

Bioinformatik

QUASAR

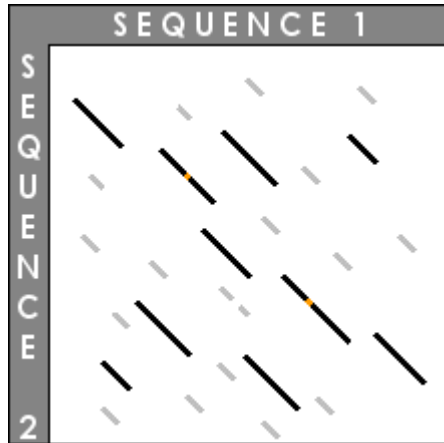
Der Celera Assembler

Ulf Leser

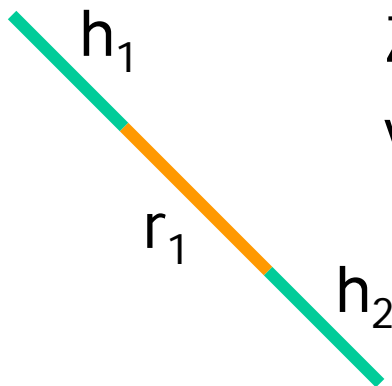
Wissensmanagement in der
Bioinformatik



FASTA - 1. Schritt



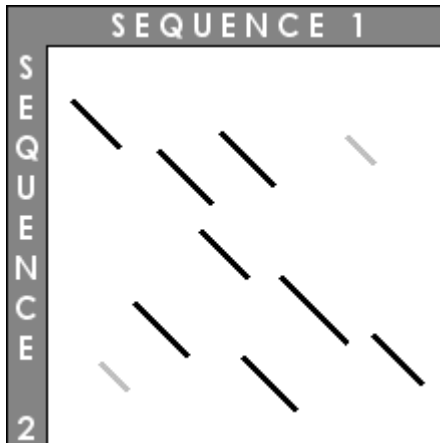
- Bestimme alle exakt Matches der Länge k (*hot-spot*)
 - BLAST lässt hier Mismatches zu, FASTA nicht



Zusammenfassen, falls

$$v(h_1) + v(h_2) + |r_1| \cdot r > \max(v(h_1), v(h_2))$$

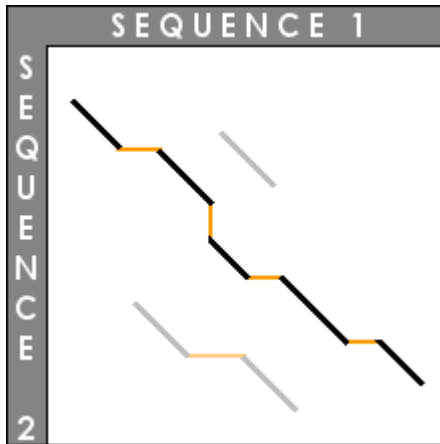
FASTA - 2. Schritt



- Bisher erfolgte Bewertung der Regionen
 - Jede Übereinstimmung zählt e
 - Jedes Mismatch kostet r

- Für jede Region erfolgt nun eine separate Bewertung anhand einer **Substitutionsmatrix**
 - Erst PAM, heute BLOSUM
- Sei **INIT₁** die Region mit der höchsten Bewertung
- Das ist kein echtes Alignment
 - Keine Gaps, nur Bereiche voller Mismatches

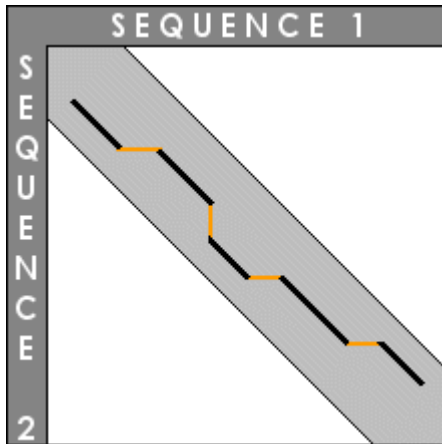
FASTA - 3. Schritt



- Regionen in unterschiedlichen Diagonalen werden zu einem längeren Alignment (mit höherer Bewertung) zusammengefasst
- Entspricht Einfügen von Gaps

- Regionen mit Bewertung unterhalb eines Schwellwerts werden nicht berücksichtigt
- Gaps werden negativ bewertet (linearer Gapscore)
- Der Score der besten Menge von Regionen wird $INIT_n$
 - „Unoptimiert“, also Summe der Regionen minus Penalty für Gaps
- Noch nicht klar, was die beste Menge von Regionen ist
 - Optimierungsproblem

FASTA - 4. Schritt



- Wie gut ist das im Vergleich zum Optimum?
 - Unklar
- Also: Berechnung eines alternativen Alignments

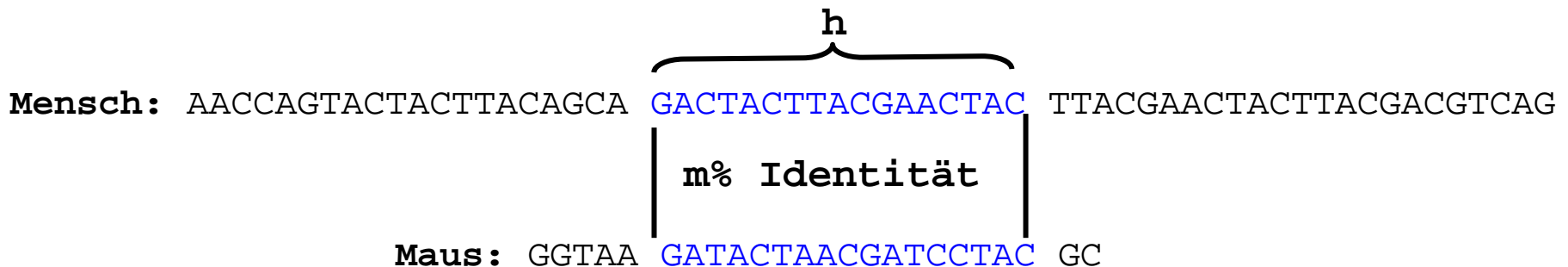
- K-Band Alignment
 - Berechnung rund um $INIT_1$
 - Das war die beste Region
 - $K=16$, also Betrachtung von 32 Diagonalen
 - Optimales Alignment in diesem Band wird **OPT**
- **FASTA Ergebnis**
 - $INIT_1, INIT_n, OPT$

