 6.2: Histogram of the log-likelihood score in the GAL data set for the motif lengths set to 6, 8, and 17bp.

6.2.4 Discussion

Studying the performance of our motif finding algorithm on these ten yeast regulons, gives us sufficient confidence in the quality and robustness of our implementation. In most case we are able to retrieve the true motif models, not always as the best scoring, but mostly within the five best scoring. Even if the true motif is not found as the best scoring motif, it is still the motif that is most consistently found over the 100 runs. This means that if we find a best motif that is only found in a very limited number of runs, it is good practice to scroll down the list and see if there is a motif that is more consistently found over the different runs.

Concerning the definition of the motif length we learn from these examples that setting the motif length to 8bp, gives satisfactory results in most cases. In the cases where there is a very strong motif in the data set (MET, INO and GCN), this motif is found even when the motif length is set much longer than the true motif length. However, the example of the GAL regulon shows us that defining the correct motif length has profound impact on the detection process. Trying different lengths and looking at the distribution of the motif scores reveals in this example clearly what length is preferred.

Another remark, we should make here is that the most difficult data sets to analyze are the TUP and YAP regulons, which have been constructed from microarray data, while the others have been built from experimentally verified genes. The microarray data sets do contain more noise and are therefore inherently more difficult data sets.

6.3 Higher-order background models: A case study in prokaryotes[2]

As various bacterial genomes become available, one can build species-specific background models for all bacteria. Such a wealth of data also allows us to study the influence of these background models in detail [69]. This type of study is especially oriented to those working in the field of comparative genomics. If one likes to search for conserved motifs in distantly related species it becomes important to use the right background model for each sequence in the data set. We started by extracting all intergenic sequences of all of the 107 sequenced prokaryotic genomes from the Genbank annotation using the modules from *INCLUSive* (see also Chapter 8). For each gene in an organism, the intergenic region is defined and stored. This results in 107 sequence sets from which we create the specific higher-order background models.

6.3.1 Comparison of the background models

One interesting aspect of these 107 background models is to check how the transition matrices[3] of two species relate to each other. Therefore we create a scatter plot of the corresponding values in the transition matrices of two matrices. Here we show two particular examples. In Figure 6.3 we compare the transition matrices of *Esscheria coli* and *Salmonella typhimurium*. This figure clearly show that these two species are not only evolutionary closely related but also in terms of background composition. As a result, exchanging both background models when searching for motifs will only have a minor effect on the performance.

A completely different picture appears in Figure 6.4 where the background models of *E.coli* and *Streptomyces coelicolor* are plotted against each other.

---

[2] This work has been described in Trends in Microbiology [69].

[3] an example of a transition matrix is given in Table 5.1

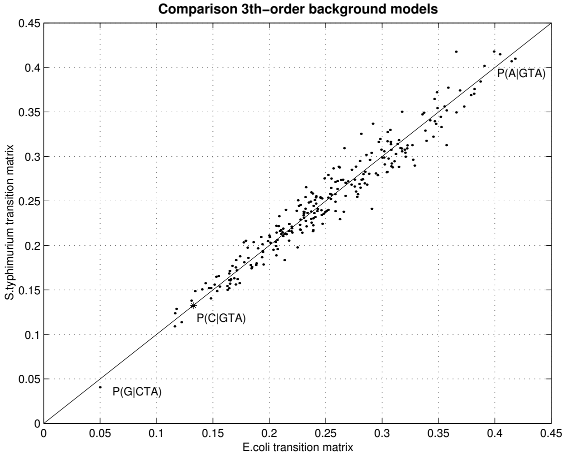


Figure 6.3: Comparison of third-order transition matrices from *E.coli* and *S.typhimurium* intergenic sequences.

Here it is clear that both organisms have a completely different background composition. The single nucleotide frequency already indicates that the *E.coli* intergenic region is AT-rich ([]) while the *S.coelicolor* intergenic region is predominantly GC-rich ([]). This effect seems even more expressed when looking at the third-order transition matrix. If we, for instance, look at the oligonucleotides `GTAA` and `GTAC`, we learn from Figure 6.4 that `GTAA` appears four times more in *E.coli* than in *S.coelicolor* while the opposite holds for `GTAC`. As we will show in the next paragraph, exchanging these two background models has a profound effect on the performance of the motif detection algorithm.

6.3.2 Influence of the background model on the motif detection

As a model system, we use the well-known σ^{54} transcription factor which is present in many bacteria. The σ^{54} factor controls several processes such as assimilation of ammonia, hydrogen uptake, nitrogen fixation, flagellar assembly and arginine catabolism. The factor recognizes a specific -12/-24 type promoter sequence[4]. The consensus of this recognized site is reported as `TGGCACG-n4-TTGCWn` [9]. The consensus consists of two conserved parts (7bp and 5bp) separated by smaller non-informative spacer of four nucleotides.

To test the influence of the background models two test sets are created. The first data set consists of 15 intergenic regions from genes in *E.coli* for which the binding site of σ^{54} factor was experimentally verified. For the second set 16 genes from *Pseudomonas aeruginosa* are chosen, of which some genes con-

---

[4] A -12/-24 promoter sequence means that the factor binds to sites located at respectively 12 and 24 upstream of the start of the gene.

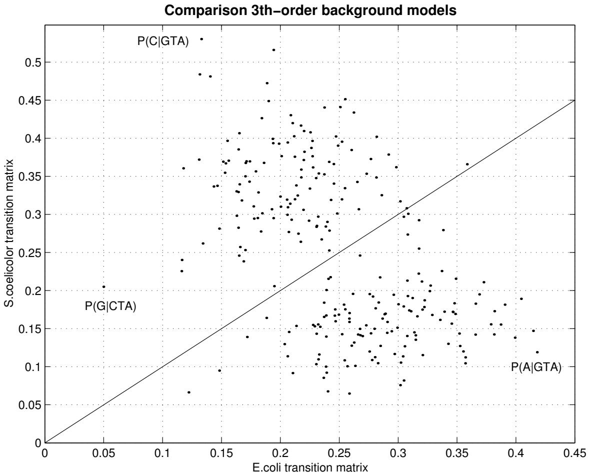


Figure 6.4: Comparison of third-order transition matrices from *E.coli* and *S.coelicolor* intergenic sequences.

tain a proven binding site. The other genes were selected because a screening of the genome revealed a putative binding site in their upstream region and those genes are either orthologues of the verified *E.coli* genes or their function is related to the known σ^{54} targets. The sequence length of the selected intergenic regions varies between 68 and 1253bp.

Our Gibbs sampling approach is applied to both data sets with two different motif lengths, 7 and 17bp. We also test single nucleotide and third-order background models from four different organisms: *E.coli*, *Lysteria monocytogenes*, *P.aeruginosa* and *S.coelicolor*. For each combination of sequence set, motif length and background model we run do 100 runs in which we search for one motif. To analyze the results we order each list of 100 motifs according to their log-likelihood score. We then check if the best scoring motif looks like the σ^{54} consensus and we report also how many times a similar motif is found in the list of 100 motifs.

Table 6.15 summarizes the results of the motif search in the *E.coli* data set and Table 6.16 shows the results of the motif search in the *P.aeruginosa* data set. For each parameter combination we give information on the highest scoring motif found and we also compare this with the known σ^{54} motif. The information given is the log-likelihood score, the consensus score, the number of instance of the motif and the number motifs in the list of 100 that are similar to the highest scoring motif.

In both cases the best results are obtained with the third-order species-specific background model. Using a background model from a different species has severe impact on the performance. When using a non-specific background model the results are totally wrong except for the case of the *L.monocytogenes* background model in the *E.coli* data set. When using the wrong background

Table 6.15: Highest scoring motifs found in the *E.Coli* data set.

Background Model	Consensus	Log-Likelihood Score		Consensus Score		Rank	#Instances	#Hits/100
	Highest	Highest	σ^{54}	Highest	σ^{54}	σ^{54}	Highest	σ^{54}
order 3, Length 7								
E.coli	TGGCACr	121.37	121.37	1.55	1.55	1	17	23
L.monocytogenes	TGGCACr	132.2	132.2	1.50	1.50	1	19	53
P.aeruginosa	TwmTTAA	122.71	/	1.15	/	/	47	0
S.coelicolor	nwnwnAw	145.64	/	0.70	/	/	138	0
order 0, Length 7								
E.coli	TGGCACr	124.16	124.16	1.55	1.55	1	17	21
L.monocytogenes	TGGCACr	131.43	131.43	1.55	1.55	1	17	52
P.aeruginosa	wTAAmAr	122.36	/	1.02	/	/	63	0
S.coelicolor	ATwwTnA	139.4	/	0.81	/	/	124	0
order 3, Length 17								
E.coli	TGGCACrAywmnTGCAT	167.48	167.48	1.10	1.10	1	13	37
L.monocytogenes	TGGCACrAywmnTGCAT	179.44	179.44	1.10	1.10	1	13	36
P.aeruginosa	wnwwwnAnnnmATwATw	151.02	/	0.51	/	/	63	0
S.coelicolor	wwwwnynkyrmwnnwnw	200.21	/	0.28	/	/	132	0
order 0, Length 17								
E.coli	TGGCACrAywmnTGCAT	177.17	177.17	1.10	1.10	1	13	23
L.monocytogenes	TGGCACrAnwnnTGCwT	195.17	195.17	1.08	1.08	1	14	57
P.aeruginosa	wwwwwnAksrmAnnwww	159.87	/	0.45	/	/	79	0
S.coelicolor	wwwwwynnnnmnwnwww	193.54	/	0.29	/	/	132	0

Table 6.16: Highest scoring motifs found in *P.aeruginosa* data set.

Background Model	Consensus	Log-Likelihood Score		Consensus Score		Rank	#Instances	#Hits/100
	Highest	Highest	σ^{54}	Highest	σ^{54}	σ^{54}	Highest	σ^{54}
order 3, length 7								
E.coli	CGCGsCk	126.41	/	1.36	/	/	47	0
L.monocytogenes	sCsnGsC	153.9	/	1.10	/	/	125	0
P.aeruginosa	wTTGGCA	111.43	111.43	1.41	1.41	1	16	15
S.coelicolor	nTkmwAw	121.08	/	0.08	/	/	66	0
order 0, length 7								
E.coli	ssCGGCs	140.69	/	1.27	/	/	82	0
L.monocytogenes	sCsCGsC	162.26	/	1.04	/	/	154	0
P.aeruginosa	TTTTnCk	110.66	109.06	1.28	1.42	2	23 (18)	4
S.coelicolor	wmTwwTT	118.46	/	0.89	/	/	56	0
order 3, length 17								
E.coli	yCGsGsCsknCssnnG	152.86	/	0.57	/	/	83	0
L.monocytogenes	CGsnGmnsnnnnssCss	194.07	/	0.36	/	/	192	0
P.aeruginosa	nTGGCACGsnwnTTGCT	142.92	142.92	1.09	1.09	1	11	37
S.coelicolor	nwwnkrnwyryynAwnn	178.09	/	0.38	/	/	71	0
order 0, length 17								
E.coli	ssnnGsCnnnsnCsssn	167.62	/	0.49	/	/	121	0
L.monocytogenes	ssnnsnsnsnnnnsnss	204.6	/	0.33	/	/	211	0
P.aeruginosa	TTsywnnynyTTTGnnn	134.1	124.65	0.76	1.23	4	17 (8)	14
S.coelicolor	nwwnTnnwnrnTwnTnr	158.52	/	0.42	/	/	?	0

model, the MotifSampler finds a motif model that is very degenerated but that still has a large log-likelihood score because a large number of instances is found. This can be explained by the fact the all segments in the sequences that partly match the motif model will be seen as completely different from the background.

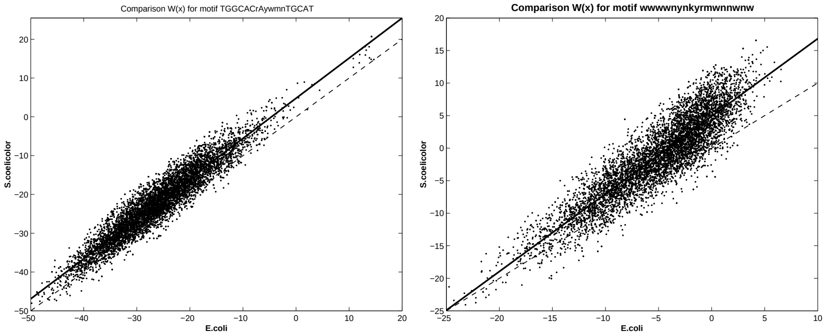


Figure 6.5: Comparison of log of the segment scores $W(\mathbf{x})$ for two different motifs `TGGCACrAywmnTGCAT` (left) and `wwwwnynkyrmwnnwnw`(right) and the *E.coli* and *S.coelicolor* third-order background model. The thick line is the regression curve fitted to the data and the dashed line indicates $y = x$.

To explain these results more in detail we look at the scores $W(\mathbf{x})$ of all segments $\mathbf{x}$ of length 17bp in the *E.coli* data set. $W(\mathbf{x})$ is calculated with the true motif, `TGGCACrAywmnTGCAT`, in the *E.coli* data set and also with the best scoring motif, `wwwwnynkyrmwnnwnw`, found with the *S.coelicolor* background model (see Table 6.15). As background models to compute $W(\mathbf{x})$ we chose the third-order background models of *E.coli* and *S.coelicolor*. We compare for each motif how the scores $W(\mathbf{x})$ evolve with respect to the different background models. This is visualized in Figure 6.5 where we plot $\log W(\mathbf{x})$. In the left plot of Figure 6.5 $W(\mathbf{x})$ is computed with the true motif and in the right plot the degenerated motif is used. In both cases, the scatter plots shows that there is a linear correlation between the scores from both background models. Therefore, we fit a regression line through the points. This linear regression curve is displayed as a full line in both plots of Figure 6.5. As a reference, we also show with the dashed line the one-to-one correspondence. We can see that in both cases the scores with the wrong background model are on average higher than those with the correct background model. This explains why we find more instances of the motif with the wrong background model. In case of the true motif the effect is moderate since the motif itself is rather strong. This is shown by the clear distinction between the high scoring segments and the cloud of other points. The effect is more pronounced when looking at the scores generated by the second motif. First of all is the range of the scores generated with this motif smaller since it is rather uninformative and looks similar to the background. Second, many instances have scores that are biased to larger values when using the wrong background model. This means that the score of much more segments $\mathbf{x}$ is assumed to rise above the background.

Therefore more segments will be selected as being motif instances.

6.4 Cell cycle in yeast

The final example concerns the analysis of real microarray data. We use the cell cycle microarray experiment conducted by Spellman et al. [97] which is considered as a benchmark data set in microarray analysis. Over the recent years, many researchers, who have been working on the analysis of common motif in sets of coexpressed genes, have used this data set to test their algorithm [18, 20, 113, 96, 102, 118, 122].

6.4.1 Clusters of coexpressed genes

The microarray experiment consists of 18 samples taken during two cell cycles in *S.cerevisiae*. The cell cycle is the process of cells growing and dividing to generate new cells that contain a copy of the DNA of the mother cell. One cell cycle consists of the succession of the four phases: **G1**, **S**, **G2** and **M**. In the G1 phase the cell grows untill it is large enough to start the cell division process. When the cell has grown sufficiently and has reached a critical size, its gene expression profile is altered to enter the synthesis phase. Once the cell has entered the S phase the cell cycle is started. First a bud, the future daugther cell, appears and this bud the gradually enlarges. At the end of the S phase the cell contains twice the number of chromosomes as in the G1 phase and now enters the G2 phase, that bridges the gap between DNA synthesis and the start of mitosis. Finally the cell enters the last phase, the M phase or mitosis. Here the daughter cell is separated from the mother cell and two cells genetically identical cell have emerged. The new daugther cell is much smaller than the mother cell and will now enter the cell cycle by growing during the G1 phase.

As a preprocessing step all genes with a low variance over their respective expression profiles are filtered out. This results in a set of 2467 genes with 18 measurements. *Adaptive Quality-based Clustering* (AQBC) developed by De Smet et al. [26] is used to define clusters of coexpressed genes. To retrieve clusters with highly coherent expression profiles, we need to limit the number of genes in a cluster and we thus set the quality criterium of AQBC to 0.99. The minimal number of genes in the cluster is set to be ten. This results in a set of 38 clusters. Some of those clusters show a very specific expression profile that can be related to the different phases in the cell cycle, while others look more like noise. We select the four clusters for further analysis because of their specific profile that relates them to the cell cycle. The expression profiles of these selected clusters are given in Figure 6.6. The first three clusters with numbers 3, 4, and 28 in Figure 6.6 exhibit a behavior that more or less corresponds to the periodic cycles in the experiment. The last cluster, with number 24, consists of genes that are highly over-expressed at the beginning

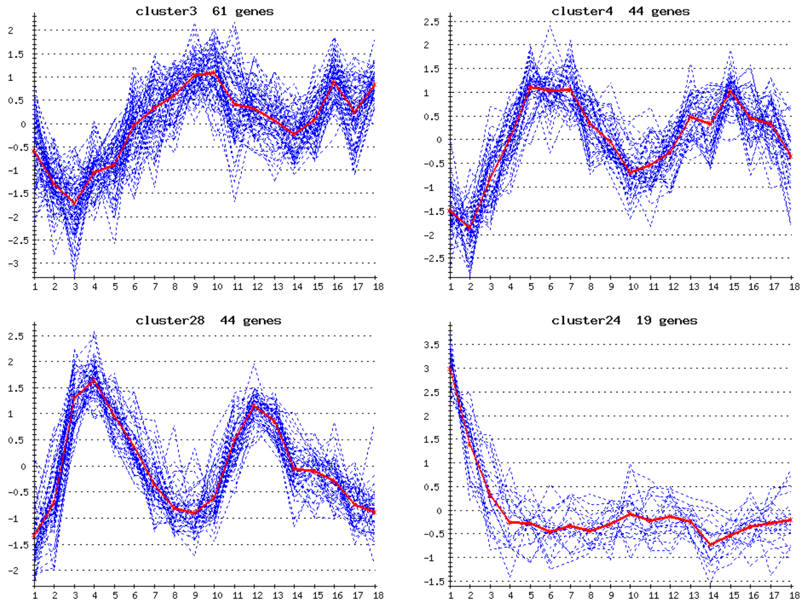


Figure 6.6: Selection of four clusters of coexpressed genes found with AQBC in the yeast cell cycle data set from Spellman et al. [97]. Measurements were done at 18 time points during two cell cycles. For each cluster, the normalized expression profiles of the genes in the cluster are plotted. In thick line the average profile of the cluster is plotted. The first three clusters show a periodic behavior that coincides with the cell cycle. The fourth cluster contains genes that have a high expression at the first time point and that are then switched off.

of the experiment and are then switched off after three time points.

6.4.2 Motif finding

To search for over-represented motifs in each of these cluster, we extract the upstream region using `RSA-tools` [112]. For each gene we select a maximum of 800bp upstream. If necessary, the sequence is truncated such that their is no overlap with any preceding gene. The sequences are used in this format without any modification like purging or masking low complexity regions Höhl et al. [44]. On each of these data sets we run the MotifSampler. We choose to try out the MotifSampler with the length of the motif set between 5 and 17bp, such that we cover the range of known binding sites in yeast. To make sure that we extract the stronger motifs first, we choose a lower prior equal to 0.2. The background model is the third-order model from the yeast intergenic sequence, which we have also used in the case study with the ten regulons in Section 6.2. In each run we search for three distinct motifs. After 100 repetitions, we rank the 300 motifs found according to their log-likelihood score.

For the evaluation of the retrieved motifs we compare them first with the results described by Spellman et al. [97]. Next we compare all retrieved motifs with the known motifs stored in SCPD (`http://sigma.cshl.org/`) and in TRANSFAC [117]. We also look at the results of other motif finding programs. Especially, the motifs found with AlignACE by Tavazoie et al. [102] in the mitotic cell cycle experiment of Cho et al. [22] who used Affymetrix oligonucleotide microarrays.

Cluster 28

Let us start with the most pronounced periodic cluster first. The profile in this cluster shows a first peak at the fourth time point and another one at the 12th time point. These peaks correspond to the transition from G1 to S phase in the cell cycle. Late in G1 two transcription factors, MBF and SBF, are activated by the protein kinase complex *Cln3p-Cdc28p*. The two transcription facors bind respectively to the *MluI* cell cycle box (MCB) and the *Swi4/6* cell cycle box (SCB). It is expected that most genes in this cluster are regulated through these two factors.

If we look at the best scoring motif for each of the different motif lengths, we see that the same consensus sequence is found with a very high consistency. The topscoring motif is very strong in this cluster, because it is found in over 90% of the runs for each of the motif lengths. The consensus `ACGCGT` corresponds perfectly to the consensus of the expected MCB.

Table 6.17: Best scoring motifs in cluster 28

consensus	#inst.	ll	#sim.
`CGCGT`	102	277.89	75
`rACGCG`	82	277.07	100
`rACGCGT`	96	305.67	100
`rACGCGTn`	89	296.11	100
`wrACGCGTn`	84	316.20	100
`rwrACGCGTn`	83	330.30	100
`nnnnACGCGTn`	81	337.96	100
`nnnnACGCGTnw`	79	334.52	100
`nnnnnACGCGTyw`	70	315.34	100
`nnnnnrACGCGTnw`	63	305.99	97
`nnnnnrACGCGTywn`	60	300.63	94
`wnAnnnnACGCGTnwn`	49	290.42	97
`nnwnAnnnnACGCGTnw`	49	298.44	85

Comparison of the long list of retrieved motifs with those in SCPD and TRANSFAC reveals that also the other motif, SCB, is found that is very specific for the cell cycle. Although, motif is not always found among the best scoring motifs, it is found it with a certain consistency. One of the reasons this motif is found at a lower frequency is that the consensus of SCB, `CsCGAAA`, partially overlaps the consensus of MCB. Since MCB is a very strong motif in this set, some of the instances of the SCB motif can be masked by instances of the MCB motif. It will be thus more difficult to pick the remaining instances. This is confirmed if we compare the instances found in both cases.

Cluster 3

The largest cluster in our analysis contains 61 genes. Visual inspection of the profiles of the genes in the cluster gives the impression it is also the most noisy cluster. The expression of the genes in this cluster peaks in the M phase of the cell cycle. Linking to the work of Tavazoie et al. [102], the cluster shows a similar profile as cluster 7 in their article.

Table 6.18: Best scoring motifs in Cluster 3

consensus	#inst.	ll	#sim.
`GACTC`	19	97.99	1
`TAAGTT`	23	113.98	2
`AAAATTT`	41	152.91	2
`AAAATTTT`	40	148.07	3
`wTTATTwTT`	30	151.30	2
`TTTTTTTTTT`	40	167.72	65

Unlike in the previous example, we do not find a strong motif. In Table 6.18 we list the highest scoring motif found for the lengths 5 to 10bp. The first thing, we notice, is that the topscoring motifs are only found in one or two runs, except for the motif `TTTTTTTTTT`. What we do find if we look for motifs that occur in more runs, is long stretches of `A`'s or `T`'s, which do not carry any regulatory function. This is somewhat disappointing, we therefore try to change the parameters of the MotifSampler to find out if we are not missing any information. Again, we do only find stretches of `A`'s or `T`'s. To further cross-validate our results, we also perform *oligo-analysis* with the sequences in this cluster. This reveals that the most frequent oligos are `tgaaaaat` and `aaaattt`. The first motif corresponds to the RRPE motif found in cluster S12 in the article of Hughes et al. [47], where they apply AlignACE on a large set of functionally related genes. The second one corresponds to the best scoring motif that we also find with length 7 and 8bp, but this motif was not very strong in our tests.

Cluster 4

This cluster also shows an average profile that coincides with the periodicity of the cell cycles. Here the expression peaks occur at the transition from S phase to G2 in the cell cycle, which is slightly later than the genes in cluster 28.

