Vergleichende Untersuchung zur Verbesserung von Methoden der Genvorhersage

Diplomarbeit
vorgelegt von

Maike Tech
aus Lübeck

angefertigt
am Institut für Mikrobiologie und Genetik
der Georg-August-Universität zu Göttingen
2003

Referent : Prof. Dr. G. Gottschalk
Korreferent : Prof. Dr. B. Morgenstern
Tag der Abgabe der Diplomarbeit : 15. Mai 2003
Letzter Tag der mündlichen Prüfung : 27. Juni 2002

Verwendete Abkürzungen und fremdsprachliche Ausdrücke

AS	**A**mino**s**äure
BLAST	**B**asic **L**ocal **A**lignment **S**earch **T**ool
BLASTN	BLAST-Unterprogramm (s.o.) zum Datenbank-Abgleich auf Nucleotidebene
BP	**B**ase **P**air (Basenpaar)
Crit	CRITICA
Default	Standard
DNA	**D**eoxyribo**n**ucleic **A**cid (deutsch: DNS - **D**esoxyribo**n**ucleins**ä**ure)
DPS	**D**NA-**P**rotein **S**earch Program
downstream	in 3'-Richtung (deutsch: stromabwärts)
File	Datei
FN	**F**alse-**N**egative (Falsch-Negative)
FP	**F**alse-**P**ositive (Falsch-Positive)
GL2.02	GLIMMER2.02
GL2.10	GLIMMER2.10
Hardcodiert	im Quellcode festgelegt, nicht dynamisch
High-Scoring	Treffer mit hohem Scorewert
HSP	**H**igh-scoring **S**egment **P**air (Ausdruck für locale BLAST-Alignments mit hohem Score und ohne Lücken)
HSS	**H**igh-**S**coring **S**egment
HTML	**H**yper**T**ext **M**akeup **L**anguage
Frame-Shift	Verschiebung im Leserahmen
Libraries	Bibliotheken (verwendet auch für Programmbibliotheken)
Linux-Cluster	Zusammenschluß mehrerer Rechner unter dem Betriebsystem Linux

Verwendete Abkürzungen und fremdsprachliche Ausdrücke

Manual	Bedienungsanleitung
Multiprozessor-System	System mit mehr als einem Prozessor
MBP	**m**ega **b**ase **p**air
ORF	**O**pen **R**eading **F**rame (offener Leserahmen)
Over-Prediction	Vorhersage von False-Positiven
PBS	**P**ortable **B**atch **S**ystem: System zur Organisation der Zuteilung von Ressourcen an einzelne Prozesse
RBS	**R**ibosom-**B**inding **S**ite (Ribosom-Bindestelle)
Readme-Datei	Datei, die Beschreibungen des zugehörigen Programms und Hilfen zur Benutzung enthält
SC	**S**tart-**C**orrectness (Abschnitt 3.2, S. 64)
Score	Wert der einem Treffer je nach Güte zugeordnet wird (deutsch: Punktzahl)
SD	**S**hine-**D**algarno (**S**hine-**D**algarno-Sequenz)
SEN	**Sen**sitivität (Abschnitt 3.2, S. 64)
SPEZ	**Spez**ifität (Abschnitt 3.2, S. 64)
upstream	in 5'-Richtung (deutsch: stromaufwärts)
TN	**T**rue-**N**egative (True-Negative)
Tool	häufig als synonym für kleinere Programme verwendet (deutsch: Werkzeug)
TP	**T**rue-**P**ositive (True-Positive)
TP_{SC}	TP mit korrekter Startposition
ZC	ZCurve

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Genvorhersage ist ein wichtiger Schritt bei der Analyse und der Annotation eines Genoms. Durch die Sequenzierung erhält man die Nucleotidsequenz als eine lange Kette von Basen, dargestellt durch die Buchstaben A (für Adenin), T (für Thymin), G (für Guanin), C (für Cytosin). In dieser sollen protein-codierende Regionen, die Gene, gefunden werden. Die Schwierigkeit bei der Vorhersage besteht darin, daß die Gene in einem von sechs Leserahmen liegen können (drei in Vorwärts- und drei in Rückwärtsrichtung) und daß zwischen informationstragenden Bereichen der Sequenz solche liegen, von denen vermutet wird, daß sie nicht informationstragend sind. Das Verhältnis von informationstragender DNA zu nicht informationstragender variiert zwischen den Organismen. In prokaryotischen Genomen wird etwa ein Anteil von 80% als codierend eingestuft. Im Unterschied dazu ist der codierende Anteil in eukaryotischen Genomen mit etwa 10% wesentlich niedriger. Eukaryotische Genome enthalten zwischen den codierenden Regionen, die als Exons bezeichnet werden, große Bereiche, die nicht codieren, die Introns. Nach der Transkription werden die Introns während des Splicens entfernt, so daß die mRNA nur die Exons enthält. Mikrobielle Genome enthalten keine Introns, wodurch die Genvorhersage etwas weniger schwierig ist. In der vorliegenden Arbeit wird nicht auf die Genvorhersage in eukaryotischen Genomen eingegangen, die Methoden der Vorhersage an sich unterscheiden sich jedoch im Grunde wenig. Viele Ausführungen und die Idee der vorliegenden Arbeit können daher auf die Genomanalyse bei Eukaryoten übertragen werden.

1.1 Vorhersage codierender Regionen in mikrobiellen Nucleotidsequenzen

Die potentiellen Gene in prokaryotischen Genomen können als eine Teilmenge der ORFs (Open Reading Frames, Offene Leserahmen) betrachtet werden. Ein ORF ist definiert als Anzahl n unmittelbar aufeinanderfolgender Tripletts $t_1...t_n$, wobei t_1 das Startcodon und t_n das Stopcodon ist. Jeder ORF beginnt mit einem Startcodon aus der Menge {ATG, GTG, TTG, CTG} und endet mit dem nächsten terminalen Codon aus der Menge {TAG, TAA, TGA}, das im gleichen Leserahmen liegt. Es handelt sich jedoch nur dann um ein Gen, wenn ein Protein codiert wird.
Die genaue Kenntnis vom Start- und Endpunkt eines Genes ist sehr wichtig, da sich daraus die Aminosäuresequenz ergibt. Ist die Aminosäuresequenz bekannt, lassen sich hieraus physiologische und biochemische Eigenschaften des codierten Proteins ableiten. So können zum Beispiel durch das Auffinden von Bereichen, die viele hydrophobe Aminosäuren enthalten, transmembrane Helices membrangebundener Proteine lokalisiert werden. Auch die exakte Vorhersage der Startposition ist unerläßlich, denn hier sind häufig wichtige funktionelle Eigenschaften codiert. Signalpeptide können zeigen, wo ein Protein in der Zelle lokalisiert ist (Nielsen *et al.*, 1997), eine Ribosom-Bindestelle, die vor dem Startcodon liegt, kann Aufschluß über die Stärke der Initiation der Translation geben (Barrik *et al.*, 1994) und der N-Terminus des Genes kann Informationen über die Lebensdauer des Proteins enthalten (Varshavsky, 1996).
Es ist wesentlich schwieriger, den korrekten Genstartpunkt zu bestimmen, als den Endpunkt. Als Genstart tritt am häufigsten das Codon ATG auf, das für Methionin codiert. Die weiteren möglichen Startcodons (GTG, TTG, CTG) codieren deutlich weniger häufig Genstarts. Alle potentiellen Startcodons treten allerdings auch innerhalb von Genen auf, zudem sind die Regionen, die upstream (in 5'-Richtung) von Genstarts liegen, wenig konserviert, das heißt charakteristische Muster, wie Ribosom-Bindestellen (RBS), variieren stark und können auch fehlen. Anders ist das bei den terminalen Codons TAG, TAA und TGA. Sie kommen nur in Ausnahmefällen innerhalb eines Genes vor, wo sie dann für eine Aminosäure codieren, beispielsweise in mi-

tochondrialer DNA, wo das Codon TGA für Tryptophan codiert. In solchen Fällen codiert das betreffende Codon dann nicht für einen Genstop.

1.1.1 Methoden der Vorhersage

Die einfachste Möglichkeit der Genlokalisation in bakteriellen Genomen ist es, zunächst die Position möglicher ORFs festzustellen. Hierbei wird vor allem nach langen ORFs gesucht. Sie enthalten mit hoher Wahrscheinlichkeit eine codierende Region. Diese muß nicht den gesamten ORF umfassen, da für den wahren Genstart mehrere Positionen innerhalb eines ORFs in Frage kommen (siehe oben). Um in den ORFs mögliche codierende Regionen zu finden, kann dann eine Ähnlichkeitssuche durchgeführt werden. Die langen ORFs werden dabei gegen Datenbanken abgeglichen, um Ähnlichkeiten zu bereits bekannten Genen festzustellen. Wenn aus einer Nucleotidsequenz beim Übersetzen eine Aminosäuresequenz resultiert, die signifikante Ähnlichkeit zu einem bekannten Protein zeigt, ist diese mit hoher Wahrscheinlichkeit codierend (Gish und States, 1993). So können aber nur Gene gefunden werden, die Ähnlichkeit zu bereits bekannten aufweisen, was nach heutigen Erkenntnissen der Genomanalyse etwa 60-80% der Gene in neusequenzierten Genomen umfaßt (Borg *et al.*, 1992; Koonin *et al.*, 1996; Frishman und Mewes, 1997).

Um auch Gene zu finden, die keine Ähnlichkeit zu bereits bekannten zeigen, wird die erwähnte Suche nach Ähnlichkeiten in Datenbanken mit statistischen Berechnungen kombiniert. Es wurde gezeigt, daß sich die Tri- und Hexanucleotid-Verteilung in codierenden Regionen signifikant von der in nicht-codierenden Regionen unterscheidet (Fickett, 1982; Frishman *et al.*, 1998). Daneben gibt es weitere Nucleotidmotive, die mit höherer Wahrscheinlichkeit in einer codierenden Region als in einer nicht-codierenden auftreten. Diese Verteilungen sind spezies-spezifisch und müssen daher für jeden Organismus individuell bestimmt werden. Durch die nicht-zufällige Verteilung von Mustern in codierenden und nicht-codierenden Regionen ist es möglich, Sequenzen intrinsisch, also aus sich selbst heraus, ohne vorherige Ähnlichkeitssuche zu analysieren.

Die Statistik macht sich zunutze, daß genetische Veränderungen in der Evo-

lution nur überdauern, wenn sie der Spezies einen Überlebensvorteil verschaffen oder zumindest keine schädlichen Auswirkungen haben. Das heißt die zufälligen Veränderungen durch Mutationen, sind nur von Bestand, wenn sie entweder die Funktionalität lebenswichtiger Proteine nicht beeinträchtigen oder eine abgewandelte bzw. neue hervorbringen. Ein Beispiel sind synonyme Codons: Durch Basenaustausch kann die Nucleotidsequenz verändert werden, ohne daß dadurch die Aminosäuresequenz und damit das Protein beeinflußt wird. Will man die Beobachtungen zur Identifikation von Genen ausnutzen, so gilt es Regionen zu finden, in denen die Verteilung von Mutationen als nicht-zufällig erkannt wird.

1.1.2 Automatisierte Verfahren

Die Gesamtheit eines Genomes ist zu umfangreich, um sie manuell zu bearbeiten, vor allem im Hinblick auf die Genvorhersage. Um die steigende Zahl vollständig sequenzierter Genome zu bewältigen, ist man auf die Hilfe von leistungsstarken Computern angewiesen, die automatisiert Sequenzvergleiche sowie statistische, linguistische und Mustererkennungs-Algorithmen ausführen. Sehr verbreitet ist die Verwendung von Markov-Ketten[1], wie beispielsweise in den Programmen GeneMark oder Glimmer. Auf diesem Wege kann effektiv die Wahrscheinlichkeit berechnet werden, mit der aufeinander folgende Nucleotide oder Tripletts in einem codierenden bzw. einem nicht-codierenden Kontext auftreten (Staden und McLachlan, 1982; Claverie und Bougueleret, 1986).

Eines der Probleme bei der automatisierten Genvorhersage ist die Vorhersage sehr vieler False-Positiver (FP, Falsch-Positive), das heißt ORFs, die vermutlich nicht für ein Protein codieren. Diese Problematik tritt vor allem bei Programmen mit rein intrinsischer Analyse, beispielsweise Glimmer, auf. Die intrinsische Analyse basiert ausschließlich auf der Auswertung der Genomsequenz selbst. Es werden keine Datenbank-Abgleiche mit dort deponierten Sequenzen durchgeführt. Andere Programme wie Orpheus und Critica sind auf Datenbank-Abgleiche angewiesen, wobei eine statistische Analyse angeschlossen wird. Die Gesamtvorhersage soll hierbei möglichst un-

[1] siehe Abschnitt 2.3.1, Erweiterte Markov-Modelle: IMMs und ICMs, S. 35

abhängig von der Ähnlichkeitssuche gehalten werden, da die Genvorhersage auf Basis von Ähnlichkeiten, wie eingangs erwähnt, die Problematik birgt, daß keine gänzlich unbekannten Gene gefunden werden können. Ein weiterer Grund, den Einfluß der Ähnlichkeitssuche möglichst präzise zu kontrollieren, ist die Gefahr einer transitiven Fehlerfortpflanzung. Bereits vorgenommene Annotationen, mit denen die ORFs verglichen werden, können fehlerbehaftet sein, was zu einer Fortpflanzung der Fehler in der eigenen Annotation führen kann. Werden die Vorhersagen nicht biochemisch verifiziert, führt das zu einer exponentiellen Zunahme der Fehler.

1.2 Ziel dieser Arbeit

Im Rahmen der Arbeit wurde ein neues Verfahren zur Vorhersage von Genen in prokaryotischen Genomen entwickelt. Es wurde das Programmpaket YACOP (**Y**et **A**nother **C**ombination **O**f **P**redictions) implementiert, das die Vorhersagen unterschiedlicher Programme (im Folgenden auch Tools genannt) durch bool'sche Operationen zu einer Gesamtvorhersage kombiniert. Dabei sollten Sensitivität und Spezifität der Vorhersage erhöht werden. Das vorrangige Problem war die Minimierung der Anzahl False-Positiver, dabei wurde die Vorhersage aller "wahren" Gene (True-Positives, TP) angestrebt. Außerdem sollte die Erkennung der Startposition verbessert werden, indem evaluiert wurde, welches der Programme den größten Anteil an Startpositionen vorhersagt, die mit der bestehenden Annotation übereinstimmen. Weiterhin sollte ermittelt werden, inwieweit die Qualität der Vorhersage von genomischen Parametern abhängt. Ein sehr hoher oder sehr niedriger G/C-Gehalt könnte die Vorhersage verschlechtern, da die meisten Programme für Genome mit einem G/C-Gehalt von etwa 50% optimiert sind.
Es werden die vier Vorhersagetools CRITICA, ORPHEUS, ZCURVE und GLIMMER vorgestellt und verglichen. Zur Beurteilung ihrer Performanz diente der Vergleich mit den Einträgen der Annotation der aktuellen GeneBank-Datenbank. Die Programme erzielen unterschiedliche Ergebnisse, bei der Genvorhersage für eine Sequenz. Dabei treten sowohl False-Positive als auch False-Negative (FN) auf. Die Vorhergesagte beinhaltet also ORFs, die wahrscheinlich nicht für ein Protein codieren (False-Positive) und einige ORFs,

die vermutlich codierend sind, werden nicht vorhergesagt (False-Negative). Durch geeignete Kombination der Ergebnisse konnte die Performanz der Vorhersage deutlich erhöht werden, so daß YACOP bereits routinemäßig im G_2L Göttingen[2] eingesetzt wird. Die Qualität der Vorhersagen von YACOP kann durch die Anwendung auf bereits annotierte Genome und den anschließenden Vergleich der Ergebnisse mit der bestehenden Annotation belegt werden.

[2]Göttingen Genomics Laboratory, http://www.g2l.bio.uni-goettingen.de/

Kapitel 2

Methoden und Material

Es folgt eine Beschreibung der Tools, die zur Genvorhersage in prokaryotischen Genomen kombiniert wurden. Dabei wird auf die Anwendung und die Algorithmen eingegangen. Um die Tools effektiv einzusetzten, ist das Verständnis ihrer Arbeitsweise bei der Vorhersage unerläßlich. Alle Tools, die bei den Untersuchungen eingesetzt wurden, sind ausschließlich zur Genvorhersage in mikrobiellen Sequenzen geeignet. Im einzelnen werden Critica, Orpheus, ZCurve und Glimmer sowie das Zusatztool RBSfinder vorgestellt. Desweiteren wird kurz auf die verwendeten Referenz-Genome eingegangen.

2.1 CRITICA105b - Coding Region Identification Tool Invoking Comparative Analysis

(Badger und Olsen, 1999)

Im Verlauf der Untersuchungen hat sich gezeigt, daß die Vorhersagen von Critica für die getesteten Genome deutlich besser sind, als die anderer Programme. Die Beschreibung des Algorithmus ist sehr detailliert, um ein grundlegendes Verständnis des komplexen Programms zu schaffen. Die folgende Darstellung ist angelehnt an die Veröffentlichung von Badger und Olsen, 1999.

2.1.1 Strategie von CRITICA

CRITICA kombiniert zwei unterschiedliche Ansätze, zum einen wird im **komparativen Ansatz** nach Ähnlichkeiten zu bereits bekannten Genen gesucht, zum anderen werden statistische Methoden (**nicht-komparativer Ansatz**) eingesetzt. Beide Ansätze werden im Folgenden beschrieben.

Komparativer Ansatz

Zunächst werden die Eingabesequenzen auf Nucleotidebene mit einer Datenbank abgeglichen. Dazu wird das Programm BLASTN (ALTSCHUL et al., 1990) verwendet, wobei die Erwartungswerte E und E2, welche die Wahrscheinlichkeit angeben, die betreffende Sequenz zufällig zu finden, standardmäßig auf 1×10^{-4} gesetzt werden. Diese dienen als Schwellenwerte bei der Datenbanksuche. Alle Treffer zwischen Eingabesequenz und Sequenzen aus der Datenbank für die ein Erwartungswert berechnet wird, der kleiner als der gesetzte Schwellenwert ist, werden von BLAST akzeptiert und als Ergebnis zurückgegeben. Treffer mit Sequenzen des Organismus, aus dem die Eingabesequenz stammt und Alignments, die Lücken enthalten, werden verworfen. Es verbleiben lokale Alignments, die sogenannten HSPs (high-scoring segment pairs). Definitionsgemäß werden im Kontext von BLAST Alignments ohne Lücken mit einem hohen Anteil identischer Nucleotide oder ähnlicher Aminosäuren als HSP bezeichnet. Um von CRITICA akzeptiert zu werden, müssen die HSPs einen genügend großen Anteil identischer Aminosäuren aufweisen, um mit signifikanter Wahrscheinlichkeit für homologe Proteine zu codieren.
Beim Abgleich mit einer Datenbank werden die Eingabesequenzen mit Ähnlichen Sequenzen aligniert. Die alignierten Sequenzen, also Eingabe- und Treffersequenz, werden dann in allen sechs Leserahmen in die entsprechende Aminosäuresequenzen übersetzt und sowohl auf Nucleotid-, als auch auf Aminosäureebene auf Unterschiede untersucht. Zeigt eine Eingabesequenz auf Aminosäureebene größere Identität zur Treffersequenz als auf Nucleotidebene, wird sie im Kontext von CRITICA als codierend gewertet. Das ist der Fall, wenn synonyme Codons an übereinstimmenden Positionen in Eingabe- und Treffersequenz auftreten. Das Vorkommen synonymer Codons in den

alignierten Sequenzen kann auf Zufall beruhen, es liegt jedoch die Vermutung nahe, daß die Veränderung in der Sequenz Bestand hatte, weil sie keinen Einfluß auf die Aminosäuresequenz und damit das resultierende Protein hatte. Handelt es sich dagegen um eine nicht-codierende Region, sollten Mutationen zufällig auf alle Positionen in der Sequenz verteilt sein. Damit ist die Wahrscheinlichkeit höher, daß sich die Aminosäuresequenzen unterscheiden.
Zur Analyse werden die alignierten Sequenzen in Tripletts zerlegt. Es werden die Unterschiede in den einzelnen Tripletts zwischen Eingabe- und Treffersequenz gezählt. Die Anzahl der Unterschiede in den zugehörigen Aminosäuresequenzen wird ebenfalls festgestellt. Daraus wird ein zusammenfassender Score für jedes Triplett berechnet, der dessen Beitrag zur Wahrscheinlichkeit widerspiegelt, daß die Region ein Protein codiert.

Beispiel zur Bestimmung des komparativen Scores: Im Beispiel (Tabelle 2.1, S. 10) werden zwei alignierte Sequenzen gezeigt. Die Tripletts TTG und CTA (erste Spalte) unterscheiden sich in zwei Nucleotiden, codieren aber für die gleiche Aminosäure. Sie sind also synonym. Aus einer heuristisch ermittelten Scoring-Matrix, die zum CRITICA-Algorithmus gehört (Tabelle 2.3, S. 15), wird dann der entsprechende Score für alignierte Tripletts, die sich in zwei Nucleotiden unterscheiden und für die gleiche Aminosäure codieren, abgelesen. Der Score hängt dabei auch davon ab, wie viele alignierte Tripletts im Kontext betrachtet werden. Die Entwicklung der Scoring-Matrizen wird im folgenden Abschnitt beschrieben.
Im Beispiel ist dem Score der Wert 164 zugeordnet. Er wird aus der Scoring-Matrix für 32 informationstragende Tripletts abgelesen. Tripletts, die in beiden Sequenzen gleich sind, erhalten einen Score von null, weil sie für die gleiche Aminosäure codieren und somit ihr Informationsgehalt im Hinblick auf die Genvorhersage gleich null ist. Synonyme Tripletts, wie AGT und AGC (Serin), erhalten immer einen positiven Score. Sie werden als Hinweis gewertet, daß es sich hier um eine codierende Region handelt. Die Differenz auf Nucleotidebene bestimmt dabei die Höhe des Scores. Synonymen Tripletts, die sich in zwei Nucleotiden unterscheiden, wird ein höherer Score zugeordnet, als solchen mit einem Nucleotid Unterschied. Nicht-synonymen

Tripletts wird ein negativer Wert zugeordnet. Dieser ist um so negativer, je ähnlicher die Tripletts sind. Analog zu synonymen, wird nicht-synonymen Tripletts mit zwei Nucleotiden Unterschied ein höherer (positiverer) Score zugeordnet, als jenen, die sich in einem Nucleotid unterscheiden. Die einzelnen Scores der alignierten Sequenzen werden dann zu einem zusammenfassenden Score aufaddiert, solange die Summe nicht null oder negativ wird (Tabelle 2.1 (5) und (6), S. 10).

Tabelle 2.1: **Beispiel zur Bestimmung des komparativen Scores.** (1) a/b Nucleotidsequenz, die verglichen werden: a) Eingabesequenz, b) mittels BLASTN gefundene Treffersequenz; (2) Anzahl unterschiedlicher Nucleotide in den alignierten Tripletts; (3) a/b die durch (1) a/b codierten AS-Sequenzen; (4) Vergleich der AS-Sequenzen; (5) der komparative Score (aus heuristische ermittelten Scoring-Matrizen für 32 betrachtet Tripletts abgelesen, siehe Tabelle 2.3, S. 15); (6) Summe der aufaddierten Scores (von links nach rechts).

(1)a	**Eingabesequenz**	TTG	GCT	GAT	GCC	TTT	AAC	GGC
b	**Treffersequenzen**	CTA	GAC	GAT	GCC	TTC	ACC	GGA
(2)	**Triplettunterschiede**	2	2	0	0	1	1	1
(3)a	**AS-Sequenz (1)a**	Leu	Ala	Asp	Ala	Phe	Asn	Gly
b	**AS-Sequenz (1)b**	Leu	Asp	Asp	Ala	Phe	Thr	Gly
(4)	**Aminosäuren Vergleich**	=	$\neq$	=	=	=	$\neq$	=
(5)	**komparativer Score**	164	-16	0	0	52	-36	52
(6)	**aufaddierter Score**	164	148	148	148	200	164	216

Zum Erstellen der Scoring-Matrizen wurden zunächst anhand eines Modells Wahrscheinlichkeiten dafür berechnet, daß zwei alignierte Tripletts, die sich in ein, zwei oder drei Nucleotiden unterscheiden, für die gleiche bzw. verschiedene Aminosäuren codieren (Tabelle 2.2, S. 11). Die Autoren haben bei der Berechnung alle Nucleotide und alle Nucleotid-Unterschiede als gleichwertig betrachtet. Alle 61 Tripletts (ohne Stopcodons) wurden gleich gewichtet. Das hier vorgestellte Modell wurde für das Genom von *Salmonella typhimurium* entwickelt, das einen G/C-Gehalt von etwa 50% hat. Es ist noch nicht untersucht, ob bei einem G/C-Gehalt, der stark davon abweicht, eine unterschiedliche Gewichtung sinnvoller wäre, da die Wahrscheinlichkeit, daß zwei Tripletts die gleiche Aminosäure codieren, umso höher ist, je mehr Nucleotide gleich sind. Die ermittelten Wahrscheinlichkeiten gehen dann in

die Berechnung der komparativen Scores für die Scoring-Matrizen ein, die im Folgenden beschrieben werden.

Tabelle 2.2: **Wahrscheinlichkeiten dafür, daß Tripletts, die sich in ein, zwei oder drei Nucleotiden unterscheiden, für die gleiche bzw. verschiedene Aminosäuren codieren.** Die Werte wurden am Modellorganismus *Salmonella typhimurium* entwickelt.

Anzahl unterschiedlicher Nucleotide	**Wahrscheinlichkeit, daß die Tripletts synonym sind**	**Wahrscheinlichkeit, daß die Tripletts nicht synonym sind**
0	1,000	-
1	0,255	0,745
2	0,018	0,982
3	0,008	0,992

Der komparative Score gibt einen Hinweis darauf, ob ein Triplett t_i der Eingabesequenz in einer codierenden Region liegt. Er wird in diesem Kontext definiert als Logarithmus der Wahrscheinlichkeit, die Kombination in einer codierenden Region zu finden, dividiert durch die Wahrscheinlichkeit, sie in einer nicht-codierenden Region zu finden (Gleichung (2.1)).

$$S_{komparativ} = \log \frac{P(t_i)_{codierend}}{P(t_i)_{nicht-codierend}} \tag{2.1}$$

$S_{komparativ}$	komparativer Score
t_i	Triplett t an einer beliebigen Position i, wobei i der Index der Triplettsequenz ist
$P(t_i)_{codierend}$	Wahrscheinlichkeit, daß ein Triplett t_i in einem codierenden Kontext liegt
$P(t_i)_{nicht-codierend}$	Wahrscheinlichkeit, daß ein Triplett t_i in einem nicht-codierenden Kontext liegt

Da die Wahrscheinlichkeiten $P(t_i)$ von vielen Faktoren abhängen, können sie nicht direkt berechnet werden. Sie werden aus fünf heuristisch ermittelten Scoring-Matrizen, die mit 8, 16, 32, 64, 128 markiert sind, abgelesen. Der Matrizenname bezieht sich hierbei auf n_0, die approximierte Anzahl der

Tripletts, die im Hinblick auf die Genvorhersage als informationstragend angesehen werden (n). Beim Erstellen der Matrizen werden Länge und der Grad der Aberration zwischen den alignierten Sequenzen einbezogen (ähnlich wie ALTSCHUL, 1993).