Vergleich

- BLAST und FASTA waren lange gleichberechtigte Konkurrenten
 - Beide sind hochgradig heuristisch
 - Beides sind **Exklusionsmethoden** – Seeds finden und erweitern
 - BLAST ist unstrittig schneller
 - Welches sensitiver ist, ist umstritten
 - FASTA benötigt im Kern perfekte Matches (hot spots)
 - Sehr problematisch für Proteine
 - Unterschiedlicher Score in Schritt 1 und 2 mutet seltsam an
 - Warum eine zweistufige Erweiterung von Seeds?
 - Von Hot Spots zu Regionen zu Bereichen
- BLAST heute dominierend

BLAT Szenario

- Vergleich einer Maus-cDNA Q mit einer humanen cDNAs
- Hintergrundwissen über Menschen und Mäuse
 - Wenn es eine homologe Teilsequenz gibt, wird diese im **Schnitt zu $m\%$ identisch sein und eine durchschnittliche Länge von h** haben
 - Frameshifts (Insertions/Deletions) sehr unwahrscheinlich
- Frage
 - Wie lang müssen Seeds aus Q sein, damit wir mit **sehr hoher Wahrscheinlichkeit mindestens einen Treffer in einer homologen Region finden**, wenn es diese gibt?



BLAT – Statistik

- Ziel: **Minimales k und Anzahl von Hits abschätzen**
 - m: Erwartete Anzahl Matches in zwei Sequenzen
 - Z.B.: 99% für Maus-Mensch cDNAs, 90% für Proteine
 - h: Durchschnittliche Länge homologer Regionen
 - g: Größe der Datenbank (in Basen)
 - q: Länge der Querysequenz
 - a: Größe des Alphabets (DNA oder Protein)
- Berechnung
 - Wahrscheinlichkeit, dass **ein beliebiges k-mer** aus der Suchsequenz mit **seinem Gegenstück** in der homologen Datenbanksequenz perfekt matched
 - $p_1 = m^k$
 - Wahrscheinlichkeit, dass **mindestens ein nicht-überlappendes k-mer** aus T mit dem entsprechenden k-mer in Q perfekt matched (wenn Q,T homolog)
 - $p = 1 - (1 - p_1)^z = 1 - (1 - m^k)^z$

Trefferwahrscheinlichkeiten

Table 3. Sensitivity and Specificity of Single Perfect Nucleotide K-mer Matches as a Search Criterion

	7	8	9	10	11	12	13	14
A. 81%	0.974	0.915	0.833	0.726	0.607	0.486	0.373	0.314
83%	0.988	0.953	0.897	0.815	0.711	0.595	0.478	0.415
85%	0.996	0.978	0.945	0.888	0.808	0.707	0.594	0.532
87%	0.999	0.992	0.975	0.942	0.888	0.811	0.714	0.659
89%	1.000	0.998	0.991	0.976	0.946	0.897	0.824	0.782
91%	1.000	1.000	0.998	0.993	0.981	0.956	0.912	0.886
93%	1.000	1.000	1.000	0.999	0.995	0.987	0.968	0.957
95%	1.000	1.000	1.000	1.000	0.999	0.998	0.994	0.991
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999

- $q=100$, m und k variabel (a und g hier nicht notwendig)
- Werte sind Wahrscheinlichkeit, dass **mindestens ein perfekter Match in der Region** vorkommt
 - Voraussetzung: $q > h$
- Ein Match reicht – Extensionsphase wird die **ganze Region finden**
- Beispiel
 - Bei erwarteter Sequenzähnlichkeit von $M=97\%$ findet man in einer homologen Region T von 100 Basen praktisch immer einen perfekten Match der Länge 13 mit einem Substring von Q

Variante 1 – Hits mit Mismatches

- BLAT kann auch Hits mit **höchstens einem Mismatch** erlauben
 - Wahrscheinlichkeit, dass mindestens ein k-mer in einer homologen Regionen perfekt oder mit einem Mismatch matched
 - $P_1 = k \cdot m^{k-1} \cdot (1-m) + m^k$
 - Restliche Formeln entsprechend
- Ergebnis
 - **Wesentlich längere Seeds** möglich
 - Dafür wird die Suche erschwert (Indizierung kompliziert)

Table 5. Sensitivity and Specificity of Single Near-Perfect (One Mismatch Allowed) Nucleotide K-mer Matches as a Search Criterion

	12	13	14	15	16	17	18	19	20	21	22
A. 81%	0.945	0.880	0.831	0.721	0.657	0.526	0.465	0.408	0.356	0.255	0.218
83%	0.975	0.936	0.904	0.820	0.770	0.649	0.591	0.535	0.480	0.361	0.318
85%	0.991	0.971	0.954	0.900	0.865	0.767	0.719	0.669	0.619	0.490	0.445
87%	0.997	0.990	0.983	0.954	0.935	0.867	0.833	0.796	0.757	0.634	0.591
89%	1.000	0.997	0.995	0.984	0.976	0.939	0.920	0.897	0.872	0.775	0.741
91%	1.000	1.000	0.999	0.996	0.994	0.979	0.971	0.962	0.950	0.890	0.869
93%	1.000	1.000	1.000	0.999	0.999	0.996	0.994	0.991	0.988	0.963	0.954
95%	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999	0.994	0.992
97%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
B. K	12	13	14	15	16	17	18	19	20	21	22
F	275671	68775	17163	4284	1070	267	67	17	4.2	1.0	0.3