Tavazoie et al. [102] showed in their analysis that they found two significant motifs in this cluster that had not been described before. They refer to them

Table 6.19: Interesting motifs found in Cluster 4

consensus	#inst.	ll	rank	#sim.
`TAAACAA`	33	134.10	2	24
`AyAAACAA`	43	151.14	1	32
`AAAyAAACA`	33	149.93	3	38
`kwCGTGT`	32	146.92	1	46
`kTCGTGTn`	32	151.97	2	27
`nTTTCGnGT`	23	133.95	4	13

as M14a, `TTTsGykT`, and M14b, `TGTTTsTT`. Our analysis reveals similar motifs when searching for motifs of length 7, 8 and 9bp. These motifs are summarized in Table 6.19. The first motif is the reverse complement of M14b. The second motif resembles M14a. However, if we search for longer motifs these two motifs are not found any more and the MotifSampler only retrieves longer stretches of `A`'s or `T`'s.

Cluster 24

The fourth cluster contains 19 genes that have a specific profile that does not exhibit any periodic behavior. The expression of these genes is very high at the start of the cycle but is turned off for the remainder of the cycle. Comparing with the work of Tavazoie et al. [102], we do not directly find a corresponding cluster with a similar profile.

For the motif lengths from 5 to 9bp, the best scoring motifs exhibit a large degree of similarity as is shown in Table 6.20. The core of this motif can be written as `ATGAAAC`. A similar motif is also found when searching for longer motifs, although it will not be the best one. Comparing these motifs to the motifs in SCPD reveals that `ATGAAAC` corresponds to the consensus of the *STE12* factor. This factor is involved in pheromone response, cell type specific transcription and signal transduction. There is even more evidence about this motif being a real motif. If we look at the genes in SCPD that are known to be regulated by STE12, we find that one of those genes, *YFL026*, also belongs to this cluster. Finding an overrepresented motif that is experimentally verified to regulate a specific gene in the cluster, is to our opinion strong proof that this factor influences the genes in this cluster.

Table 6.20: Best scoring motifs in Cluster 24

consensus	#inst.	ll	#sim
`AAACA`	13	43.28	15
`TGAAAC`	20	62.11	14
`ATGAAAC`	17	68.02	16
`ATGAAACn`	15	72.15	23
`TGAAACrAw`	14	82.56	14

If we analyze the larger motifs we also find a strong motif that is shared among the different lengths. Although this motif has only seven instances in this set of 19 sequences, the high number times of finding this motifs indicates that it is a consistent optimum. In Table 6.21 we do not only give the best scoring motif but also some examples of motifs that are similar to the best scoring but are more conserved. Comparison of the motifs with the sites in SCPD reveals a partial similarity of the first part of the motif with the binding sites of the TATA-binding protein (TBP). The second part `CAGATA` resembles the NIT2 motif `TATCTm` stored in TRANSFAC. Especially the two better conserved examples are very similar to these two boxes. The seven genes that have this motif in their upstream region are: FUS2, ERG24, GYP7, KTR2, RPS16B, FIG1 and RUS161.

Table 6.21: Most consistent motifs of 14 to 17bp in Cluster 24

consensus	#inst.	ll	rank	#sim
`TATATGkGTCAsAT`	7	67.87	3	9
`TATATGkGTCAsATA`	7	71.09	1	15
`CATrTATGkGTCAsAT`	7	75.96	2	9
`CATrTATGkGTCAsATA`	7	79.63	1	13
`ATATATGnGTCAGATA`	7	62.07	-	-
`nATATATGnnTCAGATA`	6	64.82	-	-

6.5 Conclusion

To evaluate on a larger scale the performance of our algorithm we apply the MotifSampler to several data sets in different organisms.

In the first tests we use the G-box data set and a set of random genes to evaluate the robustness of our motif finding algorithm. First we show how the data set that is used to create the background model influence the motif detection process. This analysis shows us that using a carefully created independent background model significantly increases the performance. Next, we assess the accuracy of the predicted the motif instances by comparing them with the true motif instances. The tests show that the higher-order background models slightly increase the accuracy compared to the single nucleotide model. Finally, the robustness of the algorithm is tested under increasing noisy conditions. Again we can show that the use of an intergenic background model has a positive effect on the motif detection process. Even if less than half of the sequences in the data set contain the true motif our algorithm is able to detect these true motif instances.

In the second set of tests we work with ten regulons from yeast. These regulons are well studied and help us greatly in assessing the performance of our motif finding algorithm. These tests with our algorithm give very satisfactory results. In most case, we are able to detect the known motifs in these sets either as the best scoring motif in the set. If the motif is not found as the best scoring motif, it is often found in the top five and occurs in significantly more runs than the top scoring motif.

We also study the usage of higher-order background models in prokaryotes. As model system we studied the σ^{54} element in *E.coli* and *P.aerugonisa*. This study clearly shows how the usage of the wrong background model seriously deteriorates the results. Another consequence of this study lies in the application of Gibbs sampler for phylogenetic footprinting where a species-specific background model could improve results when aligning sequences from different species.

In the last example we analyze the benchmark microarray experiment of the cell cycle in yeast. With adaptive quality-based clustering four clusters are identified that show a profile which seems to be cell cycle specific. In two of these clusters we identify motifs that are specific for these clusters and of which the function is also related to their expression profile. In one cluster we find two motifs that also have been detected with AlignACE. In the last cluster no significant motif is found that has any regulatory function.

CHAPTER 7

Searching for known transcription factor binding sites[1]

Sometimes it might be useful to find the potential binding sites of a known transcription factor in one or more sequences. In this chapter we introduce two methods to accomplish this search. The first method, ***MotifLocator****, is based on the classical method of scoring a sequence with a position weight matrix. Our implementation uses a background corrected position weight matrix to score the segments in a sequence. The second method,* ***MotifScanner****, is based on the probabilistic sequence model introduced in the MotifSampler to estimate the number of motif instances in a sequence. After a short description of the algorithms, we compare both methods and discuss their advantages and disadvantages with respect to their parameters. Next, we apply these methods to two large sets of promoter sequences in yeast and human. We show how the sequence composition influences the motif detection process and we also show distribution patterns of where retrieved instances are found. Finally, a methodology based on a binomial statistic is introduced to compute the statistical significance of the number of motif instances in a set of coregulated genes. Here we use the yeast regulons and yeast cell cycle clusters to illustrate that the method is applicable to real life problems.*

7.1 PWM-based approach to locate motif instances

The classical method to find instances of a known motif model is to transform the matrix of counts to a position-specific weight matrix or PWM. In Section 3.3

---

[1] This work has been partially published in the TOUCAN paper in Nucleic Acids Research [1].

we introduced the basic method to score a sequence with a PWM which is rather straight forward. Each segment in the sequence is scored by adding up the scores of individual bases in the segment. The segments that have a score higher than a predefined threshold are then selected as being motif instances.

Instead of scoring directly with a PWM, we choose to apply the scoring scheme from the MotifSampler. Given the PSFM, $\boldsymbol{\Theta}$, and the background model, $\mathcal{B}_m$, we compute the score of the segment being generated by the motif model and compare this with the score of the segment being generated by the background model. This corresponds to the predictive update formula of the Gibbs sampler as stated in Eq. 5.4. However, in this case we only work with the logarithm of this equation. Thus, for each segment $\mathbf{x}$ of length W in the sequence S, we compute the corresponding score as

$$\begin{aligned} W(\mathbf{x}) &= \log\left(\frac{P(\mathbf{x}|\boldsymbol{\Theta})}{P(\mathbf{x}|S,\mathcal{B}_m)}\right) \\ &= \sum_{j=1}^{W}[\log(\boldsymbol{\theta}_j^{b_j}) - \log(P(b_j|S,\mathcal{B}_m))]. \end{aligned} \tag{7.1}$$

Next, we like to apply a threshold to these scores to extract the relevant instances. To be able to define a reliable threshold over different motif models we need to normalize the scores. The preferred method is to rescale the scores such that they have values between 0 and 1. First we compute the minimal and maximal value of $W(\mathbf{x})$ over all possible segments $\mathbf{x}$ as

$$\begin{aligned} W_{\text{min}} &= \min_{\mathbf{x}} W(\mathbf{x}) \\ W_{\text{max}} &= \max_{\mathbf{x}} W(\mathbf{x}). \end{aligned}$$

Once minimum and maximum are found, the scores $W(\mathbf{x})$ are rescaled as

$$\bar{W}(\mathbf{x}) = \frac{W(\mathbf{x}) - W_{\text{min}}}{W_{\text{max}} - W_{\text{min}}}. \tag{7.2}$$

This results in a distribution of scores over the full sequence set, with scores between 0 and 1. On these scores we can impose a threshold and select all instances with a score higher than this threshold.

7.2 Using the probabilistic model to locate motif instances

In the previous section, we presented a slightly modified version of the classical PWM scoring scheme. However there is also another possibility to find the number of instances of a motif in a sequence. Within the core of the MotifSampler we use the probabilistic sequence model to estimate the number of times an instance of a specific motif model is found in a sequence. It thus makes sense to use the same methodology to find the number of instances of a known motif in a given sequence. The core of the methodology was already introduced in Section 5.2. Here we summarize the main results.

The goal is to compute the expected number of instances of a motif model $\mathbf{\Theta}$ in a sequence S given the background model $\mathcal{B}_m$. The expected number of instances can be computed as

$$\begin{aligned} E_{(S,\mathbf{\Theta},\mathcal{B}_m)}(Q) &= \sum_{c=0}^{\infty} c \times P(Q=c|S,\mathbf{\Theta},\mathcal{B}_m) \\ &= \sum_{c=0}^{\infty} c \times \frac{P(S_k|Q=c,\mathbf{\Theta},\mathcal{B}_m)P(Q=c|\mathbf{\Theta},\mathcal{B}_m)}{P(S|\mathbf{\Theta},\mathcal{B}_m)}. \end{aligned}$$

In this equation there are three terms which need further elaboration. The denominator $P(S|\mathbf{\Theta},\mathcal{B}_m)$ serves as a normalizing constant. The first part the numerator is the probability that the sequence is generated given the motif model $\mathbf{\Theta}$, the background model $\mathcal{B}_m$ and the fact that the number of instances is known. This can be easily computed with Equation 5.9. The final term is the prior probability of finding c copies. Using this equation, we introduced a new parameter in the algorithm namely the prior γ_1 to construct the complete prior distribution with the terms $P(Q=c|\mathbf{\Theta},\mathcal{B}_m)$. The parameter γ_1 will be the most most important parameter of our algorithm.

Given the previously defined formulas we can find the number of instances of $\mathbf{\Theta}$ in sequence S using the procedure outlined in Program 6. As stated before, the algorithm as it is described here only needs one parameter to be set: the prior γ_1. In the next sections we elaborate on the effect of this parameter on the performance of the algorithm.

Program 6: Basic procedure of the MotifScanner.

```
1. Score each segment $\mathbf{x}_l$ in $S$ with the motif model $\mathbf{\Theta}$.
2. Score each segment $\mathbf{x}_l$ in $S$ with background model $\mathcal{B}_m$.
3. Initialize prior distribution $[1-\gamma_1, \gamma_1]$.
4. Compute $P(Q=0|S,\mathbf{\Theta},\mathcal{B}_m)$ and $P(Q=1|S,\mathbf{\Theta},\mathcal{B}_m)$
5. While $P(Q=i|S,\mathbf{\Theta},\mathcal{B}_m) > \epsilon$
    (a) augment $i$
    (b) update $P(Q=c|S,\mathbf{\Theta},\mathcal{B}_m)$ for $c = 0 \dots i$.
6. Compute the expected number of instances as $E_{(S,\mathbf{\Theta},\mathcal{B}_m)}[Q]$.
7. Select the $Q$ best scoring positions as motif instances.
```

7.2.1 Computational complexity

To quantify the computational complexity we should look at the different steps in the algorithm. To estimate the number of instances in one sequence, we need to score the sequence once with the motif model and once with the background model. Both steps scale linearly with the length of the sequence. This is also illustrated in Figure 7.1, where the average computation time as a function of the sequence length is plotted. For different sequences ranging from 100 to 2000bp, we measured the time to estimate the number of instances of one motif in a sequence.

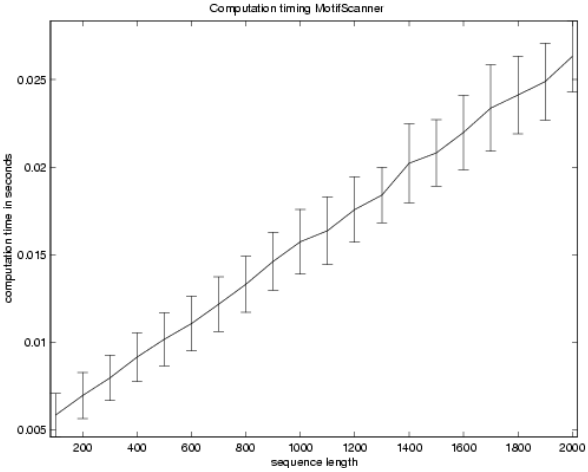


Figure 7.1: Average computation time of MotifScanner in function of an increasing length of the input sequences. The time given is the time needed to screen one sequence with one motif.

7.3 Comparison of MotifLocator and MotifScanner

In the previous sections, we introduced two methods to search for instances of known motifs in a given DNA sequence. Although both methods have the same goal, there are some significant differences. While the MotifLocator looks at each site individually, the MotifScanner takes the context of the sequence in which the motif instances are hidden into account.

7.3.1 Threshold vs. prior

Both algorithms need one specific parameter to be defined. In MotifLocator, the threshold should have a value between zero and one. The closer to one the less instances are selected as being motif instances. The prior γ_1 in the MotifScanner is also a value between zero and one, but here it is a probability instead of threshold. In this case the higher the prior the more likely it will be to select an increasing number of instances. It is thus interesting to study the behavior of both algorithms with respect to their parameter.

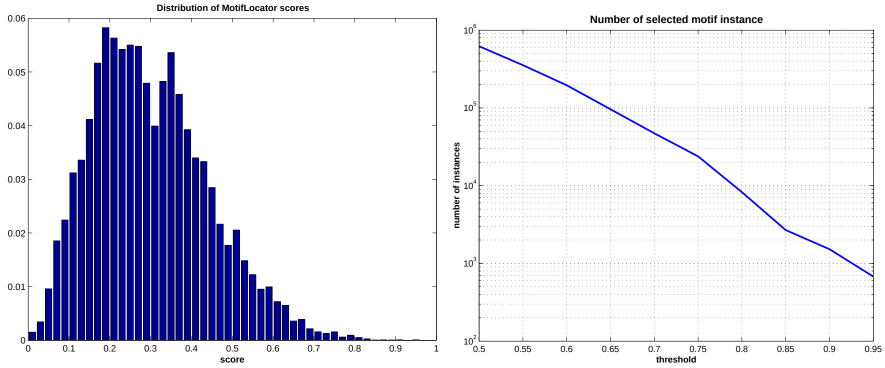


Figure 7.2: Distribution of scores $\bar{W}(\mathbf{x})$ (left) with the MCB motif in the yeast upstream regions and the number of instances found above a given threshold (right). The curve on the right shows that number of retained instances will increase exponentially with a decreasing threshold.

To illustrate this we screen the full upstream sequence set in yeast for the MCB motif with the MotifLocator. The normalized scores with values between zero and one are distributed as shown in Figure 7.2. To select motif instances, all segments with a score higher than a predefined threshold are chosen. The left figure shows how many instances are selected above the given threshold. The number of instances found with our PWM-based MotifLocator grows exponentially with decreasing threshold.

A similar test is performed with the MotifScanner implementation on the same data set. The curve in Figure 7.3 shows a different kind of behavior compared to the MotifLocator results. As expected, the number of instances increases with an increasing prior, but the increase is no longer exponential. The increase in number of instance is the sharpest between 0.0005 and 0.1. Looking at the actual number of instances found, we see that for a prior of 0.1 or 0.2 the number of instances corresponds to a threshold of 0.9.

7.3.2 Influence of the sequence length

Using the probabilistic sequence model assumes that the motif instances are hidden in background noise. Practically, this formulation implies that if the sequence gets longer, the motif instances will be buried more in background noise. It will thus be harder to detect the motif instances, unless they are significantly different from the background and have a strong match with the motif model. Since the PWM approach scores each segment individually, the

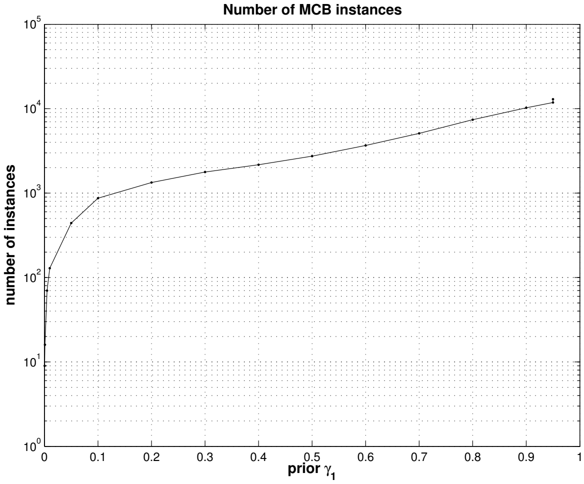


Figure 7.3: Number of instances of the MCB motif found with MotifScanner in the yeast upstream regions in function of increasing prior γ_1.

number of instances increases monotonously with the length of the sequence. On average, this increase in number of instances is linear. However, the number of added instances depends on the relation between the local sequence content and the specificities of the motif model. In contrast, the number of instances found with the MotifScanner does not increase monotonously with the sequence length. Depending on the motif model, background model and prior the number of instances detected changes together with the length of the sequence.

To illustrate the influence of the sequence length we create 20 related sequence sets. We start from a random selection of 4000 genes in human. From these genes, we select 100bp just upstream of the start site to create the first set. Next, we extend the selected upstream region each time an extra 100bp to create the next sequence set. This results in 20 sets where the sequence length increases from 100 to 2000bp in steps of 100bp. We screen these 20 data sets with three motifs from TRANSFAC: SP1, E2F and TBP. MotifLocator was tested with a threshold of 0.85 and 0.9 and MotifScanner with a prior of 0.2 and 0.5.

In Figure 7.5 we show the number of instances found of the three different motifs: SP1, E2F and TBP. Analysis of these tests shows that each of the three motifs exhibits a different type of behavior. For the SP1 sites there are two trends. The first one is that the number of instances grows significantly when changing the threshold of the MotifLocator from 0.9 to 0.85. On the other hand, the change in number of instances is only limited when augmenting the prior from 0.2 to 0.5 for the MotifScanner. The increase in number of instances in the longest sequence is more than a factor five for the MotifLocator. The second observation is that the number of motifs found with the MotifScanner increases rapidly in the shorter sequences but settles to a more

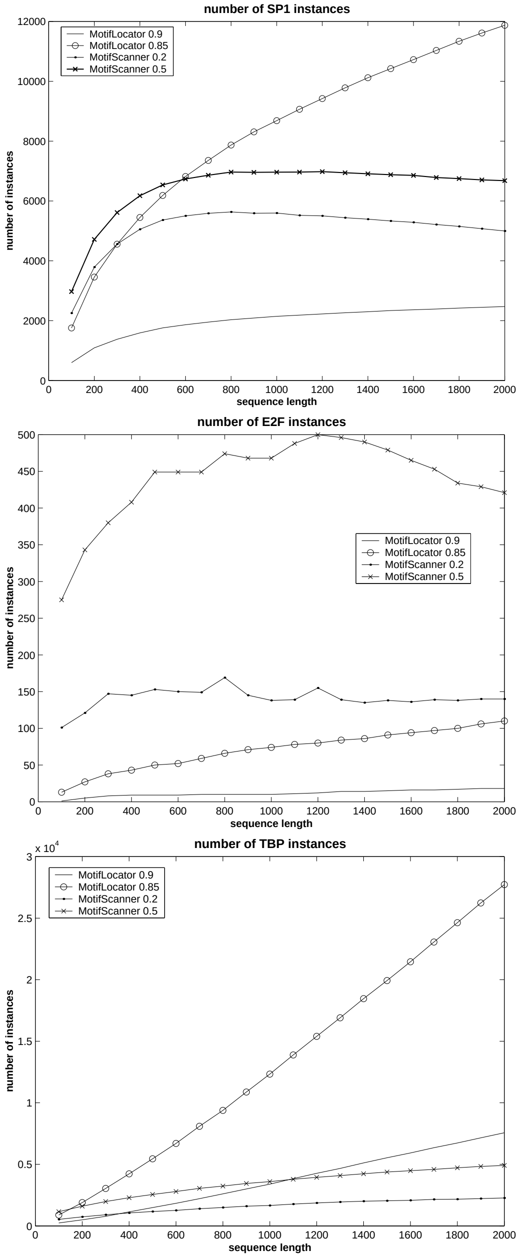


Figure 7.4: Number of instances found in the human upstream sequences as a function of the sequence length for three different motifs: SP1 (top left), E2F (top right) and TBP (bottom). Each motif model exhibits a specific behavior depending on the composition of the motif model.

or less constant number for the sequences longer than 600bp. In the E2F case only a very limited number of instances is found with the MotifLocator. The number goes from 2 to 20 for a threshold of 0.9. With a threshold of 0.85 approximately fives times more instances are found, ranging from 10 to 100. However, in this example these numbers are much lower than the number of instances found with the MotifScanner. When using a prior of 0.2, the number of instance found stays constant around 140 instances. When the prior is increased to 0.5, the number of instances increases with a factor three to four. In the last example we search instances of TBP, a short AT-rich motif. Here the number of instances increases monotonously with both MotifLocator and MotifScanner. In this case, the number of instances found with MotifScanner in the longer sequences (1200 to 2000bp) is for both priors lower than the number of instances found with the MotifLocator.

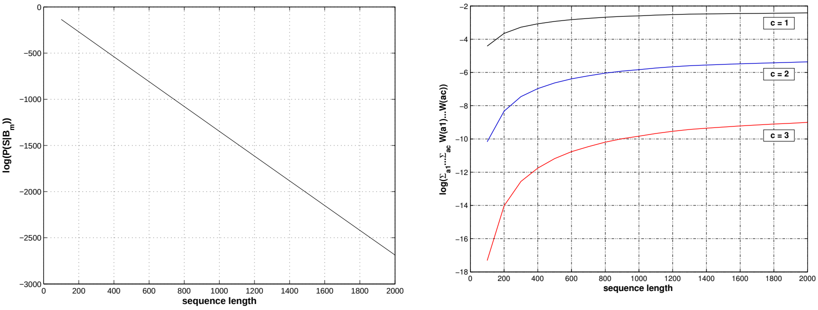


Figure 7.5: Effect of the sequence length on $P(S|Q = c, \mathbf{\Theta}, \mathcal{B}_m)$. (left) The score $\log\left(P(S|\mathcal{B}_m)\right)$ decreases linearly with the increasing sequence length. This means that the probability decrease exponentially. (right) Evolution of $\log\left(\sum_{a_1}\cdots\sum_{a_c} W(\mathbf{x}_{a_1})\ldots W(\mathbf{x}_{a_c})\right)$ as a function of the sequence length for c equal to 1, 2 and 3. For larger c the scores decrease while they increase with the sequence length.

To explain the effect of the sequence length in MotifScanner we should look at the formulas. If we use Equation 5.5, the probability of a sequence having c motif instances given the motif model $\mathbf{\Theta}$ and the background model $\mathcal{B}_m$ is computed as

$$P(S|Q = c, \mathbf{\Theta}, \mathcal{B}_m) \propto \underbrace{P(S|\mathcal{B}_m)}_{I} \underbrace{\sum_{a_1}\cdots\sum_{a_c} W(\mathbf{x}_{a_1})\ldots W(\mathbf{x}_{a_c})}_{II}. \qquad (7.3)$$

In this formula there are two parts: (I) the probability of the sequence being generated by the background model and (II) the contribution of all combinations of c motif instances in the sequence. It is clear that in both terms the sequence length plays a role, but the effect is not the same. This is illustrated in Figure 7.5 where the average of the logarithm of the two parts in the equation are shown. The logarithm of the background score decreases linearly with an increasing sequence length, the longer the sequence the smaller becomes the term $P(S|\mathcal{B}_m)$. This figure also shows how the contribution of all possible combinations of c instances changes in function of the sequence length and this for three values of c. Interesting to see is that the curves display a plateau as the sequence length increases. This effect implies that the longer the sequence is, the smaller the relative contribution of all possible combinations of a motif becomes.