Die Erstellung der Scoring-Matrizen erfolgte mit Hilfe der nun vorgestellten heuristischen Methode. Es wurden jeweils drei Matrizen generiert (ein, zwei bzw. drei Nucleotidunterschiede in den Tripletts), für unterschiedliche n_0 (8, 16, 32, 64, 128). Die folgenden Herleitungen beziehen sich auf das Beispiel einer Matrix für alignierte Tripletts mit jeweils einem Nucleotid Unterschied. Die Matrizen für alignierte Tripletts mit zwei bzw. drei Nucleotiden erfolgt analog.
Gleichung (2.2) beschreibt die Funktion für das zufällige Auftreten von n alignierten Tripletts in Eingabe- und Treffersequenz, die sich in ein, zwei bzw. drei Nucleotiden unterscheiden und bei deren Übersetzung m identische Aminosäuren resultieren. p sei die Wahrscheinlichkeit, daß die Tripletts für die gleiche Aminosäure codieren. Die Werte für p sind in Tabelle 2.2, S. 11 angeführt. Trägt man die Funktion graphisch auf, so erhält man eine Gerade (Abbildung 2.1, feingestrichelte Linie). Für identische Tripletts ist $p = 1$. Daraus folgt für identische Sequenzen $m = n$ (Abbildung 2.1, grob gestrichelte Linie). Abbildung 2.1 zeigt die Auftragung für betrachte Tripletts mit einem Nucleotid Unterschied.

$$m = np \tag{2.2}$$

m	Anzahl identischer Aminosäuren
n	Anzahl alignierter Tripletts mit 1, 2 oder 3 Nucleotiden Unterschied
p	Wahrscheinlichkeitswert aus Tabelle 2.2, S. 11

Gleichung (2.3) beschreibt eine Approximation der maximalen heuristischen Werte m bei n (Abbildung 2.1, dargestellt als **x**), deren Wahrscheinlichkeit nicht signifikant größer ist als die Wahrscheinlichkeit $P < 0,0001$ [1]. Jeder Wert m, der an der Position n über der approximierten Kurve, das heißt dem approximierten Schwellenwert, liegt, wird als signifikant angenommen

[1] siehe Abschnitt 2.1.1, Berechnung statistisch signifikanter Wahrscheinlichkeiten, S. 18

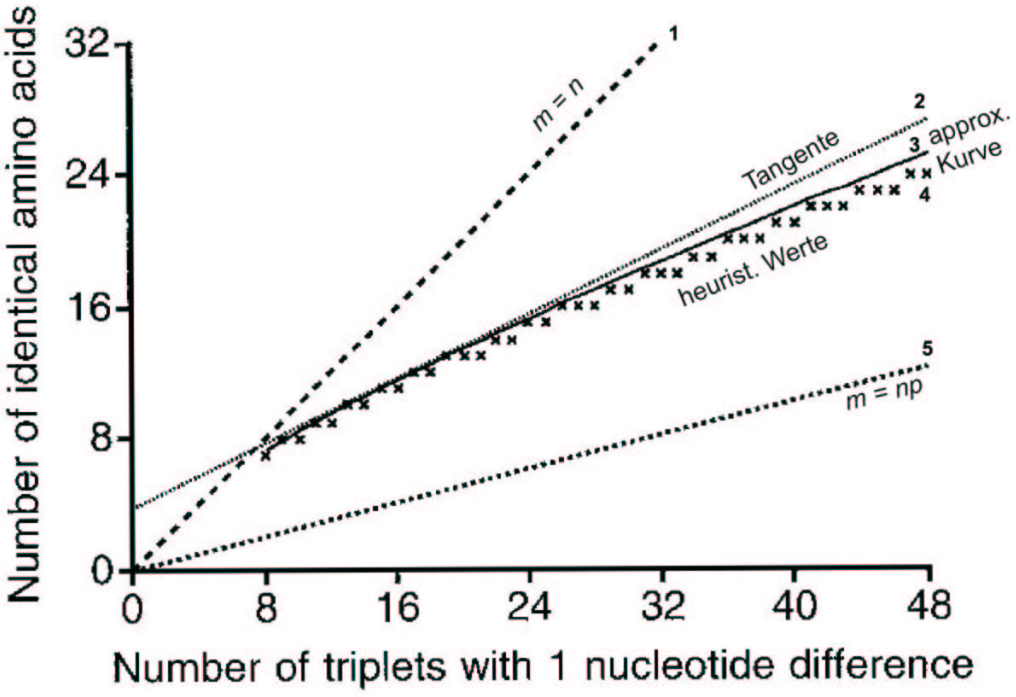

Abbildung 2.1: **Anzahl identischer Aminosäuren m im paarweisen Alignment von n Aminosäuren**. Von oben: **1** Graph zur Funktion der zufälligen Identität. Sind beide Aminosäuren gleich, so ist $p = 1$, daraus folgt $m = n$; **2** $m = an + b$, mit $a = p + \frac{Z}{2}\sqrt{\frac{pq}{n_0}}$ und $b = \frac{Z}{2}\sqrt{n_0 pq}$, Tangente an der approx. Kurve, die Tangente ist als Schwellenwert anzusehen, alle Wertpaare (n, m), die oberhalb liegen, werden als statistisch signifikant eingestuft; **3** $m = np + Z\sigma$, mit $\sigma = \sqrt{np(1-p)}$ approximierte Kurve; **4** Heuristische Werte; **5** $m = np$ Funktion zufällig identischer Aminosäuren bei Nucleotidunteschieden. Die Werte p sind in Tabelle 2.2, S. 11 aufgeführt. Alle Wertpaare (n, m) liegen in dem Bereich zwischen den Geraden **(1)** und **(5)**. Aus Badger und Olsen, 1999.

(Abbildung 2.1, durchgezogene Linie). Es wird also angenommen, daß alle Sequenzen, deren Wertpaare (n, m) über der approximierten Kurve liegen, in einen codierenden Kontext einzuordnen sind.

$$m = np + Z\sigma, \tag{2.3}$$

mit

σ = $\sqrt{np(1-p)}$ (Standardabweichung)

Z ist eine Variable, deren Wert so gewählt wird, daß die Kurve oberhalb der heuristischen Werten angesiedelt ist

Die Tangente der approximierten Kurve an der Stelle $n = n_0$ wird durch Gleichung (2.4) beschrieben. n_0 ist dabei die optimale Anzahl informationstragender Tripletts, die betrachtet werden und entspricht der Bezeichnung der jeweiligen Matrix. Wenn in den alignierten Sequenzen $n \approx n_0$ Tripletts mit einem, zwei bzw. drei Nucleotiden Unterschied gefunden werden, so wird die entsprechende Matrix n_0 betrachtet. Im Beispiel wird als Näherung die Tangente bei $n = 16$ verwendet (Abbildung 2.1, gepunktete Linie), da hier keine signifikanten Wertpaare (n, m) unter der Gerade auftreten. Für $n = 16$ bei der Betrachtung der Tripletts mit einem Nucleotid Unterschied ist $Z = 4,3$.

$$m = an + b, \tag{2.4}$$

mit

$$a = p + \frac{Z}{2}\sqrt{\frac{pq}{n_0}}$$

und

$$b = \frac{Z}{2}\sqrt{n_0 pq}$$

Die Autoren ordneten den Wertpaaren (n, m) der Matrizen Scorewerte zu. Ist M der gewählte Score, der das Auftreten identischer Aminosäuren beschreibt, so gibt es einen Score N für nicht-identische Aminosäuren. M bedingt N durch die Funktion (2.5).

$$N = \frac{a}{a-1}M \tag{2.5}$$

N	Score für das Auftreten nicht-identischer Aminosäuren
M	Score für das Auftreten identischer Aminosäuren
a	Steigung aus der Funktion (2.4)

Da $0 < a < 1$ gilt, wird $a - 1$ immer negativ und damit auch der Score für nicht-identische Aminosäuren. Der Wert M wird relativ groß gewählt und das entsprechende N berechnet. Die Berechnung erfolgt jeweils für $n_0 = 8, 16, 32, 64, 128$. Der Wert λ (Karlin und Altschul, 1990)[2] wird für M mit der Wahrscheinlichkeit p und für N mit $1 - p$ berechnet. Er ist

[2] siehe Abschnitt 2.1.1, Berechnung statistisch signifikanter Wahrscheinlichkeiten, S. 18

invers proportional zum Betrag von M und N.
Da die Werte M und N jeweils für Tripletts mit einem, zwei und drei Nucleotidunterschieden auftreten, gibt es auch jeweils drei Werte λ für ein gegebenes n_0. Zur Vereinfachung werden M und N so gewählt, daß jedes $\lambda \approx 0,015$ ist. Die Werte M und N sind in Tabelle 2.3 aufgeführt.

Tabelle 2.3: **Scoring-Tabelle** für das Alignment von Tripletts. Die Autoren gaben die Werte M für identische Aminosäuren vor. Hieraus resultieren nach Gleichung (2.5), S. 14 die angegebenen Werte N für nicht-identische. n_0 ist die approximierte Anzahl betrachter Tripletts und entspricht dem Namen der verwendeten Scoring-Matrix ($n_0 \approx n$). Im optimalen Fall ist $n_0 = n$.

n_0, optimale Anzahl betrachter Tripletts	**Score für alignierte Tripletts mit 0, 1, 2 bzw. 3 Nucleotidunterschieden**						
	0	**1**		**2**		**3**	
		M (AS =)	**N** (AS ≠)	**M** (AS =)	**N** (AS ≠)	**M** (AS =)	**N** (AS ≠)
8	0	76	-99	218	-48	243	-28
16	0	65	-59	197	-31	230	-22
32	0	52	-36	164	-16	207	-14
64	0	41	-23	141	-10	171	-7
128	0	32	-16	108	-5	135	-4

Kombination verschiedener komparativer Scores für eine Eingabesequenz: Wurden beim Datenbank-Abgleich mit BLAST mehrere HSPs zu einer Eingabesequenz gefunden, so gibt es zwei Möglichkeiten darauf zu reagieren. Werden die Treffer als verwandt eingestuft, so werden die jeweiligen Scores ermittelt und dann der Mittelwert berechnet. Dies ist häufig der Fall, da die HSPs mit der Eingabesequenz verwandt sind und so oft auch mit den anderen alignierten Sequenzen. Sind die HSPs voneinander unabhängig, so werden die Scores addiert, beispielsweise wenn Austausche an verschiedenen Triplettpositionen vorliegen. Durch die Kombination beider Berechnungen wird eine höhere Sensitivität erreicht.
Ein beträchtlicher Teil des komparativen Scores wird durch Austausch von Basen in der dritten Position bedingt. Die Unterschiede treten ebenfalls in dem entsprechenden Frame auf dem komplementären Strang auf, wobei die

Werte des jeweils nicht-codierenden Frames auf null gesetzt werden.

Nicht-komparativer Ansatz

Der komparative Ansatz ist nicht geeignet, Gene zu lokalisieren, die keine Ähnlichkeit zu bereits bekannten aufweisen. Daher haben die Autoren neben der vergleichenden Analyse, einen statistischen Ansatz verfolgt. Es wird eine abgewandelte Version der Dicodonmethode (Claverie und Bougueleret, 1986) eingesetzt. Dabei wird jedem Index i ein Score zugeordnet, der eine Funktion des Tripletts t_i und des vorangehenden Tripletts t_{i-1} ist:

$$S_{Dicodon}(t_i, t_{i-1}) = \text{nint}\left(\alpha\frac{1}{\lambda_C}\ln\left(\frac{f_{codierend}(t_i|t_{i-1})}{f_{nicht-codierend}(t_i|t_{i-1})}\right)\right) \tag{2.6}$$

$S_{Dicodon}$	Dicodonscore
$f_{codierend}(t_i\|t_{i-1})$	Häufigkeit eines Tripletts t_i, wenn es auf Triplett t_{i-1} folgt, wobei t_i und t_{i-1} in codierenden Regionen liegen.
$f_{nicht-codierend}(t_i\|t_{i-1})$	Häufigkeit eines Tripletts t_i, wenn es auf Triplett t_{i-1} folgt, wobei t_i und t_{i-1} in nicht-codierenden Regionen liegen.
λ_C	λ-Parameter (Karlin und Altschul, 1990) zur Skalierung von (2.6), so daß S mit dem zugehörigen komparativen Score addiert werden kann. Der Parameter wird für jedes Genom neu berechnet. Die Werte liegen im Bereich $\lambda \approx 0,015$ (Abschnitt 2.1.1, S. 14 und S. 18)
α	empirisch ermittelter Wert ≤ 1, der kompensiert, daß hier die Triplettsequenz als eine Folge von Markov-Modellen erster Ordnung gewertet werden, während Karlin und Altschul von der Unabhängigkeit der Tripletts ausgehen.
nint	Die Scores werden mit Hilfe der Funktion nint (nearest integer) zum nächsten ganzzahligen Wert gerundet.

Iterativ erfolgt nun die weitere Analyse. Für die erste Iteration werden nur ORFs als codierend gewertet, die einen signifikant hohen komparativen Sco-

re[3] aufweisen. Dabei bleibt der größte Teil der DNA unklassifiziert. Die Dicodonhäufigkeiten werden in den klassifizierten Regionen ermittelt und darüber für die nicht-klassifizierten abgeschätzt. Dadurch wird die Sequenz nur bedingt abhängig vom komparativen Schritt in codierende und nicht-codierende Regionen eingeteilt. In jedem weiteren Iterationsschritt erfolgt eine Neueinteilung der Sequenz auf Basis der Häufigkeiten, die in der vorangehenden Iteration ermittelt wurden. Standardmäßig werden drei Iterationen durchgeführt.

Der nicht-komparative Score und der komparative werden addiert, so daß jedem Triplett ein einziger Gesamtscore zugeordnet ist. Die Scores einer Region werden aufaddiert, so lange sie positiv oder null sind. Erreicht die Summe der Triplettscores einen signifikanten Schwellenwert, wird die Region als codierend angesehen. Das Aufaddieren wird fortgeführt bis ein Stopcodon oder das Ende der Sequenz erreicht ist. Wird ein Triplett erreicht, dem ein negativer Score zugeordnet ist, so wird er aufaddiert, sofern er durch einen folgenden positiven Wert ausgeglichen werden kann. Da in einem Genom immer ein Stopcodon am Ende der codierenden Region steht, wird der 3'-Bereich (C-terminal) jedes potentiellen Gens bis zum nächsten Stopcodon oder bis zum letzten vollständigen Codon der Sequenz verlängert. Ähnlich wird mit dem initialen Codon verfahren, wobei wiederum berücksichtigt werden muß, daß Startcodons auch innerhalb einer codierenden Region auftreten können. Der Score für ein mögliches Startcodon t wird berechnet nach Gleichung (2.7).

$$S_{Initiator}(t) = \mathrm{nint}\left(\frac{1}{\lambda_C}\ln\left(\frac{f_{Initiator}(t)}{f(t)}\right)\right) \tag{2.7}$$

$S_{Initiator}$	Initiatorscore
$f_{Initiator}(t)$	Häufigkeit eines Tripletts t unter den Vorkommen aller vorhergesagten Startcodons
$f(t)$	Häufigkeit eines Tripletts t unter allen Vorkommen der Tripletts aus {ATG, GTG, TTG}

Weiterhin wird ein Score berechnet, der die Qualität der Ribosom-Bindestelle (RBS) bewertet, sofern eine solche in der Startregion gefunden werden kann.

[3] siehe Abschnitt 2.1.1, Der komparative Score, S. 11f

Die Shine-Dalgarno Sequenz (SD-Sequenz), wird hierbei definiert als vier oder mehr aufeinander folgende Basen nach dem Muster RGGRGGTGAT (R = A oder G; SHINE und DALGARNO, 1974), die nicht weiter als 16 Basen upstream (in 5'-Richtung) des Startcodons liegen darf. Der Score für die Shine-Dalgarno Sequenz wird berechnet nach Gleichung (2.8).

$$S_{SD}(s) = \text{nint}\left(\frac{1}{\lambda_C}\ln\left(\frac{f_{SD}(s)}{f(s)}\right)\right) \tag{2.8}$$

S_{SD}	SD-Score
$f_{SD}(s)$	Häufigkeit des längsten Matches aus dem oben angeführten Muster in solchen Sequenzen, die maximal 16 Basen upstream von potentiellen Genstarts mit hohem Score liegen
$f(s)$	Häufigkeit des längsten Matches aus dem oben angeführten Muster in solchen Sequenzen, die maximal 16 Basen upstream von potentiellen Genstarts mit niedrigerem Score liegen

Dabei werden nur Startpositionen einbezogen, deren P-Werte in der Größenordnung eines empirisch ermittelten Schwellenwertes liegen. Wenn keine Ribosom-Bindestelle nach der oben angeführten Definition gefunden werden kann, wird der Score analog für andere Motive in dieser Region berechnet. Ausgewählt wird der Genstart mit dem höchsten Score, wobei aber die Möglichkeit besteht, verschiedene potentielle Starts einzubeziehen, deren Score in einem definierten Intervall liegt. Der P-Wert des resultierenden Scores muß unter dem standardmässig mit 1×10^{-4} gesetzten Schwellenwert liegen.

Berechnung einer Signifikanzschwelle

Um eine mit BLAST alignierte Region (HSP) als codierend einzustufen, muß der ihr zugeordnete Score statistisch signifikant sein. Die Signifikanz wird mit einem Satz Gleichungen bestimmt, die von KARLIN und ALTSCHUL entwickelt wurden. Eine Approximation der theoretischen Extremwertverteilung von S, dem größten Score aus der Menge der HSPs mit der Länge N, wird durch die Parameter K und λ (KARLIN und ALTSCHUL, 1990) definiert. Ist ein Score als signifikant eingestuft, wird er als nicht zufällig angesehen.

Mit diesen Parametern läßt sich nach Gleichung (2.9) für jedes HSP einen P-Wert berechnen, der die Wahrscheinlichkeit angibt, daß es zufällig einen Score größer gleich S aufweist. Umso weniger wahrscheinlich es ist, einen Score der Größe S zu finden, desto wahrscheinlicher ist es, daß das Auftreten nicht zufällig ist. Für die Wahrscheinlichkeit $P(S)$ gilt:

$$P(S) = 1 - e^{-KN^{\lambda S}} \tag{2.9}$$

$P(S)$	Wahrscheinlichkeit einen Score $\geq S$ zu finden
K	Parameter, der die approximierte Verteilung des größten Scores definiert
λ	wie K
N	Länge der Sequenz

K und λ werden aus den Wahrscheinlichkeiten, das unterschiedlichen Tripletts für die gleiche Aminosäure codieren, (Tabelle 2.2, S. 11) und den beobachteten Häufigkeiten der Nucleotidunterschiede in den HSPs berechnet. Da die beobachteten Häufigkeiten für viele HSPs relativ niedrig sind, fügen die Autoren von CRITICA eine fixe Anzahl an Tripletts, mit Nucleotidunterschieden hinzu, die sich aus gemittelten Werten früherer Analysen bakterieller Genome zusammensetzt: 36 identische Tripletts, 9 Tripletts mit 1 Base Unterschied, 4 mit 2 Unterschieden und 1 Triplett mit 3 Unterschieden.

2.1.2 Anwendung von CRITICA

Unterprogramme und Kommandozeilen-Parameter

CRITICA ist in mehrere Unterprogramme aufgeteilt, die sich durch den Aufruf von vier Skripten anwenden lassen, denen mehrere Parameter übergeben werden können. Um das Programmpaket richtig zu nutzen und die Ergebnisse richtig interpretieren zu können, ist es notwendig diese Parameter zu kennen und zu verstehen, was sie bewirken. Dies gilt insbesondere, wenn man eine Sequenz untersuchen will, die Merkmale aufweist, welche von der Mehrheit bekannter bakterieller Genome abweichen. Es folgt eine kurze Beschrei-

bung der Unterprogramme mit einer Aufzählung der zugehörigen Parameter. Für nähere Informationen liegt dem CRITICA-Paket eine HTML-Seite bei.

Das erste Skript, `blast-contigs`, zerlegt die Sequenz in 3000 BP große Fragmente. Diese werden nacheinander gegen eine Datenbank abgeglichen. Die BLAST-Ergebnisse werden in einem temporären Verzeichnis abgelegt, welches automatisch unter dem Pfad angelegt wird, unter dem das Genom liegt. Als Parameter kann mit `-db=` der Pfad einer nicht-redundanten Nucleotid-Datenbank übergeben werden, wie unter 2.1.2, S. 22 gezeigt. Die Standard-Datenbank sollte im Skript hardcodiert oder als Umgebungsvariable `CRITICA_BLASTDB` gesetzt werden.

Das zweite Skript, `make-blastpairs`, konvertiert die BLAST-Ausgabe in Multiple-FASTA-Format, das die Eingabesequenzen alternierend mit den jeweils alignierten Sequenzen aus der BLAST-Suche enthält. Mit `-exclude=` kann eine bestimmte Gattung oder eine Spezies übergeben werden, die generell als Treffer bei der BLAST-Suche ausgeschlossen wird, bzw. um Treffer mit dem Organismus selbst zu vermeiden. Meist ist dies allerdings nicht notwendig, da Treffer mit mehr als 97% Identität generell ausgeschlossen werden.

Das dritte Skript, `scanblastpairs`, nimmt als Input ein File in Multiple-FASTA-Format, wie es von `make-blastpairs` generiert wird. Das Skript schreibt eine Triplett-Tabelle, mit den alignierten Tripletts zu jedem Triplett der Sequenz.

Als letztes wird `iterate-critica` aufgerufen. `iterate-critica` ruft iterativ `critica` das Kernstück des Programmpaketes auf und übergibt ihm die gesetzten Parameter bzw. Werte, die zuvor berechnet wurden. Die Anzahl der Iterationen wird durch `-iterations=` festgelegt. Standardmäßig werden drei Iterationen durchgeführt. Mit dem Parameter `-scoring-matrix=` kann dem Programm eine Scoring-Matrix übergeben werden. Weiterhin können für die Initiation der Iterationen Dicodonscores (mit `-inital-dicod=`), Initiator-Scores (mit `-initial-init`), SD-Scores (mit `initial-sd=`) und Promotor-Scores (mit `-initial-prom=`) übergeben werden. Andernfalls werden die Scores für jedes Genom neu berechnet[4]. Mit dem Parameter `-no-sdscore`

[4]siehe Abschnitt 2.1.1, Nicht-komparativer Ansatz, S.16

kann man festlegen, daß die Iterationen ohne SD-Scores durchgeführt werden. Arbeitet man beispielsweise mit Organismen, die wenige oder keine bekannten SD-Sequenzen haben, sollte die Berechnung der SD-Scores durch Übergabe des Parameters übersprungen werden. Wird der Parameter `-prom-find` übergeben, so wird eine zusätzliche Promotorsuche durchgeführt. Dieses Feature ist für die, in dieser Arbeit verwendete Version (CRITICA105b), allerdings noch nicht vollständig getestet. Mit `-threshold=` kann der maximale P-Wert gesetzt werden, der von dem Programm akzeptiert wird. Der Defaultwert ist dieser 1×10^{-4}. Der Wert α (Gleichung (2.6), S. 16), der standardmäßig auf 1,0 gesetzt ist, kann mit `-alpha=` vorgegeben werden. Mit dem Parameter `-fraction-coding=` kann der vermutete Anteil codierender DNA gesetzt werden. Standardmäßig wird 0,8 (80%) angenommen. Wenn wenige Daten aus der komparativen Analyse vorliegen, kann mit dem Parameter `-add-longorfs=` eine Länge übergeben werden. Bei der ersten Iteration werden alle ORFs, deren Länge (in Tripletts) die gesetzte Länge überschreitet, zur Ergebnismenge addiert, um eine bessere Datenlage als Ausgangspunkt der Iterationen zu haben. Mit `-genetic-code=` können alternative Codes verwendet werden. Es wird eine Nummer nach NCBI Standard übergeben. Der Parameter `-strict-threshold` legt fest, daß beim Verlängern der potentiell codierenden Regionen zum nächsten terminalen bzw. initialen Codon der Schwellenwert nicht überschritten werden darf. Ist der Wert `-frameshift-threshold=` gesetzt, sucht CRITICA nach möglichen Frameshifts. Übergeben wird die Anzahl der erlaubten Nucleotide, zwischen zwei codierenden Regionen, die in verschiedenen Leserahmen liegen. Der Parameter `-quick-stats` bewirkt, daß für K und λ (Abschnitt 2.1.1, S. 18) die Näherungswerte $K = 0,2$ und $\lambda = 0,015$ verwendet werden. Dadurch wird eine Steigerung der Performanz erreicht, da die Berechnung der Werte sehr zeitaufwendig ist. Es muß allerdings darauf hingewiesen werden, daß dabei Verluste bei grenzwertigen Vorhersagen auftreten können.

Manueller Aufruf der Unterprogramme

Alle hier angegebenen Filenamen sind willkürlich festzulegen. Gleiche Filenamen in der Beschreibung stehen für gleiche Files. Mit “>” wird die Aus-

gabe vom Standard-Outputdevice (meistens der Bildschirm) in eine Datei umgeleitet. In eckigen Klammern angegebene Parameter sind optional.

1. `blast-contigs genome.fasta [-db nt] > genome.blast`

   (genome.fasta ist die Sequenz in FASTA-Format.)

2. `make-blastpairs [-exclude=string] genome.blast > genome.blast.pairs`

3. `scanblastpairs genome.fasta genome.blast.pairs genome.triplets`

   (genome.triplets ist die Ausgabedatei.)