Inhalt der Vorlesung

- Quasar
 - Schnellere Hitsuche für BLAST
- Der Celera Assembler
 - Alignment & Assembly in the real world

Quasar

- Burkhardt, S., Crauser, A., Ferragina, P., Lenhof, H.-P., Rivals, E. and Vingron, M. (1999). "q-gram Based Database Searching Using a Suffix Array (QUASAR)". RECOMB 1999.
 - BLAST ist zu langsam
 - Quasar ist auch für Sequenzen mit hoher Ähnlichkeit gemacht (wie BLAT)
 - Benutzt BLAST in der Alignment / Extension-Phase, aber hat schnellere Methode zum Finden der Hotspots

Quasar Grundidee

- Search-Phase sucht Regionen mit Länge w und hoher Ähnlichkeit
 - D.h. Regionen mit Editabstand $< k$
- q -Gramme sind Substrings der Länge q
 - q -Gramme = q -mere
- Dann gilt das **q -Gramm Lemma**
 - Gegeben zwei Sequenzen X, Y der Länge w mit Editabstand $< k$. Dann haben X und Y *mindestens* $t = (w + 1) - (k + 1) * q$ q -Gramme gemeinsam.
- Beweis
 - X und Y besitzen jeweils $(w - q + 1)$ q -Gramme der Länge q
 - Wenn in einem Substring Z in X ein Mismatch mit Y auftaucht, dann ist dieser Mismatch in q q -Grammen enthalten
 - X enthält höchstens k Mismatches zu Y
 - Diese können höchstens $k * q$ q -Gramm-Matches „zerstören“
 - Damit bleiben mindestens $t = (w - q + 1) - k * q = (w + 1) - (k + 1) * q$ perfekt matchende q -Gramme übrig
 - qed.

Beispiel

ACACAGGT
ACAGAGGT

- Editabstand = $k = 1$
- Also müssen mindestens $t = (w + 1) - (k + 1) * q$ q -Gramme identisch sein
 - $w = 8; k = 1$
 - $q = 3: t = 3: ACA, AGG, GGT$
 - $q = 4: t = 1: AGGT$
 - $q = 5: t < 0$

Algorithmus

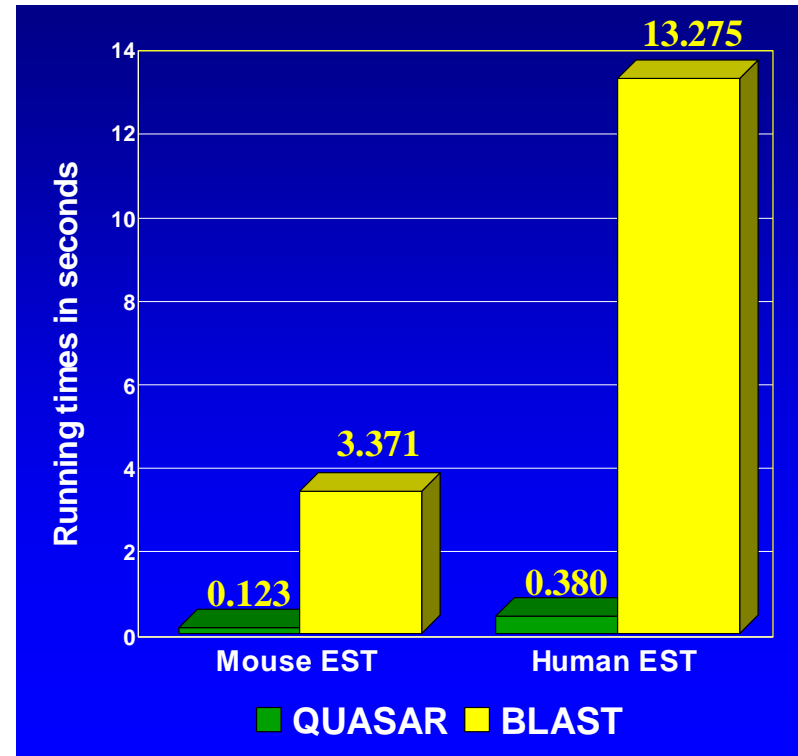
- Quasar verbessert nur die Suchphase von BLAST
 - Gegeben: Gewünschte Seedlänge w , Seedgüte k
 - Berechne t
 - Schiebe ein **Window W der Länge w** über Datenbank D und Query Q
 - Zähle identische q -Gramme in beiden Fenstern
 - Wenn W mehr als t q -Gramme enthält, ist W ein Hit für Blast
 - Diese werden BLAST zur weiteren Verarbeitung übergeben

Tricks & Komplexität

- Beachte: Wir suchen jetzt exakte Hits in der DB – also können wir einen Index aufbauen
- Sei v die maximale Anzahl eines q -Grams in DB
- **Aufbau eines Suffixarrays** für Sequenzen in DB
 - Zeit: $O(|DB|)$
 - Nachteil: Platzbedarf $O(9^* |DB|)$ laut Paper
 - Es gibt $|DB|$ Suffixe; wir brauchen jeweils einen Pointer in die Datenbank
- Aus diesem **Index** für alle q -Gramme der DB erzeugen
 - Suffixarray linear scannen ($O(|DB|)$)
 - Alle Suffixe mit identischem Präfix der Länge q zusammenfassen
 - Test kostet $O(q)$ – man muss nur mit dem Vorgänger vergleichen
 - Zusammen: $O(|DB|^*q)$
- Suche nach allen q -Grammen in einem Fenster der Größe w ist dann konstant pro q -Gramm und damit $O(|Q|^*v)$