7.3.3 Discussion

Now that we have gained some insight in the behavior of both algorithms, it is time to compare their advantages and disadvantages. The main advantage of MotifLocator is its straight-forward methodology and implementation. The disadvantage is that the number of instances grows exponentially with a decreasing threshold. The tests show that a slight change in the threshold can have a profound effect on the number of instances retrieved. MotifScanner does not suffer from this problem. In the range from 0.1 to 0.9 of the prior shows a steady increase in the number of instances but the same increase in number of instances is realized with MotifLocator by changing the threshold from 0.9 to 0.8 (as shown in Figures 7.2 and Figures 7.3). This means that the MotifScanner is less influenced by the change in parameter then the MotifLocator. The effect of the sequence length is however less predictable and largely depend on the type of motif. The probabilistic sequence model is one of the strengths of the MotifScanner but it contains also its main weakness. The influence of the sequence length on the model could have some undesirable effects especially if the sequence is very short. When this is the case, it is possible to extract motif instances from the sequence that only look very distantly like the motif model. Since the sequence is short, the noise level is rather limited and it is easier for a segment to rise above the background noise. The examples of SP1, E2F and TBP show that each type of motif displays its own characteristic behavior and there does not exist one parameter definition that covers all possibilities.

Which method gives the most satisfactory results depends on the question asked. If one likes to find all possible instances of a motif within a sequence it is better to use MotifLocator with a lower threshold. If one, however, is not interested in all possible instances but rather in those instances that rise above the background noise and really match the motif model, it is better to use the MotifScanner with preferably a low prior γ_1.

7.4 Large scale promoter analysis

The recent availability of completed genomes has triggered the interest of many researcher to datamine this large amount of data to gain better insight in the chromosomal organization of genes. We also try to contribute to this analysis by studying the promoter regions. Therefore, we test in the next paragraphs our motif detection programs on two large data sets of upstream or promoter regions, one in yeast and one in human. We study for each of these sets how the nucleotide distribution influence the motif detection process and how the motif instances themselves are distributed over the upstream region. As motif models we use respectively the matrices from SCPD, *Saccharomyces cerevisiae Promoter Database* [124], and the vertebrate models from TRANSFAC [117].

7.4.1 Yeast upstream sequences

Data sets

The primary data set consists of 6450 sequence of maximal 800bp upstream of the translation start of all the genes in *S.cerevisiae*. The selected sequences do not overlap with any preceding ORF and were downloaded with RSA-tools [112]. The yeast motif models are downloaded from the specialized database SCPD. The downloaded file contains the matrices of 24 different motif models. The length ranges from 5 to 20bp.

Nucleotide composition of the upstream regions

Let us first look at the composition of these upstream regions. We align all sequences on the right site (the translation start site of the gene) and compute the frequency of A,C,G or T at each position from -800 to 0. The measured nucleotide composition is shown in Figure 7.6. At first we should point at the peak of A's at position 0 that corresponds to the first position of the translation start site. Next, this plot nicely shows the difference in AT-content and GC-content in the upstream region. The proportion of each nucleotide remains almost constant over the region -800 to -200bp. Around position -200 the level of T first increases slightly compared to the level of A but while moving closer to TLS there is a clear increase of the level of A. The GC-content remains almost constant over the full region. This constant composition implies that for this region the same background model can be used without any strange side effects.

Localization of motif instances

Recently, Hampson et al. [39] have explored the yeast promoter regions and discovered special distribution patterns of oligonucleotides in these regions. They found that some words occur more frequently in a region close to the start site of the gene. We decide to perform a similar analysis on the full set of upstream sequences with the motifs from SCPD. Therefore, we use MotifScanner with a prior γ_1 set to 0.2 to screen these upstream sequences. The number of instances varies from 391 instances of GAL4 to 3437 instances of TBP. For each motif, the position of the corresponding instances relative to

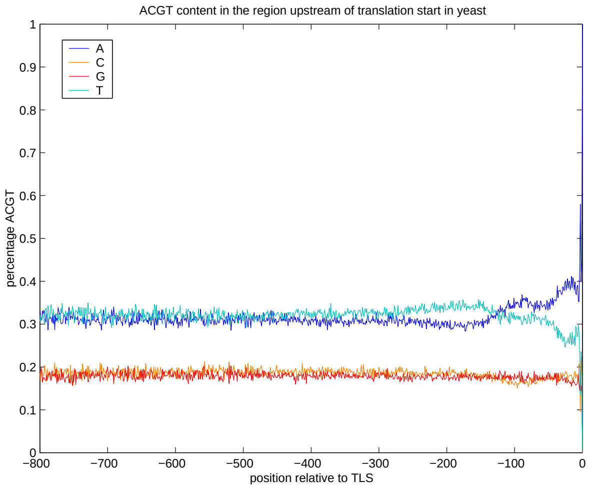


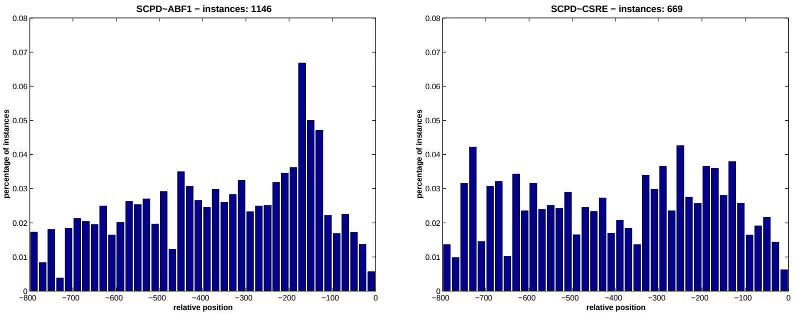
Figure 7.6: Distribution of `A`, `C`, `G` and `T` over the upstream region of all genes in the yeast genome. The upstream region are aligned at the start site of the gene (indicated by the `A`at position 0). At each position the number of `A`, `C`, `G` and `T` is counted and these are values are normalized.

the start site is recorded. From this set of relative start positions a distribution pattern is created for all 24 motifs in SCPD.

In Figures 7.7 to 7.9 we report the 24 resulting distribution patterns of the retrieved instances. There are several types of patterns present in this analysis. First there are several motifs for which a small number of instances was found and that display a rather noisy distribution pattern. Examples of this class are GAL4 (3), MIG1 (9), PDR1/PDR3 (10), RLM1 (16) and SMP1 (19). The largest class of motifs have a large number of instances and displays a more uniform distribution over the upstream region. Among these examples, there are some examples that show a minor increase in the number of instances closer to the start site. The motifs that fall in this last class are MCB (7), ROX1 (17), SCB (18), TBP (22) and UASPHR (23). Finally, there are two motifs, ABF1 (1) and REB1 (14), that exhibit a pronounced peak in their distribution. The instances of these two motifs are dominantly found in the region -200 to -100bp relative to the start site of the gene. If we look at their respective consensus sequences, `TCAGnnnnmACG` and `yyACCCG`, there is no connection between them. However, if we look at their function, it is not so surprising to find them at these positions in the sequence. ABF1 is a transcriptional activator and plays a role in DNA replication. REB1 is a RNA-polymerase enhancer binding protein and cooperates with both Pol-I and POL-II promoters.

7.4.2 Human promoter sequences from Ensembl

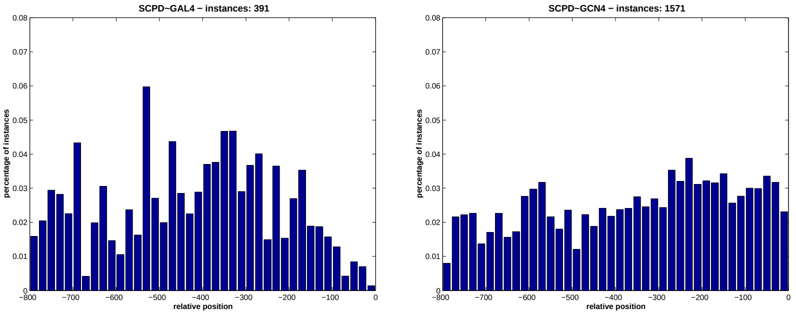
As a second case study, we analyze the upstream regions of a large set genes selected in the human genome.


SCPD~ABF1 − instances: 1146

SCPD~CSRE − instances: 669

1. ABF1

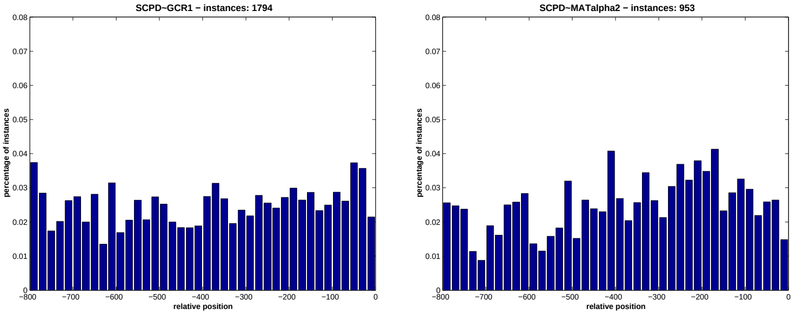
2. CSRE


SCPD~GAL4 − instances: 391

SCPD~GCN4 − instances: 1571

3. GAL4

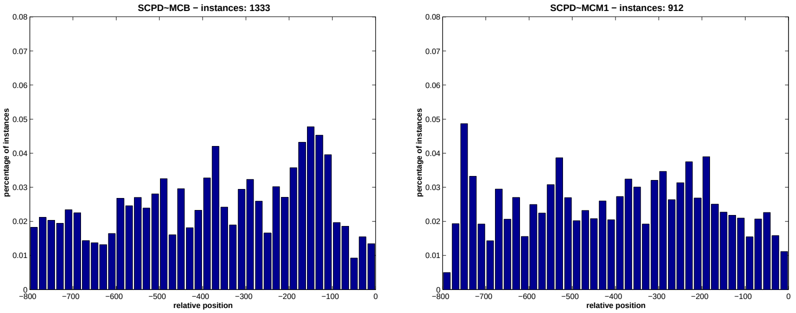
4. GCN4


SCPD~GCR1 − instances: 1794

SCPD~MATalpha2 − instances: 953

5. GCR1

6. MATα_2

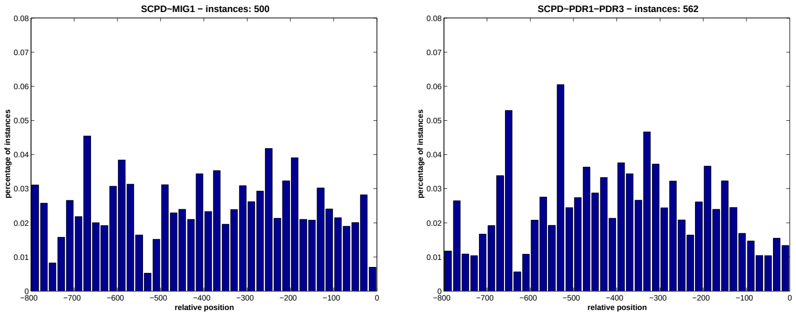

SCPD~MCB − instances: 1333

SCPD~MCM1 − instances: 912

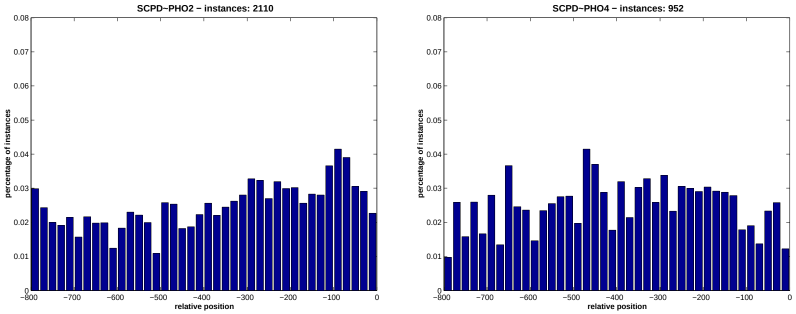
7. MCB

8. MCM1

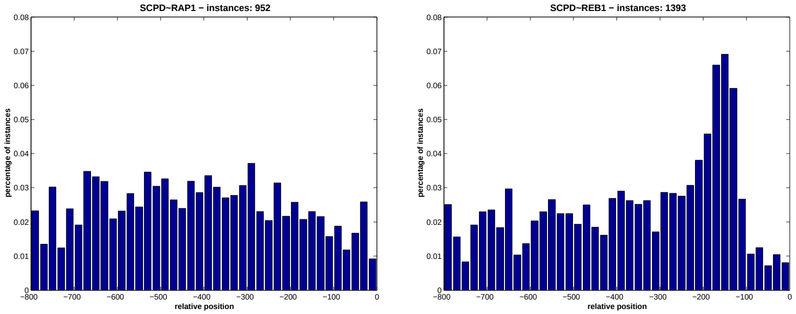
Figure 7.7: Distribution patterns of SCPD motif models in yeast upstream regions.



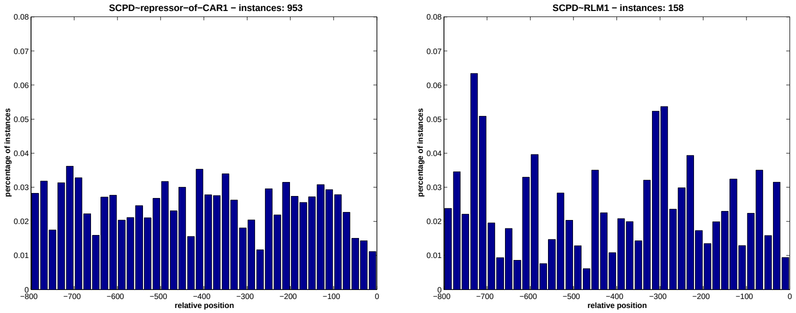
9. MIG1 **10**. PDR1/PDR3



11. PHO2 **12**. PHO4

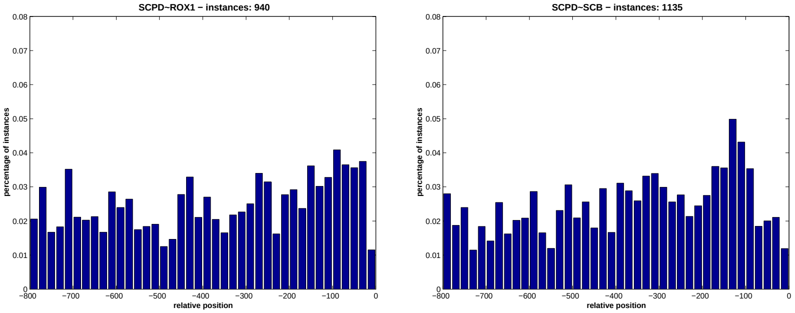


13. RAP1 **14**. REB1



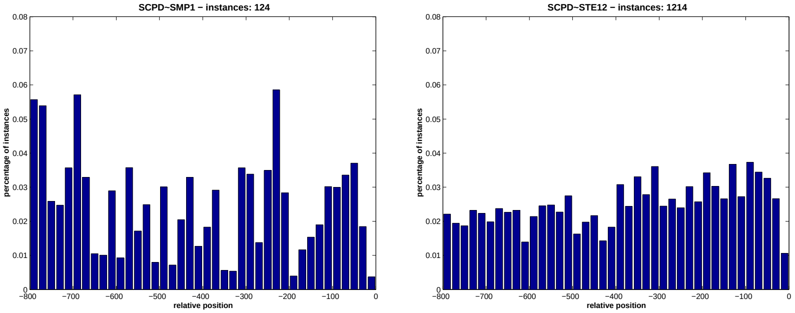
15. Repr. of CAR1 **16**. RLM1

Figure 7.8: Distribution patterns of SCPD motif models in yeast upstream regions.



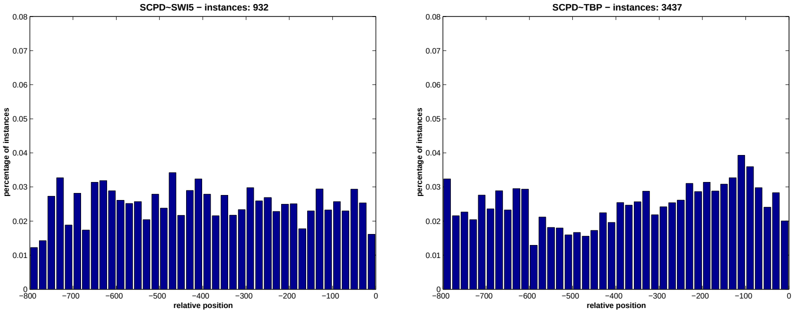
17. ROX1

18. SCB



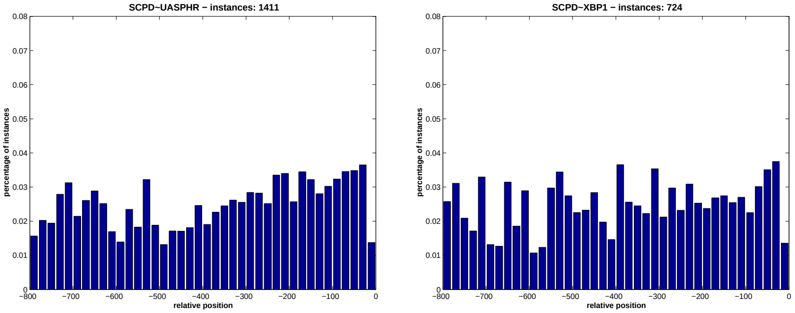
19. SMP1

20. STE12



21. SWI5

22. TBP



23. UASPHR

24. XBP1

Figure 7.9: Distribution patterns of SCPD motif models in yeast upstream regions.

data sets

A reference set is created by randomly selecting 4000 genes from Ensembl [46]. For each of these genes the annotated start of exon 1 is chosen as a reference point. In Ensembl, exon1 is annotated by mapping the 5‘-most transcript of the gene to the genome sequence. The final sequence set is then created by taking 2000bp upstream of the identified start site and also 200bp downstream. This results in a set of 4000 sequences of 2200bp. In this set of sequences we search for instances of the vertebrate motif models in TRANSFAC [117]. There are 400 motif models in this data set with the length varying from 6bp to 31bp. Since motifs in TRANSFAC contain both the core of the recognition site as the flanking nucleotides the average quality, measured by consensus score (Eq 5.10), of the motifs is rather low. While within the scoring system of TRANSFAC a distinction is made between core and flanking nucleotides, we use the motif models as they are provided. In the following paragraphs, we will only discuss a few examples that illustrate the points we like to explain in this thesis.

Quality assessment of retrieved instances

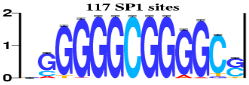
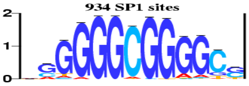
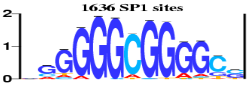
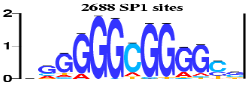
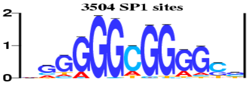
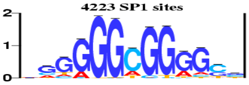
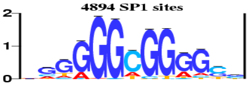
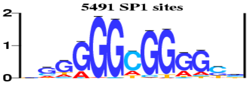
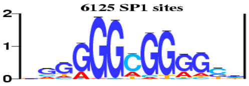
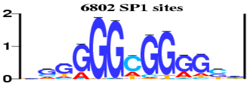
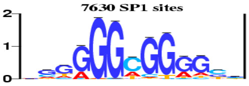
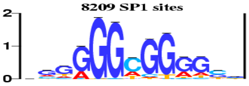
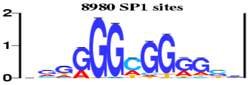
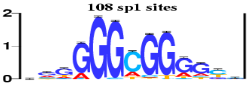
In a first set of tests we use the full sequence set and use the MotifScanner to search for one specific motif, SP1. SP1 belongs to the class of zinc finger transcription factor and is a well described factor (see Cook et al. [25] for a review). SP1 is a typical proximal promoter element and plays an active role in many different cell types. The SP1 binding site is GC-rich and has as consensus `nGGGGGCGGGGyn` (from TRANSFAC).

To evaluate the influence of the prior on the retrieved instances, we build a motif model from those retrieved instances. Table 7.1 shows how the constructed motif models evolves with an increasing prior. For each motif we report the number of instances that make up the model, the motif logo and the consensus score of the motif model. As expected the number of instances increases when we augment the prior and simultaneously the respective scores decrease as more noise is added to the model. The consecutive motif logos also shows this trend of decreasing level of conservation. For comparison we have also included the same information of the true SP1 motif as it is stored in TRANSFAC. The SP1 motif is constructed from 108 binding sites reported in human, mouse and rat. The quality of this motif is not very high since its consensus score is even lower than the score of the motif built from the 8904 sites found with a prior γ_1 of 0.99. From this comparison it is clear that defining the right prior is not evident from this table.

Nucleotide composition of upstream regions

Let us look again how the composition of the upstream region changes as we move from position -2000 to position +200. In this case the distribution, shown

Table 7.1: Motif models built from the retrieved SP1 sites.

prior γ_1	#inst.	consensus	cs
0.01	117		1.4026
0.05	934		1.3243
0.1	1636		1.2167
0.2	2688		1.1201
0.3	3504		1.0698
0.4	4223		1.0329
0.5	4894		0.9988
0.6	5499		0.9733
0.7	6125		0.9464
0.8	6802		0.9195
0.9	7630		0.8871
0.95	8209		0.8625
0.99	8980		0.8291
TRANSFAC^a	108		0.8145

^{a} motif model built from sites in TRANSFAC in Figure 7.10, behaves very different from the distribution observed in yeast (see Figure 7.6). The region far upstream of the start of exon 1 (-2000 to -800) is AT-rich and the proportion of AT and GC remains rather constant. However, if we move closer to start site the sequence composition change completely from AT-rich to GC-rich. One more remark we should make is that the start site as annotated in Ensembl seems to carry information that is common to most of these start sites as is indicated by the peaks around position 0. This might indicated that there is, in at least a large subset of these genes, a specific signal present at the start site which is most likely the transcription start. After the start site the composition remains at a more GC-rich level. It is clear that this difference in background composition over the sequence will have an important influence on the motif detection process.

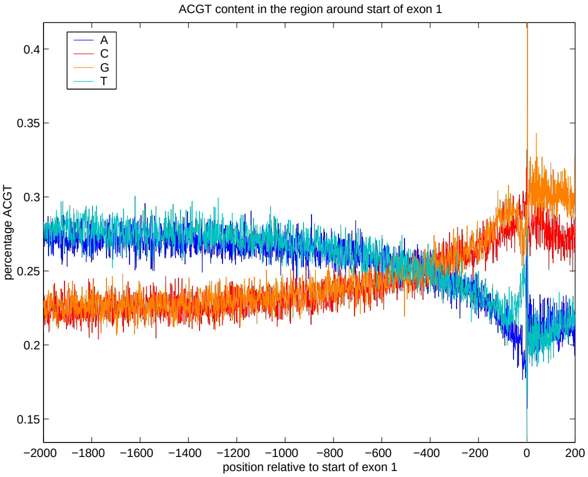


Figure 7.10: Distribution of `A`, `C`, `G` and `T` over the upstream region of the 4000 genes selected in ensemble. The region far upstream (-2000 to -700) is clearly AT-rich. Between positions -600 and -400 the nucleotide composition change from AT-rich to GC-rich. From there to just in front of the start, the level of GC keeps increasing. At the positions close to the start site a change in composition is observed that might indicate the presence of a specific signal. After the start site, the level of GC starts to drop and the level of AT slightly increases.