4. `iterate-critica genome genome.fasta genome.triplets`

   (genome3.cds ist die Ausgabedatei der letzten Iteration)

Beispiel für die Ausgabe von CRITICA

```
AP000398   197  2083 1.40e-146   32  10385  14620   52 ATG   58  7    AGGT
AP000398  2278  3102  1.52e-09  128  -1601   4131   52 ATG  -46  0       -
AP000398  3139  3378  1.65e-07  128      0   1608   52 ATG   61  7    GGAG
AP000398  3497  3982  5.84e-19  128    506   3304  -88 GTG   79  8   AGGTG
AP000398  3982  4515  2.23e-07  128  -1141   2915   52 ATG  137  8  AGGTGA
```

contig start end P-Wert M comp dicod initiator sd_{sc} sd_{sh} sd_{pat}

contig	Name des eingegebenen Contigs bzw. Genomes
start	Position des Startcodons (1.Base)
end	Position des Stopcodons (1.Base)
P-Wert	Der Score ist um so besser desto niedriger er ist. Berechnung siehe (2.9), S. 19
M	verwendete Scoring-Matrix, siehe Abschnitt 2.1.1, S. 11f
comp	Komparativer Score, siehe Abschnitt 2.1.1, S. 11
dicod	Dicodonscore $S_{Dicodon}$, Berechnung siehe (2.6), S. 16
initiator	Initiator-Score $S_{Initiator}$, Berechnung siehe (2.7), S. 17 und initiales Codon

sd_{sc}	Score für die beste Shine-Dalgarno-Sequenz , S_{SD}, Berechnung siehe (2.7), S. 17
sd_{sh}	Anzahl der Nucleotide zwischen dem Ende der SD-Sequenz und dem Startcodon
sd_{pat}	SD-Muster

2.2 ORPHEUS2

(Frishman *et al.*, 1998)

2.2.1 Strategie von ORPHEUS

Orpheus verknüpft ebenso wie Critica Ähnlichkeitssuche mit statistischen Methoden. Dabei wird besonderer Augenmerk auf die korrekte Vorhersage der Genstartpunkte gelegt. Teilsequenzen, die eine signifikante Ähnlichkeit zu bereits bekannten Genen aufweisen, werden zur Feststellung charakteristischer Eigenschaften der codierenden Regionen und Ribosom-Bindestellen (RBS) des Organismus genutzt. Die Autoren extrapolieren diese Eigenschaften auf die Menge aller Gene, um statistische Vorhersagen für das gesamte Genom zu machen.

Zunächst wird eine Ähnlichkeitssuche mit der gesamten Sequenz durchgeführt. Dazu wird das Tool dps (DNA-Protein Search Program, Huang, 1996) verwendet. Es übersetzt die Nucleotidsequenz in allen sechs Leserahmen in die jeweilige Aminosäuresequenz und gleicht diese mit einer nichtredundanten Proteindatenbank ab[5]. Die Ausgabe von dps enthält Informationen über die Qualität des Treffers in Form eines Ähnlichkeits-Scores, Start- und Endpositionen, den Leserahmen und das Alignment selbst. Dabei können Alignments auch in mehrere High-Scoring Fragmente zerteilt werden, denen jeweils separate Ähnlichkeits-Scores, Koordinaten und Leserahmen zugeordnet sind. In diesem Fall wird für das zusammengefaßte Alignment ein Gesamt-Ähnlichkeitswert ermittelt.

Alignierte Regionen mit einer Länge > 300 BP, die nur einen Leserahmen umfassen und eine genügend große Identität aufweisen (Ähnlichkeits-Score > 750), werden als "Seed-ORFs" ausgewählt. Sie werden upstream zum nächsten potentiellen Startcodon aus {ATG, GTG, TTG} und downstream zum nächsten Stopcodon aus {TAG, TAA, TGA} im gleichen Leserahmen verlängert. Mit der Menge der Seed-ORFs werden dann die Parameter für das sogenannte "Coding-Potential" festgelegt. Das Coding-Potential bezieht sich hierbei auf die unterschiedliche Triplettverteilung in codierenden und

[5] vgl. BLASTX, der Unterschied besteht darin, daß BLASTX keine kompletten Genome (3-5 MBP und mehr) abgleichen kann.

nicht-codierenden Regionen.
Im nächsten Schritt werden die ORFs weiter upstream verlängert. Dazu wird der Genstart zum nächsten Startcodon in 5'-Richtung verschoben, wobei die Region zwischen dem alten und dem neuvorhergesagten Start den Coding-Parametern entsprechen muß, das heißt, die Triplettverteilung muß der einer codierenden Region entsprechen. Mit Hilfe der Seed-ORFs werden dann Codon-Usage Tabellen, das arithmetische Mittel des Coding Potentials und die Standardabweichung berechnet.
Alignments, die mehrere Leserahmen enthalten, werden nicht als Seed-ORFs verwendet. Sie dienen zur Detektion von Frame-Shifts.

Berechnung von Coding-Potential und Coding-Qualität

Die Häufigkeit eines Tripletts t_i, $f(t_i)$, wird zur Berechnung des statistischen Gewichtes des Codons $W(t_i)$ benötigt. Das Triplett wird mit $b_i^1 b_i^2 b_i^3$, notiert, wobei b_i^1 das erste Nucleotid von t_i, b_i^2 das zweite und b_i^3 das dritte bezeichnet. Die Häufigkeit des Tripletts wird entsprechend mit $f(b_i^1 b_i^2 b_i^3)$ angegeben. Das Gewicht $W(t_i)$ wird mit $W(b_i^1 b_i^2 b_i^3)$ notiert und ist definiert als der Logarithmus der Häufigkeit des Tripletts (Gleichung (2.10)).

$$W(b_i^1 b_i^2 b_i^3) = \log f(b_i^1 b_i^2 b_i^3) \tag{2.10}$$

$W(b_i^1 b_i^2 b_i^3)$	Gewichtung eines Tripletts $t_i = b_i^1 b_i^2 b_i^3$
$f(b_i^1 b_i^2 b_i^3)$	Häufigkeit eines Tripletts $t_i = b_i^1 b_i^2 b_i^3$
i	Positionsindex in der Triplettsequenz

Das primäre Coding-Potential Q einer Sequenz wird definiert als Summe der Gewichtungen aller n Tripletts (Gleichung (2.11)).

$$Q(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3) = \sum_{k=1}^{n} W(b_k^1 b_k^2 b_k^3) \tag{2.11}$$

$Q(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3)$	Coding-Potential der Sequenz $b_1^1 b_1^2 b_1^3 .... b_n^1 b_n^2 b_n^3$, wobei $b_1^1 b_1^2 b_1^3$ die Nucleotide eines Tripletts t_1 und $b_n^1 b_n^2 b_n^3$ die Nucleotide eines Tripletts t'_n bezeichnen.
n	Länge der Sequenz in Tripletts

Das Coding-Potential muß normalisiert werden, um Fragmente unterschiedlicher Länge miteinander vergleichen zu können. Die Normalisierung erfolgt nach Gleichung (2.12).

$$R = \frac{Q - \mu n}{\sigma\sqrt{n}} \tag{2.12}$$

R Normalisiertes Coding-Potential

mit μ, dem mittleren Triplettgewicht, gegeben durch

$$\mu = \sum_{b_1 b_2 b_3 = AAA}^{TTT} f(b_1) f(b_2) f(b_3) W(b_1 b_2 b_3)$$

und σ der Standardabweichung, gegeben durch

$$\sigma^2 = \sum_{b_1 b_2 b_3 = AAA}^{TTT} f(b_1) G(b_2) f(b_3) [W(b_1 b_2 b_3)]^2,$$

wobei

$f(b)$ die Häufigkeit der Base b im Genom

und

$W(b_1 b_2 b_3)$ das Gewicht des Tripletts $b_1 b_2 b_3$ ist.

Das normalisierte Coding-Potential wird nun genutzt, um die Coding-Qualität $\Omega(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3)$ zu berechnen (Gleichung (2.13)). Durch die Berechnung der Coding-Qualität soll der Einfluß lokaler Nucleotidanordnungen gering gehalten, und Leserahmen sowie Strang mit einbezogen werden.

$$\begin{aligned} \Omega(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3) &= R(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3) \\ - \max \{ R(b_0^3 b_1^1 b_1^2 ... b_{n-1}^3 b_n^1 b_n^2) R(b_1^2 b_1^3 b_2^1 ... b_n^2 b_n^3 b_{n+1}^1) \} \end{aligned} \tag{2.13}$$

$\Omega(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3)$ Coding-Qualität der Triplettsequenz 1 bis n.

$R(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3)$ Coding-Potential der Triplettsequenz 1 bis n, wobei $b_1^1 b_1^2 b_1^3$ die Nucleotide eines Tripletts t_1 bezeichnen und $b_n^1 b_n^2 b_n^3$ die Nucleotide eines Tripletts t_n'.

$R(b_0^3 b_1^1 b_1^2 ... b_{n-1}^3 b_n^1 b_n^2)$ Coding-Potential der Sequenz einen Leserahmen versetzt im Vergleich zu $R(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3)$ (upstream)

$R(b_1^2 b_1^3 b_2^1 ... b_n^2 b_n^3 b_{n+1}^1)$ Coding-Potential der Sequenz einen Leserahmen versetzt im Vergleich zu $R(b_1^1 b_1^2 b_1^3 ... b_n^1 b_n^2 b_n^3)$ (downstream)

max Maximum

Nach der Berechnung der Coding-Qualität wird die Einteilung der Sequenz durch die Seed-ORFs überprüft. Regionen, die nach der Ähnlichkeits-Suche als nicht-codierend eingestuft wurden, werden mit Hilfe der Coding-Qualität nach weiteren potentiell codierenden ORFs durchmustert. Um akzeptiert zu werden, müssen betrachtete ORFs mindestens eine Länge von 100 Tripletts und eine genügend hohe Coding-Qualität aufweisen.
Alle vorhergesagten ORFs werden in 5'-Richtung so weit wie möglich verlängert, wobei Überlappungen bis 6 BP erlaubt sind. Die Verlängerung erfolgt in 99 BP-Schritten, solange die Coding-Qualität der Regionen genügend groß ist. Die Schrittweite von 99 BP wurde gewählt, um eine ausreichende Signifikanz der berechneten Coding-Qualität zu gewährleisten (Fickett und Tung, 1992). Dadurch erhält man eine Menge an potentiell codierenden Regionen mit offenem Start.

Auffinden der Genstarts

Um die Startpositionen zu bestimmen, wird eine gewichtete RBS-Matrix berechnet. Diese wird iterativ mit Hilfe der ORFs angepasst, denen die besten Scores zugeordnet werden (jeweils die besten 80%) und damit bei jeder Iteration verbessert. Für die initiale Berechnung werden alle mutmaßlich codierenden ORFs ausgewählt, die nur ein mögliches Startcodon aufweisen und mindestens 30 BP Abstand zum nächsten potentiell codierenden ORF haben. Die ausgewählten ORFs werden in der Region -20 bis +3 an den jeweiligen Startcodons aligniert. Mit Hilfe dieser Sequenzen kann nun eine gewichtete RBS-Matrix generiert werden. Mit $L = 6$ BP wird die Länge möglicher SD-Boxen abgeschätzt. Die Nucleotidhäufigkeit einer Base b an der Position j ist durch $f(b, j)$ angegeben, wobei j Werte im Bereich $-20...-L$ annehmen kann (bezogen auf den Anfang des aktuell betrachteten ORFs). Damit wird ein positionsabhängiger Informationsgehalt für alle Positionen im Bereich $-20...-L$ berechnet nach Gleichung (2.14) (Schneider *et al.*, 1986). Der Informationsgehalt bezieht sich hierbei auf Hinweise für eine mögliche Ribosom-Bindestelle.

$$H(j) = \sum_{b=A}^{T} f(b, j) \log(f(b, j)/f(b)) \tag{2.14}$$

$H(j)$	Positionsabhängiger Informationsgehalt für die Position j mit $j = -20... - L$, bezogen auf den betrachteten ORF.
$f(b, j)$	Positionsabhängige Nucleotidhäufigkeit einer Base b an Position j mit $j = -20... - L$, bezogen auf den betrachteten ORF.
b	Base aus {T, A, G, C}
$f(b)$	Häufigkeit der Base b im Genom

Es werden putative SD-Segmente an den Positionen mit maximalem Informationsgehalt für jeden potentiell codierenden ORF aus der Menge der alignierten ORFs ausgewählt (Gleichung (2.15)). Dabei wird jeweils eine Box der Länge L an der Position j betrachtet. Die jeweiligen Informationsgehalte der Nucleotide einer mutmaßlichen SD-Box $H(k)$, mit $k = j...(j+L-1)$ und $j = -20... - L$, werden aufaddiert. Aus den Summen der Informationsgehalte der Boxen wird diejenige mit der maximalen ausgewählt. Die zugehörige potentielle SD-Sequenz ist die wahrscheinlichste für den betrachteten ORF.

$$\left(\sum_{k=j}^{(j+L-1)} H(k) \right) \xrightarrow[j = -20... - L]{} \max \tag{2.15}$$

Die Summe der positionsabhängigen Informationsgehalte der einzelnen putativen SD-Boxen eines ORFs wird gegen ein Maximum geführt. Dabei ist j der Positionsindex, der innerhalb eines Fensters der Länge $(20 - L)$ liegt. Als Position des Fensters wird -20 bis $-L$ gegeben (in bezug auf den betrachteten potentiellen Genstart). k bezeichnet die Position innerhalb einer Box der Länge L, die im Bereich $j...(j + L - 1)$ liegt.

Anschließend wird die wahrscheinlichste Position einer SD-Box für alle als codierend angenommenen ORFs bestimmt. Dabei werden wiederum die Regionen upstream der möglichen Genstarts der ORFs betrachtet. Die Berechnungen sind in zwei Schritte aufgeteilt und erfolgen iterativ. Zunächst werden die positionsabhängigen Gewichte der Nucleotide in den potentiellen SD-Segementen bestimmt nach Gleichung (2.16).

$$W(b, k) = \log \left(\frac{N_{SD}(b, k) + 0,5}{\max_b N_{SD}(b, k) + 0,5} \right) \tag{2.16}$$

$W(b, k)$	Gewicht einer Base b bei Position k, wobei k die Werte 1... L, bezogen auf die betrachtete Box, annehmen kann.

$N_{SD}(b,k)$	Anzahl der Vorkommen einer Base b bei Position k in allen betrachteten potentiellen SD-Boxen der aktuellen Iteration.
$\max N_{SD}(b,k)$	Anzahl übereinstimmender Nucleotide an Position k in allen betrachteten potentiellen SD-Boxen

Damit werden die Scores der SD-Boxen für die erste Matrix berechnet nach Gleichung (2.17).

$$\Delta(b_1...b_L) = \sum_{k=1}^{L} W(b_k, k) \tag{2.17}$$

$\Delta(b_1...b_L)$	Score der SD-Box

Für jede Sequenz wird das SD-Segment der Länge L mit den maximalen Scorewerten gesucht, wobei die Matrix in jeder Iteration mit verbesserten Werten neu berechnet wird. Der vorhergesagte Genstart downstream des SD-Segmentes wird als wahrscheinlichste Startposition angenommen.
Anschließend folgt die zweite Stufe der Berechnungen, die den Abstand der SD-Sequenz vom Startcodon einbezieht. M sei eine mögliche Position der SD-Box innerhalb der RBS-Sequenz und $N(M)$, die Anzahl der Vorkommen. Die maximale Anzahl wird mit $N_{\max}$ bezeichnet. Die Berechnung der positionsabhängigen Gewichte $W(M)$ einer Box erfolgt nach Gleichung (2.18).

$$W(M) = \log\left(\frac{N(M) + 0,5}{N_{\max} + 0,5}\right) \tag{2.18}$$

$N(M)$	Anzahl der Vorkommen der SD-Box
$N_{\max}$	maximal aufgetretene Anzahl

Die Stärke des SD-Signals wird nun definiert nach Gleichung (2.19).

$$\Delta(b_1...b_L) = \sum_{k=1}^{L} W(b_k, k) + W(M) \tag{2.19}$$

Die Berechnung des abstandsabhängigen SD-Signals erfolgt ebenfalls iterativ, wobei die Werte in jeder Iteration verbessert werden.
Abschließend wird jedem offenen ORF ein Startcodon zugeteilt. Dieses sollte downstream einer potentiellen SD-Box mit genügend hohen Scorewerten liegen, wobei der Abstand die Länge 20 BP nicht überschreiten darf. Sollte kein Startcodon in diesem Rahmen gefunden werden, so wird der nächst kleinere ORF verwendet.

Behandlung von sehr kurzen ORFs

Standardmäßig sagt ORPHEUS nur ORFs > 80 Codons vorher. Das Problem dabei ist, daß sehr kurze ORFs häufig höhere Scores aufweisen als längere und diese so aus der Menge der potentiellen Gene verdrängen. Um dem entgegen zu wirken, werden in den ersten Iterationen nur ORFs größer 2000 BP akzeptiert. Die minimale Länge akzeptierter ORFs wird dann in jeder weiteren Iteration reduziert, wobei keine Vorhersagen aus der vorangehenden Iteration, das heißt längere potentiell codierende ORFs, ausgeschlossen werden.

2.2.2 Anwendung von ORPHEUS

Unterprogramme und Kommandozeilen-Parameter

Zunächst wird das unabhängige Programm `dps` aufgerufen. Hier müssen der Pfad zu einer nicht-redundante Protein-Datenbank und der einer BLOSUM-Matrix übergeben werden (Abschnitt 2.2.2, S. 31). Die Form der BLOSUM-Matrix ist in dem Manual des Programms vorgegeben. Empfohlen wird die Verwendung einer BLOSUM62-Matrix. Mit dem Parameter `-f` muß ein Schwellenwert gesetzt werden. Empfohlen wird hier 600. Weiterhin muß mit `-c` die maximale Anzahl zu alignierender Sequenzen angegeben werden. Hier wird 999999 empfohlen. Optional können weitere Parameter übergeben werden, auf die hier nicht näher eingegangen wird.
Anschließend wird mit dem Programm `starter` die Basenverteilung im Genom ermittelt und ausgegeben. Der Modus des Programms wird mit dem Parameter `-q` übergeben. Ist die Verteilung bekannt, kann dieser Schritt übergangen werden.
Die Basenverteilung wird für die weiteren Programmaufrufe benötigt. Zunächst wird das Programm `orpheus2` aufgerufen, dem mit `-genome` das Genome, die Ausgabe von `dps` und die Basenhäufigkeiten übergeben werden. Außerdem muß mit dem Parameter `-wcu` ein File übergeben werden, in dem die Codon-Usage gespeichert wird. Mit dem Parameter `-minlength` kann eine Mindestlänge für ORFs vorgegeben werden. Der Defaultwert ist auf 240 BP gesetzt. Im Rahmen von YACOP wurde die Mindestlänge auf 150 BP gesetzt. Für die Vorhersage kürzerer Gene ist ORPHEUS nicht geeignet.

Wird die Mindestlänge auf einen niedrigeren Wert als 150 BP gesetzt, so steigt die Anzahl falsch-positiver Vorhersagen unverhältnismässig stark an. Der Aufruf des Programms wird wiederholt, wobei nun das File, in dem die Codon-Usage festgehalten ist, mit dem Parameter `-rcu` übergeben wird. Die Ausgabe des Programms `dps` wird mit dem Parameter `-sure` übergeben. Das Programm schreibt seine Ausgabe bei diesem Aufruf in ein File mit dem Namen `genome.orfaln`. Das File wird in das Verzeichnis, in dem die Sequenz in Fasta-Format liegt, geschrieben. `genome` bezeichnet dabei den Namen des Fasta-Files ohne Suffix.
Dann erfolgt ein erneuter Aufruf des Programms `starter`, bei dem mit dem Parameter `-m` der Modus angezeigt wird, in dem die Gewichtungen berechnet werden. Diese werden mit der Ausgabeumleitung in das File `genome.weights` geschrieben. Zu beachten ist, daß die Basenhäufigkeiten ohne Leerzeichen zum jeweiligen Parameter übergeben werden müssen.
Als letztes wird wiederum das Programm `orpheus2` aufgerufen. Im Unterschied zum letzten Aufruf muß hier mit dem Parameter `-w` das File mit den Gewichtungen übergeben werden. Die Ausgabe des Programms `dps` wird ohne Parameter übergeben. Außerdem kann mit dem Parameter `-excl` ein Ausgabefile des Programms `tRNAScanSE` übergeben werden. Hiermit kann unter Umständen das Ergebnis verbessert werden, dies konnte allerdings anhand der getesteten Genome nicht gezeigt werden, da die Ergebnisse für die getesteten Genome durch `tRNAScanSE` nicht verändert wurden.

Manueller Aufruf der Unterprogramme

1. `dps genome.fasta nr BLOSUM62 -f 600 -c 999999 > genome.dps`
2. `starter -q genome.fasta`
3. `orpheus2 -genome genome.fasta -a 0,25 -c 0,25, -t 0,25 -g 0,25 [-minlength 150] -wcu genome.codon_usage genome.dps`

   (Ausgabedatei genome.orfnuc)
4. `orpheus2 -genome genome.fasta -a 0,25 -c 0,25, -t 0,25 -g 0,25 [-minlength 150] -rcu genome.codon_usage`

```
-sure genome.dps
```

5. `starter -m -a0,25 -c0,25, -t0,25 -g0,25`
   `genome.orfaln > genome.weights`

6. `orpheus2 -genome genome.fasta -a 0,25 -c 0,25, -t 0,25`
   `-g 0,25 [-minlength 150] -rcu genome.codon_usage`
   `-w genome.weights genome.dps [-excl genome.tRNA]`

Beispiel für die Ausgabe von ORPHEUS

ORPHEUS produziert bei jedem Aufruf des Programms `orpheus2.pl` drei Files, die in das Verzeichnis geschrieben werden, in dem das Genom liegt. Dabei enthält das File mit dem Suffix `.orfprot` die Aminosäuresequenzen der vorhergesagten Gene, das File `*.orfnuc` die zugehörigen Nucleotidsequenzen und das File `*.orfaln` alle Alignments, die eine Ribosom-Bindestelle upstream eines Startcodons aufweisen, in CLUSTALW-Format. Als Beispiel wird hier nur die Ausgabe der Nucleotidsequenzen gezeigt.

```
>orf424 500777 502582 gi|15617054|ref|NP_240267.1| (NC_002528) exodeoxyribonuc
lease V 67 kD polypeptide [Buchnera sp. APS]^Agi|1 1132309|sp|P57530|EX5A_BUCA
I Exodeoxyribonuclease V alpha chain^Agi|10039119|dbj|BAB13153.1| (AP001119) e
xodeoxyribonuclease V 67 kD polypeptide [Buchnera sp. APS]
ATGCTTATTTTATTAAAAAAAGCTGTAAAATTGAAAATTATACGTCCAAT
TGATTTTTATTTTTCTCAATTTATCGCACAAAAAAATAATATTGTTATGT
TAGTTGCTGCTTGTGTAAGTTATGAAAGCAGTAGAGGTTATATTTCTCTG
CCAATTAAATATTTTGAAAAACATTATTTTTTTTCTAGTTCAAATGAAGT
```

2.3 GLIMMER2.02 und RBSfinder

(Salzberg *et al.*, 1999)

Glimmer2.02 baut auf Glimmer1.0 auf. Daher wird hier zum Verständnis auch auf die ältere Version eingegangen. Im Verlauf dieser Arbeit wurde von den Autoren die neuste Version, Glimmer2.10, veröffentlicht. Es wurde kein Paper publiziert, das die Unterschiede zu den älteren Versionen beschreibt, daher wird hier nicht näher auf diese Version eingegangen.

2.3.1 Strategie von GLIMMER

Die Strategie der Genvorhersage von Glimmer basiert alleine auf der intrinsischen Analyse. Der Vorteil ist, daß das Programm nicht auf Datenbank-Abgleiche angewiesen ist und somit sehr viel schneller und stabiler arbeitet als andere Programme, wobei der Speicheraufwand geringer ist. Damit ist Glimmer auch für die Anwendung auf einfachen Arbeits-PCs mit geringer Rechenleistung geeignet.

Glimmer1.0 verwendet eine erweiterte Form von Markov-Ketten für die Genvorhersage. Markov-Modelle einer Ordnung k sagen ein Nucleotid an der Position i auf Basis der vorangehenden k Nucleotide vorher. Dem System wird eine Trainingsmenge T übergeben, aus der es lernt, mit welcher Wahrscheinlichkeit eine Base in einem codierenden Kontext auf k vorangehende Basen beliebiger Kombination folgt. Das heißt, es werden die Häufigkeiten aller Oligomere der Länge $k+1$ festgestellt, worüber dann die Wahrscheinlichkeiten berechnet werden können. Dabei ist zu beachten, daß Glimmer inhomogene Modelle generiert. Für jede der drei Positionen in einem Triplett wird ein anderes Modell erstellt[6]. Das jeweils betrachtete Fenster wird nicht von einer Base zur nächsten verschoben, sondern springt zum nächsten Triplett. Der letzten Base eines Oligomeres, das häufig in der Trainingsmenge auftritt, wird eine hohe Wahrscheinlichkeit zugeordnet, bei seltenen Oligomeren erhält die letzte Base eine niedrige Wahrscheinlichkeit. Mit Hilfe der Wahrscheinlichkeiten können dann den ORFs der Sequenz Scorewerte zugeordnet werden. ORFs, deren Scorewert einen Schwellenwert überschreitet,

[6] vgl. GeneMark, 3-periodische Markov-Ketten, Borodovsky und Mcininch, 1993.

werden als potentiell codierend angenommen.
Im Unterschied zu anderen Ansätzen verwendet GLIMMER1.0 interpolierte Markov-Modelle (IMMs), deren Ordnung nicht fix ist. Durch die variable Länge der gelernten Oligomere (siehe oben) wird eine höhere Flexibilität erreicht. Da es nötig ist, daß ausreichend Vorkommen der $(k+1)$mere in der Trainingsmenge enthalten sind, um die Wahrscheinlichkeiten aller $(k+1)$mere mit hinreichender statistischer Sicherheit zu lernen, ist es von Vorteil, sich nicht auf Oligomere einer bestimmten Länge festzulegen. Für die Genvorhersage sind möglichst lange Oligomere ideal. Für große k reicht aber oft die Datengrundlage der Trainingsmenge nicht aus. Ein Markov-Modell $k$terOrdnung einer DNA-Sequenz bedingt 4^{k+1} Wahrscheinlichkeiten, die aus der Trainingsmenge gelernt werden müssen. Da die Anzahl der zu lernenden Wahrscheinlichkeiten exponentiell mit der Ordnung steigt, muß k begrenzt werden. GLIMMER1.0 verwendet Markov-Ketten nullter bis achter Ordnung. Es werden nur Oligomere betrachtet, für die eine ausreichende Datengrundlage vorhanden ist, wobei Oligomeren, die häufiger auftreten, ein höheres Gewicht zugeordnet wird als weniger häufigen Oligomeren. Für Oligomere, die nicht in ausreichender Anzahl gefunden werden, wird das jeweils nächst kürzere mit signifikanter Häufigkeit zur Vorhersage herangezogen.