Ergebnisse

- Test Parameters
 - 6% Error
 - $w = 50$, $q = 11$
 - scan with BLAST
 - averaged for 1000 queries
- ~30 times faster than BLAST
- Bei etwa gleichbleibender Sensitivität
- Klappt nur für sehr ähnliche Sequenzen
- Nur schnell, wenn der q -Gramm Index komplett in Hauptspeicher passt



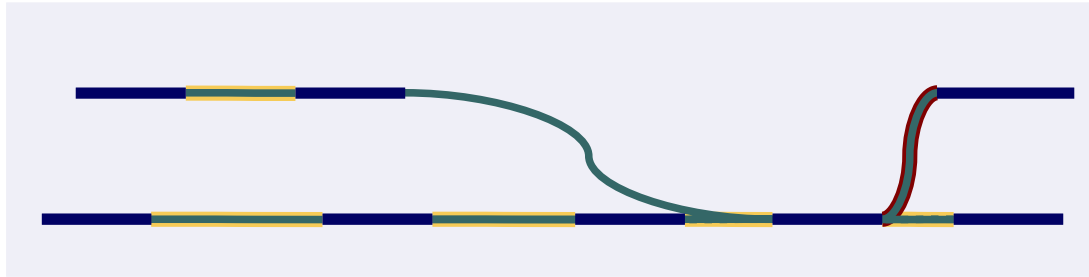
Zusammenfassung

- Engere Eingrenzung des Verwendungszwecks lohnt sich
- Gapped-Blast
 - Weniger Extensions durch Zwei-Hits Strategie
 - Gaps durch spezielle Smith-Waterman Variante
- BLAT
 - Statistik hochähnlicher Sequenzen
 - Schnellerer Algorithmen, gleiche Sensitivität wie BLAST
- Quasar
 - Ähnliche Idee wie BLAT
 - BLAT argumentiert statistisch, Quasar geht auf Nummer sicher

Weitere Varianten

- Locality-sensitive Hashing (Buhler, 2001)
 - Effiziente Suche nach k-meren mit höchstens n Fehlern
- Oasis (VLDB, 2003)
 - Smith-Watermann kombiniert mit A^* -Algorithmus
 - Als nächstes zu expandierende Zellen werden mit A^* Algorithmus gewählt
- Patternhunter, Biohunter (Bioinformatics, 2002)
 - Kommerzielle Software
 - Sucht nicht nach exakten Seeds der Länge k , sondern nach non-consecutive Areas der Länge w mit mindestens k Hits
 - Deutlich schneller als BLAST, bessere Sensitivität
- ... Ein sehr aktives Gebiet

Celera Assembler



- Quellen

- Seminararbeit Alexander Fehr, Christian Brandt
- Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., Kravitz, S. A., Mobarry, C. M., Reinert, K. H., Remington, K. A., et al. (2000). "A whole-genome assembly of Drosophila." *Science* 287(5461): 2196-204.
- Daniel Huson, Uni Tübingen. Vorlesungsskript "Algorithms in Bioinformatics: Genome Assembly", 2004

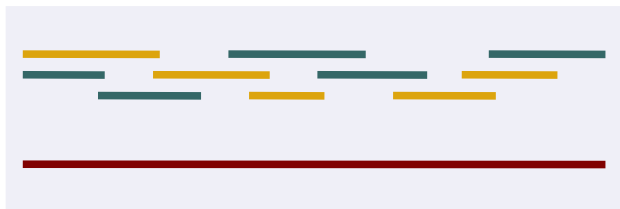
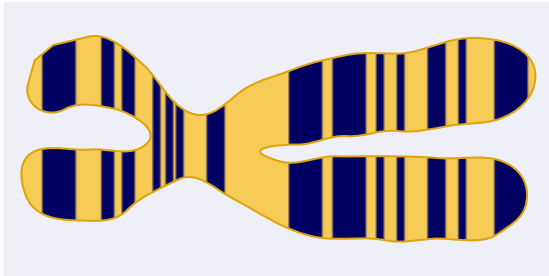
Celera

- 1998 gegründet von Craig Venter und Applera Corporation
- Ziel: vollständige Sequenzierung des menschlichen Genoms in 3 Jahren
- Sequenzierte Genome
 - Drosophila (März 2000)
 - Mensch (Februar 2001)
 - Maus (April 2001)