Analysis of proximal promoter region

Having gained extra knowledge about the sequence composition, it is necessary to have a closer look at the influence on the motif detection. For this purpose we select from the 2200bp sequences only the region between position -500 and the start site. From this new sequences we construct the appropriate background models. The difference between the background model from the full sequences or the proximal promoters is illustrated by the shift in SNF,

$$\begin{aligned}\text{full sequence SNF} &= [0.2543, 0.2419, 0.2464, 0.2574] \\ \text{proximal promoter SNF} &= [0.2336, 0.2635, 0.2659, 0.2370].\end{aligned}$$

The shift seems only minor, but the SNF of the full sequence set is the average over the AT-rich and GC-rich regions and this explains the almost equiprobable SNF in the full sequence set.

Let us now look at the number of motifs found in the sequence set. Together with the already discussed SP1, we also investigate another proximal promoter element, the `TATA`-binding protein (TBP). The binding site of TBP is represented in TRANSFAC as `TATAAATW`. TBP is thus an AT-rich motif and this is different from the background composition in the proximal promoter region, while SP1 is GC-rich and thus more like the background composition. These two very distinct motifs are useful to evaluate both algorithms. We run both MotifScanner and MotifLocator on this data set thereby testing the influence of the background model and the parameters of the respective algorithms.

Table 7.2: Number of instance found of TBP and SP1 as a function of the background model and the prior γ_1.

	instances of TBP				**instances of SP1**			
	0^a	1	2	3	0	1	2	3
MotifScanner								
0.0001^b	0	0	0	0	0	0	0	0
0.0005	0	0	0	0	1	1	0	0
0.001	0	0	0	0	36	10	1	0
0.005	2	24	1	0	439	272	101	31
0.01	33	107	4	1	728	547	296	159
0.05	535	647	251	161	1774	1441	1084	858
0.1	1253	1355	691	503	2453	2030	1636	1357
0.2	2112	2168	1505	1277	3224	2768	2413	2080
0.3	2797	2782	2184	1971	3796	3317	2992	2667
0.4	3372	3316	2734	2543	4256	3751	3434	3159
0.5	3835	3801	3260	3088	4647	4145	3870	3585
0.6	4253	4265	3785	3672	5014	4550	4319	4078
0.7	4729	4715	4296	4214	5394	5001	4787	4585
0.8	5262	5268	4933	4874	5840	5465	5332	5147
0.9	6001	5969	5810	5791	6529	6170	6105	5988
0.95	6591	6578	6436	6442	7072	6748	6726	6642
0.99	7674	7656	7562	7551	7947	7762	7723	7688
MotifLocator								
0.9^c	2864	2960	1799	1439	2071	1558	974	585
0.85	8859	8830	5796	5262	7045	5596	4245	3402

[a] order of the background model
[b] prior γ_1 of the MotifScanner
[c] threshold of the MotifLocator

The results of all theses tests are reported in Table 7.2. The first part are the number of instances found with MotifScanner in function of the background model and the prior γ_1 and the last two lines are the results with MotifLocator with threshold at 0.9 and 0.85. The first observed trend in these data is the expected increase in the number of instances as the prior increases. The second trend is that the number of instances decreases if we increase the order of the background model. These higher-order background models were actually introduced for this purpose, since we like to make a better distinction between binding sites and the background. The effect of the higher-order background model is most pronounced when using the MotifLocator.

Another interesting observation is that the number of instance of TBP sites is higher than the number of SP1 sites with the MotifLocator while contrary behavior is observed with the MotifScanner. Looking at the sequence composition, which is GC-rich, finding more TBP binding sites is unexpected a priori.

Localization of binding sites

Now that we have a found the instances of the motif in the proximal promoter region it is also interesting to find out where that these motif are located relative to the start site. Figure 7.11 displays the distribution patterns of respectively the SP1 sites (left) and TBP sites (right). In both examples we show the distribution pattern of the instances found with the MotifScanner using prior values of 0.1 and 0.95. In case of the SP1 motif, the distribution shows a peak in the region -150 to -50, relative the start site. This peak gets less expressed if the prior increases from 0.1 to 0.95. In case of the TBP motif, the instances are more evenly distributed over the complete region. However, there is a slight increase in instances in the region further upstream. This not unexpected since TBP is a more AT-rich motif and the region close to the start site is rather GC rich as was shown in Figure 7.10.

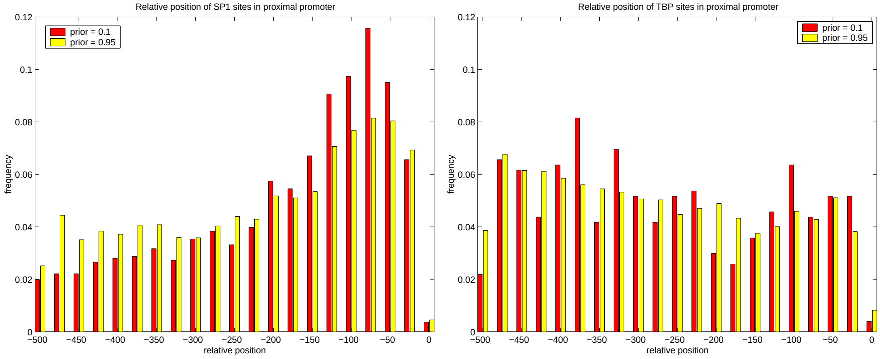


Figure 7.11: Relative position of the SP1 sites (left) and TBP sites (right) in the proximal promoter region of the 4000 human genes.

7.4.3 Discussion

To end this section, we like to expatiate on some problems we encountered in defining the optimal value for the prior with MotifScanner. Defining the best parameter for each motif model is a tedious task. As a comparison we refer to the program *match* that is used in TRANSFAC to score sequences with the stored matrices. In this program there are predefined thresholds with each of the matrices. To define these thresholds both promoter regions and exon 2 are scored with the PWMs and the distribution of the scores is compared. When we use MotifScanner such an approach is not really feasible. The parameter γ_1 depends on the motif model and the background model. It is known that the background composition in the promoter region and the exons is significantly different. Scoring exon 2 with a background model based on the promoter region will give very skewed results (remember the case study in prokaryotes in Chapter 6). Using the background model of each region separately is also not a good solution, a motif that is very unlikely to occur in the promoter region might be more likely to occur in the exon 2. The methodology proposed with *match* is thus not really appropriate to define a good value for the prior. The best approach would be to screen a reliable test set of promoter sequences in which true binding sites are annotated and measure the sensitivity and specificity. The main problem is that most sequences in such a reliable set are also part of the set that has been used to build the motif model.

Moreover, the question remains if it is a good approach to define a threshold based on the scores of the matrix in exon 2. If you score a segment with a PWM, the score is totally independent of the region in which the segment is located. This means that if the same oligomer, that corresponds to the binding site, is present in the promoter region and exon 2, both sites will have exactly the same score. It is exactly this problem we try to solve with MotifScanner. Our approach aims at finding those instances that rise above the background noise. This is also illustrated in the examples where we showed that the length of the sequence has a profound impact on the motif detection process. The longer the sequence, the more noise is present and the harder it is to locate the true motif instances. On the other hand, if the sequence is rather, short instances that only slightly resemble the binding site, will be picked up. Finding a good trade off is the main objective here.

Another aspect that influences the detection step is the nucleotide composition of the sequence set. As is shown in the human promoter sequence set, there is clear distinction between the proximal promoter region and the sequence composition further upstream. In the yeast upstream region the composition over the full upstream region is almost constant. Building a background model from a sequence set where the background composition changes, is thus not always the most proper approach. Defining the correct region in which to search for motifs will certainly improve the motif detection process. In higher eukaryotes a good approach will be to define synthetic regions that have been conserved between distantly related species (e.g. human and mouse, fly and mosquito).

7.5 Searching for known motifs in sets of coregulated genes

In this thesis, our main focus is on motif detection in sets of coregulated genes. A logical step is to apply the search for known motif models to such sets of coregulated genes. Therefore we need to assess the statistical significance of the number of motif instances in the sequence set. To exemplify the proposed methodology we analyze the yeast regulons and the cell cycle clusters introduced in the previous chapters.

7.5.1 Finding statistical significant instances

Given a set of sequences $\mathcal{S} = \{S_k | K = 1 \ldots N_s\}$, the goal is to identify if a motif $\mathbf{\Theta}$ has a significantly overrepresented number of instances in this set compared to the expected number of instances in this data set. To compute the statistical significance of a motif we use the methodology proposed by van Helden et al. [111] and later also added to *TOUCAN* [2] by Aerts et al. [1].

At first, we need to compute the expected frequency, $F_e\{\mathbf{\Theta}\}$, of the number of instances of $\mathbf{\Theta}$ in genome. After screening the reference sequence set with the motif model, we can compute the expected frequency as

$$F_e\{\mathbf{\Theta}\} = \frac{\text{total number of instances}}{\text{total number of start positions}}.$$

With this expected frequency we can compute the expected number of instances of $\mathbf{\Theta}$ in the sequence set $\mathcal{S}$ as

$$E(\text{inst}\{\mathbf{\Theta}\}) = F_e\{\mathbf{\Theta}\} \times 2 \times \sum_{k=1}^{N_s} (L_k - W + 1) = F_e\{\mathbf{\Theta}\} T. \tag{7.4}$$

We multiply with two to account for both strands in the analysis. The next step is to compute the statistical significance of the observed number of instances. Using a binomial model, the probability of observing exactly n instances of $\mathbf{\Theta}$ in the sequence set is

$$P(\text{inst}\{\mathbf{\Theta}\} = n) = \frac{T!}{(T-n)! \times n!} \times (F_e\{\mathbf{\Theta}\})^n \times (1 - F_e\{\mathbf{\Theta}\})^{T-n}.$$

The probability of observing n or more instances is then

$$\begin{aligned} P(\text{inst}\{\mathbf{\Theta}\} \geq n) &= \sum_{j=n}^{T} P(\text{inst}\{\mathbf{\Theta}\} = j) = 1 - \sum_{j=0}^{n-1} P(\text{inst}\{\mathbf{\Theta}\} = j) \\ &= 1 - \sum_{j=0}^{n-1} \frac{T!}{(T-j)! \times j!} \times (F_e\{\mathbf{\Theta}\})^j \times (1 - F_e\{\mathbf{\Theta}\})^{T-j}. \end{aligned} \tag{7.5}$$

---

[2] a tool for the analysis of *cis*-regulatory sequences

The lower this probability is the more unlikely it is to find this number of instances in this data set. We could impose a threshold on this probability to define significantly overrepresented motifs. We should select those motifs for which the probability of finding the observed number of instances is smaller than the threshold.

Since the probability in Eq. 7.5 depends on the size of the motif model, van Helden et al. [111] proposed to transform this probability to a significance coefficient

$$\mathrm{sig} = -\log_{10}\big(P(\mathrm{inst}\{\mathbf{\Theta}\} \geq n) \times D\big), \tag{7.6}$$

with D the total number of tests. In the case of oligo analysis, D is chosen equal to the distinct number of oligonucleotides of length W. In our case we could use the total number of matrices present in our evaluation set. The highest significance coefficient corresponds to most overrepresented motif in this sequence set.

7.5.2 Expected frequency in yeast genome

As test case we look at the motifs from SCPD in the yeast upstream sequences. We start with the MotifScanner and screen all the sequences with the SCPD motifs thereby testing multiple values for the prior γ_1. To illustrate the influence of the prior on the expected frequency we show in Table 7.3 the expected frequency computed after screening with a prior γ_1 of 0.01 and 0.95. The last column gives the ratio between the expected frequency of the motif found with prior 0.95 and with prior 0.01. This table illustrates that there is a great difference among the different motifs. On the one hand, the expected frequency –thus also the number of instances– of TBP increases with a approximately a factor ten. At the other hand, the number of STE12 sites increases almost with a factor 1000 from as little as thirteen instances found with a prior of 0.01 to almost 13,000 instances with a prior of 0.95. It is thus very important to compare to the correct reference set if we like to compute the statistical significance of the number of instances of a motif in the given data set.

7.5.3 Yeast regulons

To test the described methodology, we first screen each of the ten regulons with the SCPD matrices. The prior γ_1 is in each example set to 0.2. We use the implementation inside Toucan to calculate the statistical significance of motif instances. Table 7.4 gives an overview of all motifs found in the ten yeast regulons that have a significance coefficient greater than 0.

First there are several regulons in which the motifs are found that correspond to the true motifs in this regulon. Theses regulons are GAL, GCN, PHO, PDR and TUP. Looking at the scores, it is also clear that the level of the significance coefficient, ranging from 3.86 to 9.08, is rather high and is several orders of

Table 7.3: Expected frequency of SCPD motifs in yeast upstream sequences.

id	$\gamma_1 = 0.01^a$	$\gamma_1 = 0.95^a$	**ratio**^{b}
ABF1	2.68E-05	1.66E-03	62.01
CSRE	6.88E-06	1.70E-03	247.02
GAL4	1.18E-05	6.18E-04	52.44
GCN4	8.21E-06	2.03E-03	247.10
GCR1	4.10E-06	2.11E-03	516.12
MATα_2	9.53E-06	1.84E-03	193.18
MCB	2.04E-05	1.87E-03	91.73
MCM1	1.73E-05	1.51E-03	86.99
MIG1	7.99E-06	1.23E-03	153.60
PDR1/PDR3	2.30E-05	8.93E-04	38.85
PHO2	2.21E-06	2.12E-03	958.71
PHO4	1.16E-05	1.67E-03	143.59
RAP1	5.02E-05	1.11E-03	22.15
REB1	2.67E-05	1.55E-03	57.92
repr. of CAR1	2.46E-05	1.56E-03	63.21
RLM1	2.10E-06	2.91E-04	138.15
ROX1	1.17E-05	1.79E-03	153.16
SCB	5.85E-06	1.85E-03	316.19
SMP1	2.44E-06	2.69E-04	110.20
STE12	2.06E-06	2.06E-03	1001.62
SWI5	8.31E-06	1.84E-03	221.96
TBP	1.88E-04	2.29E-03	12.19
UASPHR	9.59E-06	1.98E-03	206.68
XBP1	4.63E-06	1.52E-03	327.52

[a] expected frequency in yeast upstream region
[b] ratio of the expected frequencies

magnitude higher than of the other motifs found. Among those others we find the motifs RLM1 and SMP1 that are found in a very limited number of instances. That these motifs are identified as being significant can be explained by the fact that their expected frequency is so low that finding even two instances in a sets is already marked as very unlikely. In the MET data set the PHO4 motif is found as being significant, although at a lower level. The reason might be that the consensus of the PHO4 motif, `nnCACGTkkk`, is fairly similar to the consensus of the known motif, `TCACGTG`. However, there is no motif found that corresponds to the Met31p and Met32p binding site, `AAAACTGTGG`. Looking at the PHO4 motif model in SCPD, it seems constructed from the high-affinity form `GCACGTGGG` and the low-affinity form `GCACGTTTT`. In the sets INO, NIT and YAP, there is no significant motif found at all. This not so surprising since there is no motif found in SCPD with similar consensus to the three motifs in these regulons. Therefore we also search for the specific yeast motifs stored in TRANSFAC. In this case, again there is no significant motif found in the NIT and INO regulons. In the YAP regulon only the motif *F$UAY_Q2* is found to be significant. However, this factor is part of the purine utilization pathway in

Table 7.4: Significant motifs found in yeast regulons with MotifScanner ($\gamma_1 = 0.2$)

	ID[a]	**Consensus**	n[b]	$P(\text{inst} \geq n)$	**Sig**[c]
GAL	GAL4	`CGGrnnACwnTnsnCCG`	11	5.19E-11	9.08
	RLM1	`nrTTCTATwwATAGAyyn`	2	2.51E-2	0.397
	repr. of CAR1	`AGCCGCCrn`	4	5.79E-2	0.033
GCN	GCN4	`TGACTn`	43	2.37E-9	7.25
	RLM1	`nrTTCTATwwATAGAyyn`	7	1.06E-3	1.59
	MATα2	`CATGTAATT`	16	2.40E-2	0.24
HAP	ROX1	`yCnATTGTTnTn`	5	4.41E-2	0.13
INO	-	-	-	-	-
MET	PHO4	`nnCACGTkkk`	8	5.77E-3	0.94
NIT	-	-	-	-	-
PDR	PDR1/PDR3	`TCCGCGGA`	10	9.79E-8	5.708
	ROX1	`yCnATTGTTnTn`	6	7.23E-3	0.84
	SMP1	`nnGCTkCTATwwATAGnAwn`	2	2.15E-2	0.367
	MCB	`ACGCGT`	6	3.16E-2	0.199
PHO	PHO4	`nnCACGTkkk`	10	5.66E-7	5.17
TUP	MIG1	`CCCCrsnTwTwn`	14	6.06E-6	3.86
	UASPHR	`nTTwnTnCCTCk`	16	2.04E-2	0.33
	GAL4	`CGGrnnACwnTnsnCCG`	6	4.18E-2	0.02
	RAP1	`rCACCCAkACAy`	11	4.33E-2	0.01
YAP	-	-	-	-	-
	F$UAY_Q2	`rnCGsmnsmskCCGAy`	7	3.57E-3	0.794

[a] Identifier of the motif model in SCPD or TRANSFAC.
[b] Number of instances found in the data set.
[c] Significance coefficient computed with Eq. 7.6.

Aspergillus nidulans and is thus probably not specific to yeast.

This analysis indicates that it is possible to find true overrepresented motifs in such a set of promoter sequences using the proposed methodology. However, the limiting factor is now the availability of reliable motif models.

7.5.4 Cell cycle clusters

To further evaluate the method, we repeat a similar analysis as in the previous example with the four clusters identified in the yeast cell cycle (see Section 6.4). In Table 7.5 we display all the significant motifs found in the four clusters from the cell cycle data. The strongest motifs are clearly the MCB and SCB motifs in cluster 28 and 4. For cluster 28 this coincides with the results of the MotifSampler where the MCB motif is also found as the best scoring motif. The retrieval of SCB and MCB in cluster 4 is not confirmed by the MotifSampler results where two unknown motifs were identified.

In cluster 24 the motifs RLM1 and MIG1 are found as the most significant, while they have respectively only four and five instances in the data set. The third motif in the data set is the STE12 motif which we had already identified with the MotifSampler. The fact that it is only found in only eight out of 19

Table 7.5: Significant motifs found with MotifScanner in cell cycle clusters ($\gamma_1 = 0.2$)

	ID^a	**Consensus**	n^b	$P(\text{inst} \geq n)$	**Sig**^{c}
Cl. 3	REB1	`TTACCCG`	26	1.01E-3	1.62
	MIG1	`CCCCrsnTwTwn`	11	9.42E-3	0.65
Cl. 4	SCB	`CnCGAAA`	29	1.00E-8	6.66
	MCB	`ACGCGT`	27	2.48E-6	4.26
Cl. 24	RLM1	`nrTTCTATwwATAGAyyn`	4	1.74E-3	1.42
	MIG1	`CCCCrsnTwTwn`	5	2.16E-2	0.32
	STE12	`ATGAAACn`	8	3.80E-2	0.08
Cl. 28	MCB	`ACGCGT`	71	0	2.15E+9
	SCB	`CnCGAAA`	30	3.24E-9	7.13
	SMP1	`nnGCTkCTATwwATAGnAwn`	4	1.42E-3	0.49

[a] Identifier of the motif model in SCPD.
[b] Number of instances found in the data set.
[c] Significance coefficient computed with Eq. 7.6.

sequences could indicate that the sequence set might contain genes that do exhibit the same expression profile but that are not regulated by the same factor. Finally, in cluster 3, there are two significant motifs found, REB1 and MIG1.

7.6 Conclusions

In this chapter, we introduce two methods **MotifLocator** and **MotifScanner** that have the same goal, namely finding instances of known motifs in DNA sequences, but that are intrinsically very different. MotifLocator is based on a PWM scoring scheme, while MotifScanner uses the probabilistic sequence model to estimate the number of motif instances. In the first set of tests, we discuss how the parameters of the algorithm influence the performance of the algorithms. One important aspect of the probabilistic sequence model is that the background noise depends on the sequence length and this has an impact on the motif detection process. While the number of instances found with a PWM method increases linearly with the sequence length, the number of instances found with the MotifScanner can in some cases decrease when the sequences grow. This effect is due to the growing level of noise present in a longer sequence. This also means that it becomes harder to detect the motif instances in this noise and only those instances that are strong enough will rise above the noise.

To illustrate the potential of our implementation we analyze two large sets of promoter sequences. The first set consists of all upstream regions in the yeast genome, the other data set is a selection of 4000 upstream sequences in human. In this study we not only look at the motifs hidden in these sequences but also at the nucleotide composition of the selected regions and how this influences the motif detection process. While in yeast the upstream region has a rather constant distribution, the selected human upstream sequences display a significance chance over this long region. Defining the correct region in which to search for motifs seriously influences the motif detection process. The screening of these promoter sequences with known motif models reveals that some of the motifs tend to bind at some prefered region close to the start of the gene.

The two described scoring methods can also be applied on data sets of coregulated genes to find motifs in such a set that have a significantly overrepresented number of instance when compared to a reference data set. To quantify the significance we compute the probability of finding the counted number of instance with respect to the expected number of instances in the set. As an illustration of the applicability of this methodology, we screen the yeast regulons and the four cell cycle clusters with the SCPD motif models. The first conclusion of this analysis is that we are able to find the overrepresented motifs in these data set. However, we also see that we are completely dependent on the quality of the database at hand. If the true motif is present in the database, it is found in all examples as the most significantly overrepresented motif in the sequence set.

CHAPTER 8

INCLUSive: Integrated Search for Regulatory Elements[1]

Throughout this thesis, we have emphasized that our motif finding algorithm was especially designed to analyze sets of coregulated genes. In this chapter we discuss the integration of tools to analyze the promoter region of groups of coregulated genes, starting from expression measurements. To illustrate the process from clustering to motif finding we use a microarray experiment in which the gene expression in response to mechanical wounding in Arabidopsis was measured. These data are first clustered with adaptive quality-based clustering. Next, we try to identify the upstream sequences by locating the genes on the Arabidopsis genome. In the final step we use MotifSampler and MotifScanner on these data sets to search for overrepresented motifs. To end this chapter, we present some results by others who have used the MotifSampler within their own experiments.

8.1 Basics of INCLUSive

Let us in Figure 8.1 redraw the flowchart that we introduced in Figure 1.1 in which the procedure to analyze gene expression data is outlined. In the version of INCLUSive as it is described here is part of the flowchart has been, at least partially, solved. The part in gray is not a part of the current implementation of INCLUSive, but the functional annotation will be included in the near future (summer 2003, see also Aerts et al. [1]). This part of our work

---

[1] INCLUSive as it is presented here has appeared as an Application Note in Bioinformatics [108]. An updated version of INCLUSive will appear in the websoftware issue of Nucleic Acids Research [24].

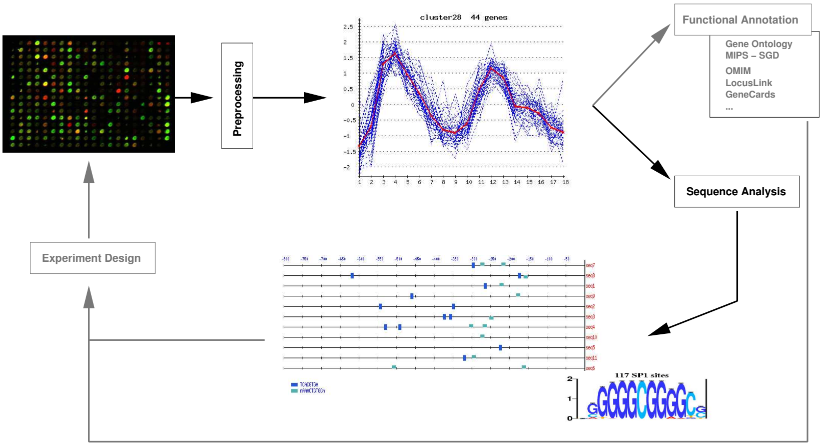


Figure 8.1: Flowchart of INCLUSive. The process start from several microarray experiments. The data in these experiments are then preprocessed and clustered to find groups of genes that show a similar expression profile over the experiment. The next step is the analysis of the clusters. To study the cluster at a sequence level, the promoter region is identified. Statistically overrepresented motifs, that might be potential transcription factor binding sites, are retrieved from such a set of sequences. These motifs are also checked and functionally annotated. The first version of INCLUSive does not include the parts in gray in this figure that are also part of the flowchart in Figure 1.1.

is mainly driven by practical concerns. As our algorithms were evolving from prototype to usable version, we felt the need to make these tools accessible for biologists. In the first stage of our project we choose to make the algorithms available through a simple web interface where interested users could upload their data and try out our algorithms. This allowed us already to communicate about first results [105]. It also allowed us to interact with day-by-day users who could help identify the weaknesses of our methods. As soon as our algorithms had evolved beyond prototypical designs we tried to make them compliant with standards in bioinformatics (e.g. output in GFF). We also tried to make the link between the different tools to automate the process flow from expression to regulation. This set of integrated tools is accessible through our **INCLUSive** web interface [108]:

`http://www.esat.kuleuven.ac.be/~dna/Software.html.`

To find groups of tightly coexpressed genes we use **Adaptive Quality-Based Clustering** (AQBC) [26]. For each of the clusters identified with AQBC, the user can retrieve the upstream sequences. We have implemented several modules that enable us to automatically parse annotated sequence records and identify the upstream sequences for each gene in the cluster. Once the set of upstream sequences has been constructed, the user can perform a motif search with both MotifSampler and MotifScanner.