GLIMMER2.02 verwendet Interpolierte Context Modelle (ICMs). Diese sind eine erweiterte Form der IMMs. Eine Base b_i wird hierbei durch Basen an beliebiger Position im vorangehenden $k$mer vorhergesagt. Das heißt, es müssen nicht alle Basen der Kette mit denen im betrachteten $k$mer übereinstimmen. Dadurch wird die Flexibilität im Vergleich zu den IMMs erhöht und die Tatsache mit einbezogen, daß die Aussagekraft einer Base stark von ihrer Position innerhalb eines Codons abhängt. Da viele Basenaustausche an der dritten Position im Codon irrelevant für die resultierende Aminosäuresequenz sind, konnte die Vorhersage durch die Verwendung von ICMs deutlich verbessert werden.

Erweiterte Markov-Modelle: IMMs und ICMs

Eine DNA-Sequenz kann als zufällige Folge von Symbolen eines Alphabetes $\mathbb{A}$ mit den Elementen A,G,T,C betrachtet werden. Die Basen werden dargestellt durch die Variablen $b_1, b_2 ... b_n$, die einen Zustand aus $\mathbb{A}$ annehmen können. Dabei bezeichnet n die Länge der Sequenz und b_i eine Base an der Position i, mit $i > 0$ und $i \leq n$. Die Variable b_i folgt mit einer Wahrscheinlichkeit P auf die vorangehende Markov-Kette der Ordnung k. Wenn beispielsweise die Base A mit einer Wahrscheinlichkeit von 0,2 auf A folgt, ist $P = 0,2$ für eine Markov-Kette der Ordnung $k = 1$. Für eine Folge AAAAA beträgt die Wahrscheinlichkeit dann $P = (0,2)^5$. Die Wahrscheinlichkeit, daß eine Variable b_i einen bestimmten Zustand aus $\mathbb{A}$ annimmt, hängt in diesem Fall nur von dem der vorangehenden Variable b_{i-1} ab. Betrachtet man eine DNA-Sequenz, so muß für Markov-Modelle erster Ordnung eine Matrix mit $4^{1+1} = 16$ Wahrscheinlichkeiten, $P(a|a), P(a|g), ...$ generiert werden. Bei Markov-Ketten zweiter Ordnung werden die Zustände von zwei vorangehenden Variablen b_{i-2} und b_{i-1} betrachtet und es muß eine Matrix mit 4^{2+1} Wahrscheinlichkeiten berechnet werden.
Nimmt man an, daß das Auftreten einer Base b_i von zwei vorangehenden Basen b_{i-2} und b_{i-1} abhängt, so liefert ein Modell zweiter Ordnung bessere Ergebnisse als ein Modell erster Ordnung. Hängt die Base aber nur von der unmittelbar vorangehenden Base b_{i-1} ab, so liefern beide Modelle das gleiche Ergebnis. Ein Modell höherer Ordnung ist daher grundsätzlich zu bevorzugen. Dabei tritt allerdings das schon erwähnte Problem auf, daß die Anzahl der zu lernenden Wahrscheinlichkeiten exponentiell mit der Ordnung steigt, während die Datengrundlage für Ketten höherer Ordnung meist schlechter ist. Durch die Verwendung von interpolierten Markov-Modellen, ist das System, wie oben beschrieben, im Vergleich zu früheren Ansätzen flexibler. Es können Ketten höherer Ordnung verwendet werden, soweit die Datengrundlage dies zuläßt. Wenn keine ausreichende Datengrundlage besteht, wird auf Modelle niedrigerer Ordnung zurückgegriffen. Durch die Verwendung von ICMs wird das System noch flexibler.
Der gemeinsame Informationsgehalt zweier beliebiger Basen b und c wird

dabei definiert nach Gleichung (2.20).

$$I(b;c) = \sum_i \sum_j P(b_i, c_j) \log \left(\frac{P(b_i)P(c_j)}{P(b_i, c_j)} \right) \tag{2.20}$$

$I(b;c)$	Gemeinsamer Informationsgehalt der Basen b und c
$P(b_i)$	Wahrscheinlichkeit des Auftretens von Base b an der Position i
$P(c_j)$	Wahrscheinlichkeit des Auftretens von Base c an der Position j
$P(b_i, c_j)$	Verbundwahrscheinlichkeit des Autretens von b an Position i, wenn c an Position j auftritt.

Um die ICMs für ein Markov-Modell der Ordnung k zu generieren, werden alle Oligomere der Länge $k+1$ aus der Trainingsmenge gelernt. Sei b_1 die Basenverteilung an der Position 1 der Oligomere, b_2 die Verteilung an Position 2 und so weiter bis b_{k+1}. Es werden die gemeinsamen Informationsgehalte $I(b_1; b_{k+1}), I(b_2; b_{k+1}), ...I(b_k; b_{k+1})$ ermittelt. Von diesen Werten wird das Maximum $I(b_j; b_{k+1})$ ausgewählt. Die Gesamtmenge V aller Oligomere der Länge $k+1$ wird nun in vier Untermengen $V_V^A, V_V^C, V_V^G, V_V^T$ für die vier Nucleotide, die an Position j auftreten können, aufgeteilt. Der Index $_V$ bezeichnet dabei die Menge aus der V_V^b mit $b \in$ {A,C,G,T} hervorgegangen ist ($V_V^b \subset V$). Das heißt, die Menge V_V^A umfaßt alle Oligomere der Länge $k+1$ aus der Menge V mit einem A an Position j, die Menge V_V^C alle mit einem C an Position j usw. Mit den vier Teilmengen wird dann wiederum genauso verfahren. Es wird in jeder der Mengen, V_V^b mit $b \in$ {A,T,C,G}, die Position ausgewählt, die den höchsten Wert I mit der Base b_{k+1} aufweist. Daraus werden neue Untermengen gebildet, für die gilt $V_{V_V^b}^c \subset V_V^b$ mit $b, c \in \{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$ (Abbildung 2.2, S. 37). Vergleichbar ist der Prozess mit dem Aufbau einer Baumstruktur, wobei jeweils die Tiefe des Baumes die Gewichtung ausmacht. Der Vorgang wird solange fortgeführt, bis eine vorher festgesetzte maximale Baumtiefe erreicht ist, oder die Größe der Mengen zu klein wird, um Wahrscheinlichkeiten für die letzte Position zu berechnen (Beispiel 2.3.1, S. 37).

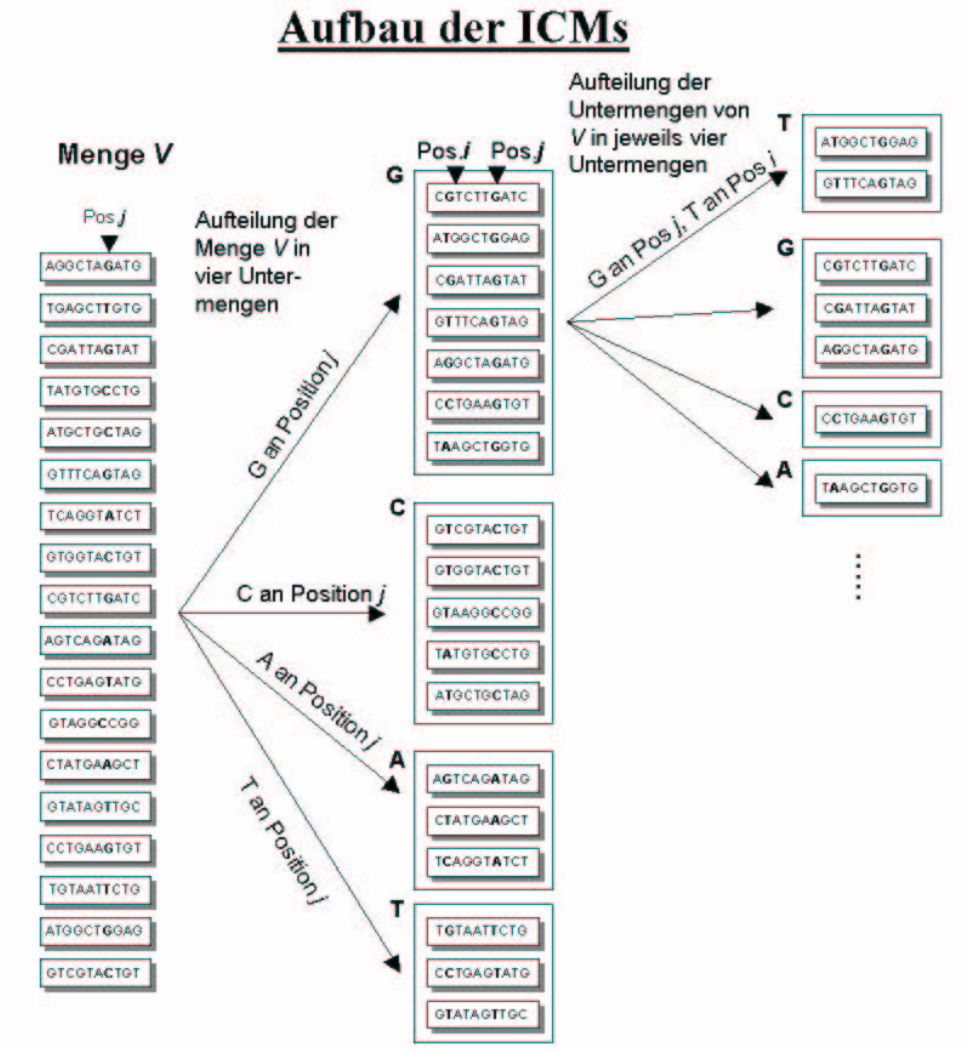

Abbildung 2.2: **Abbildung zum Aufbau der ICMs:** Für jede Menge von Oligomeren einer Länge $k+1$ wird jeweils die Position j identifiziert, für die der maximale Informationsgehalt I mit der Base $k+1$ bestimmt wurde ($I(b_j, b_{k+1})$). Die Menge wird dann in vier Untermengen aufgeteilt, die den an Position j gefunden Basen entsprechen. Für die Untermengen werden dann wiederum die Positionen mit maximalem I in Kombination mit der Base an Position $k+1$ gesucht. Die Untermengen werden dann analog zur ersten Menge aufgeteilt.

Trainieren von IMMs und generieren der Matrizen

Zum Trainieren der Modelle werden zunächst lange ORFs in der Sequenz gesucht, die mit hoher Wahrscheinlichkeit codierende Regionen enthalten. Mit diesen Trainings-ORFs werden die Häufigkeiten von Motiven der Länge 1 bis $k+1$ für alle sechs Leserahmen festgestellt. Dabei wird der Leserahmen durch die jeweils letzte Base festgelegt. Optional kann ein weiteres Modell für nicht-codierende Regionen generiert werden. Dies ist aber nicht zwingend notwendig. Die Häufigkeit einer Sequenz $S = s_1...s_{k+1}$ wird mit $f(S)$ angegeben. Das Auftreten einer Base s_i im Kontext von $s_{i-k}, s_{i-k+1}, ...s_{i-1}$ wird mit $S_{i,k}$ bezeichnet. Die Matrix mit den Wahrscheinlichkeiten $P_k(S_i)$ für das Auftreten der Base s an der Position i nach einer Folge von k Basen wird dann berechnet nach Gleichung (2.21).

$$P_k(S_i) = P(s_i|S_{i,k}) = \frac{f(S_{i,k})}{\sum_{b\epsilon\{A,G,T,C\}} f(S_{i,k}, b)} \tag{2.21}$$

$P_k(S_i)$	Wahrscheinlichkeit für das Auftreten einer Base s_i in der Sequenz S
$f(S_{i,k})$	Häufigkeit der Base an der Position i nach einer Folge von k Basen

Hat GLIMMER eine gewichtete Matrix mit Hilfe der Trainingsmenge T generiert, so wird die Wahrscheinlichkeit, daß ein Modell M eine Sequenz S beinhaltet mit $P(S|M)$ angegeben (Gleichung (2.22)).

$$P(S|M) = \sum_{i=1}^{n} \mathrm{IMM}_k(S_i) \tag{2.22}$$

$P(S\|M)$	Wahrscheinlichkeit des Vorkommens des Modells M in der Sequenz S
n	Länge der Sequenz
k	Ordnung des Modells
S_i	Oligomer, das an Position i endet

Das Markov-Modell $k$ter Ordnung, $\mathrm{IMM}_k(S_i)$, wird berechnet nach Gleichung (2.23).

$$\mathrm{IMM}_k(S_i) = \lambda_k(S_{i-1}) \bullet P_k(S_i) + [1 - \lambda_k(S_{i-1})] \bullet \mathrm{IMM}_{k-1}(S_i) \tag{2.23}$$

$\lambda_k(S_{i-1})$	Gewichtung des $k$meres, das an Position $i-1$ endet
$P_k(S_i)$	Wahrscheinlichkeit mit der eine Base an der Position i in einem Modell $k$ter Ordnung auftritt

Das heißt, die Vorhersagen eines Modells achter Ordnung sind eine Kombination der Vorhersagen der Modelle nullter bis siebter Ordnung. Der Wert $\lambda_k(S_{i-1})$ bezeichnet die Gewichtung und ist somit ein Maß dafür, wie häufig das betrachtete (k+1)mer in der Trainingsmenge aufgetreten ist. Überschreitet die Häufigkeit eines Oligomeres einen Schwellenwert, der bei GLIMMER1.0 standartdmäßig mit 400 festgesetzt ist, so wird $\lambda_k(S_{i-1}) = 1,0$ gesetzt. Wenn der Schwellenwert überschritten ist, wird die Berechnung abgebrochen, da man mit $\sim$ 95%ger Wahrscheinlichkeit annehmen kann, daß der bisher berechnete Wert um höchstens $\pm 0,05$ vom wahren Wert (wenn alle Vorkommen des Oligomeres betrachtet würden) abweicht. Tritt ein Motiv nicht genügend häufig in einer Sequenz auf, wird ein zusätzliches Kriterium zur Berechnung von λ herangezogen. Es werden die Häufigkeiten der Basen ($f(S_{i,k}, \mathrm{A})$, $f(S_{i,k}, \mathrm{C})$, $f(S_{i,k}, \mathrm{T})$, $f(S_{i,k}, \mathrm{G})$) und die der Markov-Kette der nächst kleineren Ordnung ($f(S_{i,k-1}, \mathrm{A})$, $f(S_{i,k-1}, \mathrm{C})$, $f(S_{i,k-1}, \mathrm{T})$, $f(S_{i,k-1}, \mathrm{G})$) betrachtet. Mit dem χ^2-Test wird berechnet, ob die Häufigkeiten der Basen in dem Muster mit den Wahrscheinlichkeiten der Markov-Kette der Ordnung $k-1$ entsprechen. Ist dies der Fall, so wird der längeren Kette ein niedrigerer λ-Wert zugeordnet. Andernfalls erhält sie einen größeren Wert. Es wird ein Wert c berechnet, der mit dem χ^2-Test konform ist. Dieser geht in die Berechnung des Gewichtes ein. $\lambda_i(S_{x-1})$ wird berechnet nach Gleichung (2.24).

$$\lambda_i(S_{x-1}) = \begin{cases} 0 & \text{, falls } c < 0,50 \\ \frac{c}{400} \sum_{b \epsilon \{A,G,C,T\}} f(s_1 s_2 ... s_i b) & \text{, falls } c \geq 0,50 \end{cases} \tag{2.24}$$

Behandlung von Überlappungen

Es hat sich gezeigt, daß von GLIMMER vorhergesagte Startpositionen oft zu weit in 5'-Richtung angesiedelt werden. Dadurch kommen Überlappungen mit Genen zustande, die in der Nachbarschaft vorhergesagt wurden. Es muß nun entschieden werden, welchem potentiellen Gen die überlappende Region zugeordnet wird. GLIMMER1.0 löste dieses Problem, indem der Scorewert für beide Gene in der Überlappung berechnet wurde und die Längen der Gene verglichen wurden. Kann die Überlappung durch Verschieben des Starts nicht aufgelöst werden, so wurde das Gen mit dem niedrigeren Scorewert in

der Überlappung aus der Vorhersage entfernt.
GLIMMER2.02 hat einen differenzierteren Ansatz. Überlappen sich zwei Sequenzen A und B, so werden wie in GLIMMER1.0 jeweils die Scorewerte für die Überschneidung berechnet. Das weitere Vorgehen hängt jedoch von der Orientierung der beiden potentiellen Gene ab. Es werden vier Fälle unterschieden.

1. Angenommen A weist jeweils einen höheren Score auf als B. Im ersten Fall überschneiden sich die Gene A und B gegenläufig. Dabei liegen jeweils die terminalen Regionen von A und B in der überlappenden Region.

   $$A \ {}^{5'}\!\longrightarrow\!{}^{3'}$$
   $$\qquad {}^{3'}\!\longleftarrow\!{}^{5'} \ B$$

   Die Verschiebung des Startes von B würde die Überschneidung nicht auflösen. Ist A signifikant länger als B, so wird B verworfen. Andernfalls, werden beide Gene beibehalten. Sie werden mit der Kennzeichnung versehen, daß sie eine Überlappung aufweisen.

2. Im zweiten Fall ist die Orientierung von A und B gleich, wobei der Start von A weiter in 5'-Richtung liegt.

   $$A \ {}^{5'}\!\longrightarrow\!{}^{3'}$$
   $$\quad B \ {}^{5'}\!\longrightarrow\!{}^{3'}$$

   Hier kann nur die Verschiebung des Startes von B die Überlappung auflösen. Ist keine Verschiebung des Startes von B möglich, wird B verworfen, sofern es signifikant kürzer ist als A. Ist das nicht der Fall, so werden beide Gene behalten und wie im ersten Fall gekennzeichnet.

3. Im dritten Fall ist die Orientierung wie im zweiten mit dem Unterschied, daß hier der Start von B weiter in 5'-Richtung liegt.

$$A \ ^{5'}\!\longrightarrow^{3'}$$
$$B \ ^{5'}\!\longrightarrow^{3'}$$

Nur die Verschiebung des Startes von A kann hier die Überlappung auflösen. Da A aber den höheren Score aufweist, wird der Start nur verschoben, wenn die Überlappung sehr kurz ist. Andernfalls wird B verworfen.

4. Im letzten Fall ist die Orientierung gegenläufig, wobei sich die 5'-Enden überlappen. Beide Startpunkte können verschoben werden, um die Überlappung aufzulösen.

$$^{3'}\!\longleftarrow^{5'} A$$
$$B \ ^{5'}\!\longrightarrow^{3'}$$

Der Start von B wird solange verschoben, bis die Region von B einen höheren Score als die von A aufweist oder die Überlappung aufgelöst ist. Wird B beim Verschieben der Startposition zu kurz, so wird es verworfen.

2.3.2 Das Zusatzmodul RBSfinder

(Suzek *et al.*, 2001)

Um die Vorhersage der Startpositionen zu verbessern kann Glimmer mit dem Zusatzmodul RBSfinder kombiniert werden. Der Algorithmus modifiziert bereits vorhergesagte Startpositionen und ist somit auf ein Genvorhersage-Tool angewiesen. Die derzeitige Implementation ist für die Ausgabe von Glimmer angepaßt. Die Modifikationen der vorhergesagten Startpositionen beruht auf der Suche nach Ribosom-Bindestellen (RBS). Diese enthält meist eine 6 BP lange SD-Sequenz (Shine-Dalgarno-Sequenz), die komplementär zum 3'-Ende der 16S rRNA des betrachteten Organismus ist.

RBSfinder wählt zunächst eine "Seed-Sequenz" der Länge L aus, mit deren Hilfe ein wahrscheinlichkeitstheoretisches Modell für die Ribosom-Bindestellen trainiert wird. R sei das reverse Komplement zum 3'-Terminus der 16S rRNA mit einer Länge von 15 BP. Die Sequenz R kann dem Algorithmus übergeben werden. Andernfalls wird sie nach der Tompa-Methode ermittelt

(Tompa, 1999). Dabei werden die Regionen upstream der Startcodons nach konservierten Motiven durchmustert, für die jeweils ein z-Score berechnet wird. Die Sequenz, die den größten z-Score aufweist, wird dann R zugeordnet.
Die Menge S enthält alle Infixe der Länge L, die in R enthalten sind. Es wird eine weitere Menge U zusammengestellt. Diese umfaßt alle Regionen der Länge 30 BP, die 5'-wärts der vorhergesagten Genstarts liegen. In der Menge U werden die Häufigkeiten der Elemente aus S festgestellt. Die Sequenz aus S mit der größten Häufigkeit in U wird als Seed-Sequenz ausgewählt.

Trainieren der Modelle

Beim Trainieren der Modelle wird angenommen, daß ein gewisser Anteil der Startpositionen von dem vorgeschalteten Programm korrekt vorhergesagt wird. Die Regionen -10 bis -40 BP upstream der potentiellen Genstarts werden nach konservierten Motiven durchmustert. Aus jeder dieser Sequenzen wird ein Motiv Seq_i ausgewählt, das den größten Ähnlichkeitswert zur Seed-Sequenz Seq_{seed} aufweist. Die Berechnung des Ähnlichkeitswertes erfolgt nach Gleichung (2.25).

$$S(\mathrm{Seq}_{seed}, \mathrm{Seq}_i) = \sum_{j=1}^{L} Bp\mathrm{sim}(\mathrm{Seq}_{seed}[j], \mathrm{Seq}_i[j]) \tag{2.25}$$

Seq_{seed}	Seed-Sequenz
Seq_i	potentielle RBS-Sequenz
$\mathrm{Seq}_i[j]$	Position j innerhalb der Sequenz Seq_i
i	Positionsindex im Rahmen -10 bis -40
L	Länge der Seed-Sequenz

mit

$$Bp\mathrm{sim}(\mathrm{Seq}_{seed}, \mathrm{Seq}_i) = \begin{cases} 3 & \text{falls } \mathrm{Seq}_{seed}[j] = \mathrm{Seq}_i[j] \text{ und } \mathrm{Seq}_{seed}[j] \ \epsilon \ \{\mathrm{G}, \mathrm{C}\} \\ 2 & \text{falls } \mathrm{Seq}_{seed}[j] = \mathrm{Seq}_i[j] \text{ und } \mathrm{Seq}_{seed}[j] \ \epsilon \ \{\mathrm{A}, \mathrm{T}\} \\ 1,5 & \text{falls } \mathrm{Seq}_{seed}[j] = \mathrm{A} \text{ und } \mathrm{Seq}_i[j] = \mathrm{G} \end{cases}$$

Die Funktion Bpsim bezieht sich auf die Anzahl an Wasserstoffbrücken, das heißt die Stärke der Bindungen, die zwischen der rRNA und der betrachteten Sequenz ausgebildet werden können. Die Sequenzen, mit der größten

Ähnlichkeit zur Seed-Sequenz, werden ausgewählt. Liegt der Wert über einem festgesetzten Schwellenwert, so wird die Sequenz einer Trainingsmenge T zugeordnet.
Alle Sequenzen, die in T enthalten sind, werden aligniert und es wird eine Wahrscheinlichkeitsverteilung berechnet nach Gleichung (2.26) und Gleichung (2.27).

$$P(b,k) = f(b,k)/N \tag{2.26}$$

mit	$b \in$ {A,T,G,C}
	$L \geq k \geq 1$
$P(b,k)$	Wahrscheinlichkeit des Vorkommens von Base b an Position k
$f(b,k)$	Häufigkeit des Vorkommens von Base b an Position k
N	Anzahl der Sequenzen in der Trainingsmenge T

$$P_{pos}(i) = f_{pos}(i)/N \tag{2.27}$$

mit	$W \geq i \geq L$
W	Größe des betrachteten Fensters
$P_{pos}(i)$	Wahrscheinlichkeit, daß eine Sequenz der Trainingsmenge i Nucleotide upstream eines Startcodons auftritt
$f_{pos}(i)$	Häufigkeit des Auftretens einer Sequenz der Trainingsmenge T, i Nucleotide upstream eines Startcodons

Mit den Wahrscheinlichkeiten wird nun für jede Sequenz Seq_i ein kombinierter Score C_i berechnet nach Gleichung (2.28).

$$C_i = P_{pos}(i) \times \text{Score}_i \tag{2.28}$$

$$\text{mit} \qquad \text{Score}_i = \prod_{k=1}^{L} P(\text{Seq}_i[k], k)$$

$$\text{Seq}_i[k] \in \{\text{A,T,G,C}\} \qquad \text{Base } b \text{ an Position } k \text{ in Seq}_i$$

Für jedes Gen wird die Sequenz Seq_{max} mit maximalem Score C als Kandidat für die Ribosom-Bindestelle ausgewählt. Liegt der Score über einem Schwellenwert C_0, so wird die Sequenz als Ribosom-Bindestelle behandelt. C_0 wird berechnet nach Gleichung (2.29).

$$C_0 = \frac{\text{Score}_{max} + \text{Score}_{min}}{2} \times min(P_{pos}) \tag{2.29}$$

$Score_{max}$	maximaler Score aus der Trainingsmenge
$Score_{min}$	minimaler Score aus der Trainingsmenge
$min(P_{pos})$	minimales P_{pos} aus Gleichung (2.27)

Nachdem putative Ribosom-Bindestellen für alle alternativen Startpositionen eines Genes gefunden sind, wird festgelegt, welcher Start am wahrscheinlichsten ist. Es werden zwei Kriterien einbezogen: zum einen C_i, der Score der einzelnen Ribosom-Bindestelle, zum anderen die Bewertung des Startcodons. Der Score muß dabei über dem Schwellenwert C_0 liegen. Das Triplett ATG wird den anderen Tripletts, die als Startcodon auftreten, vorgezogen, auch wenn der zugehörigen Ribosom-Bindestelle ein niedrigerer Score zugeordnet ist. Weiterhin wird das Triplett GTG dem Triplett TTG vorgezogen.