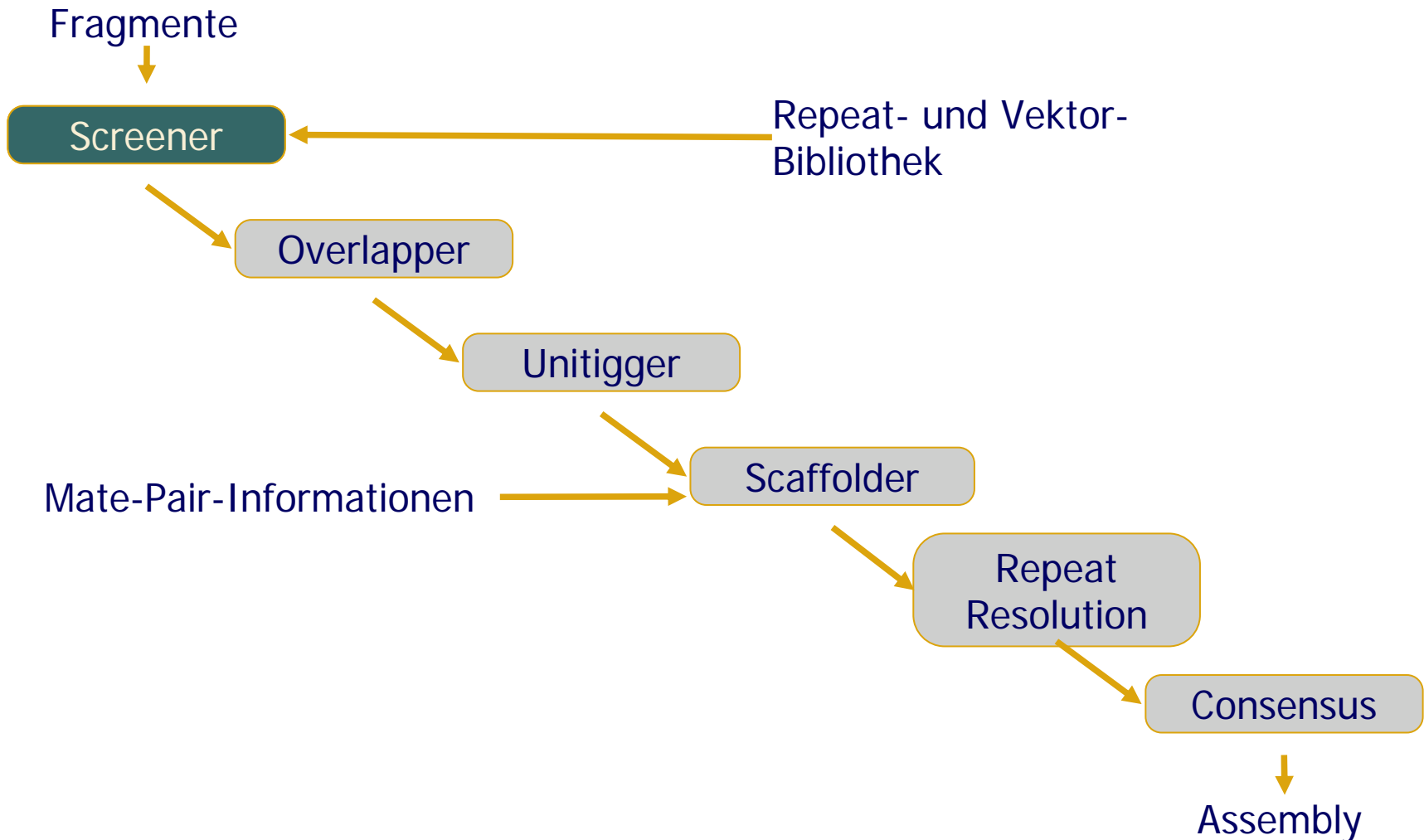
CELERA

Whole Genome Shotgun



- Zerschneiden eines kompletten Chromosoms in Einzelstücke
- (An)sequenzierung jedes Bruchstücks
- **Assembly** der Einzelsequenzen zur Konsensussequenz
- Konkret
 - Clone der Länge 2kb, 10kb, 50kb und 150kb
 - "Double Barrel" Shotgun
 - **Ansequenzieren** von beiden Seiten

Assembler - Überblick



Overlapper

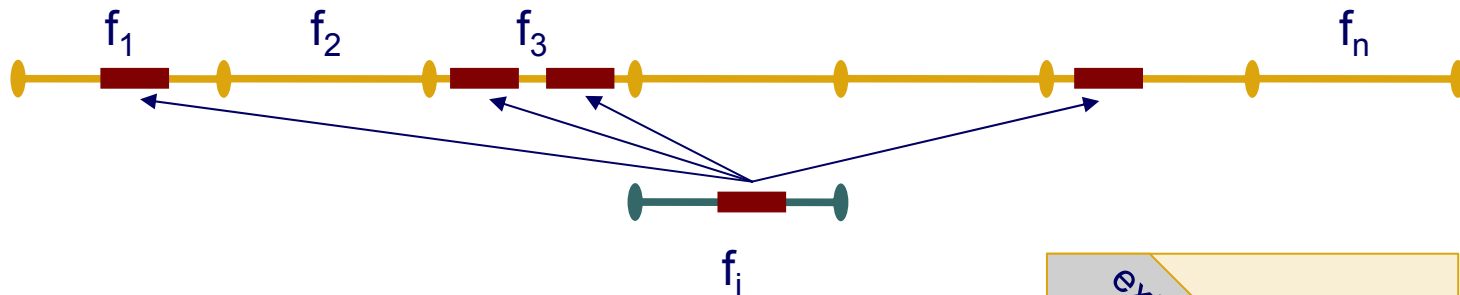
- Suche nach Überlappungen zwischen Fragmenten
- Möglichkeiten wie Fragmente überlappen können:



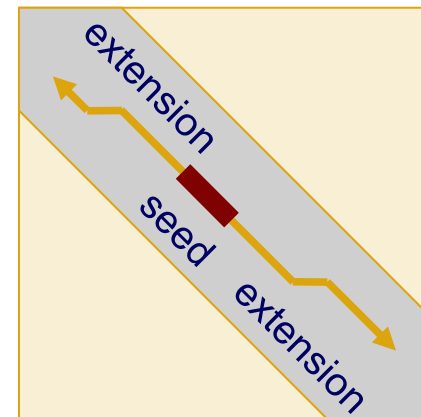
- Dynamische Programmierung?
 - Zu langsam
 - Nur high-scoring Overlaps relevant

Seed and Extend

- "Seed and Extend" Ansatz (ähnlich BLAST)
 - Konkatination aller Fragmente
 - Seeds: Suche exakte Matches der k-mere von f_i auf konkatenierter Sequenz