As an example of this integrated data flow, we analyze a small microarray experiment in *A.thaliana*[2]. In this microarray experiment, Reymond et al. [86] tested the effect of mechanical wounding of leaves on the expression of a particular set of genes in *A.thaliana*. We use this data set to guide us through the different steps of INCLUSive.

8.2 Defining groups of coexpressed genes

In recent years, a variety of clustering methods have been developed and adapted to find groups of coexpressed genes from microarray measurements. These algorithms include classical methods as hierarchical clustering [31], k-means [102], and self-organizing maps [100] but also more advanced methods as the self-organizing tree algorithm [40], model-based clustering [121], quality-based clustering [26, 43] and many more. Each of these methods addresses one or more problems specific to the process of clustering gene expression profiles.

8.2.1 Adaptive quality-based clustering

Within the INCLUSive framework, we use the in-house developed Adaptive Quality-Based Clustering algorithm [26]. AQBC is an extension of the algorithm proposed by Heyer et al. [43] and is in essence a heuristic two-step method that sequentially identifies clusters. In the first step of the algorithm the center of the cluster is located by searching for a region in the high dimensional data space with a high density of data points. In the second step the radius of the cluster is estimated. The radius is estimated by fitting a model to the data points with an EM algorithm. This model assumes that the distribution of points belonging to a cluster is clearly distinguishable from the distribution of the points belonging to the background.

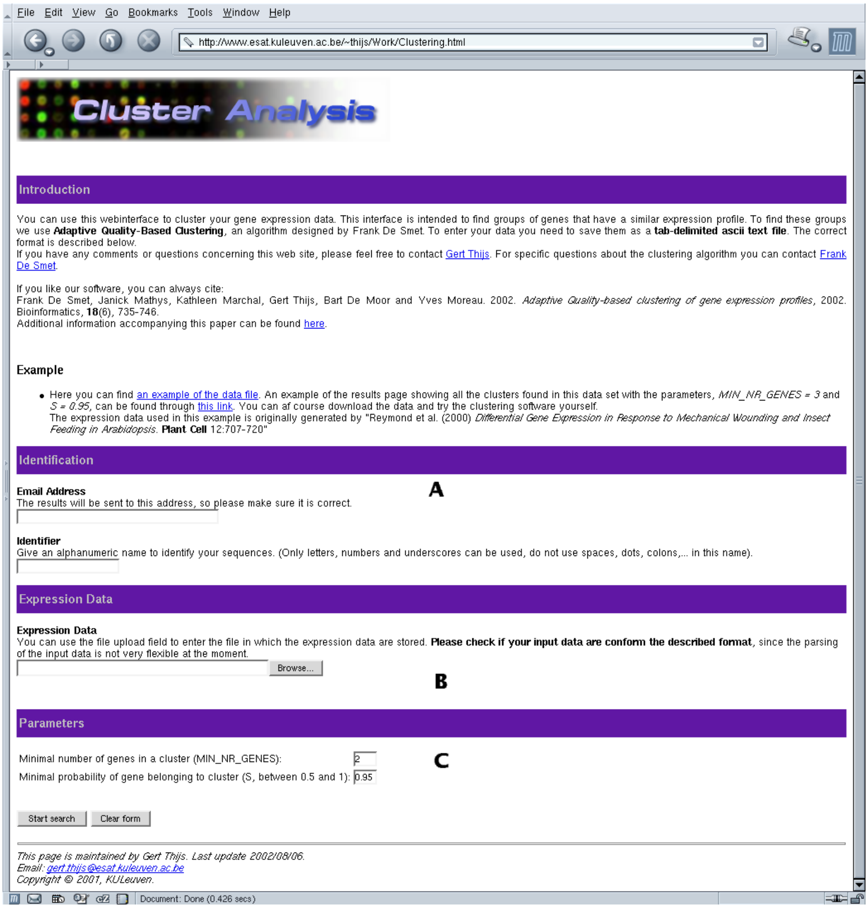
Figure 8.2 is a screenshot of the web interface of as we have implemented it on top of the algorithm. AQBC needs two parameters to be set by the user: (1) the quality criterion and (2) the minimal number of genes in a cluster. The data should be submitted in a simple text file with each row containing the expression measurements of one gene. The input data are checked and send to a server that will apply the AQBC algorithm. The results are sent by email to the users.

8.2.2 Clustering example: mechanical wounding in plants

To illustrate the steps in INCLUSive, we use a small data set of microarray experiments in *A.thaliana*. Reymond et al. [86] have investigated the expres-

---

[2] Results with the first version of the MotifSampler on this data set have been published in the Journal of Computational Biology [106].



Introduction

You can use this webinterface to cluster your gene expression data. This interface is intended to find groups of genes that have a similar expression profile. To find these groups we use **Adaptive Quality-Based Clustering**, an algorithm designed by Frank De Smet. To enter your data you need to save them as a **tab-delimited ascii text file**. The correct format is described below.
If you have any comments or questions concerning this web site, please feel free to contact Gert Thijs. For specific questions about the clustering algorithm you can contact Frank De Smet.

If you like our software, you can always cite:
Frank De Smet, Janick Mathys, Kathleen Marchal, Gert Thijs, Bart De Moor and Yves Moreau. 2002. *Adaptive Quality-based clustering of gene expression profiles*, 2002. Bioinformatics, **18**(6), 735-746.
Additional information accompanying this paper can be found here.

Example

- Here you can find an example of the data file. An example of the results page showing all the clusters found in this data set with the parameters, *MIN_NR_GENES = 3* and *S = 0.95*, can be found through this link. You can af course download the data and try the clustering software yourself.
  The expression data used in this example is originally generated by "Reymond et al. (2000) *Differential Gene Expression in Response to Mechanical Wounding and Insect Feeding in Arabidopsis*. **Plant Cell** 12:707-720"

Identification

Email Address
The results will be sent to this address, so please make sure it is correct.

Identifier
Give an alphanumeric name to identify your sequences. (Only letters, numbers and underscores can be used, do not use spaces, dots, colons,... in this name).

Expression Data

Expression Data
You can use the file upload field to enter the file in which the expression data are stored. **Please check if your input data are conform the described format**, since the parsing of the input data is not very flexible at the moment.

Parameters

Minimal number of genes in a cluster (MIN_NR_GENES): 2
Minimal probability of gene belonging to cluster (S, between 0.5 and 1): 0.95

This page is maintained by Gert Thijs. Last update 2002/08/06.
Email: gert.thijs@esat.kuleuven.ac.be
Copyright © 2001, KULeuven.

Figure 8.2: Screen shot of the adaptive quality-based clustering web interface. The interface consists of three parts: (A) fields to identify the user and the data set; (B) field to upload the data file; (C) definition of the parameters of AQBC. The data set should be a tab-delimited ascii file with the gene identifiers and the expression data. The parameters of the algorithm are the quality criterion and the minimal number of genes in a cluster.

sion of genes in response to mechanical wounding of plant leaves. The array contains a set of 150 known defense related genes and 16 control genes for which the expression was unlikely to vary. Leaves of six- to seven-week-old plants were wounded. mRNA was extracted at eight time points up to 24 hours after wounding.

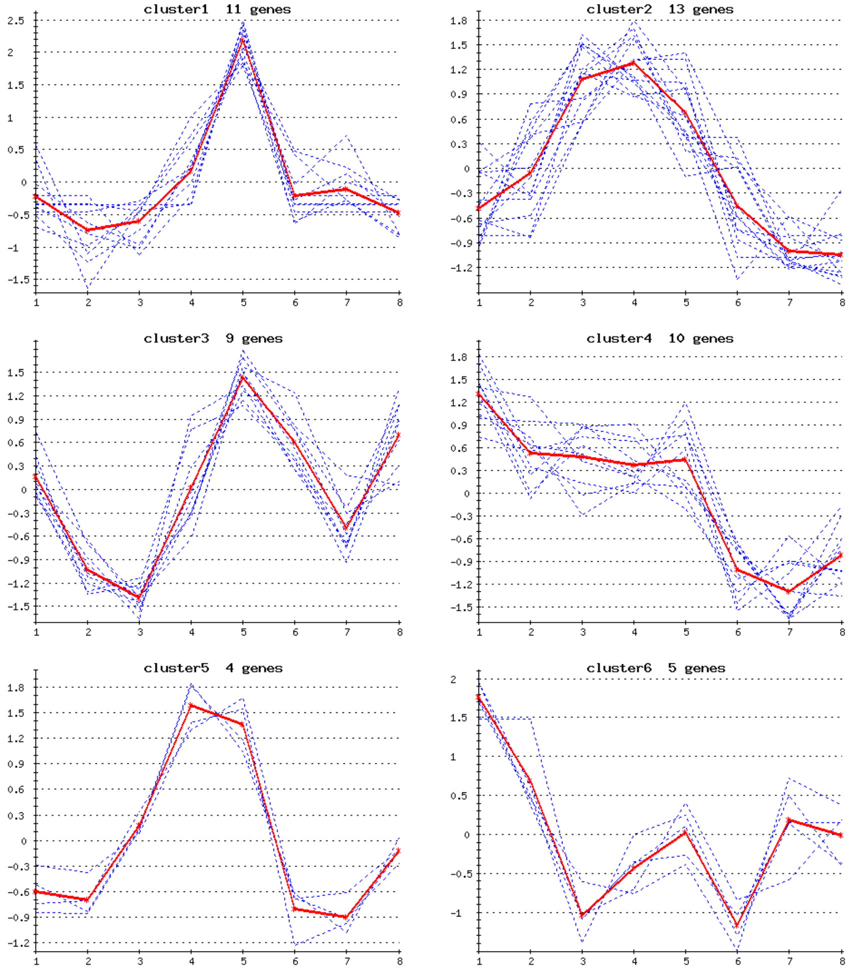


Figure 8.3: Six clusters found in the wounding data set with AQBC. As parameters, the quality criterion is set to 0.95 and the minimal number of genes in a cluster is set to 4. For each cluster, the normalized expression profile of the genes together with the mean profile (thick line) of the cluster is plotted.

AQBC is used to find groups of genes that have a similar expression profile. As parameters of AQBC we set the quality criterion to 0.95 and the minimal number of genes within a cluster should be four. This results in six distinct clusters. The expression profiles of all the genes belonging to the respective clusters are shown in figures 8.3.

8.3 Upstream sequence retrieval

Once we have clustered the expression profiles, we can start analyzing the genes that belong to a cluster with an interesting profile. We are mainly interested in the composition of the regulatory region upstream of the gene. To this end we need to extract this upstream region from sequence databases. In this section we describe the steps involved in the retrieval of the upstream sequences starting from the identification of the genes spotted on the array and the information present in the sequence databases. The process consists of the following elementary steps: (1) identification of the spotted genes, (2) locating the genes on a genomic sequence and (3) retrieval of the corresponding intergenic region[3].

8.3.1 Parsing of annotated sequences

Sometimes it is possible to directly extract the intergenic region from the information given for each spotted gene, but mostly we will need to align the gene of interest on genomic DNA to locate the correct intergenic region. The processing of the sequence data in these steps is done with the help of modules from *bioperl* (`http://www.bioperl.org`) [98]. We build a gene model on top of the sequence object provided in *bioperl*. A gene object is generated by iterating twice over all annotated features in the DNA sequence. The basic tags *gene*, *CDS* and *mRNA* are used in the first iteration to build the core of the gene objects from the annotation. In the second iteration all other features related and/or overlapping with an initial gene are added to the gene object to refine the gene object.

As an example of such a sequence in GenBank format, we show here the annotation of the gene Eli3 stored as an mRNA sequence in the entry with *X67816* as accession number.

```
LOCUS       ATELI31                 1340 bp    mRNA    linear   PLN 30-JUN-1993
DEFINITION  A.thaliana mRNA for Eli3-1.
ACCESSION   X67816 S51267
...
FEATURES             Location/Qualifiers
     source          1..1340
                     /organism="Arabidopsis thaliana"
                     /strain="ecotype Columbia"
                     /db_xref="taxon:3702"
                     /clone_lib="lambda gt10"
     gene            46..1119
```

---

[3] The described procedure was conceived in 2000 when no large annotated genome databases, like Ensembl, were available. Nowadays (spring 2003) promoter sequences can be retrieved from these specialized databases. However, for some species the described method is still useful.

```
                     /gene="Eli3-1"
     CDS             46..1119
                     /gene="Eli3-1"
                     /codon_start=1
                     /protein_id="CAA48027.1"
                     /db_xref="GI:16267"
                     /db_xref="SWISS-PROT:Q02971"
                     /translation="MGKVLQKEAFGLAAKDNSGVLSPFSFSRRATGEKDVRFKVLFCG
                     ICHTDLSMAKNEWGLTTYPLVPGHEIVGVVTEVGAKVKKFNAGDKVGVGYMAGSCRSC
                     DSCNDGDENYCPKMILTSGAKNFDDTMTHGGYSDHMVCAEDFIIRIPDNLPLDGAAPL
                     LCAGVTVYSPMKYHGLDKPGMHIGVVGLGGLGHVAVKFAKAMGTKVTVISTSERKRDE
                     AVTRLGADAFLVSRDPKQMKDAMGTMDGIIDTVSATHPLLPLLGLLKNKGKLVMVGAP
                     AEPLELPVFPLIFGRKMVVGSMVGGIKETQEMVDLAGKHNITADIELISADYVNTAME
                     RLAKADVKYRFVIDVANTMKPTP"
BASE COUNT      354 a    261 c    347 g    378 t
ORIGIN
        1 caaatactta cttttgaatc cgttttttc attgtttgat cgattatggg aaaggttctt
        ...
//
```

The first lines of the entry give the identification of the sequence and also the complete classification of the organism (not shown) and reference to the literature (not shown). The second part of such an entry describes the annotated features. A gene is annotated from positions 46 till 1119 in this sequence. This is indicated with the feature tags *gene* and *CDS*, where *CDS* refers to the coding sequence and the corresponding translation of the gene (cfr. Table 2.1). At the end of the record the sequence itself is given. In this example the gene and corresponding coding sequence have both the same identifier, but this is mostly not the case. To build a reliable gene object from an annotation which lacks consistent naming, we have defined a set of rules on top of the sequence parser from *bioperl*. An overlapping gene and CDS annotation that do not bear the same identifier, are assumed to be parts of the same gene if they show sufficient overlap with each other. Other annotated features that help identifying a gene are the *primary transcript*, *mRNA* and *5'UTR* or *3'UTR* annotations. Other features are added to the gene model but they do not directly contribute to the quality of the gene object.

8.3.2 Gene indexing and upstream region identification

The first step in the creation of the upstream sequence sets is the identification of the genes present in a cluster. Each measurement in the expression data set comes from one spot on the microarray. Normally, each spotted clone corresponds to one particular gene and this is indicated with a specific identifier. These identifiers could refer to genes as they are given in *UniGene* or to ESTs or cDNA sequences stored in the sequence databases. Within INCLUSive we assume that the user has made the mapping from the specific identifier of the spot on the array to the sequence accession number. The accession number and gene name should be part of the expression data set entered at the AQBC web interface. Based on this accession number and gene name, the system will automatically try to locate the gene in the GenBank record.

Table 8.1: Overview of the genes in the six clusters in the wounding data set.

Nbr.	Genes[a]		
1	X99793-AWI31	L40031-CCOAMT	X67816-ELI3
	AJ000470-GPX2	J04537-HMG1	AC001645-JIP
	n.a.-MEKKK	U09958-NIT2	M34107-PR3AIII
	X91957-GER2	AF001168-LECRK	
2	AF057044 ACX1	M92353-ASA1	L22585-ASB
	D78598-CYP83B1	D14007-FAD7	Z26426-GST1
	D44465-GST5	Y13577-JR3	Y10617-OPR1
	U59508-PRODH	U18993-TSA	M23872-TSB
	Z26519-CM1		
3	AB003040-ERF	S697270-GS	AJ012571-GST8
	AP000413-HADH	AF123254-MFP2	Y11607-MP2C
	J03240-NR	AB008854-PED1	Z56278-SPS
4	M84697-α-TUB	U40857-AIG2	M84706-β-TUB
	M17132-HIST	AF021346-NDR1	n.a.-REM2
	D45848-TCH1	L34546-TCH3	U27609-TCH4
	AB008106-ERF4		
5	X74604-HSC70	X52428-OEC	AF059581-SAHH
	M33217-SAM		
6	U47029-EREC	AJ000469-GPX1	Z97337-HCNL1
	AC008113-PEL	X79195-TGG2	

[a] genes are identified by accession number and gene name.

The accession number and the corresponding gene name of the genes belonging to the respective clusters found in the plant wounding data set are shown in Table8.1. In most cases the accession number refers to a cDNA sequence stored in genome databases. In such a case there is only one gene present in the entry and the gene name merely helps to match and to verify the annotated gene identifier. However, in a few cases the accession number refers to a longer, fully sequenced part of the *A.thaliana* genome. In this case the name of the gene is necessary to identify the gene of interest. We can apply some form of pattern matching to automatically identify the matching gene name in the database entry, but the user should check the matching names. Sometimes the gene name is given as an abbreviation while the full name is given in the annotation. This type of annotation and the lack of standard gene names makes the automatic identification of genes a very hard job to accomplish.

The next step is the localization of the intergenic or upstream region of the identified genes. When the gene of interest is stored as an EST or cDNA sequence there is no upstream region present in the entry. In this case we need to find the gene on a genomic sequence. If the spotted gene is located in a longer sequence and if the upstream located gene is also annotated, it is possible to immediately extract the full intergenic region. If no upstream gene is found in the annotation, part of the upstream region closest to the gene might be selected and the gene is send to the next step in the selection procedure.

8.3.3 Aligning genes on genomic sequences

The genes that have been identified and where no intergenic region is found in the previous steps, are now further processed. It is necessary to locate those genes on the genomic sequence and to retrieve the upstream region from the annotated genome sequence. Figure 8.4 gives the flowchart of the steps to follow to accomplish this retrieval. We start by aligning the gene sequence with the genomic sequences by performing a BLAST[4] search [5]. Since we are only interested in those hits that are almost identical with our query gene, we can run Blast with the default parameter settings. The Blast search results are summarized in a list of high-scoring pairs (HSP) between query sequence (the gene of interest) and a subject sequence from the data base. Each HSP is checked for the length of the matching sequence and its similarity score. To be considered for detailed analysis, a HSP should be at least 100bp long and have 90% similarity. If this rule is obeyed, the subject sequence is downloaded and all the genes annotated on the sequence are located. Next we check if the positions of the match between query and subject overlaps with one of the indexed genes. If there is an overlapping gene found, we consider this as being the match with the gene of interest. Next, the upstream or intergenic sequence of the aligned gene is located depending on the presence of an upstream annotated gene.

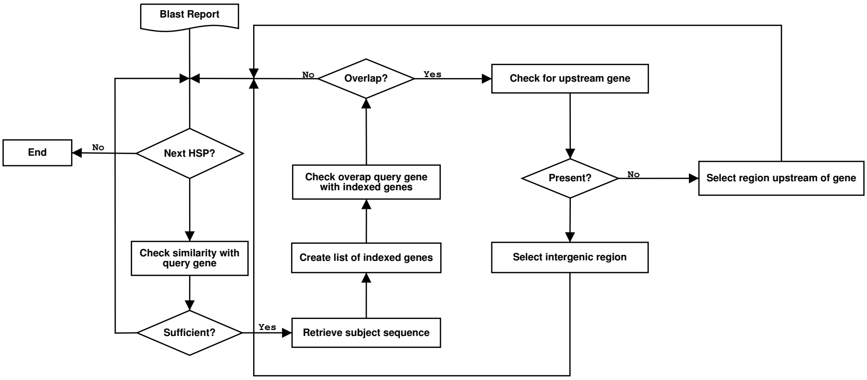

Blast Report
No
Overlap?
Yes
Check for upstream gene
End
No
Next HSP?
Check overap query gene with indexed genes
Present?
No
Select region upstream of gene
Check similarity with query gene
Create list of indexed genes
Select intergenic region
Sufficient?
Yes
Retrieve subject sequence

Figure 8.4: Flowchart of the automatic identification of intergenic region from the result in blast reports.

In the final step the user will receive a list of all potential intergenic and upstream regions. This list might contain multiple entries for one particular gene, since there might be several database entries available which contain the gene of interest. The user should then indicate which sequences he trusts most. The interface allows making a subselection and truncating the sequences at specific positions.

---

[4] Basic Local Alignment Search Tool.

In our plant wounding example we use the described strategy to find the upstream region of all the genes in the six clusters. Most genes are located without much problems and the corresponding upstream region is easily retrieved. For those genes where we cannot find the upstream region automatically, we perform the same strategy in a manual fashion. Manually checking each individual step allows us to retrieve sequences that fall beyond the capacity of our automated system. Examples are sequences that were poorly annotated. Also defining translation or transcription start becomes more accurate when we double check each step. However, this becomes very time consuming when the data sets become larger. The automated approach significantly speeds up the process and we still achieve sufficient accuracy to compile reliable data sets.

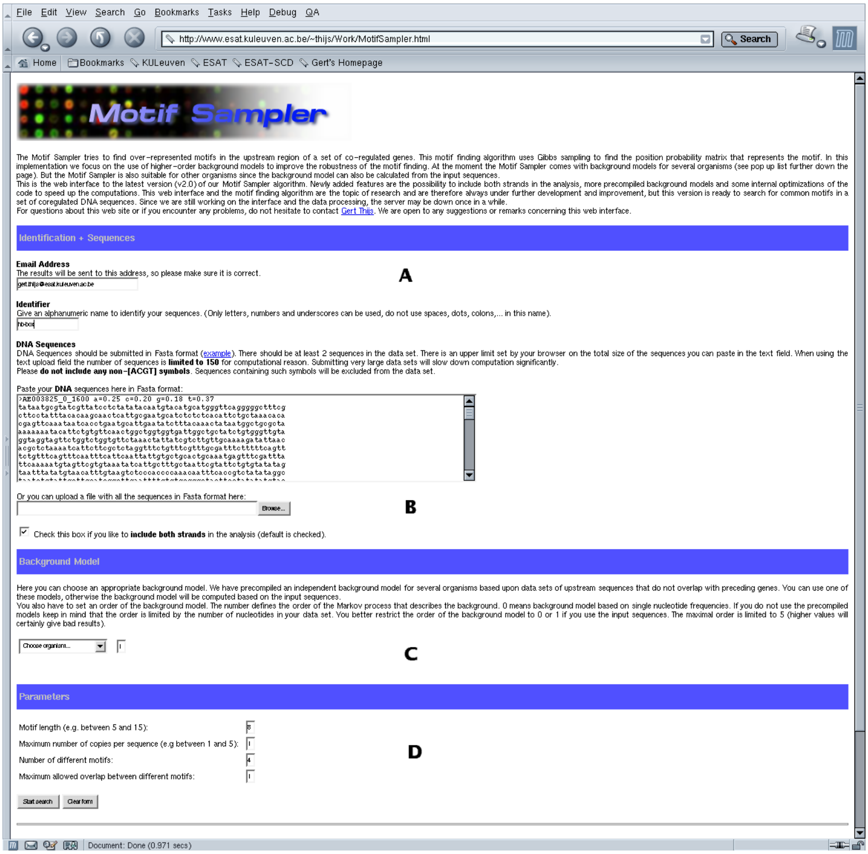
8.4 MotifSampler

In the previous section we explained how we can generate a set of upstream sequences based on the genes present in a cluster with a specific expression profile. So now we will try to find the motifs which are over-represented in these sets of sequences using our Gibbs sampling algorithm.