2.3.3 Anwendung von GLIMMER und RBSfinder

Kommandozeilen-Parameter

Als erstes erfolgt der Aufruf des Unterprogramms `long-orfs`, dem die Sequenz in FASTA-Format übergeben wird. Die Ausgabe entält eine Liste aller langen ORFs, die in der Sequenz enthalten sind. Als Parameter kann mit `-g` eine minimale ORF-Länge vorgegeben werden. Der Defaultwert ist 500. Durch Übergabe des Parameters `-l`, wird das Genom als nicht zirkulär betrachtet, was zur Auswirkung hat, daß keine vorhergesagten ORFs über das Ende der Sequenz hinausgehen können. Weiterhin kann angegeben werden, wie weit zwei Vorhersagen sich maximal überlappen dürfen. Die Angabe kann mit `-o` in BP übergeben werden oder mit `-p` prozentual. Standardmäßig sind Überlappungen bis 30 BP (bzw. 10%) erlaubt.
Dem zweiten Programm, `extract`, wird wiederum die Sequenz in FASTA-Format übergeben. Außerdem wird die Ausgabe des Programms `long-orfs` übergeben, die Start- und Endpositionen der ORFs enthält. `extract` gibt die Teilsequenzen zu den angegebenen Positionen aus der Gesamtsequenz aus. Die Übergabe des Parameters `-skip` bewirkt, daß die Sequenzen jeweils ohne Startcodon ausgegeben werden. Mit dem Parameter `-l` kann eine minimale Länge übergeben werden. ORFs, die diese Länge unterschreiten, werden nicht mit ausgegeben. Die drei Basen des Startcodons werden hier-

bei immer mitgezählt.
Das dritte Programm, `build-icm`, generiert ICMs auf Basis der Sequenzen, die von `extract` aus der Eingabesequenz extrahiert wurden.
Als letztes wird das Kernprogramm, `glimmer2`, aufgerufen. Dem Programm werden die Sequenz in FASTA-Format und die Markov-Modelle übergeben. Damit berechnet es für jeden ORF die Scorewerte und darauf basierend seine Vorhersage. Mit dem Parameter `-g` kann hier eine minimale ORF-Länge vorgegeben werden, die zugelassen wird. Mit `-i` kann ein File mit Koordinaten von Regionen übergeben werden, die ignoriert werden sollen. Wie schon bei dem Programm `extract`, kann auch `glimmer2` mit dem Schalter `-l` veranlaßt werden, das Genom als nicht zirkulär zu behandeln. Mit den Schaltern `-o` bzw. `-p` können analog zu `long-orfs` maximale Überlappung angegeben werden. Durch Übergabe des Parameters `+f` wird das jeweils erste Codon eines ORFs (das am weitesten 5'-wärts liegende) als Startcodon betrachtet. Mit dem Parameter `-s` kann die Sequenz einer Ribosom-Bindestelle übergeben werden, um das Auffinden von korrekten Startpositionen zu verbessern. Die Energie der Ribosom-Bindestelle kann ebenfalls einbezogen werden, um die Vorhersage der Startpositionen zu verbessern. Dazu muß der Parameter `-f` übergeben werden. Diese Funktion ist in der hier verwendeten Version noch nicht voll getestet. Weiterhin kann mit `-t` ein Schwellenwert angegeben werden, den die Scores der potentiellen Gene überschreiten müssen.

Manueller Aufruf der Unterprogramme

1. `long-orfs genome.fasta > genome.orfs`
2. `extract genome.fasta genome.orfs > genome.extract`
3. `build-icm < genome.extract > genome.model`
4. `glimmer2 genome.fasta genome.model > genome.out`
5. `perl rbs_finder.pl genome.fasta genome.outR genome.rbs`

   (`genome.outR` muss aus `genome.out` generiert werden.)

Beispiel für die Ausgabe von GLIMMER

Die Ausgabe von GLIMMER2.02 enthält jeweils eine ID und das Frame, in dem der ORF liegt, Start- und Endpositionen des potentiellen Genes und des ORFs, so wie deren Längen (`Lengths`). Außerdem sind Scores für das Gen, die Frames und ein unabhäniger Score enthalten.

		Orf	Gene		Lengths		Gene	-- Frame Scores -						Indep	
ID#	Fr	Start	Start	End	Orf	Gene	Score	F1	F2	F3	R1	R2	R3	Score	
	R1	303	303	196	108	108	0	_	99	_	0	_	_	0	0
	F3	225	228	320	96	93	0	_	99	0	_	_	_	0	0
	R3	529	511	395	135	117	0	_	99	_	_	_	0	0	0
	R1	846	837	706	141	132	0	_	99	_	0	_	_	0	0

Beispiel für die Ausgabe von RBSfinder

Die Ausgabe von RBSfinder enthält neben der alten und neuen Startposition, das zugörige Startcodon und das putative Motiv der Ribosom-Bindestelle (`Pattern`) mit Positionsangabe. Außerdem wird die Verschiebung des alten Startes unter `shift` sowie die Position des Stopcodons angegeben.

	NewStart	Stop	RibosomeBindingSite				OldStart	
GeneID	Position	Position	Pattern	Position	NewCodon	Shift	Codon	Position
1	197	2080	AAAAG	184	ATG	33	TTG	164
2	2924	2772	---	0	ATG	0	ATG	2924
3	2278	3099	ATGAG	2265	ATG	0	ATG	2278
4	3139	3375	TGGAG	3128	ATG	0	ATG	3139

2.4 ZCURVE1.0

(Guo *et al.*, 2000)

2.4.1 Strategie von ZCURVE

ZCurve verfolgt wie Glimmer einen rein intrinsischen Ansatz. Die Genvorhersage basiert auf der Z-Curve Repräsentation von DNA (Zhang und Zhang, 1991). Dabei wird die lokale Komposition der Nucleotidsequenz in einem mehrdimensionalen Raum dargestellt.
Zunächst wird eine Trainingsmenge für den Algorithmus zusammengestellt, die ermöglichen soll, daß ZCurve entscheiden kann, ob die Wahrscheinlichkeit, daß eine Region ein Protein codiert, signifikant ist. Hierbei werden zwei Fälle unterschieden. Hat das Genom einen G/C-Gehalt $< 56\%$, werden alle ORFs ≥ 500 BP ohne Überlappungen mit anderen ORFs aus der Sequenz extrahiert und als "Seed"-ORFs verwendet. Für Genome mit einem G/C-Gehalt $\geq 56\%$ wurde gezeigt, daß häufig nicht genügend ORFs dieser Länge ohne Überlappungen extrahiert werden können, um den ZCurve-Algorithmus zu trainieren. Daher wird hier die sogenannte "Methode der neun-dimensionalen Super-Sphäre" angewendet. Diese wird in Abschnitt 2.4.1, Erstellen der Trainingsmenge für Hoch-G/C-Genome, S. 50, näher beschrieben.
Der eigentliche Algorithmus beruht, wie oben erwähnt, auf der Z-Curve-Darstellung der DNA. Hierzu werden zunächst die Häufigkeiten der Basen A_i, C_i, G_i, T_i mit $i \ \epsilon \ \{1, 2, 3\}$ in Abhänigkeit von ihren Positionen im Triplett festgestellt. Dabei bezeichnen A_1, C_1, G_1, T_1 die Häufigkeiten der Basen an der ersten Position im Triplett, A_2, C_2, G_2, T_2 die an der zweiten und A_3, C_3, T_3, G_3 die an der dritten. Aus den Häufigkeiten wird nun nach Gleichung (2.30) ein Punkt P_i im dreidimensionalen Raum V_i gewonnen. x_i, y_i, z_i seien die Koordinaten von Punkt P_i.

$$\begin{cases} x_i &= (A_i + G_i) - (C_i + T_i) \\ y_i &= (A_i + C_i) - (G_i + T_i) \\ z_i &= (A_i + T_i) - (G_i + C_i), \end{cases} \tag{2.30}$$

mit

$$x_i, y_i, z_i \quad \epsilon \quad [-1, 1]$$

und

$$i \quad \epsilon \quad \{1, 2, 3\}$$

Analog werden die Häufigkeiten für Dinucleotide festgestellt. Dabei werden jeweils die Basen an den Positionen 1 und 2 im Triplett sowie diejenigen an den Positionen 2 und 3 zusammen betrachtet. Die ermittelten Häufigkeiten werden mit $f_{1,2}(AA), f_{1,2}(AC)...f_{1,2}(TT)$ und $f_{2,3}(AA), f_{2,3}(AC)...f_{2,3}(TT)$ bezeichnet. Die Z-Transformation erfolgt dann nach Gleichung (2.31), wobei x_j^X, y_j^X, z_j^X die Koordinaten der Punktes P_j^X im dreidimensionalen Raum V_j^X mit $X\epsilon\{A, C, G, T\}$ und $j = 1, 2$ bzw. $j = 2, 3$ sind.

$$\begin{cases} x_j^X & = (f_j(XA) + f_j(XG)) - ((f_j(XC) + f_j(XT)) \\ y_j^X & = (f_j(XA) + f_j(XC)) - ((f_j(XG) + f_j(XT)) \\ z_j^X & = (f_j(XA) + f_j(XT)) - ((f_j(XG) + f_j(XC)) \end{cases} \tag{2.31}$$

mit

$$X \quad \epsilon \quad \{A, T, C, G\}$$

und

$$j = [1, 2] \text{ oder } j = [2, 3]$$

Vereinigt man die Räume V_i und V_i^X, so erhält man einen 33-dimensionalen Raum V mit den Koordinaten u_k.

$$\begin{cases} u_1 = x_1, & u_2 = y_1, & u_3 = z_1 \\ u_4 = x_2, & u_5 = y_2, & u_6 = z_2 \\ u_7 = x_3, & u_8 = y_3, & u_9 = z_3 \end{cases} \tag{2.32}$$

$$\begin{cases} u_{10} = x_{1,2}^A, & u_{11} = y_{1,2}^A, & u_{12} = z_{1,2}^A \\ u_{13} = x_{1,2}^C, & u_{14} = y_{1,2}^C, & u_{15} = z_{1,2}^C \\ u_{16} = x_{1,2}^G, & u_{17} = y_{1,2}^G, & u_{18} = z_{1,2}^G \\ u_{19} = x_{1,2}^T, & u_{20} = y_{1,2}^T, & u_{21} = z_{1,2}^T \end{cases} \tag{2.33}$$

$$\begin{cases} u_{22} = x^A_{2,3}, & u_{23} = y^A_{2,3}, & u_{24} = z^A_{2,3} \\ u_{25} = x^C_{2,3}, & u_{26} = y^C_{2,3}, & u_{27} = z^C_{2,3} \\ u_{28} = x^G_{2,3}, & u_{29} = y^G_{2,3}, & u_{30} = z^G_{2,3} \\ u_{31} = x^T_{2,3}, & u_{32} = y^T_{2,3}, & u_{33} = z^T_{2,3} \end{cases} \tag{2.34}$$

Jeder ORF kann nun durch einen Punkt im Raum V dargestellt werden. Diese Punkte lassen sich mit Hilfe der Fisher Diskrimierung räumlich in verschiedene Gruppen einteilen. Die Diskriminierung entspricht dabei einer Super-Ebene, die durch einen Vektor c mit 33 Komponenten $c_1, c_2, ...c_{33}$ beschrieben wird. Basierend auf der Trainingsmenge wird ein Vektor c_0 als Schwellenwert definiert.

Durch positive und negative Beispiele lernt das System zu entscheiden, ob ein Punkt zu der Menge der ORFs gehört, die wahrscheinlich für ein Protein codierenden oder zu der Menge der ORFs, die wahrscheinlich nicht für ein Protein codieren. Die Positiven entsprechen dabei der Menge der Seed-ORFs, die Menge der Negativen leiten die Autoren aus den Seed-ORFs ab. Der Anteil nicht-codierender DNA in mikrobiellen Genomen liegt oft nur bei 10-20% und ist somit zu gering, um eine signifikante Trainigsmenge zusammenzustellen. Daher ist es nötig, solche Negativ-Beispiele künstlich zu generieren. Die Basen der Seed-ORFs werden dazu mit Hilfe eines Monte Carlo Algorithmus mindestens 20.000× verschoben. Im Unterschied zu nicht-codierenden Sequenzen sind codierende durch reguläre Strukturen ausgezeichnet. Diese werden durch das zufällige Mischen zerstört, und man erhält so eine Zufallssequenz. Das Komplement der resultierenden Sequenz wird dann von den Autoren als Negativ-Beispiel verwendet.

Die Entscheidung codierend/nicht-codierend wird dann für jeden ORF mit Hilfe der Relation $c * u > c_0 / c * u < c_0$, mit $c = (c_1, c_2, ...c_{33})^T$ und $u = (u_1, u_2, ...u_{33})^T$ getroffen (T zeigt die Transposition der Matrix an). u wird dabei berechnet nach den Gleichungen (2.33) und (2.34). c und c_0 (siehe oben) werden mit Hilfe der positiven und negativen Beispiele berechnet. Anders formuliert wird die Ungleichung als $Z(u) > 0 / Z(u) < 0$, mit $Z(u) = c * u - c_0$ ausgedrückt. Dabei entspricht $Z(u)$ einem Score (Z-Score oder Z-Index), der für jeden ORF berechnet wird und definiert, ob ein ORF in der Menge der Positiven bzw. der Negativen liegt. Als putative Gene wer-

den zunächst alle ORFs mit einem Z-Score > 0 behalten. ORFs mit einem Z-Score < 0 werden verworfen.

Erstellen der Trainingsmenge für Hoch-G/C-Genome

Genome mit einem G/C-Gehalt $\geq 56\%$ weisen häufig nur sehr wenige ORFs auf, die für die Menge der Seed-ORFs geeignet sind. Um als Seed-ORF deklariert zu werden, muß ein ORF eine Mindestlänge von 500 BP haben und darf von keinem anderen ORF überlappt werden. Beim Vergleich von Genomen mit unterschiedlichem G/C-Gehalt ist erkennbar, daß bei einem G/C-Gehalt von $\geq 56\%$ ein signifikanter Anstieg des Verhältnis, ORFs mit einer Länge ≥ 500 BP zu ORFs mit einer Länge ≥ 500 BP ohne Überlappungen, erfolgt. Daher wählten die Autoren die "Methode der neun-dimensionalen Super-Sphäre", um die Menge der Seed-ORFs für Genome mit einem G/C-Gehalt $\geq 56\%$ zusammenzustellen. Dazu muß zunächst der Ursprung O des neun-dimensionalen Raumes berechnet werden. Dies erfolgte mit Hilfe von annotierten Genen ≥ 500 BP der Organismen *Caulobacter crescentus, Deinococcus radiodurans* R1 *chromosome 1, Halobacterium sp.* NRC-1 und *P.aeruginosa* PA01[7], mit denen die neun Parameter nach Gleichung (2.32) berechnet wurden. Die Mittelwerte dieser Auswahl werden im Algorithmus als Konstanten vorgegeben. Die neun Parameter aus Gleichung (2.32) werden für alle ORFs berechnet. Liegt der so berechnete Punkt eines ORFs innerhalb eines Radius r um den Ursprung O, so wird dieser als Seed-ORF verwendet. Der Radius r wird stufenweise vergrößert, bis mindestens 250 Seed-ORFs vorliegen.

Behandlung von Überlappungen

ZCurve behandelt Überlappungen, die mindestens 1/5 der Länge eines der beteiligen ORFs umfassen. Weist der längere der ORFs einen höheren Z-Score auf, so wird er als codierend betrachtet. Der kürzere ORF wird verworfen. Hat dagegen der kürzere ORF einen höheren Z-Score, so werden beide ORFs als codierend eingestuft. Ist der Score des kürzeren ORFs wesentlich höher, so der längere ORF verworfen und nur der kürzere wird als

[7] Annotation der GeneBank-Datenbank Release 129.0. (GenBank/EMBL/DDBJ)

potentielles Gen bewertet.
Treten zwei Überlappungen zusammen auf, werden die Z-Scores aller drei ORFs verglichen. Wenn der Z-Score des zweiten ORFs kleiner ist als die des ersten und dritten ORFs, wird der mittlere ORF verworfen. Ist dies nicht der Fall, so werden beide Überlappungen einzeln behandelt (wie oben beschrieben).

Vorhersage der Startpositionen

Die positionsspezifische Basenverteilung in der Startregion ist im Hinblick auf die ZCurve-Darstellung, deutlich verschieden von der Verteilung in anderen codierenden Regionen. Dies machen sich die Autoren bei der Vorhersage der potentiellen Genstarts zu nutzen. Zunächst erfolgt ein Schritt zur Filterung der Seed-ORFs mit den wahrscheinlichsten Startvorhersagen. Die Koordinaten der Punkte, die für Startregionen berechnet werden, können durch eine Gerade approximiert werden, die durch 2.35 definiert ist.

$$\begin{cases} x_n = v_1 \times n + b_x, & n \ \epsilon \ [-13, -7] \\ y_n = v_2 \times n + b_y, & n \ \epsilon \ [-13, -7] \\ z_n = v_3 \times n + b_z, & n \ \epsilon \ [-36, +20] \end{cases} \tag{2.35}$$

n	Position der Base, das erste Nucleotid des Startcondons ist an Position 0
v_1, v_2, v_3	Steigungen der Geraden
b_x, b_y, b_z	Zu v_1, v_2, v_3 gehörende Achsenabschnitte

Dabei werden die Parameter v_1 und v_2 in der Region -13 bis -7 relativ zum Genstart (angepaßt für verschiedene Genome) berechnet und v_3 in der Region -36 bis +20. Ein vierter Parameter, v_4, wird nach (2.36) für die Regionen -90 bis 0 und 1 bis 89 berechnet.

$$v_4 = -Z_{up} \times Z_{down}, \tag{2.36}$$

Z_{up}	Z-Score in der Region -90 bis -1
Z_{down}	Z-Score in der Region 0 bis 89

Da die Z-Scores Z_{up} und Z_{down} nicht für ORFs, sondern nur für Fragmente berechnet werden, sind neue Mengen von Positiven und Negativen Beispielen

nötig. Die Positiven entsprechen hierbei jenen Startcodons der Seed-ORFs, für die die höchsten Z-Scores berechnet werden. Die Negativen entsprechen allen potentiellen Starts up- und downstream der Seed-Starts, die nicht als Genstart angesehenen werden, für die also ein niedriger Z-Score berechnet wird. Die vier Parameter v_1, v_2, v_3, v_4 spannen einen vierdimensionalen Raum auf. Die Fisher-Diskriminierung wird nun wie oben beschrieben für alle Positiven und Negativen berechnet, dabei werden nur die ORFs mit den maximalen Werten beibehalten. Diese werden als wahre Genstarts angenommen, die anderen werden verworfen.
Nachdem eine Menge von ORFs gefunden wurde, bei denen es sich wahrscheinlich um Gene handelt, werden die vier Parameter für alle putativen Genstarts berechnet. Außerdem werden noch zwei weitere Parameter v_5 und v_6 berechnet. v_5 bezieht das Startcodon selbst mit in die Betrachtung ein. Für putative Genstarts mit einem ATG als Startcodon ist $v_5 = 0,78$, mit GTG ist $v_5 = 0,14$ und für TTG $0,07$. Für alle anderen Codons, die als Genstart auftreten, ist $v_5 = 0,01$. Der Parameter v_6 bezieht sich auf die Längen der ORFs und ist definiert als $v_6 = e^{l/l_0}$, wobei l der Abstand zwischen dem betrachteten Start und der am weitestens upstream liegenden potentiellen Startposition, die dem hier zu Grunde gelegten Seed-ORFs, zugeordnet ist. l_0 bezeichnet die Länge des längsten ORFs.
Mit Hilfe der sechs Parameter können nun erneut über den Algorithmus zur Fisher Diskriminierung (wie oben beschrieben) die Koeffizienten und ein Schwellenwert berechnet werden. Ebenfalls analog werden für die putativen Genstarts Scores berechnet. Als wahrer Start wird jeweils derjenige angenommen, der den höchsten Score aufweist.

2.4.2 Anwendung von ZCURVE

Aufruf in der DOS-Box

Das Tool ZCurve ist nur als ausführbares Programm für win32-Systeme verfügbar. Der Source-Code in der Sprache `FORTRAN` wurde nicht veröffentlicht. Der Programmablauf ist interaktiv gestaltet. Beim Aufruf wird eine DOS-Box geöffnet, in der Anweisungen gegeben werden, die notwendigen Pfade und Parameter zu setzen.

Beispiel für die Ausgabe von ZCURVE

Die Ausgabe von ZCurve beinhaltet für jeden putativen ORF eine ID (`No`), die vorhergesagten Start- und Stopkoordinaten, den Strang, die Länge und einen Score (`Z_Index`).

No	Start	Stop	Strand	Length	Z_Index
1	197	2083	1	1887	0.26675
2	2278	3102	1	825	0.24041
3	3139	3378	1	240	0.12293
4	3497	3982	1	486	0.35300

2.5 Das Programmpaket YACOP - Yet Another Combination Of Predictions

Ziel der Arbeit ist es, durch geschickte Kombination vorhandener Tools die Qualität der Vorhersagen codierenden Regionen (Spezifität und Sensitivität) zu verbessern. Insbesondere soll die Performanz der Vorhersage der Startpositionen erhöht werden. Zu diesem Zweck wurde das Programmpaket YACOP entwickelt und implementiert. im Folgenden werden Aufbau und Anwendung des Programms beschrieben. Für weitere Informationen zur Anwendung und Anpassung von YACOP für eigene Zwecke ist dem Programm eine Readme-Datei beigelegt[8].

2.5.1 Aufbau und Anwendung von Yacop

Das Programmpaket YACOP bietet die Möglichkeit, die Vorhersagen der vier vorgestellten Programme, CRITICA, GLIMMER, ZCURVE, und ORPHEUS automatisiert zu kombinieren. Die Aufrufe von drei der Programme sind im Paket integriert, ZCURVE, kann nicht automatisiert aufgerufen werden, da das Tool ausschließlich als Executable, das nur auf win32-Systemen lauffähig ist, veröffentlicht wurde. Der Aufruf von ZCURVE muß also manuell erfolgen und die Ausgabe beim Start von YACOP übergeben werden.
Aus den einzelnen Vorhersagen der Programme wird von YACOP ein Gesamtergebnis zusammengestellt. Dabei werden verschiedene Modi angeboten, nach denen die Vorhersagen der Programme kombiniert werden können. Um ein File im Multiple-Fasta-Format einzulesen, kann das Zusatzskript `combinate-multi` verwendet werden. Die einzelnen Sequenzen in der Eingabedatei werden temporär zwischengespeichert und dem eigentlichen Skript `combinate` übergeben. Dabei ist zu beachten, daß die Mindestlänge der Sequenzen bei 50 kB liegt. Weiter ist zu beachtet, daß bislang keine Möglichkeit geboten werden kann, ZCURVE beim Aufruf mit Multiple-Fasta zu integrieren.
Das Skript `combinate` ruft CRITICA und ORPHEUS standardmäßig als Subprozesse auf. Die Aufrufe können allerdings auch direkt erfolgen, falls kein

[8]Download unter http://www.g2l.bio.uni-goettingen.de/software/yacop.tar.gz

Batch-System zur Verfügung steht (im Quellcode auskommentiert). Das Programm GLIMMER wird nicht als Subprozess aufgerufen, da die Laufzeit bei weitem unter der von CRITICA und ORPHEUS liegt. Wie bereits erwähnt kann ZCURVE nicht automatisiert aufgerufen werden, so daß die Ausgabe manuell übergeben werden muß.

Beim Aufruf von `combinate` muß die Sequenz in Fasta-Format, ein .ini-File, ein Präfix und für jedes Programm, das einbezogen werden soll, ein Parameter übergeben werden. Das Präfix muß dabei dem Regulären Ausdruck `/[A-Z]{2,3}/` entsprechen. Das heißt es muß aus zwei oder drei Großbuchstaben bestehen. Das .ini-File enthält jeweils die Parameter, mit denen die anderen Programme aufgerufen werden sollen (siehe jeweils Abschnitt "Kommandozeilen-Parameter") und eine Mindestgenlänge, die man für YACOP setzen kann. Dabei ist zu beachten, daß diese Mindestlänge nicht auf die Vorhersagen von CRITICA angewendet wird. Außerdem enthält das .ini-File die Verzeichnisse, in denen die verwendeten Programme liegen. Um beispielsweise das Programm CRITICA aufzurufen, muß `combinante` der Parameter `-c` übergeben werden, analog wird für GLIMMER `-gr` und für ORPHEUS `-o` übergeben. Soll eine Vorhersage von ZCURVE einbezogen werden, so muß das unter Windows generierte Ausgabe-File mit dem Parameter `-z` übergeben werden. Für Genome, die bereits annotiert sind, besteht die Möglichkeit mit dem Parameter `-gb` ein GeneBank-File zu übergeben. Die GeneBank-Annotation wird dann mit den Vorhersagen der anderen Programme verglichen.

Nachdem alle Programme ihre Vorhersage abgeschlossen haben, werden die jeweiligen Ausgaben dem Unterprogramm `meta_pred.pl` übergeben. Je nach Modus, mit dem `meta_pred.pl` aufgerufen wird, erfolgt die Kombination der Vorhersagen. Mit dem Modus `crit_orp_gl` beispielsweise wird die Vorhersage von CRITICA komplett einbezogen und zudem die Schnittmenge der Vorhersagen von ORPHEUS und GLIMMER. Dabei werden jeweils die vorhergesagten Startpositionen von CRITICA bzw. ORPHEUS angegeben. Der Standard Modus für YACOP (mit ZCURVE) ist `crit_zc_gl`. Bei diesem werden die Vorhersagen von CRITICA mit der Schnittmenge von ZCURVE und GLIMMER kombiniert. Dabei werden jeweils die Startpositionen von CRITICA bevorzugt. Bei Vorhersagen, die nur von ZCURVE und GLIMMER gemacht

werden, wird die vorhergesagte Startposition von ZCurve angegeben. Die Begründung für diese Kombination wird im Ergebnisteil gegeben.

Yacop erzeugt ein Verzeichnis mit der Startzeit in `long`-Format als Namen. Dieses wird unter `data/` im Homeverzeichnis des Aufrufers abgelegt und dient zum Speichern aller Ergebnisse, die im Programmverlauf generiert werden. Die eigentlichen Ausgabe-Files des Paketes sind `PREFIX.tbl`, `PREFIX.fasta` und `PREFIX.sum_out`. `PREFIX.sum_out` enthält dabei die Vorhersagen aller Programme. `PREFIX.tbl` und `PREFIX.fasta` enthalten nur die Vorhersagen, die entsprechend dem Modus und der Mindestlänge einbezogen werden.
Die Objekt-orientierte Struktur des Programms erlaubt die Integration weiterer Programme. Dazu ist es nötig, jeweils ein Objekt für den Anstoß und ein Objekt zum Parsen der Ausgabe analog zu den bereits verwendeten Parsern zu implementieren. Zudem müssen `combinate` und `meta_pred.pl` für die Aufrufe angepaßt werden.