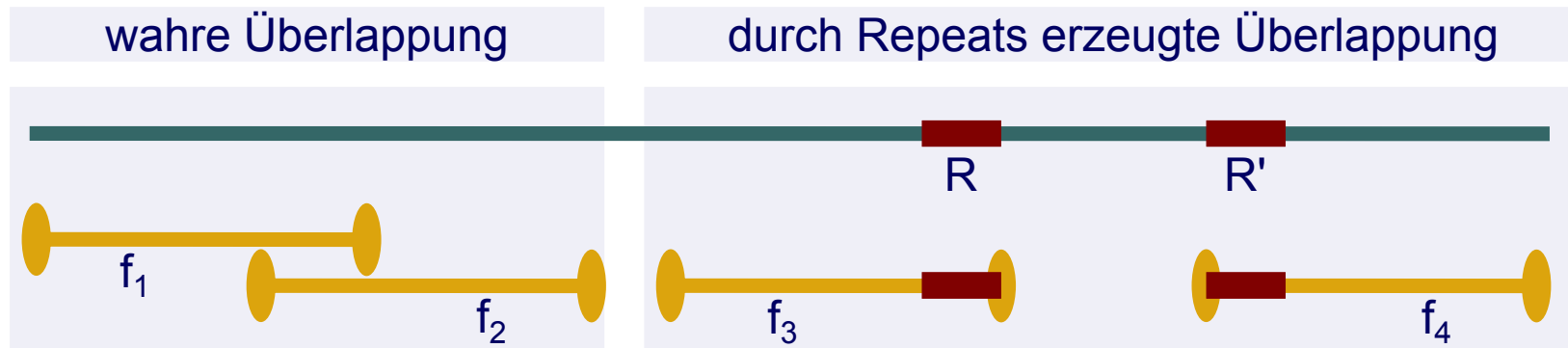


- Extension: Erweiterung der Seeds durch Banded-Alignment



Repeats

- Repeat-induced Overlaps



- Vermeiden durch
 - Screening bekannter Repeats
 - Sehr häufige k-mere ignorieren
- **Repeats bleiben das Hauptproblem des Assemblies**
 - Führen zu falsch-positiven Überlappungen
 - Bilden Verbindungen über das gesamte Chromosom
 - Zerstören/Überlagern die wahre Anordnung

Overlap Graph

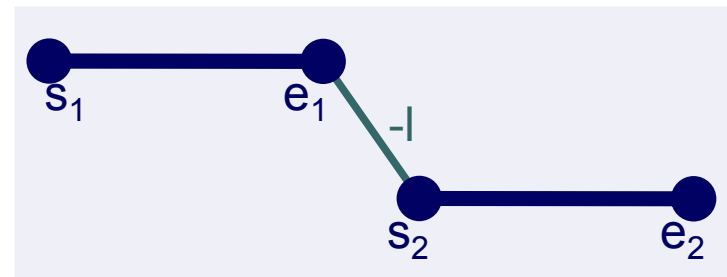
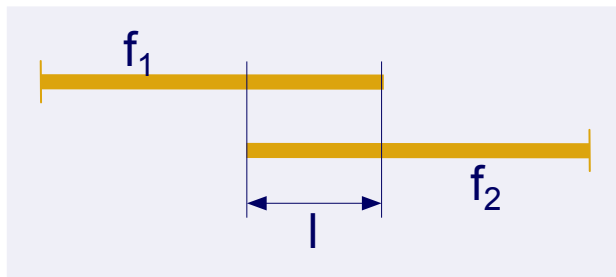
- Overlap Graph

- Knoten

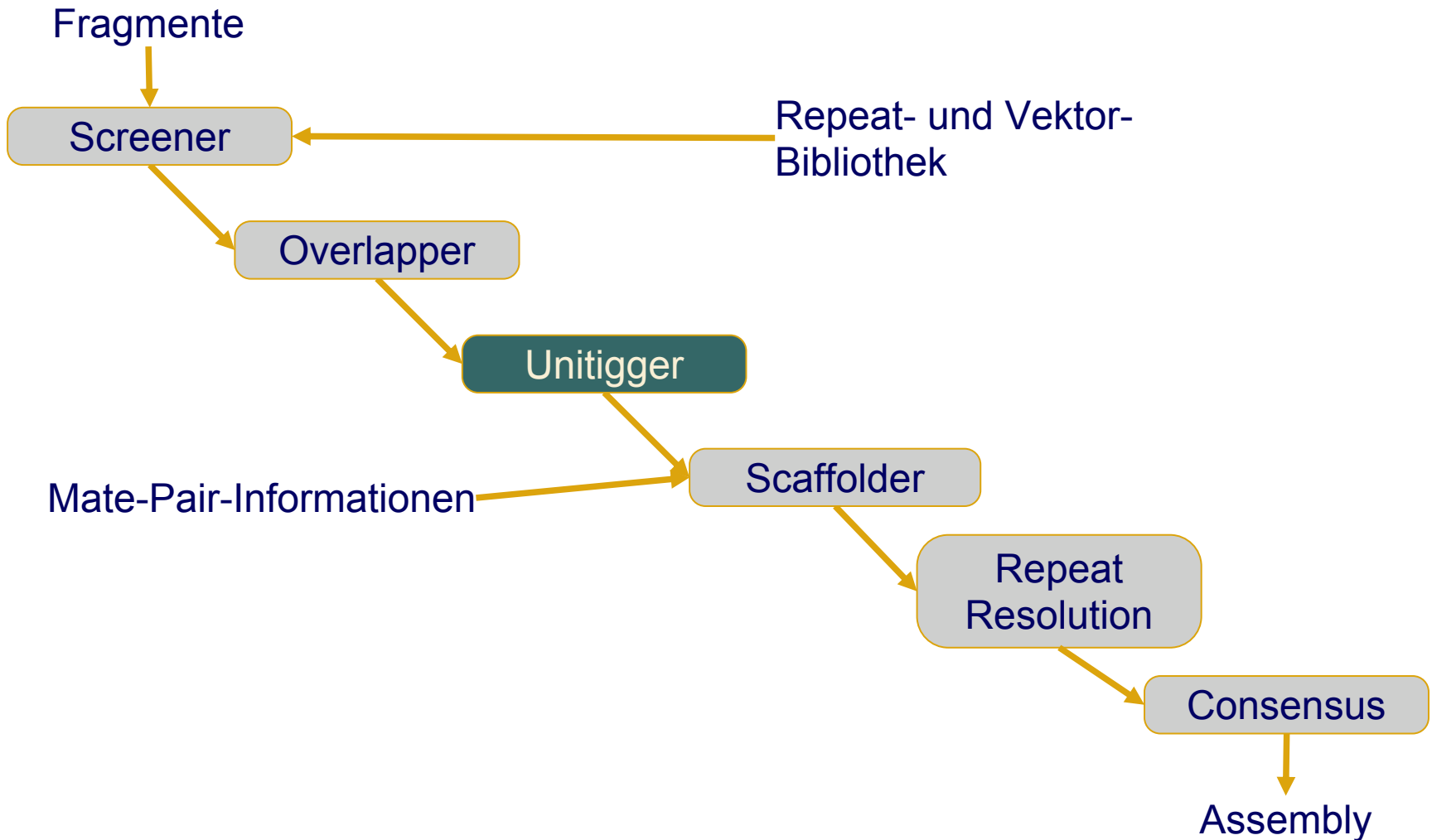
- s_i und e_i (Start- und Endknoten von Fragment f_i)