8.4.1 MotifSampler web interface

Figure 8.5 is a screenshot of the MotifSampler web interface. Besides uploading the sequences the user should indicate the background he or she would like to use. Currently we provide a list of background models for over 100 species, ranging from prokaryotic organisms to higher eukaryotes as *A.thaliana*, *D.melanogaster* and *H.sapies*. The users should also set a limited number of parameters: (1) the desired motif length, (2) the maximal number of copies to expect, (3) the number of different motifs to search for and (4) the allowed overlap between different motifs. When the users presses the submit bottom, the data are sent to a local server. On this server, one run of the MotifSampler on the submitted sequence set is carried out. The result files are then post-processed and a HTML page is generated that visualizes the results with sequences logos and an alignment plot. These final results are mailed to the user.

The web interface to MotifSampler is very useful to quickly test a data set and to get a visual representation of the obtained results, but it lacks the possibilities to do an exhaustive analysis as presented at the end of Chapter 5. Since MotifSampler is a stochastic method, one should do multiple runs to find the significant results. For such a thorough analysis the user should use the stand-alone version of MotifSampler which is written in C++ and is available for download from the INCLUSive web pages.


File Edit View Search Go Bookmarks Tasks Help Debug QA
http://www.esat.kuleuven.ac.be/~thijs/Work/MotifSampler.html
Search
Home
Bookmarks
KULeuven
ESAT
ESAT-SCD
Gert's Homepage
Motif Sampler
A
B
C
D
Browse...
Start search
Clear form
Document: Done (0.971 secs)

The Motif Sampler tries to find over-represented motifs in the upstream region of a set of co-regulated genes. This motif finding algorithm uses Gibbs sampling to find the position probability matrix that represents the motif. In this implementation we focus on the use of higher-order background models to improve the robustness of the motif finding. At the moment the Motif Sampler comes with background models for several organisms (see pop up list further down the page). But the Motif Sampler is also suitable for other organisms since the background model can also be calculated from the input sequences.
This is the web interface to the latest version (v2.0) of our Motif Sampler algorithm. Newly added features are the possibility to include both strands in the analysis, more precompiled background models and some internal optimizations of the code to speed up the computations. This web interface and the motif finding algorithm are the topic of research and are therefore always under further development and improvement, but this version is ready to search for common motifs in a set of coregulated DNA sequences. Since we are still working on the interface and the data processing, the server may be down once in a while.
For questions about this web site or if you encounter any problems, do not hesitate to contact Gert Thijs. We are open to any suggestions or remarks concerning this web interface.

Identification + Sequences

Email Address
The results will be sent to this address, so please make sure it is correct.
gert.thijs@esat.kuleuven.ac.be

Identifier
Give an alphanumeric name to identify your sequences. (Only letters, numbers and underscores can be used, do not use spaces, dots, colons,... in this name).
hb-box

DNA Sequences
DNA Sequences should be submitted in Fasta format (example). There should be at least 2 sequences in the data set. There is an upper limit set by your browser on the total size of the sequences you can paste in the text field. When using the text upload field the number of sequences is **limited to 150** for computational reason. Submitting very large data sets will slow down computation significantly.
Please **do not include any non-[ACGT] symbols**. Sequences containing such symbols will be excluded from the data set.

Paste your **DNA** sequences here in Fasta format:

```
>AE003825_0_1600 a=0.25 c=0.20 g=0.18 t=0.37
tataatgcgtatcgttatcctctatatacaatgtacatgcatgggttcaggggctttcg
cttcctatttacacaagcaactcattgcgaatgcatctctctcacattctgctaaacaca
cgagttcaaataatcacctgaatgcattgaatatctttacaaactataatggctgcgcta
aaaaaaatacattctgtgttcaactggctggtggtgattggctgctatctgtgggttgta
ggtaggtagttctggtctggtgttctaaactattatcgtcttgttgcaaaagatattaac
acgctctaaaatcattcttcgctctaggtttctgtttcgtttgcgatttctttttcagtt
tctgtttcagtttcaatttcattcaattattgtgctgcactgcaaatgagtttcgattta
ttcaaaaatgtagttcgtgtaaatatcattgctttgctaattcgtattctgtgtatatag
taatttatatgtaacatttgtaagtctcccaccccaaacaatttcaccgtctatataggc
```

Or you can upload a file with all the sequences in Fasta format here:

☑ Check this box if you like to **include both strands** in the analysis (default is checked).

Background Model

Here you can choose an appropriate background model. We have precompiled an independent background model for several organisms based upon data sets of upstream sequences that do not overlap with preceding genes. You can use one of these models, otherwise the background model will be computed based on the input sequences.
You also have to set an order of the background model. The number defines the order of the Markov process that describes the background. 0 means background model based on single nucleotide frequencies. If you do not use the precompiled models keep in mind that the order is limited by the number of nucleotides in your data set. You better restrict the order of the background model to 0 or 1 if you use the input sequences. The maximal order is limited to 5 (higher values will certainly give bad results).

Choose organism... 1

Parameters

Motif length (e.g. between 5 and 15): 8
Maximum number of copies per sequence (e.g between 1 and 5): 1
Number of different motifs: 4
Maximum allowed overlap between different motifs: 1

Figure 8.5: Screenshot of the MotifSampler web interface: (A) identification of the data set and the user; (B) sequences submission field; (C) selection of the background model; (D) parameter definition.

8.4.2 Motif finding in plant wounding clusters

In the plant wounding example, we end up with six sets of sequences of a maximum of 1000bp that can be used in the MotifSampler. However, before we can start the motif search it might be necessary to filter out long repeats in the sequences that might potentially deteriorate the motif finding results. To this end we could use the programs *RepeatMasker* (`http://ftp.genome.washington.edu/RM/RepeatMasker.html`) or *purge* [44]. This step is not provided in INCLUSive but might be useful in some cases to improve results, when large repetitious regions are present.

To find the relevant motifs in these six data sets we use the stand-alone version of the MotifSampler and apply the strategy described in this dissertation. For each of the six data sets we apply the MotifSampler with different values for the motif length (8, 10 or 12bp). The prior probability is set to 0.3 to retrieve only the stronger motifs. Each individual test is repeated 100 times and in each run three different motifs are retrieved. This results for each combination of parameters in a set of 300 different motifs. This list of 300 motifs is then further processed to find the most significant ones. Significance can be based on different measurements. Here is the list of analysis steps we apply for each data set:

1. Run MotifSampler with given parameter setting and store results in GFF and matrix file.
2. Create ranking of the motifs based on motif scores (e.g. log-likelihood).
3. Select top 5 by iteratively removing similar motifs from the list.
4. Compare selected motifs with database of known motifs.

The results of this motif search are bundled in Table 8.2 to Table 8.7 for each cluster respectively. As candidate motifs we select the five motifs with the highest log-likelihood for each of the chosen lengths (8, 10 or 12bp). For each retrieved motif we display the consensus sequence, the number of sequences in which the motif is found, the total number of motif instances, the number of similar motifs found and the motif scores: consensus score, information content and log-likelihood. The number of similar motifs is defined as the total number of motifs from the list of all motifs (900 motifs) that have a mutual information with the selected motif smaller than 0.65 (see Eq. 5.13).

This analysis shows that it is not always evident to select the motifs from this large list of retrieved motifs. Unlike in the case studies of Chapter 6, the selection of the true motifs is not straightforward. However, there are a few motifs that are found consistent over the different motif lengths in five out of six clusters. These motifs are also mostly found in the most runs. In the first cluster, the common motif is `ATATATAT`. In cluster 3 the consensus of the common motif can be deduced as `ACwATTAA`. In cluster 4 the common motif, `wGCTGGmGC`, is only found when searching for motifs of length 10 or 12bp. In cluster 5, there

Table 8.2: Top scoring motifs of 8, 10 and 12bp found in cluster 1

Consensus	**#Seq.**^{a}	**#Inst.**^{b}	**#sim**^{c}	**cs**^{d}	**ic**^{e}	**ll**^{f}
yATATATA	9	21	158	1.73	1.36	70.28
AkTACGww	8	12	5	1.39	1.37	67.05
TCwAGTCC	7	10	5	1.39	1.40	63.67
wwCTAGAT	7	13	16	1.53	1.39	60.29
AATAAkGr	8	12	3	1.54	1.41	59.15
ATATATATAy	7	15	150	1.55	1.23	63.78
wAknATATAT	7	15	9	1.45	1.16	63.32
AAAAAAACAA	8	11	6	1.68	1.34	59.03
GGrwTTGAAT	8	12	1	1.36	1.30	58.40
ATTATTnAAT	7	9	8	1.47	1.17	56.52
AyCnAwTTGrTn	8	18	5	1.20	1.03	72.65
AwATAAwwATnA	8	12	3	1.38	1.04	67.45
wyATATAnATAT	7	14	103	1.41	1.04	66.82
AArnAAAAAyAA	8	17	1	1.43	1.17	64.27
TTTGArATTTAT	7	10	16	1.52	1.22	62.74

^{a} Number of sequences in which motif is found. ^{b} Number of instances.
^{c} Number of similar motifs found. ^{d} Consensus score.
^{e} Information content. ^{f} Log-likelihood score.

Table 8.3: Top scoring motifs of 8, 10 and 12bp found in cluster 2

Consensus	**#Seq.**	**#Inst.**	**#sim**	**cs**	**ic**	**ll**
ksACCkwC	10	11	6	1.39	1.59	78.43
AnGnAAnG	8	17	12	1.28	1.36	66.67
GTTCrTAT	8	12	21	1.53	1.49	66.23
TTsAATkT	9	14	9	1.64	1.38	66.06
wGACGAAs	8	10	18	1.49	1.64	65.94
GkyAAAGTTn	7	10	4	1.41	1.37	71.17
AwTTGACTmA	7	10	2	1.37	1.23	67.89
GGAnwTkrGr	9	11	1	1.29	1.38	62.58
GAmkkwArwA	10	14	3	1.28	1.19	61.75
TTCTGTAkTT	8	10	4	1.49	1.33	61.61
ArAAGAAAAnAr	10	18	4	1.46	1.25	85.58
mAwGGAwnCGTn	9	11	11	1.18	1.18	78.50
GTArnTGAnGrn	7	9	9	1.16	1.30	71.63
nGrAGAAGAAnA	9	12	23	1.33	1.32	71.24
AGAAGnwGrAGA	8	15	19	1.29	1.34	65.54

For the legend, see Table 8.2

Table 8.4: Top scoring motifs of 8, 10 and 12bp found in cluster 3

Consensus	#Seq.	#Inst.	#sim	cs	ic	ll
`AACTGTAn`	8	9	39	1.45	1.34	52.9488
`ACwATTAA`	8	10	90	1.68	1.38	42.4619
`AATrTGAA`	9	10	49	1.53	1.37	42.3155
`mTTGTCAT`	6	7	9	1.45	1.37	42.09
`AsCGATCA`	4	4	11	1.37	1.49	41.7828
`TAAAAAkAAA`	7	8	20	1.60	1.25	55.1164
`ACwATTAAmA`	6	7	41	1.48	1.24	49.9716
`nCnGCACnAT`	6	6	9	1.33	1.35	46.7344
`AAGTCTyAAG`	4	7	7	1.36	1.27	44.1155
`AnsAsTGTGn`	6	7	9	1.19	1.30	43.6022
`TmATAsTATTAA`	10	11	13	1.23	1.01	60.8616
`TCAAnTCnnAAG`	4	6	13	1.33	1.29	52.6372
`AATAmynTTATT`	7	9	10	1.31	0.99	50.0605
`AGAkATrywACG`	5	5	5	1.28	1.24	49.5559
`mTGnATGnnTyA`	5	6	13	1.29	1.27	49.4079

For the legend, see Table 8.2

Table 8.5: Five top scoring motifs found in cluster 4

Consensus	#Seq.	#Inst.	#sim	cs	ic	ll
`kGAGkCGn`	6	12	54	1.35	1.66	69.1132
`GArTCGyC`	7	9	29	1.45	1.71	61.3259
`ACGrATyG`	6	10	40	1.48	1.62	60.3919
`ATACAkTr`	8	12	9	1.51	1.36	58.3634
`sCATCCAA`	7	11	10	1.47	1.53	55.6167
`ATCGAAyTAm`	7	12	20	1.29	1.17	65.017
`GAnnCGnnTC`	6	20	30	1.14	1.36	62.9839
`AmArAGsAAA`	8	12	2	1.39	1.32	62.3872
`nmrCGrATyG`	5	14	16	1.18	1.27	61.9134
`nwGCTGGmGC`	6	9	51	1.23	1.55	60.8755
`nkGAGkCGnCnC`	7	12	34	1.07	1.29	77.7388
`nCwGGCTyTkAC`	6	7	4	1.17	1.29	62.0885
`nkswGCTGnmGC`	6	14	20	1.01	1.23	62.0524
`TsrTTwCwGknT`	8	15	2	1.12	1.14	61.6448
`AAwAAATwCATT`	5	9	4	1.43	1.08	59.961

For the legend, see Table 8.2

Table 8.6: Five top scoring motifs found in cluster 5

Consensus	#Seq.	#Inst.	#sim	cs	ic	ll
`nCnnAGCC`	4	6	40	1.22	1.40	38.7311
`CGArATCT`	4	9	106	1.38	1.42	37.3456
`kGACTGwG`	4	10	32	1.21	1.43	36.0444
`sACTTsAr`	4	8	11	1.35	1.38	35.9269
`TCAnTTGn`	4	9	19	1.40	1.41	35.2884
`TCACAGCCTT`	4	7	29	1.26	1.30	43.4228
`CnAGATCTGn`	4	10	125	1.17	1.31	42.9451
`ACTGAnCyAn`	4	6	27	1.13	1.19	41.8416
`rmyGGsCCrT`	4	8	42	1.02	1.28	41.7759
`kykAAATCTG`	4	8	47	1.28	1.26	40.6394
`TAAnGCnTTmAn`	3	6	15	1.21	1.08	46.5792
`CAGATynAmywm`	4	14	123	1.13	1.10	42.217
`mnGGCCCATTAG`	3	4	48	1.14	1.34	41.606
`ACTGrGCCAkTA`	4	4	19	1.14	1.18	39.8769
`AnsAGATCTsnT`	3	6	116	1.26	1.33	39.4901

For the legend, see Table 8.2

Table 8.7: Five top scoring motifs found in cluster 6

Consensus	#Seq.	#Inst.	#sim	cs	ic	ll
`CrrryCyG`	5	11	6	1.13	1.38	45.5397
`AGTTCwws`	5	11	21	1.37	1.35	44.0565
`GGAGrTnG`	5	12	77	1.18	1.52	40.0167
`wCAAGTsy`	4	8	30	1.30	1.32	39.6928
`mGnGGATG`	5	7	54	1.32	1.50	39.6146
`nsnwnTGAGT`	4	12	5	1.11	1.19	47.1158
`knAnGsGGwT`	4	6	21	1.16	1.34	46.1431
`GnrTGGrTCC`	4	8	5	1.07	1.31	43.6795
`CnGAwyCnGA`	4	9	31	1.09	1.31	42.6934
`GGAGAwGGGw`	5	13	71	1.18	1.38	42.4648
`CnsATnmnsnnG`	4	9	4	0.91	1.18	48.951
`nArrwnyGGwGA`	4	8	12	0.99	1.14	47.6678
`TmAGmACknGGA`	5	10	15	0.99	1.09	47.4842
`ArrmTCAkAAys`	4	8	9	1.16	1.15	46.3071
`rsnwCCGGwnsy`	5	10	1	0.88	1.15	44.4689

For the legend, see Table 8.2

is a strong motif found in each of the runs. This motif has as common consensus `CsAGATCT`. In cluster 6 there is a common motif found when searching for motifs of 8 or 10bp. The motif can be summarized as `GGAGATGG`. Finally, in cluster 2 we do not find any motif that is consistently found in the three motif lengths.

8.5 MotifScanner

8.5.1 Web interface

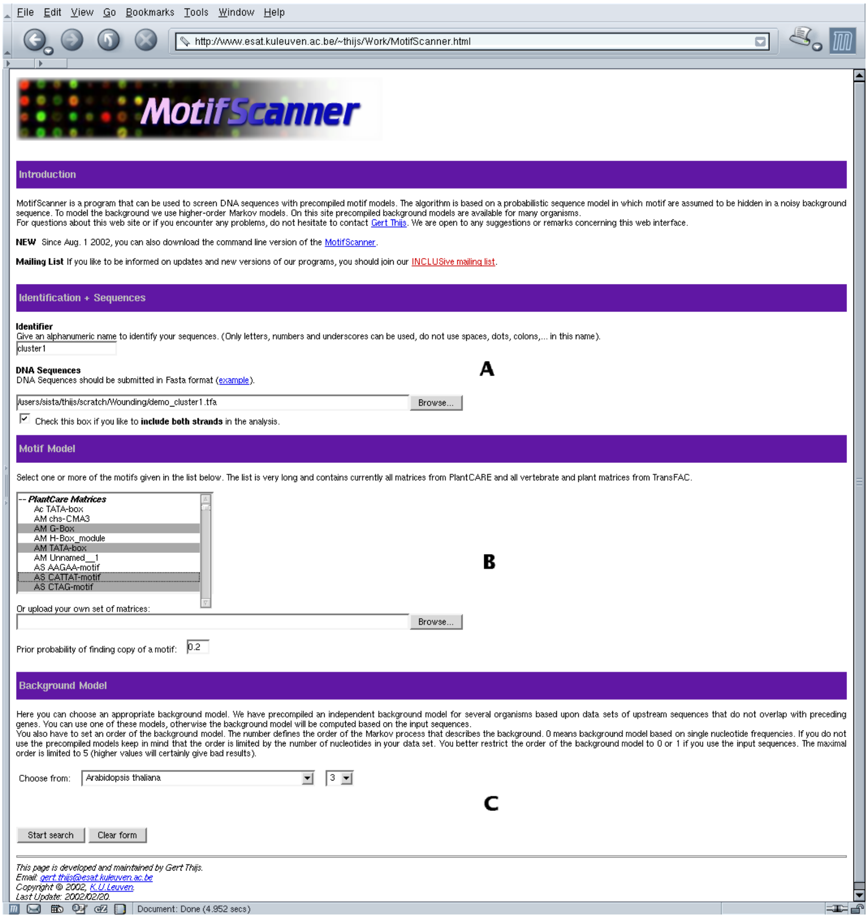
Also for MotifScanner, our program to screen for instances of known motifs, we have build a web interface. On this site we offer a large selection of motif models compiled from the sites in PlantCARE and TRANSFAC. The user has also access to over 100 precompiled background models of a wide variety of species. The user should submit a sequence set, make a selection of the motif models, select the right background model and define the prior γ_1. Figure 8.6 gives a screenshot of the MotifScanner web interface.

8.5.2 Example: plant wounding clusters

Since we also have several precompiled motif models available, it seems also useful to search for these known motifs in the upstream region of the coexpressed genes. Therefore we use the procedure explained in Chapter 7. The significance level is computed by comparing the retrieved number of motif instances with the expected number of motif instances. This expected number of motif instances is computed by scoring a set of 17000 upstream sequences of 1000bp in *A.thaliana* with the same matrices.

Motif are considered being significantly overrepresented if their significance coefficient is greater than 0. Table 8.8 gives an overview of the motifs from TRANSFAC and PlantCARE found in the six data sets. For each motif we give the accession number and/or identifier of the motif, the number of sequences in which the motif is found, the total number of motif instances and the significance level. In cluster 4 there is no motif found that is significantly overrepresented. In the other five clusters a wide variety of motifs is found to be overrepresented. Some of these motifs have been characterized in process that are related to the plant defense system.

In cluster 1, three motifs are found: SPF1_Q2, SBF1_01 and the TATA-box. The TATA-box corresponds to a repetition of `AT`'s that was also found with the MotifSampler and is probably not the real TATA-box. The SPF1 factor activates the expression sporamin and β-amylase and Ishiguro and Nakamura [48] have related this factor to a leaf cutting experiment. The other motif, SBF1, is silencer-binding factor and is found in relation to bean defense genes [55]. The PIF3 factor found in cluster 3 is the phytochrome-interacting factor which is needed for normal photoinduced signal transduction [76]. In clus-


File Edit View Go Bookmarks Tools Window Help
http://www.esat.kuleuven.ac.be/~thijs/Work/MotifScanner.html
MotifScanner
A
B
C
Browse...
Start search
Clear form

Introduction

MotifScanner is a program that can be used to screen DNA sequences with precompiled motif models. The algorithm is based on a probabilistic sequence model in which motif are assumed to be hidden in a noisy background sequence. To model the background we use higher-order Markov models. On this site precompiled background models are available for many organisms.
For questions about this web site or if you encounter any problems, do not hesitate to contact Gert Thijs. We are open to any suggestions or remarks concerning this web interface.

NEW Since Aug. 1 2002, you can also download the command line version of the MotifScanner.

Mailing List If you like to be informed on updates and new versions of our programs, you should join our INCLUSive mailing list.

Identification + Sequences

Identifier
Give an alphanumeric name to identify your sequences. (Only letters, numbers and underscores can be used, do not use spaces, dots, colons,... in this name).
cluster1

DNA Sequences
DNA Sequences should be submitted in Fasta format (example).

/users/sista/thijs/scratch/Wounding/demo_cluster1.tfa

☑ Check this box if you like to **include both strands** in the analysis.

Motif Model

Select one or more of the motifs given in the list below. The list is very long and contains currently all matrices from PlantCARE and all vertebrate and plant matrices from TransFAC.

-- ***PlantCare Matrices***
Ac TATA-box
AM chs-CMA3
AM G-Box
AM H-Box_module
AM TATA-box
AM Unnamed__1
AS AAGAA-motif
AS CATTAT-motif
AS CTAG-motif

Or upload your own set of matrices:

Prior probability of finding copy of a motif: 0.2

Background Model

Here you can choose an appropriate background model. We have precompiled an independent background model for several organisms based upon data sets of upstream sequences that do not overlap with preceding genes. You can use one of these models, otherwise the background model will be computed based on the input sequences.
You also have to set an order of the background model. The number defines the order of the Markov process that describes the background. 0 means background model based on single nucleotide frequencies. If you do not use the precompiled models keep in mind that the order is limited by the number of nucleotides in your data set. You better restrict the order of the background model to 0 or 1 if you use the input sequences. The maximal order is limited to 5 (higher values will certainly give bad results).

Choose from: Arabidopsis thaliana 3

This page is developed and maintained by Gert Thijs.
Email: gert.thijs@esat.kuleuven.ac.be
Copyright © 2002, K.U.Leuven.
Last Update: 2002/02/20.

Document: Done (4.952 secs)

Figure 8.6: Screen shot of the MotifScanner web interface. (A) Identification of the user and submission field of the sequence set. (B) Selection of motif models. (C) Definition of the higher-order background model.

Table 8.8: Significant motifs from TRANSFAC and PlantCARE found in the 6 plant wounding clusters.

cl.[a]	**id**[b]	**consensus**	**#seq**	**#inst**	**sig**[c]
1	M00702 SPF1_Q2	WMATAGTAWW	7	10	0.474
	M00149 SBF1_01	KWRTNGTTAAWWWN	9	10	0.063
	AT TATA-box	TATAwA	8	17	0.782
2	M00434 PIF3_01	GKRGGMCACGTGRMSWCK	4	5	0.101
3	M00654 OSBZ8_Q6	ACGTGTCGCSAKWC	4	6	0.467
	M00503 ATHB5_01	CAATTATTG	5	7	0.587
	AT HD-zip2	CAATsATTG	4	8	0.612
	AT CHS-unit1_m1	ACCTACCACAC	4	5	0.28
4	-				
5	M00442 ABF_Q2	NNGGMCACGTGGCNCNN	2	4	1.222
	AT G-box	GCCACGTGGnA	2	4	0.668
6	M00438 ARF_Q2	YTTGTCTC	3	4	0.072
	M00345 GAMYB_01	YAACSGMC	4	5	0.129
	M00374 O2_03	GATGAYATGG	4	5	0.293
	M00439 C1_Q2	YNAACNAYCNS	4	7	0.604
	M00506 LIM1_01	CCACNANMNNCN	4	8	0.587

[a] Cluster number.