Beispiel für die Ausgabe-Formate von Yacop

Im Ausgabefile mit dem Suffix `.sum_out` sind die gesamten Vorhersagen der verwendeten Programme zusammengefaßt. Im folgenden Beispiel wird ein bereits annotiertes Genom (*S.typhimurium.*) verwendet, wobei zu jeder Vorhersage die tatsächliche Annotation nach GeneBank angefügt wird (hier aus Platzmangel nur teilweise notiert). Außerdem werden jeweils Leserahmen und Strang angegeben.
Die Aufführung der Vorhersagen ist jeweils nach den vorhergesagten Positionen der Stopcodons angeordnet. In den Spalten mit den Programmnamen sind die von den entsprechenden Programmen vorhergesagten Startpositionen eingetragen. Für die Vorhersagen von Critica wird zusätzlich zu den Startkoordinaten der berechnete P-Wert für die Güte der Vorhersage mitgeliefert. Die Berechnung des P-Wertes erfolgt nach Gleichung (2.9), S.19. Weiterhin wird für ZCurve jeweils der Z-Score oder Z-Index des vorhergesagten ORFs angegeben (Berechnung siehe Abschnitt 2.4.1, S. 47). Die Vorhersage von Glimmer wird in Kombination mit den von RBSfinder kor-

rigierten Startpositionen angegeben, sofern RBSfinder verwendet wurde. Die Verschiebung des Startes in 3'-Richtung wird dabei jeweils in der Spalte RBS in Klammern angegeben.

```
stop comp Glimmer      Critica     Orpheus    ZCURVE    genebank RBS-Site annotation
-------------------------------------------------------------------------------------
 253 F     - (-)     -      -        -      -     -      190              thr operon...
2797 F   337 (42)   337 1.72e-286   295    337 0.23075    337   325..330 aspartokinI...
3728 F  2801 (0)   2801 1.21e-104  2801   2801 0.22190   2801 2789..2794 homoserine ...
5018 F  3734 (0)   3734 3.93e-170  3734   3734 0.25223   3734 3722..3727 threonine s...
5116 R  5887 (0)   5887 2.70e-77   5887   5887 0.27152   5887            putative cy...
5968 R  7396 (27)  7396 9.97e-90   7423   7396 0.17603   7396 5893..5989 putative AG...
```

Die beiden weiteren Ausgabe-Files enthalten nur die ORFs, die entsprechend dem eingestellten Modus und der vom Benutzer spezifizierten Mindestlänge als putative Gene gewertet werden. Dabei ist zu beachten, daß die Mindestlänge nicht für die Vorhersagen von CRITICA gilt. Da die Vorhersage von CRITICA wie im Ergebnisteil gezeigt auch im Bereich < 150 BP nur einen geringen Anteil False-Positiver enthält, akzeptiert YACOP hier alle ORFs.
Das Ausgabe-File mit dem Suffix `.tbl` enthält jeweils eine Genvorhersage mit entsprechender ID (Identifikator) pro Zeile. Die ID ist nach dem Muster des Regulären Ausdrucks `/^R([A-Z]{2,3})\d{5,6}/` aufgebaut, wobei nach dem `R` das übergebene Präfix und dann eine fortlaufende sechsstellige Nummer beginnend mit 000001 folgt. Weiterhin sind das jeweilige Contig mit Startposition und Endposition, getrennt durch Unterstriche, angegeben.

```
RBU000001 AP000398_Buchnera_sp._APS_complete_genome._197_2080
RBU000002 AP000398_Buchnera_sp._APS_complete_genome._2278_3099
RBU000003 AP000398_Buchnera_sp._APS_complete_genome._3139_3375
RBU000004 AP000398_Buchnera_sp._APS_complete_genome._3497_3979
RBU000005 AP000398_Buchnera_sp._APS_complete_genome._3982_4512
```

In dem File mit dem Suffix `.fasta` ist die ID entsprechend dem `.tbl`-File, gefolgt von der Aminosäuresequenz, aufgeführt.

```
>RBU000001
MFNLRNFDVIVVGAGHAGTEAAMASSRMGCKTLLLTQKISDLGALSCNPAIGGIGKSHLV
KEIDALGGMMAKAIDYSGIQFRILNSSKGPAVRSTRAQADKILYHETVKKILKKQNNLLI
LEAEVKDLIFKNYSVVGVLTQNEINFYSRSVVLAAGTFLGGKIHIGLKSYSAGRIGDKSA
```

```
IDLSVRLRELSLRVNRLKTGTPPRIDINTVNFNNLLIQNSDTPVPVFSFMGNVSHHPKQI
PCYLTHTNEKTHEIIRKNLDKSPIYTGFLKGLGPRYCPSIEDKIVRFPDRKSHQVFLEPE
GLSSIKVYPNGISTSLPIEVQEQIVASIKGLEKSKIIRPGYAIEYDFFDPKDLNLTLESK
LIKGLFFAGQINGTTGYEEAASQGLLAGLNAALSSKNTEGWFPRRDQAYLGVLIDDLTTQ
GTEEPYRMFTSRAEYRLSLREDNADLRLTEIGRKLGLVNDSRWIRYNQKVLNIQTEMNRL
KKNKISPISPDADILKKLYNINLIKEISMSELLKRPQIRYQDLQSLESFRTGIVDLEAIG
QIENEIKYAGYIKRQSEEIERHLKNENTFLSSIYDYNKIRGLSSEVVKKLNDYKPISIGQ
ASRISGITPAAISILLIHLKKESYKHTL
>RBU000002
MILEKISDPQKYISHHLSHLQIDLRSFKIIQPGALSSDYWTVNVDSMFFSLVLGSFFLSI
FYMVGKKITQGIPGKLQTAIELIFEFVNLNVKSMYQGKNALIAPLSLTVFIWVFLMNLMD
LVPIDFFPFISEKVFELPAMRIVPSADINITLSMSLGVFFLILFYTVKIKGYVGFLKELI
LQPFNHPVFSIFNFILEFVSLVSKPISLGLRLFGNMYAGEMIFILIAGLLPWWTQCFLNV
PWAIFHILIISLQAFIFMVLTIVYLSMASQSHKD
>RBU000003
MENLNVDMLYIAVAIMVGLASIGAAIGIGILGGKFLEGAARQPDLVPLLRTQFFVVMGLV
DAIPMIAVGLGLYMLFAIS
```

2.5.2 Technische Voraussetzungen

Im Folgenden werden die Systemanforderungen beschrieben, die zur Nutzung von YACOP erforderlich sind. Für nähere Informationen wird hier auf die Readme-Datei verwiesen.

YACOP wurde unter RedHat Linux entwickelt und getestet. Generell ist das Programm unter alle UNIX-Derivaten lauffähig, die alle weiteren Anforderungen erfüllen. Zum Ausführen des Programms, muß auf dem System die Programmiersprache Perl verfügbar sein. Der Implementierung wurde Perl5.6.0[9] zu Grunde gelegt. Die Perl-Libraries müssen unter dem Pfad `/usr/bin/perl` liegen. Andernfalls ist eine Anpassung von YACOP sowie der Tools CRITICA und ORPHEUS nötig. Weiterhin erfordert YACOP die Bereitstellung von BioPerl[10]. Getestet wurde das Programmpaket für BioPerl1.3. Das Skript `combinate` für die Aufrufe der verwendeten Vorhersagetools wurde für Multiprozessor-Maschinen entwickelt und nutzt die Umgebung PBS (Portable Batch System), die für Linux-Cluster optimiert ist. Mit PBS lassen sich die Systemressourcen effektiv ausnutzen. Aufrufe von Programmen werden dazu in eine Warteschlange gestellt und erst ausgeführt, wenn aus-

[9] http://www.perl.com
[10] http://bioperl.org/

reichend Kapazitäten gegeben sind. Dies ist vor allem beim Ausführen von Programmen mit hohen Ansprüchen an die Systemleistung oder sehr langer Laufzeit von Vorteil. Der Aufruf eines Programms, welches das Batch-System nutzen soll, erfolgt mit dem Befehl `qsub`. Im Programmpaket YACOP werden ORPHEUS und CRITICA als Batch-Prozesse gestartet. GLIMMER beansprucht bei der Ausführung wesentlich weniger Systemressourcen und kann daher direkt gestartet werden. YACOP kann ebenfalls auf Singleprozessor-Rechnern verwendet werden. Dazu muß das Skript `combinate` angepaßt werden, wie in der Readme-Datei beschrieben. Wie bereits erwähnt kann ZCURVE nur auf Windows-Maschinen verwendet werden. Daher muß der Aufruf separat erfolgen und die Ausgabe anschließend manuell beim Aufruf von YACOP übergeben werden.

2.6 Annotierte Genome zur Überprüfung der Qualität der Vorhersagen

In der Arbeit wurde die Qualität der Vorhersagen der Tools sowie deren Kombination überprüft, indem für bereits annotierte Genome Vorhersagen gemacht wurden, die anschließend mit der Annotation aus der aktuellen GeneBank-Datenbank verglichen wurden. Es muß darauf hingewiesen werden, daß die Annotation, auf die sich alle in Kapitel 3 vorgetragenen Ergebnisse stützen, zum Teil fehlerbehaftet sein kann. Das heißt beispielsweise, daß in der Annotation angegebene Startpositionen inkorrekt sein können und daß Gene fehlen können. Außerdem können Gene in der Annotation eingetragen sein, die möglicherweise ausschließlich auf der Vorhersage eines Tools beruhen, die aber nicht durch experimentelle Belege verifiziert sind und die nicht für ein Protein codieren (Pseudogene). Es ist daher für die Bewertung der Vorhersagen besonders wichtig zu wissen, mit welchem Tool die Genome ursprünglich annotiert wurden. Basiert die Annotation eines Genoms beispielsweise auf einer Vorhersage mit CRITICA, wird möglicherweise CRITICA im Vergleich die besten Vorhersagen erzeugen. In Tabelle 2.4, sind die Genome aufgeführt und es ist angegeben mit welcher Genvorhersagemethode die Annotation erstellt wurde, auf der die GeneBank-Annotation basiert.

Es wurden geeignete Genome ausgewählt, welche die Bandbreite möglicher G/C-Gehalte abdecken und mit keinem der hier verwendeten Tools annotiert wurden. Dadurch sollten sich Fehler in der Annotation auf die Bewertung aller Vorhersagen gleichmäßig auswirken und die Tendenzen nicht beeinflussen.

Tabelle 2.4: Tools, die bei der Annotation der betrachteten Genome verwendet wurden.

Buchnera sp. APS	Es wurde das Programm GeneHacker (basiert auf Hidden Markov Modellen) verwendet. Die Ergebnisse wurden mit BLAST gegen eine NCBI-Datenbank abgeglichen. Der Isoelektrische Punkt der vorhergesagten Proteine wurde zur Überprüfung mit dem Programm ISOELECTRIC bestimmt.
Pub.: Shigenobu,S., Watanabe,H., Hattori,M., Sakaki,Y., Ishikawa,H. 2000. Genome Sequence of endocellular bacteria symbiont of aphids *Buchnera sp.* APS. *Nature*, **470:**81-86.	
Clostridium acetobutylicum ATCC842	Zur Identifizierung der ORFs wurde uniorf[11] verwendet. Um als protein-codierend akzeptiert zu werden, mußte den ORFs ein significanter BLAST2-Score zugeordnet sein, ihre Dicodon-Verteilung mußte der von *C. acetobutylicum* entsprechen und sie mußten eine Länge von mind. 400BP aufweisen. Die Vorhersagen wurden durch weitere Datenbank-Abgleiche mit PSI-BLAST, SMART und IMPALA überprüft.
Pub.: Nolling J. 2001. Genome Sequence and Comparative Analysis of the Solvent-Producing Bacterium *Clostidium acetobutylicum. J. Bacteriol* **183:**4823-4838.	
Lactococcus lactis subsp. lactis IL1403	Zur Genvorhersage wurde GeneMark verwendet. Die Annotation erfolgte mit BLAST.
Pub.: Bolotin A. *et al.* 2001. The complete genome sequence of the lactic acid bacterium *Lactococcus lactis. Genome Res.* **11:**731-751.	
Heliobacter pylori 26695	Es wurde die Programme GeneMark und GeneSmith verwendet. GeneMark wurde mit einer Auswahl von ORFs > 600 BP trainiert.
Pub.: Tomb J-F. *et al.* 1997. The complete genome sequence of the gastric pathogen *Helicobacter pylori. Nature*, **388:**539-547.	

Fortsetzung Tabelle 2.4

Bacillus subtilis	Es wurde GeneMark verwendet und gezielt nach sd-Sequenzen gesucht. Zusätzlich wurde mit BLAST2X Datenbank-Abgleiche durchgeführt.
Pub.: Kunst F. *et al.*. 1997. The complete genome sequence of the Gram-positive bacterium *Bacillus subtilis*. *Nature*, **390** (6657):249-256. Medigue C., Rose M., Viari A., Danchin A. 1997. Detecting and Analysing DNA Sequencing Errors: Toward a Higher Quality of the *Bacillus subtilis* Genome Sequence. *Nature*, **390** (6657):249-256.	
Escherischia coli K-12	Es wurde GeneMark verwendet. Zusätzlich wurden mit BLASTN und BLASTX Datenbank-Abgleiche auf SwissProt Protein-Datenbanken durchgeführt.
Pub.: Blattner F. R. 1997. Complete genome sequence of *Escherischia coli* K-12. *Science*, **277** (5331): 1453-1474.	
Salmonella typhimurium LT2	Es wurden GeneMark und Glimmer verwendet. Zusätzlich wurde PSORT (Protein Localisation Prediction Software eingesetzt. Die vorhergesagten Proteine wurden gegen die Proteinfamilien-Datenbank Pfam5.5 und zur Auffindung orthologer Gene gegen die COG-Datenbank abgeglichen.
Pub.: McClelland M. *et al.* 2001. Complete genome sequence of *Salmonella enterica* serovar Typhimurium LT2. *Nature*, **413:**852-856.	
Mycobacterium tuberculosis	Es wurden tRNAScanSE und TB-parse (basiert auf Hidden Markov Modellen) verwendet. Mit BLASTN und BLASTX wurden Datenbank-Abgleiche auf den folgenden Datenbanken durchgeführt: EMBL, TREMBL, SwissProt, DOTTER.
Pub.: Cole S. T, *et al.* 1998. Deciphering biology of *Mycobakterium tuberculosis* from the complete genome sequence. *Nature*,**393** (6685):537-544.	

Kapitel 3

Ergebnisse

3.1 Allgemeine Erläuterung der Darstellung

im Folgenden sollen die Vorhersagen der verwendeten Tools dargestellt und im Hinblick auf Sensitivität (SEN), Spezifität (SPEZ) und korrekte Vorhersage der Startposition (SC) verglichen werden. Aus der Güte und Verläßlichkeit der Vorhersagen der einzelnen Programme ergibt sich deren Kombination zur Erhöhung der Performanz der Gesamtvorhersage. Dabei soll die Anzahl der True-Positiven (TP) möglichst erhöht werden, wobei die Anzahl False-Positiver (FP) minimiert werden soll. Außerdem soll die korrekte Vorhersage der Startpositionen beobachtet und verbessert werden. Es wird darauf hingewiesen, daß sich diese Bewertung der Vorhersagequalität auf die bestehende Annotation stützt, die durchaus fehlerbehaftet sein kann. Dies gilt insbesondere für die annotierten Genstarts. In Abschnitt 2.6 wurde bereits näher auf die Problematik eingegangen, eine Menge geeigneter Genome zur Bewertung der Tools zusammenzustellen.
In diesem Kapitel wird zunächst auf die oben genannten Bewertungskriterien eingegangen. Dann werden die Vorhersagen der vorgestellten Tools Critica, Glimmer, ZCurve und Orpheus vergleichend dargestellt. Dabei werden zwei verschiedenen Versionen von Glimmer (Version 2.02 und die aktuelle Version 2.10) einbezogen. Schließlich werden verschiedene Möglichkeiten zur bool'schen Verknüpfung der Vorhersagen der Tools vorgestellt und verglichen.

Bei den Untersuchungen zeigte sich, daß die Vorhersagen sehr kurzer Gene meist einen wesentlich höheren Anteil an FP enthielten. Um eine optimale Kombination der Tools zur Vorhersage zu finden, wurde daher zum einen die Performanz der Tools mit einer Mindstlänge von 150 BP bewertet und zum anderen ohne Mindestlänge. Die Vorhersagen von ORPHEUS werden davon im Ergebnisteil ausgenommen. Sie werden nur mit 150 BP angegeben, da ORPHEUS im Bereich < 150 BP bei den gegebenen Parametern eine sehr schlechte Vorhersage lieferte[1]. Im Bereich ≥ 150 BP war die Performanz der Vorhersagen mit der anderer Programme vergleichbar. Die Zahl falsch-positiver Vorhersagen im Bereich < 150 BP stieg unverhältnismäßig stark an, während nur wenige zusätzliche Gene gefunden wurden. Beispielsweise wurden für *B. subtilis* insgesamt 17240 ORFs von ORPHEUS vorhergesagt, wenn es keine Längenbeschränkung gab. Davon stimmten 3963 (96,7%) mit Genen aus der Annotation überein. Das heißt, die Vorhersage enthielt 13277 FP. Wurde die Mindestlänge auf 150 BP gesetzt, so wurden 3932 (95,9%) Gene aus der Annotation vorhergesagt, dabei beinhaltete die Vorhersage aber nur 745 falsch-positive Vorhersagen (Tabelle 3.1, S. 71). Es wurden 12522 False-Positive weniger vorhergesagt als ohne Mindestlänge.
Desweiteren wird darauf hingewiesen, daß die Differenz der Werte bei den prozentualen Angaben sowie bei den Werte SPEZ, SEN und SC zum Teil gering scheint. Es ist jedoch zu beachten, daß es sich dabei um relativ große absolute Werte handeln kann. Schon eine Verbesserung um wenige Prozent bei der Vorhersage kann eine erhebliche Verringerung des Aufwandes der Annotatoren bedeuten, da Fehler meist manuell korrigiert werden müssen. Beispielsweise wurde von CRITICA für *E. coli*, ohne Mindestlänge ein Anteil von 0,72 korrekt vorhergesagter Startpositionen ermittelt (Tabelle 3.4, S. 77). Der SC-Wert von GLIMMER2.02 betrug unter den gleichen Bedingungen 0,70. Die Differenz von 0,02 bedeutet in diesem Fall, daß knapp 100 Genstarts weniger von GLIMMER korrekt vorhergesagt wurden als von CRITICA.

[1]Bei der Anwendung von ORPHEUS ist standardmäßig eine Mindestlänge von 240 BP gesetzt, diese kann mit Hilfe eines Parameters (Abschnitt 2.2.2, S. 30) variiert werden.

3.2 Berechnung von Spezifität, Sensitivität und der Start-Correctness

Um die Vorhersagen der einzelnen Tools darzustellen und zu vergleichen, werden die Parameter Sensitivität (SEN) und Spezifität (SPEZ) eingeführt. Diese können aus TP (True-Positive), FP (False-Positive) und FN (False-Negative) berechnet werden. Die Gene aus der Annotation, die vorhergesagt werden, entsprechen den TP. Gene, die in der Annotation vorhanden sind, aber nicht gefunden werden, bezeichnet man als FN, daraus folgt, daß die gesamte Anzahl der Gene in der Annotation der Summe von TP und FN entspricht. ORFs, die von den Programmen vorhergesagt werden, aber nicht in der Annotation vorhanden sind, werden als FP bezeichnet. Es wird angenommen, das diese nicht für Proteine codieren. Die Summe von FP und TP entspricht der gesamten Vorhersage eines Tools.
Die **Sensitivität** wird nach Gleichung (3.1) berechnet und ist ein Maß dafür wieviele Gene, die in der Annotation aufgeführt sind, von einem Tool gefunden werden. In der vorliegenden Arbeit wird die Annotation entsprechend der Publikationen (Tabelle 2.4, S. 60) als Referenz verwendet. Die Sensitivität (SEN) multipliziert mit 100 entspricht dem prozentualen Anteil der TP an der gesamten Menge der annotierten Gene.

$$\mathrm{SEN} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.1}$$

SEN	Sensitvität der Vorhersage
TP	True-Positive
FN	False-Negative

Die **Spezifität** (SPEZ) zeigt, wie groß der Anteil TP an der gesamten Vorhersage ist. Sie ist defniert durch Gleichung (3.2).

$$\mathrm{SPEZ} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.2}$$

SPEZ	Spezifität der Vorhersage
TP	True-Positive
FP	False-Positive

Vergleicht man Sensitivität und Spezifität als Bewertungskriterien, so zeigt sich, daß sich beide ergänzen. Wenn ein Tool sehr viele Gene aus der Annotation vorhersagt, hat es eine hohe Sensitivität. Wird diese hohe Sensitivität aber durch eine sehr große Gesamtvorhersage und damit einen großen Anteil an FP 'erkauft', so ist die Spezifität niedrig. Angestrebt wird eine Vorhersage mit hoher Sensitivität und hoher Spezifität.
Als weiteres Bewertungskriterium wird in dieser Arbeit der Parameter SC (Start-Correctness) verwendet, der angibt, welcher Anteil der vorhergesagten Startpositionen mit den entsprechenden aus der Annotation übereinstimmt.

$$\mathrm{SC} = \frac{\mathrm{TP}_{SC}}{\mathrm{Annot.}} \tag{3.3}$$

SC	Anteil der vorhergesagten ORFs mit korrektem Start
TP_{SC}	True-Positive deren Startposition mit der
Annot.	Anzahl annotierter Gene (entspricht TP+FN)

3.3 Vorhersagen der Tools CRITICA, GLIMMER, ZCURVE und ORPHEUS

Der folgende Abschnitt ist in zwei Blöcke aufgeteilt. Zum einen werden die Ergebnisse mit einer unteren Längenbegrenzung von 150 BP dargestellt, zum anderen ohne eine Mindestlänge. Darauf folgt eine kurze Zusammenfassung der beiden Abschnitte. Alle prozentualen Angaben beziehen sich auf die jeweils betrachtete Menge der annotierten Gene (alle annotierten Gene bzw. diejenigen $\geq$ 150 BP). Zum Vergleich der Programme wurde zu jedem angeführten Parameter der Mittelwert angegeben.
Desweiteren werden die Begriffe und Längen derjenigen Gene aus der Annotation aufgelistet, die von keinem der Tools vorhergesagt wurden.

3.3.1 Vorhersagen mit Mindestlänge $\geq$ 150 BP

Bei der Vorhersage mit einer Mindestlänge von $\geq$ 150 BP (Tabelle 3.1, S. 71), sagte **CRITICA** im Mittel 91,6% der annotierten Gene vorher. Im Schnitt lag der Anteil korrekt vorhergesagter Startpositionen bei 78,2%. Die Vorhersage enthielt dabei mit durchschnittlich 1,7% den geringsten Anteil FP. Der Anteil TP lag im Bereich 89,0% (*L. lactis*) bis 91,6% (*Buchnera*). Für *C. acetobutylicum* lag er um 1,8% sowie für *H. pylori* um 3,7% über dem Schnitt. Für alle anderen Genome war der Anteil TP etwas geringer als das Mittel. Bei den verwendeten Genomen mit einem G/C-Gehalt $\geq$ 50% lag der Anteil TP_{SC} 5,7% (*E. coli*) bis 14,3% (*M. tuberculosis*) unter dem Schnitt. Für die Genome mit einem G/C-Gehalt $<$ 50% lag er im Bereich 0,4% (*B. subtilis*) bis 9,4% (*C. acetobutylicum*) über dem Durchschnitt. Der Anteil FP lag im Intervall 0,8% (*E. coli*) bis 2,8% (*L. lactis*) und schien nicht in Zusammenhang mit dem G/C-Gehalt zu stehen.
Mit den gegebenen Parametern erzeugte **GLIMMER2.02** im Mittel 98,6% TP, mit 73% TP_{SC} und 9,5% FP. Der Anteil an TP variierte von 97,5% (*E. coli*) bis 100% (*Buchnera*), der an TP_{SC} im Bereich 51,3% (*M. tuberculosis*) bis 81,7% (*Buchnera*). Bei den Genomen mit einem G/C-Gehalt $\geq$ 50% lag der Anteil TP um 0,6% (*M. tuberculosis*) bis 1,1% (*E. coli*) unter dem Mittelwert, bei den TP_{SC} um 3,6% (*E.coli*) bis 22,3% (*M. tuberculosis*).

Der Anteil TP für *L. lactis* lag 0,2% unter dem Schnitt, während der Anteil TP_{SC} 8% darüber lag. Für alle anderen Genome mit einem G/C-Gehalt $<$ 50% lag der Anteil TP um 0,1% (*B. subtilis*) bis 1,4% (*Buchnera*) und der Anteil TP_{SC} um 0,5% (*H. pylori*) bis 8,1% (*Buchnera*) über dem Durchschnitt. Der Anteil FP lag im Bereich 4,2% (*C. acetobutylicum*) bis 23,1% (*M. tuberculosis*) und stieg mit zunehmendem G/C-Gehalt an.

Die Vorhersage von **GLIMMER2.10** beinhaltete im Durchschnitt 97,7% TP, 73,5% TP_{SC} und 7,3% FP. Der Anteil TP lag im Intervall 96,7% (*E. coli*) bis 99,5% (*Buchnera*). Für *Buchnera* und *H. pylori* lag der Anteil TP über dem Durchschnitt, für alle anderen Genome lag er knapp darunter. Für die verwendeten Genome mit einem G/C-Gehalt $\geq$ 50% lag der Anteil der TP_{SC} um 3,6% (*M. tuberculosis*) bis 21,6% (*M. tuberculosis*) niedriger als das Mittel. Bei allen getesteten Genome mit einem G/C-Gehalt $<$ 50% lag der Anteil TP_{SC} um 1,4% (*H. pylori*, *B. subtilis*) bis 8% (*Buchnera*) über dem Schnitt. Der Anteil FP lag im Intervall 3,6% (*Buchnera*) bis 19,4% (*M. tuberculosis*).

Bei einer Mindestlänge von 150 BP erzeugte **ORPHEUS** durchschnittlich einen Anteil von 95,1% TP, 76,0% TP_{SC} und 18,5% FP. Der Anteil TP lag dabei im Bereich von 80,8% (*M. tuberculosis*) bis 100% (*Buchnera*), der Anteil TP_{SC} im Bereich 43,5% (*M. tuberculosis*) bis 86% (*H. pylori*). Für Genome mit einem G/C-Gehalt $\geq$ 50% lag der Anteil TP um 1,8% (*E. coli*) bis 14,7% (*M. tuberculosis*) unter dem Schnitt. Der Anteil TP_{SC} lag ebenso unter dem Mittel. Die Differenz zum Mittelwert betrug hier bis zu 32,5%. Für alle Genome mit einem G/C-Gehalt $<$ 50% lagen sowohl TP als auch TP_{SC} über dem Durchschnitt. Die FP nahmen mit steigendem G/C-Gehalt zu.