- Kanten für

- jedes Fragment f_i
 - jede Überlappung

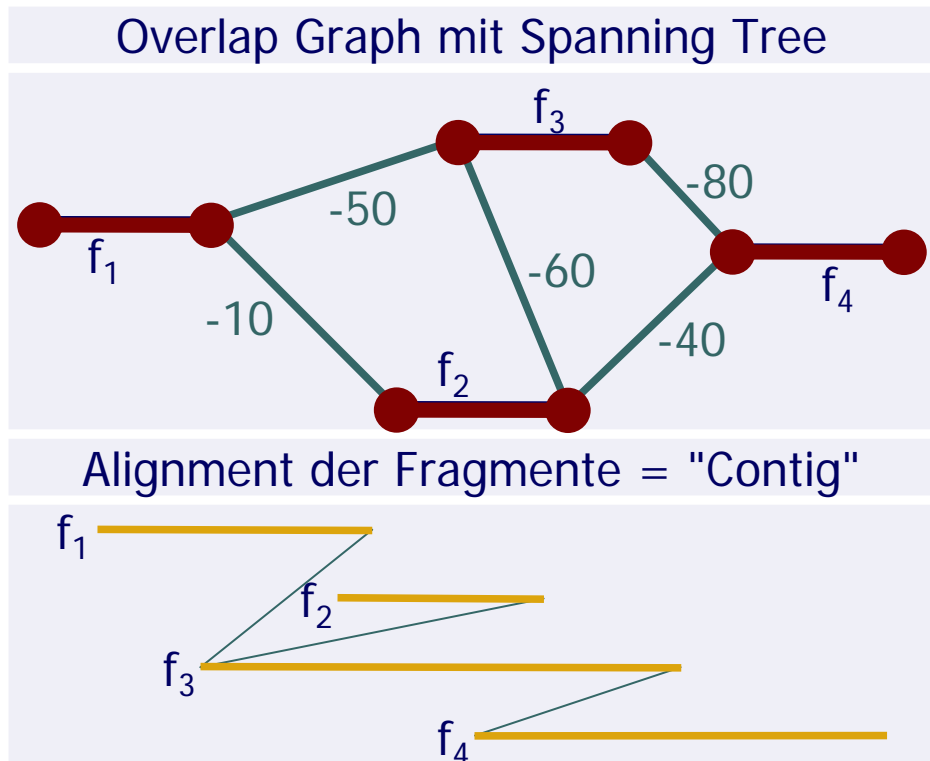


Überblick

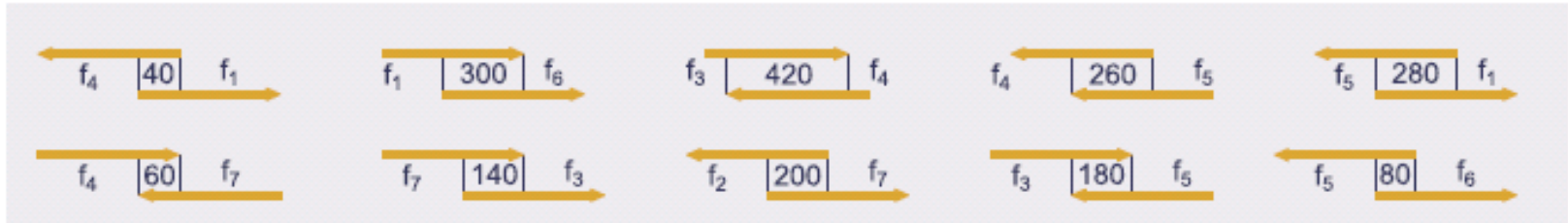


Unitigs

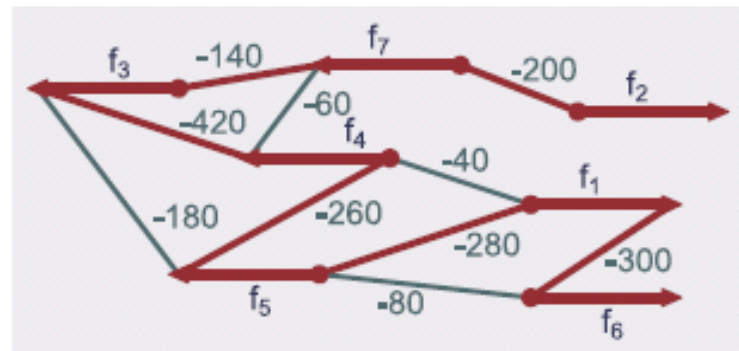
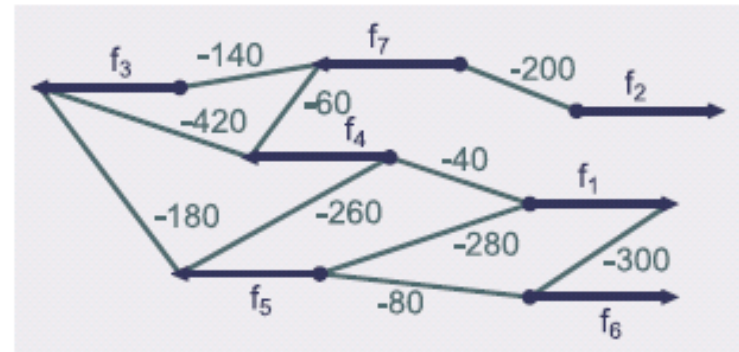
- Verbindet Fragmente zu Unitigs
 - „Unique Contigs“
- Einfache Heuristik:
Spannender Wald
 - Greedy Verfahren
 - „Leichte“ Kanten zuerst
 - Kanten, die Zyklen erzeugen, ignorieren
 - So lange, bis jede Zusammenhangskomponente abgedeckt ist