[b] Identifier of the motif model either from TRANSFAC or PlantCARE.

[c] significance coefficient (Eq. 7.6).

ter 3, we find the binding sites of a bZip protein identified in rice, that is induced by abscisic acid or dehydration and is known to be part of the plant defense system [75]. Another overrepresented motif in this cluster is a specific homedomain-leucine zipper complex ATHB5 that is involved in the control of the plant development. The other two motifs come from PlantCARE. The HD-zip2 transcription factor is annotated as being involved in the development of leaf morphology. The Chs-unit 1 module is part of a light responsive element. In the fifth clusters the light response element G-box is found together with ABF. ABF is a stress-related transcription factor that is induced by abscisic acid and salt [23]. Finally, in the sixth cluster five different motifs are found to be significantly overrepresented. The strongest factors are C1, which is involved in anthocyanin regulation [90], and LIM1, which is involved in lignin biosynthese [49].

8.6 Application of INCLUSive

Since we have made our applications available through both a web interface and later also a stand alone program, other researchers, mainly biologist, have been using these programs extensively. That our tools have been used extensively is shown in Figure 8.7 where the evolution of the number of hits per month is given for AQBC and MotifSampler. The first version of the MotifSampler web interface was started in November 2000 and ran until February 2002 when we updated the server to the next version of the program. During the two and a half year period that the MotifSampler has been online, more than 10,000 hits by almost 600 different users were registered. The web interface of AQBC was created a few months later and has been used less extensively than the MotifSampler site. We counted almost 2000 hits by 190 users from January 2001 till March 2003.

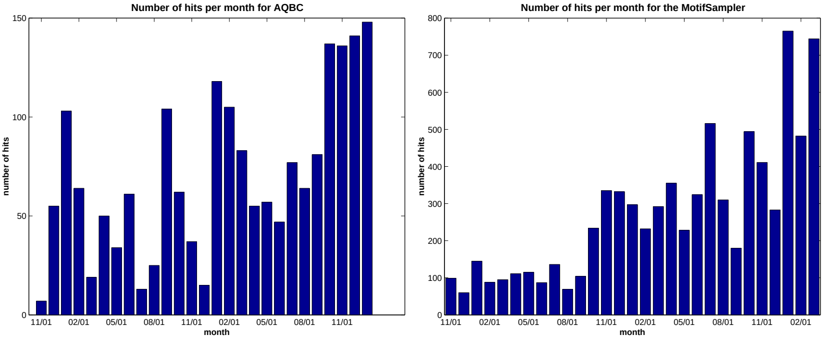


Figure 8.7: User statistics of INCLUSive. Evolution of number the number of hits per month for AQBC (left) and MotifSampler (right).

Most of the users just try out the software once or twice. However, some of these users, could use the results obtained with the MotifSampler to publish their work. In the next paragraphs, we give a short overview of the results these researchers have obtained with MotifSampler on their specific data sets. There are results from the analysis of microarray data in yeast and *Arabidopsis*. The MotifSampler has also been applied for large scale promoter analysis in *Drosophila*.

YRR1 transcription factor regulation system

Le Crom et al. [56] studied the YRR1 transcription factor that is involved in the pleiotropic drug resistance (PDR) phenomenon. They have used the web interface of MotifSampler to analyze a set of 15 genes shown to be regulated by YRR1. They have characterized in seven genes, up-regulated and involved in a specific shift, a core consensus sequence `wCCGykkww`. This consensus resembles the PDR element binding site, although the second half of the consensus sequence is less well conserved.

Gene expression in response to high light

Rossel et al. [87] performed microarray and northern-blot experiments to identify the transcription factors and signaling pathways that enable plants to limit oxidative damage caused by exposure to high light. To determine if the genes that were differentially expressed by high light stress were under similar transcriptional control, they analyzed the promoter sequences of these specific genes. Using the MotifSampler web interface and by comparing the motifs found with those stored in PlantCARE, they could identify several potential motifs.

Gene expression in response to environmental stress

Chen et al. [21] conducted microarray experiments to deduce the functions of genes encoding known and putative transcription factors involved in response to environmental stress. The mRNA levels of 402 distinct transcription factor genes in *A.thaliana* were examined at different developmental stages and under various stress conditions. Transcription factors potentially controlling downstream gene expression in stress signal transduction pathways were identified by observed activation and repression of the genes after certain stress treatments. Statistical analysis of the promoter regions of the genes responsive to cold stress indicated unambiguous enrichment of known conserved transcription factor binding sites. They also identified a highly conserved novel promoter motif in genes responding to a broad set of pathogen infection treatments.

Low-oxygen response in Arabidopsis root cultures

Klok et al. [51] identified 210 differentially expressed genes over the four time points after transfer to low-oxygen conditions. These genes include the known anaerobic proteins as well as transcription factors, signal transduction components and genes that encode enzymes of pathways not known previously to be involved in low-oxygen metabolism. Analysis of their respective promoter regions with MotifSampler revealed common binding sites.

Analysis of Unfolded Protein Response

Recently, Martinez and Chrispeels [72] have analyzed with gene expression data the breadth of the unfolded protein response in Arabidopsis. By using stress inducing agents they could identified genes that are repressed or induced by this stress. The genes that are induced are mostly genes that encode proteins that create an optimal protein-folding environment, when the there is temporary abundance of unfolded proteins. The downregulated genes mostly encode for extracellular proteins. Promoter analysis revealed that half of the upregulated genes contains regulatory elements that resembles UPR response elements in mammalians.

Promoter analysis in the *drosophila* genome

The work of Ohler et al. [80] is part of a series of papers published in *Genome Biology*[5] describing Release 3 of the *Drosophila* genome. In this particular paper, Ohler et al. [80] analyzed the promoter region of 2000 genes of which they could identify the transcription start site. They searched in the region -60 to +40 around TSS for overrepresented motifs with MEME [7] and the stand alone version of MotifSampler. Comparison of the results showed that both methods gave comparable results.

8.7 Conclusions

After the discussion of the theoretical and algorithmic aspects of our work in the previous chapters, we introduce in this chapter the practical implementation of the developed algorithms. One of the goals of our research was to design an algorithm that is applicable in real biological examples. This chapter illustrates how we provide the tools developed within our research to the interested user. First the software was only accessible through a user friendly web interface and later we also made stand alone C++ programs of our algorithms.

We illustrate the complete process from clustering expression data to motif search with the data from a small microarray experiment in Arabidopsis. With

---

[5] `http://www.genomebiology.com/`

adaptive quality-based clustering we extract six small clusters of genes from this data set. Next, we use the modules in INCLUSive to extract the intergenic region of these genes. In these six sets of upstream sequences we search for motifs with both MotifSampler and MotifScanner. With each test we could identify a few interesting motifs that are related to factors that are involved in the plant defense system.

Finally, to show that we managed to accomplish the goal of an applicable tool. The user statistics of the INCLUSive website show that the number of users has constantly grown over the last two and a half year. More importantly, we could also refer to the work of other researchers who have been working on the analysis of their specific set of promoter sequences. These researchers were able to find with MotifSampler relevant motifs that are related to the process that they have been investigating.

CHAPTER 9

Conclusions

To end this dissertation, we like to summarize our accomplishments and to give some ideas on future research directions.

9.1 Accomplishments

The main goal of our research was to develop a suite of robust and user-friendly tools for the detection of transcription factor binding sites starting from the analysis of expression data. The robustness criterion stems from the fact that microarray data are very noisy. Our algorithms should be able to cope with this noise that is propagated through the clustering to the creation of the upstream sequence set. Due to the nature of the experiments, a cluster might contain genes that are not coregulated. This implies that these genes are not activated or repressed by the common transcription factor. The other criterion for our algorithms is their user-friendliness. This implies that the algorithms should have only a limited number of parameters and if a parameter needs to be defined by the user, it should be easy to understand.

MotifSampler

To accomplish the goal to design a method to detect overrepresented motifs in the upstream region of coexpressed genes we opted to extend the basic Gibbs sampling algorithm for motif finding. The first step was to introduce higher-order background models in this Gibbs sampling framework. In Section 5.1 we introduced higher-order Markov models in the probabilistic sequences model. The second extension to the original algorithm was a procedure to estimate the number of instances of a motif in the sequence. Adding these two extensions to the Gibbs sampler resulted in a first workable implementation. This first version was used to explore the influence of the different parameters on the performance of our algorithm. We were especially interested what the convergence behavior was of algorithm and how the resulting motif models depended on the parameters. From this extended analysis, we gained sufficient insight in the algorithm to develop a general procedure to apply our Gibbs sampling method in real test cases. This resulted in a practical implementation called MotifSampler.

To illustrate the applicability and performance of our algorithm, we used four cases: (1) G-box binding sites in *Arabidopsis*, (2) ten sets of coregulated genes in yeast, (3) σ^{54} factor in *E.coli* and *P.aeruginosa*, and (4) four clusters of coexpressed genes identified in the cell cycle of yeast. Each of these data sets was used to zoom in on particular aspects of the motif finding problem. The G-box data set was used to study the accuracy of our algorithm in retrieving the correct motif instances. The same data set together with a set of random sequences was used to asses the robustness of our algorithm under noisy conditions. The yeast regulons allowed us to test how well the algorithm was able to detect the motifs hidden in the data set and how this was related to the type of motif. In the study with the prokaryotic data set, the focus was on the influence of the background model on the detection process. The final test case was a set of four clusters identified with AQBC in the yeast cell cycle. Each of these tests gave us satisfactory results. Using higher-order background models built from an independent data set significantly increases the robustness of the algorithm to noise in the data set. But, we should be careful in selecting the right background model. The study of the σ^{54} factor in *E.coli* and *P.aeruginosa* showed that when a background model is used that does not correspond to the sequence in which the binding sites are hidden that the results could be completely wrong. The analysis of the ten regulons showed us that we can identify different types of motifs. Even a long motif as GAL4 that contains a large uninformative part could be retrieved. Other shorter motifs were much harder to find. Finally, the analysis of four clusters of coexpressed genes, identified in the yeast cell cycle, gave mixed results. In two clusters we could identify binding sites that could be related to the expression profile of the respective clusters. In one cluster two specific motif were identified that also have been identified by other researchers. In the last cluster we were not able to find any overrepresented motif that resembles a potential TFBS.

MotifScanner & MotifLocator

Sometimes we are also interested in finding the instances of a motif in a new sequence. We have developed two methods to solve this problem. The first one, MotifLocator, is based on the classical scoring scheme with a PWM and the second one is based on the probabilistic sequence model introduced within the framework of the Gibbs sampling procedure. Inside MotifLocator we use a background adapted PWM scoring scheme. Instead of scoring only with the matrix we also take the background score into account. The score of a site is computed as the logarithm of the ratio of the site scored by the motif model and the site scored by the background model. These scores are then normalized and a threshold is defined to make the final selection. Within MotifScanner the expected number of motif instances is computed using the probabilistic sequence model, that we also used inside MotifSampler. Although both methods have the same purpose and do share some computational components, their behavior is significantly different. To illustrate the difference we examined several data sets. We especially analyzed two large collections of promoter sequences in yeast and human. At first we looked at the nucleotide composition in the upstream region. This analysis showed that in yeast the composition is almost constant over the selected region of 800bp. In human the composition was completely different in the region close to the start and further downstream. This implied that we better split the sequences and study the regions separately.

The methodology described in this chapter could also be used to detect overrepresented motifs in sets of coregulated genes. To define statistical significance a binomial model is used to compute how likely (or unlikely) it is to find the counted number of instances in the given data set compared to the expected number of instances. The expected number of instances could be found by screening a large reference set. This methodology has been developed concurrently with the implementation of *Toucan* [1]. In this text we applied the method on the ten yeast regulons and also on the four clusters found in the yeast cell cycle. In both cases, using this methodology we were able to identify the known motifs in most of the data sets. The main reason the method did not work on all data set is that the motifs stored in databases like SCPD and TRANSFAC do not cover all possible transcription factors in yeast.

INCLUSive

In the final part of this text we describe the integrated online framework **INCLUSive** for the analysis of microarray data and regulatory sequences. One of the main objectives of our research was to develop a tool that would allow users to easily identify the potential TFBS within their set of coexpressed genes. To facilitate the access to our algorithm we have designed a web interface to these tools. INCLUSive is an ever evolving system and can be accessed at the following page:

`http://www.esat.kuleuven.ac.be/~dna/BioI/Software.html`

INCLUSive starts with the clustering of expression data using AQBC. The user receives a clear results page on which the profiles of the genes in a cluster are shown together with the identifiers of the genes in the cluster. From this page the guided upstream sequence retrieval system can be started. This results in a set of upstream sequences in which the overrepresented binding sites can be located.

9.2 Future research directions

The algorithms as we presented them in this dissertation are only a small part in the deciphering of the *cis*-regulatory logic. The primary goal of these tools is to assist the biologist in gaining insight in the functioning of the cellular processes. In this respect they are only part of the analysis procedure as it was outlined in the introduction in Figure 1.1. In this respect is the ultimate goal the biological validation of the *in silico* results to close the loop of the analysis. In Chapter 8 we could already refer to several papers where the MotifSampler has been used by others to analyze their data. But there are also still some open problems with respect to the algorithm aspects of our approach.

Advanced motif models and combinations of factors

In the introduction the different types of transcription factor classes were described and some of these transcription factors do not bind to one continuous site but rather two sites separated by a spacer. Although the MotifSampler can handle such motifs existing of two conserved blocks separated by a spacer of fixed length (see GAL4 example in Section 6.2), it is not possible to extract motifs that are a combination of two blocks separated by a spacer that might be variable. An algorithm to search for this type of motifs should simultaneously search for the two conserved blocks and also find the length of the spacer that separates them.

Another very important aspect we have addressed in the biological outline of gene regulation is that the great diversity of cells in higher organisms does not come from the increasing number of genes but rather from the growing complexity of the system controlling the gene expression. When dealing with data from such a higher organism, it is no longer sufficient to only search for individual binding sites. It becomes more and more important to search for combinations or modules of motifs that are overrepresented within a specific sequence set. A possible approach is to search for known or unknown motifs with the proposed methods. In the next step we could try to identify patterns of co-occurrence of certain motifs. Using a combinatorial search procedure it should be possible to identify these patterns. Another approach could be to directly work on the probabilistic sequence model. The probabilistic framework should allow us to incorporate multiple motif models in the basic equation of the sequence model and from there recompute the predictive update formula of the Gibbs sampler. GuhaThakurta and Stormo [37] have introduced a similar approach with *CoBind* to search for two cooperatively binding factors. Practical issues in both approaches are the definition of the number of motif models that should make up the module, the order of the motifs in the module and how to define a score for a module. Also, the range in which factors are expected to interact with each other is still an unsolved issue.

Phylogenetic footprinting

Given the increasing availability of newly sequenced genomes, cross-species comparison becomes a more important aspect of bioinformatics research. In the Chapter 3 and in the case study in prokaryotes in Chapter 6 we have already touched upon the subject of phylogenetic footprinting. Phylogenetic footprinting is the methodology for the discovery of transcription factor binding sites in a set of orthologous regulatory regions from multiple species. The basic idea is that selective pressure causes functional elements to evolve at a slower rate than nonfunctional sequences. This means that unusually well conserved sites among a set of orthologous regulatory regions might be functional regulatory elements. Applying motif search algorithms to such a set of orthologous promoter sequences should reveal these conserved patterns.

The main problem one needs to solve when using Gibbs sampling for phylogenetic footprinting is that it should be possible to quantify the identity between sequences and use this measure to weight the sequences in the data set. Closely related species will show a high level of sequence conservation while more distant relatives will show only a small level of sequence similarity. This difference in sequence similarity will influence the motif detection process. If the sequence set contains a few very similar sequences, theses sequences might deteriorate the motif detection process severely by evolving to a local optimum that comes from the high level of similarity between these sequences. Finding the right global optimum will be much harder. In such a case a weighting mechanism should take this similarity into account to prevent the algorithm to convergence to the local optimum. A second part in this analysis is to make sure that each sequence is scored with correct background model. As was shown in the study of prokaryotes in Chapter 6, using the wrong background model can really spoil the results of the motif detection algorithm.

Combinations of different sources of information

The growing availability of different types of genomic data, should make it possible for researchers to combine these different sources of information to increase the understanding of the problem at hand. Combining different sources of information can be achieved in many ways. The first efforts to combine microarray data and sequence data have been proposed by Bussemaker et al. [18] and Holmes and Bruno [45] who try to correlate the expression data with the occurrence of specific TFBS in the upstream regions. The first attempts to integrate motif finding algorithms with other technologies have been recently published (eg. Liu et al. [65]). Also in the study of genetic networks will the information about transcription factor help in reconstructing interactions between several components.

Another source of information is textual information. The immense collection of scientific literature is and has been the primary source of anyone who studies any kind of biological process. The availability of these articles in electronic format has allowed researcher to design algorithm that search for information in these text. Recent advances in textmining help to automate the learning process and allow to automatically assign textual information to specific clusters. This will speed up the process of functional annotation and will, hopefully, help researchers to gain new insights in the biological process their studying.

The most recent advances in INCLUSive [24] are already aiming towards a further degree of integration of different information sources. This can be achieved by either direct links to external databases or by using several techniques to automate analysis steps. For instance, in the original version of INCLUSive the selection of upstream sequences was done by blasting the gene on the genomic DNA and parse this sequences to find the annotation. Recently, several genome database are now directly accessible and allow the direct retrieval of upstream sequences based on gene information, this evades the current procedure and will give more reliable results.

In this thesis we presented an integrated methodology to study *in silico* transcription factor binding sites. We hope that our work has been a positive contribution to decipher the transcriptional regulatory mechanism.

Appendix A. Glossary

This glossary has been constructed based upon information extracted from the books by Alberts et al. [2], Brown [14] and Lodish et al. [66].

5’ and 3’ Symbols used to indicate the start (5’) and end (3’) of a DNA or amino acid sequence.

activator DNA-binding protein that stabilizes the construction of the RNA polymerase complex.

alignment The process of lining up two or more sequences to achieve maximal levels of identity between sequences.

amino acid A monomeric unit of protein. There are 20 different amino acids which can be found in Table 2.1.

base pair The shortest unit of length in a double-stranded DNA sequence formed by two hydrogen-bounded complementary nucleotides.

BLAST Basic local alignment search tool. This program has been and still is the working horse of almost everybody who works with DNA and protein sequences. BLAST searches homologous sequences in a sequence database. (`http://www.ncbi.nlm.nih.gov/BLAST/`)

binding site short DNA sequence where a transcription factor can bind to the DNA.

cDNA a double-stranded DNA copy of an mRNA molecule.

cDNA microarray High-density array of spotted cDNA clones used for high-throughput parallel hybridization analysis.

cell cycle Events that happen in a cell between one division and the next.

central dogma Term invented by Francis Crick to refer to the flow of information from DNA to RNA to protein.

chromatin The complex of DNA and histones that is found in chromosomes.

chromosome Compact DNA-protein structure that contains part of the nuclear genome of a eukaryote. Sometimes this term is also used to refer to the DNA molecule that contains the prokaryotic genome.

codon Triplet of nucleotides that occurs in mRNA that is translated into a specific amino acid or that represents the starting or termination signals of protein synthesis (see Table 2.1).

consensus A nucleotide sequence that represents an ’average’ of a set of similar sequences.

DNA Deoxyribonucleic acid; contains the genetic material of a living organism.

DNA chip High-density array of oligonucleotide probes used for high-throughput parallel hybridization analysis.

double helix Base-paired double-stranded structure that is the natural structure of DNA, first characterized by Watson and Crick [116].

enhancer A regulatory sequence, located some distance away from the gene, where a transcription factor can bind that increases the rate of transcription.

Ensembl Interface developed at the EBI to browse several fully sequenced and annotated genomes. Currently (spring 2003) contains sequences and annotation of human, mouse, rat, zebrafish, fugu, fruitfly, mosquito, *C.elegans* and *C.briggsae*. (`http://www.ensembl.org/`)

eukaryote An organism whose cells contain membrane-bound nuclei.

exon Section of DNA that forms the active mRNA and is translated to the amino acid sequence.

GenBank Primary sequence database located at NCBI and shares information daily with DDBJ and EMBL. GenBank is an annotated collection of all publicly available DNA sequences. There are also links to other sources of information like SwissPrott, LocusLink, UniGene. (`http://www.ncbi.nlm.nih.gov/Genbank/`)

gene expression Series of events by which the biological information present in a gene is made available to the cell.

genetic code This code indicates which triplet codes for which amino acid (see also Table 2.1).

histone Basic protein found in nucleosome that is needed to compact DNA.

homologous genes Genes that share a common evolutionary ancestor.

HUGO Human Genome Organization. Group of publicly funded scientists involved in the Human Genome Project to map and sequence the human genome.

intron An intervening section of DNA which occurs almost exclusively within a eukaryotic gene and which is not translated to the amino acid sequence.

microarray High-density array used for high-throughput parallel hybridization analysis.

mitochondrion One of the energy-generating organelles present in eukaryotic cells.

model organism Organism that is relatively easy to study and to manipulate and that hence can be used to obtain relevant information about other organisms that are much harder to study. Typical model organisms are yeast, the nematode *C.elegans*, *A.thaliana* and mouse.

mRNA An RNA molecule that transfers the coding information for protein synthesis and is formed from a DNA template by transcription.

nucleosome The complex of histones and DNA that is the basic structural unit in chromatin.

nucleotide The monomeric unit of DNA or RNA. It is a purine or pyrimidine base attached to a five-carbon sugar to which also a mono-, di- or triphosphate is attached.

nucleus The membrane-bound structure of a eukaryotic cell in which the chromosome are contained.

oligonucleotide A short synthetic single stranded DNA molecule.

open reading frame A series of codons starting with an initiation codon and ending with a termination codon.

orthologous genes Homologous genes that are located in the genomes of different organisms.

phylogeny Classification of the organism according to their evolutionary relationships.

polypeptide A polymer of amino acids.

prokaryote An organism whose cells lack a distinct nucleus.

promoter Region on the DNA, just upstream of a gene, where the RNA polymerase binds to initiate transcription.

protein A linear polymeric compound built from amino acid monomers.

primary transcript The initial product of the transcription of one or a group of genes.

regulatory element Specific section on the DNA where a transcription factor binds to the DNA to control the gene expression.

regulon Set of promoter sequence s of genes known to be regulated by the same transcription factor.

RNA Ribonucleic acid.

RNA polymerase An enzyme that synthesizes RNA on a DNA or RNA template.

silencer A regulatory sequence, located some distance away from the gene, where a transcription factor can bind that reduces the rate of transcription.

splicing The process where introns are removed from the primary transcript.

transcription The process of the synthesis of an RNA copy of a gene.

transcription factor DNA-binding protein that controls the rate of transcription.

translation The process that takes place on the ribosomes whereby the genetic information present in an mRNA, codon by codon, is converted into a corresponding sequence of amino acids to form a protein.

Bibliography

[1] Aerts, S., Thijs, G., Coessens, B., Staes, M., Moreau, Y., and De Moor, B. (2003). Toucan: Deciphering the *cis*-regulatory logic of coregulated genes. *Nucleic Acids Res*, 31(6):1753–1764.

[2] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., and Watson, J. D. (1994). *Molecular Biology of the Cell*. Garland Publishing, Inc., New York, USA, 3rd edition.

[3] Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., and Levine, A. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci USA*, 96:6745–6750.

[4] Altman, R. and Raychaudhuri, S. (2001). Whole-genome expression analysis: challenges beyond clustering. *Curr Opin Struct. Biol*, 11:340–347.

[5] Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402.