ZCURVE sagte im Mittel 99,0% der ORFs $\geq$ 150 BP vorher. Dabei wurden durchschnittlich 81,1% der Startpositionen korrekt vorhergesagt und 13,7% FP produziert. Wie schon bei den anderen Tools beobachtet, lag der Anteil TP sowie der Anteil TP_{SC} für Genome mit einem G/C-Gehalt $\geq$ 50% unter dem Mittelwert, für die restlichen darüber. Die Differenz der TP zum Mittel bei den Genomen mit hohem G/C-Gehalt war mit 0,2% (*E. coli*) bis 1,5% geringer als bei den TP_{SC} mit 7,2% (*E. coli*) bis 16,7% (*M. tuberculosis*). Der Anteil FP lag für diese Genome im Intervall 19,7% bis 30,0%.

Für Genome mit einem GC < 50% lag der Anteil FP im Bereich 4,9% (*C. acetobutylicum* bis 14,0% (*B. subtilis*).
In Tabelle 3.1 wird außerdem angegenben, wieviele der Gene ≥ 150 BP von keinem der Tools gefunden wurde. Hierbei wurde zwei Varianten getestet, zum einen wurde GLIMMER2.02 zum andern GLIMMER2.10 einbezogen. Der Anteil nicht gefundener Gene lag bei 1,2%, wenn GLIMMER2.02 (GL2.02) verwendet wurde bzw. bei 1,3%, mit GLIMMER2.10 (GL2.10).

3.3.2 Vorhersagen ohne Mindestlänge

Bei der Verwendung von **CRITICA** ohne Mindestlänge (Tabelle 3.2, S. 72) sagte das Programm im Mittel mit 91,9% den niedrigsten Anteil TP vorher. Der Anteil TP_{SC} betrug im Schnitt 87,1%. Bei der Vorhersage wurde außerdem mit 1,8% durchschnittlich der geringste Anteil an FP erzeugt. Der Anteil TP variierten im Bereich von 88,6% (*L. lactis*) bis 92,8% (*H. pylori*), die TP_{SC} lagen im Intervall 63,8 (*M. tuberculosis*) bis 82,8% (*H. pylori*). Allgemein waren die Vorhersagen für Genome, mit niedrigem bis mittlerem G/C-Gehalt etwas besser als für Genome mit höherem G/C-Gehalt. Dies traf insbesondere auf die Vorhersage der korrekten Startposition zu. Für die Genome mit einem G/C-Gehalt ≥ 50% stimmten 63,8% (*M. tuberculosis*) bis 71,8% (*E. coli*) der TP, die CRITICA generiert hat, in der Startposition mit der Annotation überein. Der Anteil lag damit 5,7% bis 13,7% unter dem Durchschnitt. Für alle verwendeten Genome mit einem G/C-Gehalt < 50% lag der Anteil TP_{SC} um 0,1% (*B.subtilis*) bis 9,6% (*C. acetobutylicum*) über dem Schnitt. Der Anteil FP, den CRITICA produzierte, war mit 0,8% am niedrigsten für das Genome von *E. coli* und am höchsten für *L. lactis* mit 2,8%. Dabei zeigte der Anteil TP für das letztgenannte Genom mit 88,6% den geringsten Wert.
GLIMMER2.02 sagte ohne Mindestlänge im Durchschnitt 98,2% TP vorher, dabei wurden im Mittel 20,9% FP und 73,2% TP_{SC} produziert. Der Anteil TP lag im Bereich 96,9% (*E. coli*) bis 100% (*Buchnera*). Für TP_{SC} lag er im Bereich 51,2% (*M. tuberculosis*) bis 81,4% (*C. acetobutylicum*). Dabei lag der Anteil TP für Genome mit einem G/C-Gehalt ≥ 50% mit 0,4% (*M. tuberculosis*) bis 1,3% (*E. coli*) unter dem Durchschnitt. Bei den

TP_{SC} lag der Anteil 3,6% (*E. coli*) bis 22% (*M. tuberculosis*) niedriger. Für *H. pylori* (G/C-Gehalt: 38,9%) lag der Anteil TP mit 0,8% unter dem Mittel und der Anteil TP_{SC} mit 0,1%. Bei den anderen Genome mit einem G/C-Gehalt $< 50\%$ lag der Anteil TP 0% (*L. lactis*, *B.subtilis*) bis 1,8% (*Buchnera*) über dem Mittel. Der Anteil TP_{SC} lag mit 1,5% (*B. subtilis*) bis 8,2% (*L.lactis*) höher als der Durchschnitt. Für FP lag der Anteil im Bereich 14,2% (*Buchnera*, *C. acetobutylicum*) bis 31,1% (*M. tuberculosis*). Er stieg mit zunehmenden G/C-Gehalt an.

Bei den Vorhersagen von **GLIMMER2.10** lag der Anteil TP im Intervall 96,2% (*E. coli*) bis 99,6% (*Buchnera*). Im Mittel lag er bei 97,2% mit einem Anteil von 73,0% TP_{SC} und 14,1% FP. Für Genome mit einem G/C-Gehalt $\geq 50\%$ lag der Anteil TP um 1,0% unter dem Schnitt (*E. coli*) bzw. um 0,3% darüber (*M. tuberculosis*). Für die Genome mit einem G/C-Gehalt $< 50\%$ lag der Anteil TP zum Teil über dem Durchschnitt (*Buchnera* mit 99,6%) und zum zum Teil darunter (*H. pylori* mit 96,6%). Der Anteil TP_{SC} lag hier 8,6% unter dem Schnitt. Für die anderen Genome G/C-Gehalt: $< 50\%$ lag der Anteil TP_{SC} 1,4% (*B. subtilis*) bis 8,4% (*Buchnera*) höher als der Schnitt. Der Anteil FP nahm mit steigendem G/C-Gehalt zu. Er lag bei den Vorhersagen von GLIMMER2.10 im Bereich 7,4% (*Buchnera*) bis 25,7% (*M.tuberculosis*).

ZCURVE produzierte unter diesen Voraussetzungen im Mittel 98,3% TP mit 19,0% FP. Dabei betrug der mittlere Anteil TP_{SC} 80,5%. Für die Genome mit einem G/C-Gehalt $\geq 50\%$ lag der Anteil TP 0,1% (*E. coli*) bis 1,0% (*M. tuberculosis*) unter dem Schnitt. Der Anteil TP_{SC} war um 7,1% (*E. coli*) bis 16,2% (*M. tuberculosis*) geringer als das Mittel. Der Anteil TP, den ZCURVE für *H. pylori* vorhersagte, war 1,9% geringer als der Schnitt, während der Anteil TP_{SC} 0,9% niedriger lag. Bei allen weiteren Genomen mit einem G/C-Gehalt $< 50\%$ war der Anteil TP um 0,3% (*B. subtilis*) bis 1,5% (*Buchnera*) und der Anteil TP_{SC} um 2,6% (*B.subtilis*) bis 11,6% (*C. acetobutylicum*) größer. Der Anteil FP lag im Intervall 9,0% (*C. acetobutylicum*) bis 35,2% (*M. tuberculosis*). Wie bei den Vorhersagen der anderen Tools, stieg er mit zunehmenden G/C-Gehalt an.

Für **ORPHEUS** wurden die Werte aus den eingangs genannten Gründen nicht angegeben.

Weiterhin wird in Tabelle 3.2 angegeben, wieviele der annotierten Gene von keinem der verwendeten Tools vorhergesagt wurde. Es werden hier jeweils zwei Werte angegeben. Der Wert GL2.02 bezieht sich auf die gesamte Vorhersage, wenn GLIMMER2.02 verwendet wurde, analog bezieht sich der Wert GL2.10 auf die gesamte Vorhersage, wobei GLIMMER2.10 einbezogen wurde. Der Anteil nicht gefundener Gene lag bei 0,2% (GL2.02) bzw. 0,3% (GL2.10).

3.3.3 Zusammenfassung der Vorhersagen

Bei der Untersuchung wurde deutlich, daß CRITICA im Durchschnitt den geringsten Anteil an FP und TP erzeugte. Dies galt sowohl für die Vorhersage mit Längenbegrenzung, als auch ohne Mindestlänge. ZCURVE, ORPHEUS und beide Versionen von GLIMMER sagten jeweils mehr TP vorher, wobei sie allerdings auch wesentlich mehr FP produzierten.

Die Performanz der Vorhersagen von GLIMMER und ZCURVE nahm ab, wenn keine Mindestlänge vorgegeben wurde. Der Anteil TP sank im Mittel um bis zu 0,7% ohne Längenbeschränkung, während der Anteil TP um bis zu 11,4% höher war. Für CRITICA nahmen der Anteil an TP um 0,1% ab, wenn keine Mindestlänge gesetzt war, wobei Der Anteil FP um 0,1% zunahm. Die Performanz der Vorhersage ohne Mindestlänge war hier also kaum schlechter. Weiterhin war erkennbar, daß sich die Vorhersage von GLIMMER, ZCURVE und ORPHEUS mit steigendem G/C-Gehalt deutlich verschlechterten. Vor allem der Anteil korrekt vorhergesagter Startpositionen nahm ab. Die Vorhersage von CRITICA verschlechterte sich mit steigendem G/C-Gehalt deutlich weniger.

Tabelle 3.1: **Vorhersagequalität der Programme Critica, Glimmer, ZCurve und Orpheus für Gene ≥ 150 BP.** FP (False-Positive): ORFs, die nicht in der Annotation vorkommen; TP (True-Positive): Anzahl tatsächlich annotierter Gene aus der Vorhersage; TP_{SC}: Vorhersagen, bei denen die Startposition mit der Startposition aus der Annotation übereinstimmt; Annot.: Anzahl von annotierten Genen. Die prozentualen Angaben beziehen sich auf die annotierten Gene ≥ 150 BP. NF: Anzahl annotierter Gene, die von keinem der Tools gefunden wurden. Da zwei Versionen von Glimmer getestet wurden, ist jeweils NF angegeben, wenn Glimmer2.02 (GL2.02) verwendet wurde bzw. wenn Glimmer2.10 (GL2.10) verwendet wurde.

Genom	GC	Critica			Glimmer2.02			Glimmer2.10			Orpheus			ZCurve			NF	
	(%)	FP	TP	TP_{SC}	FP	TP	TP_{SC}	FP	TP	TP_{SC}	FP	TP	TP_{SC}	FP	TP	TP_{SC}	GL2.02	GL2.10
Buchnera sp.	26,3	7	515	445	28	562	456	20	562	459	43	559	458	37	562	478	0	0
Annot. 562		1,2	91,6	79,2	5,0	100	81,7	3,6	99,5	81,5	7,7	100	85,1	6,6	100	85,1	0	0
C. acetobutylicum	30,9	44	3430	3194	154	3605	2969	116	3559	2959	375	3576	3157	177	3617	3375	10	19
Annot. 3647		1,2	94,0	87,6	4,2	98,8	81,4	3,2	97,6	81,1	10,3	98,1	86,6	4,9	99,2	92,5	0,2	0,5
L. lactis	35,3	62	2006	1815	167	2218	1839	132	2183	1816	339	2209	1847	358	2240	1948	7	11
Annot. 2254		2,8	89,0	80,5	7,4	98,4	81,6	5,9	96,9	80,6	15,0	98,0	81,9	11,4	99,4	86,4	0,3	0,4
H. pylori	38,9	40	1450	1292	115	1498	1121	97	1490	1132	152	1484	1312	138	1499	1237	2	3
Annot. 1512		2,6	95,9	85,4	7,6	99,1	74,1	6,4	98,5	74,9	10,1	98,1	86,8	9,1	99,1	81,8	0,1	0,2
B. subtilis	43,5	47	3692	3177	427	3993	3039	262	3934	3029	755	3932	3244	566	4017	3391	9	17
Annot. 4044		1,2	91,3	78,6	10,6	98,7	75,1	6,5	97,3	74,9	18,7	97,2	80,2	14,0	99,3	83,9	0,2	0,4
E. coli	50,0	32	3880	3080	377	4142	2975	249	4111	2972	915	3657	2880	839	4199	4955	16	17
Annot. 4250		0,8	91,3	72,5	8,9	97,5	70,0	5,9	96,7	69,9	21,5	93,3	67,8	19,7	98,8	73,9	0,4	0,4
M. tuberculosis	65,5	84	3537	2484	899	3809	1996	755	3794	2018	1804	3142	1690	1165	3790	2502	18	21
Annot. 3888		2,2	91,0	63,9	23,1	98,0	51,3	19,4	97,6	51,9	46,4	80,8	43,5	30,0	97,5	64,4	0,5	0,5
Mittelwert		**1,7**	**92,0**	**78,2**	**9,5**	**98,6**	**73,6**	**7,3**	**97,7**	**73,5**	**18,5**	**95,1**	**76,0**	**13,7**	**99,0**	**81,1**	**0,2**	**0,3**

Tabelle 3.2: **Vorhersagequalität der Programme Critica, Glimmer, ZCurve und Orpheus.** Die prozentualen Angaben beziehen sich auf alle annotierten Gene. Abkürzungen: Siehe Tabelle 3.1. nd: nicht dargestellt.

Genom	GC	Critica			Glimmer2.02			Glimmer2.10			Orpheus			ZCurve			NF	
	(%)	FP	TP	TP_{SC}	FP	TP	TP_{SC}	FP	TP	TP_{SC}	FP	TP	TP_{SC}	FP	TP	TP_{SC}	GL2.02	GL2.10
Buchnera sp.	26,3	7	517	447	80	564	457	42	604	562	nd	nd	nd	84	563	479	0	0
Annot. 564		1,2	91,7	79,3	14,2	100	81,0	7,4	99,6	81,4	nd	nd	nd	14,9	99,8	84,9	0	0
C. acetobutylicum	30,9	53	3435	3488	520	3619	2980	276	3568	2967	nd	nd	nd	332	3630	3383	27	31
Annot. 3672		1,4	93,5	87,1	14,2	98,6	81,2	7,5	97,2	80,8	nd	nd	nd	9,0	98,9	92,1	0,7	0,8
L. lactis	35,3	63	2007	1816	372	2225	1844	253	2189	1821	nd	nd	nd	343	2245	1950	9	12
Annot. 2266		2,8	88,6	80,0	16,4	98,2	81,4	11,2	96,6	80,4	nd	nd	nd	15,1	99,1	86,1	0,3	0,5
H. pylori	38,9	42	1454	1296	360	1525	1145	276	1511	1150	nd	nd	nd	247	1509	1246	27	34
Annot. 1566		2,7	92,8	82,8	23,0	97,4	73,1	17,6	96,5	64,4	nd	nd	nd	15,8	96,4	79,6	1,7	2,2
B. subtilis	43,5	59	3696	3182	1042	4026	3061	645	3961	3048	nd	nd	nd	799	4036	3406	19	29
Annot. 4099		1,4	90,2	77,6	25,4	98,2	74,7	15,7	96,6	74,4	nd	nd	nd	19,5	98,5	83,1	0,5	0,7
E. coli	50,0	36	3882	3082	947	4155	2986	590	4124	2983	nd	nd	nd	1012	4210	3146	37	37
Annot. 4289		0,8	90,5	71,8	22,1	96,9	69, 6	13,8	96,2	69,6	nd	nd	nd	23,6	98,2	73,4	0,9	0,9
M. tuberculosis	65,5	93	3539	2486	1213	3811	1996	1000	3796	2018	nd	nd	nd	1371	3791	2503	20	23
Annot. 3895		2,4	90,9	63,8	31,1	97,8	51,2	25,7	97,5	51,8	nd	nd	nd	35,2	97,3	64,3	0,5	0,6
Mittelwert		**1,8**	**91,9**	**77,5**	**20,9**	**98,2**	**73,2**	**14,1**	**97,2**	**73,0**	**nd**	**nd**	**nd**	**19,0**	**98,3**	**80,5**	**0,7**	**0,8**

3.3.4 Annotierte ORFs, die von keinem der Tools vorhergesagt werden

Zur Bewertung der Tools und der Methode der Genvorhersage an sich, ist es wichtig festzustellen, welche Gene von keinem der Programme vorhergesagt wurden. Zur Charakterisierung der Genklasse dieser ORFs, wurde die Annotation geparst und in Tabelle 3.3.4 zusammengefaßt[2]. Es wurden dabei alle Gene der Annotation ohne Mindestlänge betrachtet, wobei die Mindestlänge der Vorhersagen auf $\geq$ 150 BP gesetzt war. Es zeigte sich, daß viele der ORFs, die unter diesen Bedingungen von keinem Tool vorhergesagt wurden, biochemisch nicht verifiziert waren. Das heißt sie waren nicht experimentell belegt, sondern wurden nur auf Grund einer Vorhersage als codierend angenommen. Diese ORFs wurden in der Annotation oft mit Begriffen wie 'hypothetical' (hypothetisch) oder 'predicted' (vorhergesagt) gekennzeichnet. Beispielsweise wurden 41 der Gene aus der Annotation von *C. acetobutylicum* von keinem Tool mit vorgegebener Mindestlänge $\geq$ 150 BP vorhergesagt. Davon waren 31 mit 'hypothetical protein' (hypothetisches Protein) und 5 mit 'predicted membrane protein' (vorhergesagtes Membranprotein) bezeichnet. Häufig handelte es sich bei nicht vorhergesagten ORFs auch um solche, die als ribosomale oder Prophagen-Proteine annotiert waren. Dies macht deutlich, daß auch mit einer unteren Längenbegrenzung von 150 BP für alle Tools, nur wenig Gene nicht gefunden wurden, denen nach der Annotation biochemisch-relevante Proteine zugeordnet waren.

[2] GLIMMER2.02 wurde hierbei nicht einbezogen

Tabelle 3.3: **Zusammenfassung der annotierten Gene, die von keinem der verwendeten Tools gefunden wurden.** Die Mindestlänge bei den Vorhersagen der Tools war mit 150 BP vorgegeben. Die Anzahl nicht gefundener Gene, bezieht sich dabei auf alle Gene aus der Annotation ohne untere Längengrenze. Die Vorhersage von GLIMMER2.02 wurde nicht einbezogen. Anz: Anzahl nicht vorhergesagter Gene; Annotation; LenI: Längen der ORFs, die nicht vorhergesagt wurden bzw. das entsprechende Längeninterall in BP; MLen: mittlere Länge in BP.

Anz	Annotation	LenI	MLen
Buchnera sp.			
	Es wurden alle annotierten Gene identifiziert.		
C. acetobutylicum			
31	hypothetical protein	84-279	158
5	predicted membrane protein	123-219	186
3	ribosomal protein	147, 111, 132	130
1	uncharacterized C4-type Zn-finger containing protein		180
1	TldD-like protein fragment		204
L. lactis			
8	unknown protein	96-207	167
5	ribosomal protein	114-147	137
3	prophage pi1 protein	147, 150, 180	159
H. pylori			
50	H.pylori predicted coding region	36-294	103
3	ribosomal protein	132, 111, 144	129
1	histidine-rich, metal binding polypeptide (hpn)		180
B. subtilis			
38	keine Annotation	63-234	129
6	regulator of the activity of phosphatase (Rap)	114-132	122
3	ribosomal protein	111, 132, 147	130
1	A subtilosin		129
1	spore coat protein		144
E. coli			
28	hypothetical protein	96-444	172
9	operon leader peptide	42-96	76
1	synthetase (pheST) operon leader	-	42
4	ribosomal protein	114-411	200
2	RNA	117, 135	126
1	regulator for sigma 32 heat shock promoters	-	588
1	in rhs element	-	219
1	modulation factor	-	165
1	protaminelike protein	-	99
1	alleviation and modification enhancement	-	192
1	destructive to membrane potential	-	153
1	peptide of chorismate mutase-P-prephenate dehydrat	-	45
1	histone	-	339
M. tuberculosis			
21	hypothetical protein	81-651	285
1	rpmJ		111
1	rpmB2		234
1	rpmH		141

3.4 Vergleich der Vorhersagen

Voraussetzung für die Entscheidung, auf welche Weise die Vorhersagen der verwendeten Tools in YACOP kombiniert werden sollten, um ein optimales Ergebnis zu erzielen, war der Vergleich der einzelnen Ausgaben mit einer Mindestlänge von $\geq$ 150 BP und ohne Mindestlänge. Die Qualität der Vorhersagen wurde durch die Parameter Spezifität (SPEZ), Sensitivität (SEN) und Start-Correctness (SC)[3] dargestellt (Tabelle 3.4 und 3.5, S. 77).

Bei den Untersuchungen wurde gezeigt, daß sich für CRITICA die Qualität der Vorhersagen mit einer unteren Längenbegrenzung von $\geq$ 150 BP und ohne Mindestlänge nahezu glich. Nur die Sensitivität lag im Mittel für die Vorhersage ohne Mindestlänge um 0,01 niedriger als mit der Bedingung $\geq$ 150 BP. Für die anderen verwendeten Tools nahme die Qualität der Vorhersage ohne Mindestlänge meist stärker ab. Für GLIMMER2.02 fiel die Spezifität im Durchschnitt um 0,08, wenn keine Mindestlänge vorgegeben wurde. Der Wert der Sensitivität war hier um 0,01 niedriger, während der Wert für den Anteil korrekt vorhergesagter Startpositionen ohne Mindestlänge um 0,11 höher lag als für $\geq$ 150 BP. Bei GLIMMER2.10 nahm die Spezifität um 0,05 ab. Sensitivität und der Wert SC waren jeweils um 0,01 niedriger. Die Werte für ZCURVE verhielten sich ähnlich, die Spezifität lag um 0,04 niedriger, Sensitivität und Vorhersage der korrekten Startposition waren jeweils wie bei GLIMMER2.10 um 0,01 niedriger, wenn keine Mindestlänge festgesetzt war.
Die Vorhersage von CRITICA wies unter beiden Bedingungen im Mittel die höchste Spezifität (0,98) auf, Sensitivität (0,91) und SC (0,78) waren niedriger als bei den anderen Tools. Die beste Sensitivität ohne Mindestlänge wurde von GLIMMER2.02 und ZCURVE erreicht. Sie lag um 0,06 über dem Wert von CRITICA. Der Anteil korrekter Startpositionen lag dabei um 0,07 (GLIMMER2.02) bzw. 0,02 (ZCURVE) höher, während die Spezifität um 0,15 (GLIMMER2.02) bzw. 0,14 (ZCURVE) niedriger war als die von CRITICA. GLIMMER2.10 war in den Werten SEN und SC um 0,01 bzw. 0,12 schlechter als die ältere Version GLIMMER2.02. Dabei lag die Spezifität um 0,05 höher.

[3] siehe Abschnitt 3.2, S. 64

Bei der Vorhersage mit Mindestlänge zeigte ORPHEUS den niedrigsten Wert für die Spezifität. Die anderen Tools lagen hier um 0,04 (ZCURVE) bis 0,14 (CRITICA) höher. Die höchste Sensitivität erreichten mit 0,99 ZCURVE und GLIMMER2.02, GLIMMER2.10 lag mit 0,01 darunter, ORPHEUS mit 0,04 und CRITICA mit 0,07. Der Wert SC für Vorhersagen mit Mindestlänge war mit 0,88 bei ZCURVE am höchsten, CRITICA zeigte einen Wert der um 0,03 niedriger war. Für beide Versionen von GLIMMER lag der Wert für die korrekte Vorhersage der Startpositionen um 0,07 niedriger, für ORPHEUS um 0,05.

Tabelle 3.4: **Vorhersagequalität der Tools bei einer Mindestlänge von ≥ 150 BP.** SPEZ: Spezifität; SEN: Sensitivität; SC: Start-Correctness; nd: nicht dargestellt. Die Werte wurden nach den Gleichungen (3.1-3.3), S. 64 berechnet.

	(1)Critica			(2)Glimmer2.02			(3)Glimmer2.10			(4)Orpheus			(5) ZCurve		
Genom	SPEZ	SEN	SC	SPEZ	SEN	SC	SPEZ	SEN	SC	SPEZ	SEN	SC	SPEZ	SEN	SC
Buchnera sp.	0,99	0,92	0,79	0,95	1,0	0,82	0,82	1,0	0,82	0,93	1,0	0,85	0,94	1,0	0,85
C. acetobutylicum	0,99	0,94	0,88	0,96	0,99	0,81	0,97	0,98	0,81	0,91	0,98	0,87	0,95	0,99	0,93
L. lactis	0,97	0,89	0,81	0,93	0,98	0,82	0,94	0,97	0,81	0,87	0,98	0,82	0,90	0,99	0,86
H. pylori	0,97	0,96	0,85	0,93	0,99	0,74	0,94	0,99	0,75	0,91	0,98	0,87	0,92	0,99	0,82
B. subtilis	0,99	0,91	0,79	0,90	0,99	0,75	0,94	0,97	0,75	0,84	0,97	0,80	0,88	0,99	0,84
E. coli	0,99	0,91	0,73	0,92	0,97	0,70	0,94	0,97	0,70	0,80	0,93	0,68	0,83	0,99	0,74
M. tuberculosis	0,98	0,91	0,64	0,81	0,98	0,51	0,83	0,98	0,52	0,64	0,81	0,44	0,76	0,98	0,64
Mittelwert	**0,98**	**0,92**	**0,78**	**0,91**	**0,99**	**0,74**	**0,93**	**0,98**	**0,74**	**0,84**	**0,95**	**0,76**	**0,88**	**0,99**	**0,81**

Tabelle 3.5: **Vorhersagequalität der Tools ohne Mindestlänge.** Abkürzungen: Siehe Tabelle 3.4.

	(1)Critica			(2)Glimmer2.02			(3)Glimmer2.10			(4)Orpheus			(5) ZCurve		
Genom	SPEZ	SEN	SC	SPEZ	SEN	SC	SPEZ	SEN	SC	SPEZ	SEN	SC	SPEZ	SEN	SC
Buchnera sp.	0,99	0,92	0,79	0,88	1,0	0,81	0,93	1,0	0,81	nd	nd	nd	0,87	1,0	0,85
C. acetobutylicum	0,98	0,94	0,87	0,87	0,99	0,81	0,93	0,97	0,81	nd	nd	nd	0,92	0,99	0,92
L. lactis	0,97	0,89	0,80	0,86	0,99	0,81	0,90	0,97	0,80	nd	nd	nd	0,87	0,99	0,86
H. pylori	0,97	0,93	0,83	0,81	0,97	0,73	0,85	0,96	0,73	nd	nd	nd	0,86	0,96	0,79
B. subtilis	0,98	0,90	0,78	0,79	0,98	0,75	0,86	0,97	0,74	nd	nd	nd	0,83	0,98	0,83
E. coli	0,99	0,91	0,72	0,81	0,97	0,70	0,87	0,96	0,70	nd	nd	nd	0,81	0,98	0,73
M. tuberculosis	0,97	0,91	0,64	0,76	0,98	0,51	0,79	0,97	0,52	nd	nd	nd	0,73	0,97	0,64
Mittelwert	**0,98**	**0,91**	**0,78**	**0,83**	**0,98**	**0,85**	**0,88**	**0,97**	**0,73**	**nd**	**nd**	**nd**	**0,84**	**0,98**	**0,80**

3.5 Kombination der Vorhersagen zur Verbesserung der Performanz durch YACOP

Auf Grund der in Abschnitt 3.4 dargestellten Ergebnisse wurden als mögliche bool'sche Verknüpfungen für die Meta-Vorhersage von YACOP die Menge CRITICA ∪ (GLIMMER ∩ ZCURVE) sowie die Schnittmenge ZCURVE ∩ GLIMMER ausgewählt. Da die Spezifität von CRITICA im Vergleich zu den anderen Tools sehr hoch war, wurde die gesamte Vorhersage bei die Zusammenstellung der Kombination für YACOP in Betracht gezogen. Dabei wurde für CRITICA keine Mindestlänge vorgegeben, da sich die Qualität der Vorhersage im Gegensatz zu den anderen Tools nicht verbessert, wenn nur ORFs ≥ 150 BP einbezogen wurden. Daher wurden die Angaben für CRITICA ∪ (GLIMMER ∩ ZCURVE) auf alle annotierten Gene bezogen, auch wenn für die Schnittmenge GLIMMER ∩ ZCURVE eine Mindestlänge vorgegeben war.

ZCURVE und beide Versionen von GLIMMER zeigten jeweils eine höhere Sensitivität als CRITICA und ORPHEUS. Dabei war die Spezifität schlechter. Daher sollten für die weiteren Untersuchungen die Vorhersagen von GLIMMER und ZCURVE kombiniert werden. Die Kombinationen (ZCURVE ∩ GLIMMER2.02 und ZCURVE ∩ GLIMMER2.10) wurden jeweils ohne Mindestlänge und mit einer Mindestlänge von ≥ 150 BP getestet. Der Wert SC, für die korrekte Vorhersage der Startposition, den ZCURVE dabei lieferte, lag in allen Fällen höher als der entsprechende Wert von GLIMMER (nicht dargestellt). Daher wurde die Vorhersage der Startposition von ZCURVE vorgezogen. ORPHEUS zeigte unter den Testbedingungen die schlechtesten Ergebnisse und wurde daher von der weitere Evaluation ausgeschlossen.

Die Auswertung der Schnittmenge ZCURVE ∩ GLIMMER (Tabelle 3.7 und 3.6, S. 81) zeigte, daß im Mittel für die Spezifität jeweils ein höherer Wert erzielt wurde als von ZCURVE oder einer der Versionen von GLIMMER alleine. Die Sensitivität war bei den Untersuchungen ohne Mindestlänge im Mittel unverändert im Vergleich zu den Einzelergebnissen. Mit Mindestlänge ≥ 150 BP lag sie durchschnittlich um 0,01 unter den Werten. Der Wert SC war für die Kombination bei der Untersuchung ohne Mindestlänge um 0,05 kleiner als für GLIMMER2.02 alleine, im Vergleich zu ZCURVE war der Wert

gleich (Kombination mit GLIMMER2.02) bzw. um 0,01 niedriger (Kombination mit GLIMMER2.10). Für die Tests mit Mindestlänge $\geq$ 150 BP lag der Wert SC der Kombination im Mittel um 0,05 höher als für GLIMMER2.02 oder GLIMMER2.10 alleine. ZCURVE erzielte im Schnitt alleine einen um 0,01 höheren Wert als in Kombination mit GLIMMER2.02. Im Vergleich zur Kombination mit GLIMMER2.10 lag der Einzelwert um 0,02 höher.
Da die Kombination ZCURVE $\cap$ GLIMMER2.02 im Schnitt höhere Werte für Sensitivität und die korrekte Vorhersage der Startpositionen erzielte, als die Kombination ZCURVE $\cap$ GLIMMER2.10 wurde die erstgenannte als Element von YACOP ausgewählt. Dabei wurde die Länge mit $\geq$ 150 BP festgesetzt, da sich hier Sensitivität und der Wert SC im Vergleich zur Vorhersage ohne Mindestlänge nicht veränderten, wobei der Wert für die Spezifität um 0,02 höher lag.
Für die Verknüpfung CRITICA $\cup$ (ZCURVE $\cap$ GLIMMER2.02) entsprachen die Werte der Spezifität und Sensitivität im Mittel denen für ZCURVE $\cap$ GLIMMER2.02. Der Wert SC wurde zweimal mit unterschiedlicher Kombination der Programme ermittelt. Zum einen wurde die vorhergesagte Startposition von CRITICA vorgezogen (SC_{crit}), zum anderen die von ZCURVE (SC_{ZC}). Sofern keine Vorhersage von CRITICA vorlag, wurde immer die Startposition von ZCURVE angegeben. Der Wert SC_{crit} lag bei der Vorhersage ohne Mindestlänge 0,01 über dem Wert SC_{ZC} und 0,02 über dem Wert der Kombination ZCURVE $\cap$ GLIMMER2.02. Wurden nur die Vorhersage von ZCURVE und GLIMMER2.02 $\geq$ 150 BP einbezogen, so lag der Wert SC_{Crit} um 0,03 höher als der Wert SC_{ZC} und der Wert SC der Schnittmenge von ZCURVE und GLIMMER2.02. Dabei ist zu beachten, daß die Werte für die Kombination CRITICA $\cup$ (GLIMMER $\cap$ ZCURVE) mit unterer Längenbegrenzung für GLIMMER $\cap$ ZCURVE auf alle annotierten Gene bezogen sind, da die Vorhersage von CRITICA ohne Mindestlänge einbezogen wurde.
Als optimale Kombination für YACOP ergab sich durch die Untersuchungen die Mengen der Vorhersagen nach der folgenden bool'schen Verknüpfung: CRITICA $\cup$ (ZCURVE $\cap$ GLIMMMER). Dabei wurde für GLIMMER und ZCURVE eine Mindestlänge von $\geq$ 150 BP gesetzt. Als Startposition wurde, wenn vorhanden, die von CRITICA vorhergesagte, vorgezogen, anderenfalls wurde die entsprechende Vorhersage von ZCURVE angegeben. Insbesondere im

Vergleich zu den einzelnen Tools, aber auch gegenüber der Kombination ZCurve ∩ Glimmer war die Qualität der Vorhersage von Yacop besser. Dies galt vor allem für die korrekte Vorhersage der Startpositionen. Hier konnte die Performanz im Vergleich zu den einzelnen Tools um bis zu 0,10 verbessert werden.

Tabelle 3.6: **Vorhersagequalität der Kombinationen mit Mindestlänge ≥ 150 BP.** GL2.02: GLIMMER2.02; ZC: ZCURVE; Crit: CRITICA; GL2.02 ∩ ZC: Schnittmenge der Vorhersagen von GLIMMER2.02 und ZCURVE; GL2.10 ∩ ZC: Schnittmenge der Vorhersagen von GLIMMER2.10 und ZCURVE; Crit ∪ (GL2.02 ∩ ZC): Summe der gesamten Vorhersage von CRITICA und der Schnittmenge der Vorhersagen von GLIMMER2.02 und ZCURVE (YACOP); SC_{ZC} Start-Correctness, wenn die Vorhersage von ZCURVE bevorzugt wurde; SC_{Crit}: Start-Correctness, wenn die Vorhersage von CRITICA bevorzugt wurde; SPEZ: Spezifität; SEN; Sensitivität. Für CRITICA ist keine Mindestlänge vorgegeben. Dementsprechend wurden SEN, SC_{Crit} und SC_{ZC} für die Werte von YACOP auf alle annotierten ORFs bezogen.

Genom	GL2.02 ∩ ZC			GL2.10 ∩ ZC			Crit ∪ (GL2.02 ∩ ZC) (YACOP)			
	SPEZ	SEN	SC_{ZC}	SPEZ	SEN	SC_{ZC}	SPEZ	SEN	SC_{ZC}	SC_{Crit}
Buchnera sp.	0,96	1,0	0,85	0,97	1,0	0,83	0,96	1,0	0,85	0,86
C. acetobutylicum	0,98	0,99	0,92	0,98	0,97	0,91	0,97	0,98	0,92	0,91
L. lactis	0,94	0,98	0,85	0,95	0,97	0,84	0,94	0,98	0,85	0,88
H. pylori	0,95	0,99	0,81	0,95	0,98	0,78	0,95	0,96	0,79	0,85
B. subtilis	0,95	0,98	0,83	0,96	0,97	0,82	0,94	0,97	0,82	0,84
E. coli	0,95	0,97	0,73	0,96	0,96	0,72	0,95	0,97	0,73	0,76
M. tuberculosis	0,91	0,96	0,64	0,92	0,96	0,63	0,91	0,98	0,64	0,68
Mittelwert	**0,95**	**0,98**	**0,80**	**0,96**	**0,97**	**0,79**	**0,95**	**0,98**	**0,80**	**0,83**

Tabelle 3.7: **Vorhersagequalität der Kombinationen ohne Mindestlänge.** Abkürzungen: Siehe Tabelle 3.6

Genom	GL2.02 ∩ ZC			GL2.10 ∩ ZC			Crit ∪ (GL2.02 ∩ ZC)			
	SPEZ	SEN	SC_{ZC}	SPEZ	SEN	SC_{ZC}	SPEZ	SEN	SC_{ZC}	SC_{Crit}
Buchnera sp.	0,92	1,0	0,85	0,95	1,0	0,85	0,93	1,0	0,85	0,86
C. acetobutylicum	0,96	0,98	0,92	0,97	0,97	0,91	0,96	0,98	0,92	0,92
L. lactis	0,93	0,98	0,85	0,94	0,96	0,83	0,93	0,98	0,85	0,86
H. pylori	0,91	0,96	0,79	0,93	0,96	0,79	0,91	0,96	0,80	0,85
B. subtilis	0,92	0,97	0,82	0,94	0,96	0,81	0,92	0,98	0,83	0,84
E. coli	0,94	0,96	0,72	0,96	0,96	0,72	0,94	0,97	0,73	0,76
M. tuberculosis	0,90	0,96	0,63	0,91	0,97	0,64	0,90	0,97	0,64	0,76
Mittelwert	**0,93**	**0,98**	**0,80**	**0,94**	**0,97**	**0,79**	**0,93**	**0,98**	**0,81**	**0,84**

Kapitel 4

Diskussion

In der vorliegenden Arbeit wurde der Ansatz verfolgt, die Performanz der Genvorhersage durch bool'sche Verknüpfung der Ausgaben mehrerer Tools zu erhöhen. Die verwendeten Tools waren CRITICA, GLIMMER, ORPHEUS und ZCURVE.

Es werden bis zu 99% der annotierten Gene von den Tools vorhergesagt. Dabei werden aber zum Teil mehr als 30% False-Positive erzeugt. Daher galt es in erster Linie, die Menge der False-Positiven zu reduzieren. Weiterhin ist die Vorhersage der Startpositionen bislang unzureichend und sollte verbessert werden. Von GOU *et al.* (2003) wurde gezeigt, daß die Schnittmenge der Vorhersagen von GLIMMER und ZCURVE im Mittel die gleiche Sensitivität aufweist, wie die Einzelvorhersagen. Die Spezifität kann durch die Kombination um bis zu 20% zunehmen. Das Meta-Tool, YACOP, basierend auf der bool'schen Verknüpfung der Vorhersage von CRITICA mit der Schnittmenge der Vorhersagen von GLIMMER und ZCURVE, mit GLIMMER, ZCURVE $\geq$ 150 BP, erzielte, sowohl für die Spezifität, als auch für die Vorhersage der korrekten Startpositionen deutlich bessere Ergebnisse. Die Anzahl TP blieb für alle getesteten Genome annähernd gleich.

Um eine optimale Kombination für YACOP zu finden, wurden die einzelnen Vorhersagen der Tools bewertet und verglichen. Die Bewertung stützte sich dabei auf die bestehende Annotation der getesteten Genome. Dieses Vorgehen muß kritisch hinterfragt werden. Es kann nicht sichergestellt werden, daß die bestehende Annotation fehlerfrei ist. Oft sind Gene nur auf Grund

einer Vorhersage annotiert und nicht durch experimentelle Arbeit verifiziert. Daher ist mit einem gewissen Anteil fehlerhafter Angaben in der Annotation zu rechnen. Dies gilt insbesondere für die annotierten Startpositionen. Bei der Betrachtung der Gene aus der Annotation, die von keinem der getesteten Tools vorhergesagt wurden, zeigt sich, daß ein Großteil als "hypothetic" (hypothetisch) oder "predicted" (vorhergesagt) gekennzeichnet ist. Dies verdeutlicht die Problematik, daß Gene aus der Annotation teilweise nicht experimentell belegt sind, wodurch die Angabe von TP, FP und FN von den wahren Werten abweichen kann. Aus Mangel an verläßlichen Daten als Referenz, mußte diese Einschränkung jedoch akzeptiert werden.
Es besteht außerdem die Möglichkeit, daß Gene von keinem bislang entwickelten computer-basierten Verfahren vorhergesagt werden. Am Genom von *E. coli*, derzeit einem der am besten untersuchten Genome, wurde jedoch gezeigt, daß hier nahezu alle biochemisch-relevanten Gene gefunden werden (Guo *et al.*, 2003).
Weiterhin kann eine Verzerrung der Bewertungen auftreten, wenn die Annotation auf einem Algorithmus beruht, der dem Vorgehen eines der verwendeten Tools ähnelt oder gleicht. In den Ausführungen der Arbeit wurden ausschließlich Genome verwendet, für die sichergestellt werden konnte, daß die Annotation nicht auf einem der getesteten Tools beruht. Dennoch können Parallelen zwischen den Verfahren, die von den Annotatoren verwendet wurden und den hier vorgestellten, nicht ausgeschlossen werden.
Die Vorhersagen aller Tools für *C. acetobutylicum* weisen auffällig gute Werte auf. Dies könnte darauf zurückzuführen sein, daß bei dieser Annotation ein größerer Anteil der Vorhersagen ohne Konsistenzprüfung übernommen wurde. Diese Vermutung liegt nahe, da in der Publikation nur ein Autor angegeben wurde.
Bei den Untersuchungen wurden in erster Linie die Mittelwerte der Parameter betrachtet, die für die Vorhersagen berechnet wurden. Dadurch wurden die beschriebenen Fehlerquellen bei der Bewertung etwas relativiert. Trotz der verbleibenden Unsicherheiten, ist der Vergleich informativ und kann zur Evaluation einer verbesserten Vorhersage durch geeignete Kombination herangezogen werden. Desweiteren wird darauf hingewiesen, daß die Tendenzen, wie sie für die Vorhersagen der einzelnen Tools dargestellt werden,

durch weitere, nicht in der Arbeit angeführte Genvorhersagen bestätigt wurden.
Bei den Untersuchungen zeigte sich, daß die Qualität der Vorhersagen mit steigendem G/C-Gehalt sank. Insbesondere nahm die Anzahl an FP zu, während die Anzahl an TP_{SC} abnahm. Für Genome mit einem G/C-Gehalt $\geq 50\%$ war die Qualität signifikant schlechter als für jene mit einem G/C-Gehalt $< 50\%$. Möglicherweise ist dies auf die Default-Einstellungen der Programme zurückzuführen. Es wurde in der Arbeit nicht untersucht, inwieweit die Qualität der Vorhersage erhöht werden kann, indem die Parameter speziell auf die Eigenheiten des betrachteten Genomes abgestimmt werden. Möglicherweise läßt sich die Performanz der Vorhersagen so im einzelnen noch verbessern.

Die Vorhersage von YACOP wurde durch einen hohen G/C-Gehalt deutlich weniger beeinträchtigt als die der einzelnen Tools. Das ist zum einen darauf zurückzuführen, daß die Vorhersage von CRITICA bei steigenden G/C-Gehalt weniger an Qualität abnahm als bei GLIMMER, ZCURVE und ORPHEUS. Zum anderen war die Qualität der Vorhersagen von GLIMMER und ZCURVE deutlich besser, für Genome, die einen hohen G/C-Gehalt aufweisen, wenn eine untere Längenbegrenzung vorgegeben wurde.

4.1 Beurteilung von Yacop und Ausblick

Die Performanz von YACOP im Vergleich zu den einzeln eingesetzten Tools ist hinreichend belegt. Es wird ein hoher Anteil TP geliefert, wobei der Anteil FP auf ein akzeptables Maß reduziert wurde. Die Sensitivität war im Vergleich zur Vorhersage der einzelnen Tools durchschnittlich um bis zu 3% geringer. Dabei war die Spezifität allerdings um bis zu 12% höher. Die Vorhersage der korrekten Startposition konnte im Mittel um bis zu 10% verbessert werden. Im Vergleich zur Schnittmenge der Ausgaben von ZCURVE und GLIMMER, konnte die Spezifität nicht gesteigert werden. Die Vorhersage der korrekten Startpositionen konnte jedoch um bis zu 4% gegenüber der Schnittmenge von ZCURVE und GLIMMER erhöht werden. Durch den routinemäßigen Einsatz von YACOP im G_2L Göttingen, konnte der Aufwand der

Annotatoren deutliche verringert werden.

Weiterhin könnte eine Steigerung der Performanz von YACOP durch zusätzliches Scannen nach Lücken in der Vorhersage erzielt werden. Da in prokaryotischen Genomen 80-90% der Sequenz als codierend angenommen werden, ist es relativ unwahrscheinlich, daß große Bereiche ohne Gene auftreten. Eine mögliche Strategie solche Lücken auszumachen, wäre das erneute Durchmustern der Meta-Vorhersage. Anschließend werden die gesamten Vorhersagen der Tools nach ORFs durchsucht, die bislang nicht von YACOP akzeptiert wurden, um diese Lücken zu füllen. Das Verfahren befindet sich derzeit in der Testphase.
Die korrekte Vorhersage der Startpositionen ist nach wie vor nicht zufriedenstellend gelöst und bedarf der Verbesserung. Hierzu wird ein neuer Ansatz in Aussicht gestellt, bei dem der Translationsstart mit Hilfe von Support Vektor Maschinen (ZIEH *et al.*, 2000) vorhergesagt wird.

Kapitel 5

Zusammenfassung

Ziel der vorliegenden Arbeit war es, verschiedene Genvorhersage-Tools für prokayotische Genome im Hinblick auf ihre Performanz zu testen und durch geeignete Kombination der Ergebnisse die Qualität der Vorhersage zu erhöhen. Als Bewertungskriterien wurden die Parameter Sensitivität (SEN), Spezifität (SPEZ) und Start-Correctness (SC) eingeführt. Mit Hilfe dieser Kriterien wurde die Performanz der Vorhersagen verglichen. Darauf aufbauend wurde eine Kombination der Vorhersagen zusammengestellt, die in dem Programmpaket YACOP implementiert wurde. Als optimale Verknüpfung wurde die gesamte Vorhersage von CRITICA mit der Schnittmenge der Vorhersagen von GLIMMER2.02 und ZCURVE $\geq$ 150 BP ermittelt.

Es konnte gezeigt werden, daß die Problematik der Genvorhersage für Prokaryoten durch die verfügbaren Programme nicht vollständig gelöst ist. Bei den Untersuchungen stimmten bis zu 56% der vorhergesagten Genstarts nicht mit der Annotation überein. Dabei enthielten die Vorhersagen bis zu 20% False-Positive. Die Qualität der Vorhersage für Genome mit einem G/C-Gehalt $> 50\%$ war deutlich schlechter als für Genome mit einem niedrigerem G/C-Gehalt.

Durch die in dieser Arbeit entwickelten bool'sche Kombination der Ausgaben, konnte die Spezifität der Vorhersage signifikant verbessert werden, ohne die Sensitivität zu verringern. Dabei wurde die Vorhersage der Startpositionen um durchschnittlich 20% verbessert. Insbesondere für Genome mit hohem G/C-Gehalt konnte die Qualität der Vorhersagen gesteigert werden. Dennoch bleibt das Auffinden der korrekten Startpositionen problematisch.

Literaturverzeichnis

[1] Altschul S. F., Gish W., Miller W., Myers E. W., Lipman D. J. 1990. Basic local alignment search tool. *Mol. Biol.* **215**:403-410.

[2] Badger J., Olsen G. 1999. CRITICA: Coding Region Identification Tool Invoking Comparative Analysis. *Mol. Biol. Evol.* **16**(4):512-524.

[3] Barrik D., Villanueba K., Childs J., Kalil R., Schneider T.D. 1994. *Nucleic Acids Res.* **22**: 1287-1295.

[4] Borodovsky M., Mcininch D. 1993. GeneMark: Parallel Gene Recognition for both DNA Strands. *Comp. Chem.* **17**:123-133.

[5] Borodovsky M., Mcininch J., Koonin E., Rudd K., Medigue C., Danchin Detection of New Genes in the Bacterial Genome Using Markov Models for Three Gene Classes. *Nucleic Acids Res.* **23**:3554-3562.

[6] Claverie J. M., Bougueleret L. 1986. Heuristic informational analysis of sequences. *Nucleic. Acids Res.* **14**:179-196.

[7] Delcher A., Harmon D., Kasif S., White O., Salzberg S. 1999. Improved microbial gene identification with GLIMMER. *Nucleic Acids Res.* **27**:4636-4641.

[8] Fickett J. W., Tung C. S. 1992. Assessments of Protein coding measures *Nucleic Acids Res.* **20**:6441-6450.

[9] Frishman D., Mewes H.-W. 1997. PEDANTIC genome analysis *Trends Genet.*, **13**:415-416.

[10] Frishman D., Mironov A., Mewes H.-W., Gelfand M. 1998. Combining diverse evidence for gene recognition in completely sequenced bacterial genomes. *Nucleic Acids Res.* **26**:2941-2947.

[11] Gish W., States D. J. 1993. Identification of protein coding regions by database similarity search. *Nat. Genet.* **3**: 266-272.

[12] Guo F.-B., Hou H.-Y., Zhang C.-T. 2000. ZCURVE: a new system for recognizing protein-coding genes in bacterial and archaeal genomes. *Nucleic Acides Res.* **31**: 1780-1789.

[13] Huang X. 1996. *Micorb. Compar. Genomics* **1**: 281-291.

[14] Nakai K., Horton P. 1999. PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends Biochem. Sci.* **24**: 34-36.

[15] Nielsen H., Engelbrecht J., Brunak S., Heijne G. 1997. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. Protein Engineering **10**:1-6.

[16] Salzberg S., Delcher A., Kasif S., White O. 1998. Microbial gene identification using interpolated Markov models. *Nucleic Acids Res.* **26**:544-548.

[17] Schneider T.D., Stormo G.D., Gold L., Ehrenfeucht A. 1986. Information content of binding sites on nucleotide sequences. *J. Mol. Biol.* **188**:415-431.

[18] Shine J., Dalgarno L. 1974. The 3' terminal sequence of *Escherichia coli* 16S ribosomal RNA: complementary to nonsense triplets and ribosom binding sites. *Proc. Natl. Acad. Sci.* USA **71:**1342-1346.

[19] Smith T.F., Waterman M.S. 1981. Identification of common molecular subsequences, *J. Mol. Biol.* **147**: 195-197.

[20] Suzek B.E., Ermolaeva M.D., Schreiber M., Salzberg S. 2001. A probabilistic method for identifying startcodons in bacterial genomes. *Bioinformatics* **17:**1123-1130.

[21] Tatusov R. L. *et al.* 2001. The COG database: new developments in phylogenetic classification of proteins, from complete genomes.*Nucleic Acids Res.* **29**:22-28.

[22] Tompa M., 1999. An exact method for finding short motifes in sequences, with application to the ribosom binding site problem. *Seventh International Conference on Intelligent Systems for Molecular Biology*, S. 262-271.

[23] Varshavsky A. 1996. *Proc. Natl. Acad. Sci. USA* **93**: 12142-12149

[24] Zhang C.-T., Zhang R. 1991. Analysis of distribution of bases in the coding sequences by a diagrammatic technique. *Nucleic Acids Res.* **19**:6313-6317.

[25] Zien A., Ratsch G., Mika S., Schölkopf B., Lengauer T., Müller K.R. 2000. Engineering support vector machine kernels that recognize translation initiation sites, *Bioinformatics* **16**:799-807.

Danksagung

Die letzte Seite dieser Arbeit soll den Personen gelten, die mich durch Leben, Studium und Arbeit während meiner Zeit in Göttingen begleitete haben.
Vorallem danke ich Prof. Gerhard Gottschalk, Prof. Burkhard Morgenstern und Dr. Rainer Merkl, die mir diese Arbeit ermöglichten und mit Geduld und Interesse den Fortgang verfolgten. Mein besonderer Dank gilt Rainer für neue Anregungen, den offenen und ehrlichen Umgang und die Hilfe bei der Wegfindung. Außerdem danke ich meinen Eltern, ohne die mein Studium nicht möglich gewesen wäre, für die Unterstützung und ihre Anteilnahme. Arnim, Heiko und Jarek danke ich für Support in technischen Fragen und für aufmunternde Worte.
Meinen Kollegen von der Prof. Schumann GmbH danke ich für viele gemeinsam verbrachte Stunden beim Arbeiten, Mensa-Essen und diversen anderen "Veranstalltungen". Insbesondere danke ich Jörg und Martin für den Beistand in allen Lebenslagen und die Hilfe beim Korrekturlesen. Meinen Mitbewohnern Bernhard, Uli und Uwe danke ich für jeden "schwarzen Kaffee, Junge, so richtig heiß und lecker". Ebenso möchte ich meinen ehemaligen Mitbewohnerinnen Anna und Merle danken, für die Zeit, die wir während unseres Studiums zusammen verbracht haben, für Merles-Welt, Nudeln mit Mais und all die anderen Kleinigkeiten. Robert und Wiebke, Euch danke ich für gemeinsam Durchlittenes beim Lernen für die Prüfungen und Eure beruhigende Wirkung.
Thomas, Andreas und Moni danke ich für einen anderen Blick auf Göttingen zur richtigen Zeit. Stepi und Matthias danke ich für geistige Anstöße und Lebensweisheiten ;) und jede Menge kontroverse Diskussionen. Meinen Geschwistern Anton und Janina sowie Hugo, Herrn Boll, Chico, Frank, Nadine, Mirco, Micha und Wiebke danke ich für unerschütterliche Freundschaft und Geduld mit mir.
Und ganz besonders danke ich Dir für alles, Nils...