[6] Audic, S. and Claverie, J.-M. (1997). Detection of eukaryotic promoters using Markov transition matrices. *Comput Chem*, 21(4):223–227.

[7] Bailey, T. L. and Elkan, C. (1995a). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51–80.

[8] Bailey, T. L. and Elkan, C. (1995b). The value of prior knowledge in discovering motifs with MEME. In *Proc. 5th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 3, pages 21–29.

[9] Barrios, H., Valderrama, B., and Morett, E. (1999). Compilation and analysis of σ^{54}-dependent promoter sequences. *Nucleic Acids Res*, 27:4305–4313.

[10] Berg, O. and von Hippel, P. (1987). Selection DNA binding sites by regulatory proteins. *J Mol Biol*, 193:723–750.

[11] Berman, B., Nibu, Y., Pfeiffer, B., Tomancak, P., Celniker, S., Levine, M., Rubin, G., and Eisen, M. (2002). Exploiting transcription factor binding site clustering to identify *cis*-regulatory modules involved in pattern fromation in *Drosophila* genome. *Proc Natl Acad Sci*, 99(2):757–762.

[12] Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E., Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., Sampas, N., Dougherty, E., Wang, E., Marincola, F., Gooden, C., Lueders, J., Glatfelter, A., Pollock, P., Carpten, J., Gillanders, E., Leja, D., Dietrich, K., Beaudry, C., Berens, M., Alberts, D., and Sondak, V. (2000). Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406:536–540.

[13] Blanchette, M., Schwikowski, B., and Tompa, M. (2000). An exact algorithm to identify motifs in orthologous sequences from multiple species. In *Proc. 8th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 8, pages 37–45, San Diego, CA, USA.

[14] Brown, T. (2002). *Genomes (2nd edition)*. Bios Scientific Publishers, Oxford, UK.

[15] Bucher, P. (1990). Weight matrix description of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J Mol Biol*, 212:563–578.

[16] Bucher, P. (1999). Regulatory elements and expression profiles. *Curr Opin Struct. Biol*, 9:400–407.

[17] Bussemaker, H., Li, H., and Siggia, E. (2000). Regulatory element detection using a probabilistic segmentation model. In *Proc. 8th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 8, pages 67–74, San Diego, CA, USA.

[18] Bussemaker, H. J., Li, H., and Siggia, E. D. (2001). Regulatory element detection using correlation with expression. *Nature Genet*, 27:167–71.

[19] Cardon, L. and Stormo, G. (1992). Expectation maximization for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *J Mol Biol*, 223:159–170.

[20] Caselle, M., Punto, F. D., and Provero, P. (2002). Correlating overrepresented upstream motifs to gene expression: a computational approach to regulatory element discovery in eukaryotes. *BMC Bioinformatics*, 3(1):7.

[21] Chen, W., Provart, N., Glazebrook, J., Katagiri, F., Chang, H.-S., Eulgem, T., Mauch, F., Luan, S., Zou, G., Whitham, S., Budworth, P., Tao, Y., Xie, Z., Chen, X., Lam, S., Kreps, J., Harper, J., Si-Ammour, A., Mauch-Mani, B., Heinlein, M., Kobayashi, K., Hohn, T., Dangl, J., Wang, X., and Zhu, T. (2002). Expression profile matrix of Arabidopsis transcription factor genes suggests their putative functions in response to environmental stresses. *Plant Cell*, 14(3):559–574.

[22] Cho, R., Campbell, M., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell*, 2:65–73.

[23] Choi, H., Hong, J., Ha, J., Kang, J., and Kim, S. Y. (2000). ABFs, a family of ABA-responsive element binding factors. *J Biol Chem*, 275(3):1723–1730.

[24] Coessens, B., Thijs, G., Aerts, S., Mathys, J., Moreau, Y., Marchal, K., De Smet, F., Engelen, K., Glenisson, P., and B. De Moor (2003). Inclusive– A web portal and service registry for microarray and regulatory sequence analysis. *Nucleic Acids Res*. in press.

[25] Cook, T., Gebelein, B., and Urrutia, R. (1999). Sp1 and its likes: Biochemical and functional predictions for a growing family of zinc finger transcription factors. *Ann NY Acad Sci*, 880:94–102.

[26] De Smet, F., Mathys, J., Marchal, K., Thijs, G., De Moor, B., and Moreau, Y. (2002). Adaptive quality-based clustering of gene expression profiles. *Bioinformatics*, 18(5):735–746.

[27] Delcher, A., Harman, D., Kasif, S., White, O., and Salzberg, S. (1999). Improved microbial gene identification with glimmer. *Nucleic Acids Res*, 27(23):4636–4641.

[28] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via EM algorithm. *J Royal. Stat. Soc.*, 39(1):1–38.

[29] DeRisi, J., Iyer, V., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686.

[30] Donald, R. and Cashmore, A. (1990). Mutation of either G-box and I-box sequences profundly affects expression from the *Arabidopsis rbsC*-1A promoter. *EMBO J*, 9:1717–1726.

[31] Eisen, M., Spellman, P., Brown, P., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA*, 95:14863–14868.

[32] Fondrat, C. and Kalogeropoulos, A. (1997). Approaching the function of new genes by detection of their potential upstream activation sequences in saccharomyces cerevisiae: application to chromosome III. *Bioinformatics*, 12(5):363–374.

[33] Frith, M., Spouge, J., Hansen, U., and Weng, Z. (2002). Statistical significance of motif represented by position spesific scoring matrices in nucleoide sequences. *Nucleic Acids Res*, 30(14):3214–3224.

[34] Gelfand, A. and Smith, A. (1990). Sampling-based approaches to calculating marginal densities. *J Am Stat Assoc*, 85:398–409.

[35] Geman, D. and Geman, S. (1984). Stochastic relaxation, Gibbs distribution and Bayesian restoration of images. *IEEE Transactions PAMI*, 6(6):721–741.

[36] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537.

[37] GuhaThakurta, D. and Stormo, G. (2001). Identifying target sites for cooperatively binding factors. *Bioinformatics*, 17:608–621.

[38] Halfon, M., Grad, Y., Church, G., and Michelson, A. (2002). Computation-based discovery of related transcriptional regulatory modules and motifs using an experimentally validated combinatorial model. *Genome Res*, 12:1019–1028.

[39] Hampson, S., Kibbler, D., and Baldi, P. (2002). Distribution patterns of over-represented k-mes in non-coding yeast. *Bioinformatics*, 18(4):513–528.

[40] Herrero, J., Valencia, A., and Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, 17:126–136.

[41] Hertz, G., Hartzell, G., and Stormo, G. (1990). Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput Appl Biosci*, 6:81–92.

[42] Hertz, G. and Stormo, G. D. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7/8):563–577.

[43] Heyer, L., Kruglyak, S., and Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. *Genome Res*, 9:1106–1115.

[44] Höhl, M., Kurtz, S., and Ohlebusch, E. (2002). Efficient multiple genome alignment. *Bioinformatics*, 18(90001):S312–S320.

[45] Holmes, I. and Bruno, W. (2000). Finding regulatory elements using joint likelihoods for sequence and expression profile data. In *Proc. 8th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 8, pages 202–210, San Diego, CA, USA.

[46] Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., Durbin, R., Eyras, E., Gilbert, J., Hammond, M., Huminiecki, L., Kasprzyk, A., Lehvaslaiho, H., Lijnzaad, P., Melsopp, C., Mongin, E., Pettett, R., Pocock, M., Potter, S., Rust, A., Schmidt, E., Searle, S., Slater, G., Smith, J., Spooner, W., Stabenau, A., Stalker, J.,

Stupka, E., Ureta-Vidal, A., Vastrik, I., and Clamp, M. (2002). The Ensembl genome database project. *Nucleic Acids Res*, 30(1):38–41.

[47] Hughes, J., Estep, P., Tavazoie, S., and Church, G. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in *Saccharomyces cerevisiae*. *J Mol Biol*, 296:1205–1214.

[48] Ishiguro, S. and Nakamura, K. (1994). Characterization of a cDNA encoding a novel DNA-binding protein, SPF1, that recognizes SP8 sequences in the 5' upstream regions of genes coding for sporamin and beta-amylase from sweet potato. *Mol Gen Genet*, 244(6):563–571.

[49] Kawaoka, A., Kaothien, P., Yoshida, K., Endo, S., Yamada, K., and Ebinuma, H. (2000). Functional analysis of tobacco LIM protein Ntlim1 involved in lignin biosynthesis. *Plant J*, 22(4):289–301.

[50] Keich, U. and Pevzner, P. (2002). Finding motifs in the twilight zone. *Bioinformatics*, 18(10):1374–1381.

[51] Klok, E., Wilson, I., Wilson, D., Chapman, S., Ewing, R., Somerville, S., Peacock, W., Dolferus, R., and Dennis, E. (2002). Expression profile analysis of the low-oxygen response in arabidopsis root cultures. *Plant Cell*, 14:2481–2494.

[52] Krogh, A. (1997). Two methods for improving performance of an hmm and their application for gene finding. In *Proc. 5th Intl. Conf. Intelligent Systems in Molecular Biology*, volume 5, pages 179–186, Halkidiki, Greece.

[53] Lawrence, C. and Reilly, A. (1990). An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7:41–51.

[54] Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214.

[55] Lawton, M. A., Dean, S. M., Dron, M., Kooter, J. M., Kragh, K. M., Harrison, M. J., Yu, L., Tanguay, L., Dixon, R. A., and Lamb, C. J. (1991). Silencer region of a chalcone synthase promoter contains multiple binding sites for a factor, SBF-1, closely related to GT-1. *Plant Mol Biol*, 16(2):235–249.

[56] Le Crom, S., Devaux, F., Marc, P., Zhang, X., Moye-Rowley, W., and Jacq, C. (2002). New insights into the pleiotropic drug resistance network from genome-wide characterization of the YRR1 transcription factor regulation system. *Mol Cell Biol*, 22(8):2642 – 2649.

[57] Lescot, M., Déhais, P., Thijs, G., Marchal, K., Moreau, Y., de Peer, Y. V., Rouzé, P., and Rombauts, S. (2002). PlantCARE, a database of plant *cis*-acting regulatory elements and a portal to tools for in silico analysis of promoter sequences. *Nucleic Acids Res*, 30:325–327.

[58] Liang, S., Fuhrman, S., and Somogyi, R. (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architecture. In *Proc. Pacific Symposium on Biocomputing*, volume 3, pages 18–29, Honolulu, Hawai.

[59] Lipschutz, R., Fodor, S., Gingeras, T., and Lockheart, D. (1999). High density synthetic oligonucleotide arrays. *Nature Genet Suppl.*, 21:20–24.

[60] Liu, J. (1994). The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *J Am Stat Assoc*, 89(427):958–966.

[61] Liu, J. (2001). *Monte Carlo strategies in scientific computing*. Springer Series in Statistics. Springer.

[62] Liu, J. and Lawrence, C. (1999). Bayesian inference on biopolymer models. *Bioinformatics*, 15:38–52.

[63] Liu, J., Neuwald, A., and Lawrence, C. (1995). Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J Am. Stat. Assoc.*, 90(432):1156–1170.

[64] Liu, X., Brutlag, D., and Liu, J. (2001). BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. In *Proc. Pacific Symposium on Biocomputing*, volume 6, pages 127–138.

[65] Liu, X., Brutlag, D., and Liu, J. (2002). An algorithm for finding protein-DNA binding sites with application to chromatin-immunoprecipitation microarray experiments. *Nature Biotech*, 20:835–839.

[66] Lodish, H., Baltimore, D., Berk, A., Zipursky, S., Matsudaira, P., and Darnell, J. (1995). *Molecular Cell Biology (3rd edition)*. Scientific American Books, New York, USA.

[67] Lukashin, A. and Borodowsky, M. (1998). Genemark.hmm: new solutions for gene finding. *Nucleic Acids Res*, 26:1107–1115.

[68] Marchal, K., Engelen, K., Brabanter, J. D., Aerts., S., Moor, B. D., Ayoubi, T., and Hummelen, P. V. (2002). Comparison of different methodologies to identify differentially expressed genes in two-sample cdna arrays. *J Biol Sys*, 10(4):409–430.

[69] Marchal, K., Thijs, G., De Keersmaecker, S., Monsieur, P., De Moor, B., and Vanderleyden, J. (2003). Genome-specific higher-order background models to improve motif detection. *Trends in Microbiol*, 11(2):61–66.

[70] Markstein, M. and Levine, M. (2002). Decoding *cis*-regulatory DNAs in the *Drosophila* genome. *Curr Opin Genetics Dev*, 12:601–606.

[71] Marsan, L. and Sagot, M.-F. (2000). Algorithms for extracting structured motifs using a suffix tree with application to promoter and regulatory site consensus identification. *J Comput Biol*, 7:345–360.

[72] Martinez, I. and Chrispeels, M. (2003). Genomic analysis of unfolded protein response in Arabidopsis shows its connection to important cellular processess. *Plant Cell*, 15(2):561–576.

[73] McCue, L., Thompson, W., Carmack, C., Ryan, M., Liu, J., Derbyshire, V., and Lawrence, C. (2001). Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic Acids Res*, 29:774–782.

[74] Moreau, Y., De Smet, F., Thijs, G., Marchal, K., and De Moor, B. (2002). Functional bioinformatics of microarray data: from expression to regulation. *Proceedings of the IEEE*, 90(11):1722–1743.

[75] Nakagawa, H., Ohmiya, K., and Hattori, T. (1996). A rice bZIP protein, designated OSBZ8, is rapidly induced by abscisic acid. *Plant J*, 9(2):217–227.

[76] Ni, M., Tepperman, J. M., and Quail, P. H. (1998). PIF3, a phytochrome-interacting factor necessary for normal photoinduced signal transduction, is a novel basic helix-loop-helix protein. *Cell*, 95(5):657–667.

[77] Nicodème, P. (2001). Fast approximate motif statistics. *J Comput Biol*, 8(3):235–248.

[78] Ogata, K., Sato, K., and Tahirov, T. (2003). Eukaryotic transcriptional regulatory complexes: cooperatively from near and afar. *Curr Opin Struct Biol*, 13:40–48.

[79] Ohler, U., Harbeck, S., Niemann, H., Nöth, E., and Reese, M. (1999). Interpolated markov chains for eukaryotic promotor recognition. *Bioinformatics*, 15(5):362–369.

[80] Ohler, U., Liao, G., Niemann, H., and Rubin, G. (2002). Computational analysis of core promoters in the *Drosophila* genome. *Genome Biology*, 3(12):research0087.1–research0087.12.

[81] Orphanides, G. and Reinberg, D. (2002). A unified theory of gene expression. *Cell*, 108:439–451.

[82] Pavy, N., Rombauts, S., Déhais, P., Mathé, C., Ramana, D., Leroy, P., and Rouzé, P. (1999). Evaluation of gene prediction software using a genomic data set: Application to *Arabidopsis thaliana* sequences. *Bioinformatics*, 15:887–899.

[83] Pe'er, D., Regev, A., Elidan, G., and Friedman, N. (2001). Inferring subnetworks from perturbed expression profiles. In *Proc. 9th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 9, pages 243–252, Kopenhagen, Denmark.

[84] Perou, C. M., Sorlie, T., Eisen, M. B., van de. Rijn., M., Jeffrey, S. S., Rees, C. A., Pollack, J. R., Ross, D. T., Johnsen, H., Akslen, L. A., Fluge, O., Pergamenschikov, A., Williams, C., Zhu, S. X., Lonning, P. E., Borresen-Dale, A. L., Brown, P. O., and Botstein, D. (2000). Molecular portraits of human breast tumours. *Nature*, 406(6797):747–752.

[85] Pevzner, P. and Sze, S.-H. (2000). Combinatorial approaches to finding subtle signals in DNA sequences. In *Proc. 8th Intl. Conf. Intelligent Systems in Molecular Biology*, volume 8, pages 269–278, San Diego, CA, USA. AAAI press.

[86] Reymond, P., Weber, H., Damond, M., and Farmer, E. E. (2000). Differential gene expression in response to mechanical wounding and insect feeding in Arabidopsis. *Plant Cell*, 12:707–719.

[87] Rossel, J., Wilson, I., and Pogson, B. (2002). Global changes in gene expression in response to high light in arabidopsis. *Plant Physiology*, 130:1109–1120.

[88] Roth, F., Hughes, J., Estep, P., and Church, G. (1998). Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole genome mRNA quantitation. *Nature Biotech.*, 16:939–945.

[89] Sagot, M. and Myers, E. (1998). Identifying satellites and periodic repetitions in biological sequences. *J Comput Biol*, 5:539–553.

[90] Sainz, M. B., Grotewold, E., and Chandler, V. L. (1997). Evidence for direct activation of an anthocyanin promoter by the maize C1 protein and comparison of DNA binding by related Myb domain proteins. *Plant Cell*, 9(4):611–625.

[91] Schena, M., Shalon, D., Davis, R., and Brown, P. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470.

[92] Schiex, T., Moisan, A., Duret, L., and Rouzé, P. (1999). Eugène: a simple yet effective gene finder for eucaryotic organisms (Arabidopsis thaliana). In *Proc of 2nd Georgia Tech International Conference on Bioinformatics - In silico Biology*, volume 2, Atlanta, USA.

[93] Schneider, T. and Stephens, R. (1990). Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res*, 18(20):6097–6100.

[94] Schneider, T., Stormo, G., Gold, L., and Ehrenfeucht, J. (1986). Information content of binding sites on nucleotide sequences. *J Mol Biol*, 188:415–431.

[95] Segal, E., Taskar, B., Gasch, A., Friedman, N., and Koller, D. (2001). Rich probabilistic models for gene expression. In *Proc. 9th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 9, pages 243–252, Kopenhagen, Denmark.

[96] Sinha, S. and Tompa, M. (2000). A statistical method for finding transcription factor binding sites. In *Proc. 8th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 8, pages 37–45, San Diego, CA, USA.

[97] Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast S. Cerevisiae by microarray hybridization. *Mol Biol Cell*, 9:3273–3297.

[98] Stajich, J., Block, D., Boulez, K., Brenner, S., Chervitz, S., Dagdigian, C., Fuellen, G., Gilbert, J., Korf, I., Lapp, H., Lehväslaiho, H., Matsalla, C., Mungall, C., Osborne, B., Pocock, M., Schattner, P., Senger, M., Stein, L., Stupka, E., Wilkinson, M., and Birney, E. (2002). The Bioperl toolkit: Perl modules for the life sciences. *Genome Res*, 12:1611–1618.

[99] Stormo, G. D. (1990). Consensus patterns in DNA. In *Methods Enzymol.*, volume 183, chapter 14, pages 211–221. Academic Press.

[100] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E., and Golub, T. R. (1999). Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc Natl Acad Sci USA*, 96:2907–2912.

[101] Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. with discussion and with a reply by the authors. *J Am. Stat. Assoc.*, 82(398):528–550.

[102] Tavazoie, S., Hughes, J., Campbell, M., Cho, R., and Church, G. (1999). Systematic determination of genetic network architecture. *Nature Genet*, 22(7):281–285.

[103] The International Human Genome Sequencing Consortium (2001). Initial sequencing and analysis of the human genome. *Nature*, 409:860–921.

[104] Thijs, G., Lescot, M., Marchal, K., Rombauts, S., De Moor, B., Rouzé, P., and Moreau, Y. (2001a). A higher order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics*, 17(12):1113–1122.

[105] Thijs, G., Marchal, K., Lescot, M., Rombauts, S., De Moor, B., Rouzé, P., and Moreau, Y. (2001b). A Gibbs sampling meyhod to detect overrepresented motifs in the upstream regions of co-expressed genes. In *Proc. 5th Annual Intl. Conf. on Computational Biology*, volume 5, pages 305–312. ACM press.

[106] Thijs, G., Marchal, K., Lescot, M., Rombauts, S., De Moor, B., Rouzé, P., and Moreau, Y. (2002a). A Gibbs sampling meyhod to detect overrepresented motifs in the upstream regions of co-expressed genes. *J Comput Biol*, 9(2):447–464.

[107] Thijs, G., Moreau, Y., Rombauts, S., De Moor, B., and Rouzé, P. (1999). Recognition of gene regulatory sequences by bagging of neural networks. In *Proc Intl. Conf. on Artificial Neural Networks*, volume 2, pages 988–993.

[108] Thijs, G., Moreau, Y., Smet, F. D., Mathys, J., Lescot, M., Rombauts, S., Rouzé, P., De Moor, B., and Marchal, K. (2002b). INCLUSive: INtegrated CLustering, Upstream sequence retrieval and motif Sampling. *Bioinformatics*, 18(2):331–332.

[109] Tolstrup, N., Rouzé, P., and Brunak, S. (1997). A branch point consensus from Arabidopsis found by non-circular analysis allows for better prediction of acceptor sites. *Nucleic Acids Res*, 25(15):3159–3163.

[110] Tompa, M. (1999). An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *Proc. 7th Intl. Conf. Intelligent Systems for Molecular Biology*, volume 7, pages 262–271, Heidelberg, Germany.

[111] van Helden, J., André, B., and Collado-Vides, L. (1998). Extracting regulatory sites from upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J Mol Biol*, 281:827–842.

[112] van Helden, J., André, B., and Collado-Vides, L. (2000a). A web site for the computational analysis of yeast regulatory sequences. *Yeast*, 16:177–187.

[113] van Helden, J., Rios, A., and Collado-Vides, J. (2000b). Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res*, 28(8):1808–1818.

[114] Vanet, A., Marsan, L., Labigne, A., and Sagot, M. (2000). Inferring regulatory elements from a whole genome. an analysis of helicobacter pylori $sigma^{80}$ family of promoter signals. *J Mol Biol*, 297(2):335–353.

[115] Venter, J., Adams, M., Myers, E., and et al. (2001). The sequence of the human genome. *Science*, 291:1304–1351.

[116] Watson, J. and Crick, F. (1953). Molecular structure of nucleic acids: A structure for Deoxyribose Nucleic Acid. *Nature*, 171:737–738.

[117] Wingender, E., Chen, X., Fricke, E., Geffers, R., Hehl, R., Liebich, I., Krull, M., Matys, V., Michael, H., Ohnhäuser, R., Prüß, M., Schacherer, F., Thiele, S., and Urbach, S. (2001). The TRANSFAC system on gene expression regulation. *Nucleic Acids Res*, 29(1):281–283.

[118] Wolfsberg, T., Gabrielian, A., Campbell, M., Cho, R., Spouge, J., and Landsman, D. (1999). Candidate regulatory sequence elements for cell cycle-dependent transcription in Saccharomyces Cerevisiae. *Genome Res*, 9:775–792.

[119] Workman, C. and Stormo, G. (2000). Ann-spec: a method for discovering transcription binding sites with improved specificity. In *Proc. Pacific Symposium on Biocomputing*, volume 5, pages 464–475, Honolulu, Hawai.

[120] Wu, C. (1997). Artificial neural networks for molecular sequence analysis. *Comput Chem*, 21(4):237–256.

[121] Yeung, K., Fraley, C., Murua, A., Raftery, A., and Ruzzo, W. (2001). Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17:977–987.

[122] Zhang, M. (1998). Promoter analysis of co-regulated genes in yeast genome. *Comput Chem*, 23:233–250.

[123] Zhang, M. (1999). Large-scale gene expression data analysis: A new challenge to computational biologists. *Genome Res*, 9:681–688.

[124] Zhu, J. and Zhang, M. (1999). SCPD: a promoter database of the yeast Saccharomyces cerevisiae. *Bioinformatics*, 15(7):607–611.

Curriculum Vitae

Gert Thijs, born in Bilzen, Belgium, on June 1, 1973, received the Master's degree in electrical engineering in 1998 at the Katholieke Universiteit Leuven, Belgium. In September 1998, he joined the ESAT-SCD group at the Department of Electrical Engineering (ESAT) at the same university. From September 1998 till December 1999 he was a research assistant of the Katholieke Universiteit Leuven. Since January 2000 he has been a research assistant with the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT-Vlaanderen). He was an invited speaker at the EMBS Satellite Symposium on Bioinformatics in Instanbul, Turkye. He has also acted as a reviewer for several articles in *Bioinformatics*. Currently, he is a member of IEEE, ISCB, KVIV and LeuvenInc.