Beispiel

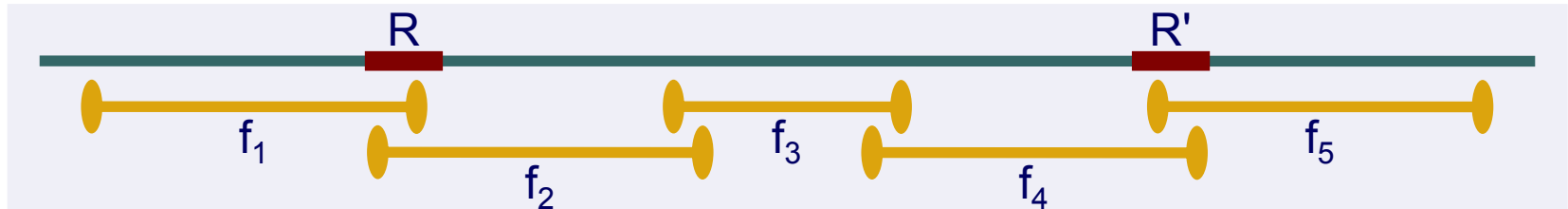


- Overlap Graph bilden
 - Nicht alle Kanten haben Richtung
- Kanten sortieren und Greedy markieren
 - -180 würde Kreis schliessen
 - Nach -140 Terminierung – danach nur noch Kreise

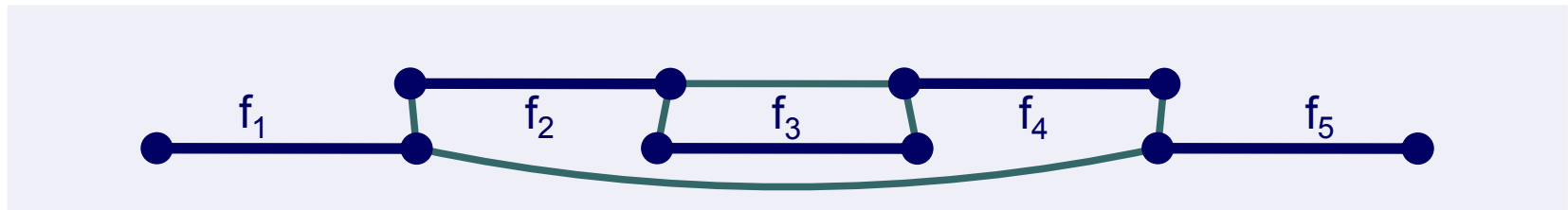


Repeats und Unitigs

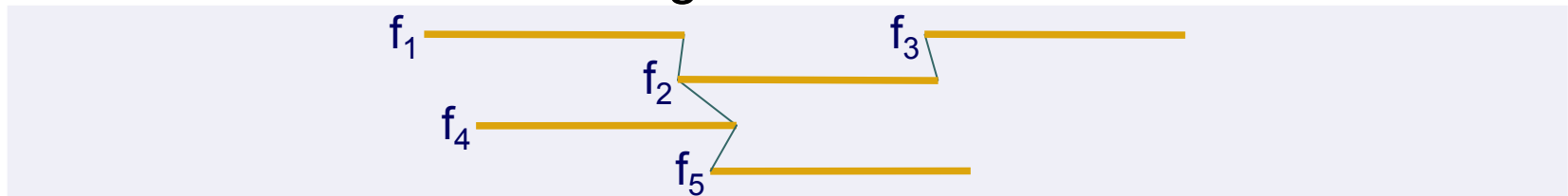
- Repeats



– Erzeugen "falsche" Overlap-Kanten



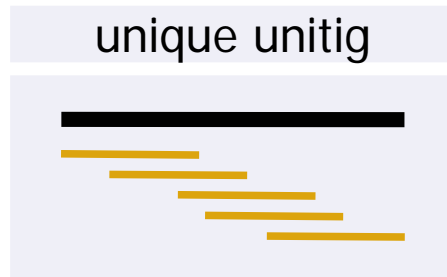
– Misassembled Contig



→ Kein konsistentes Layout möglich

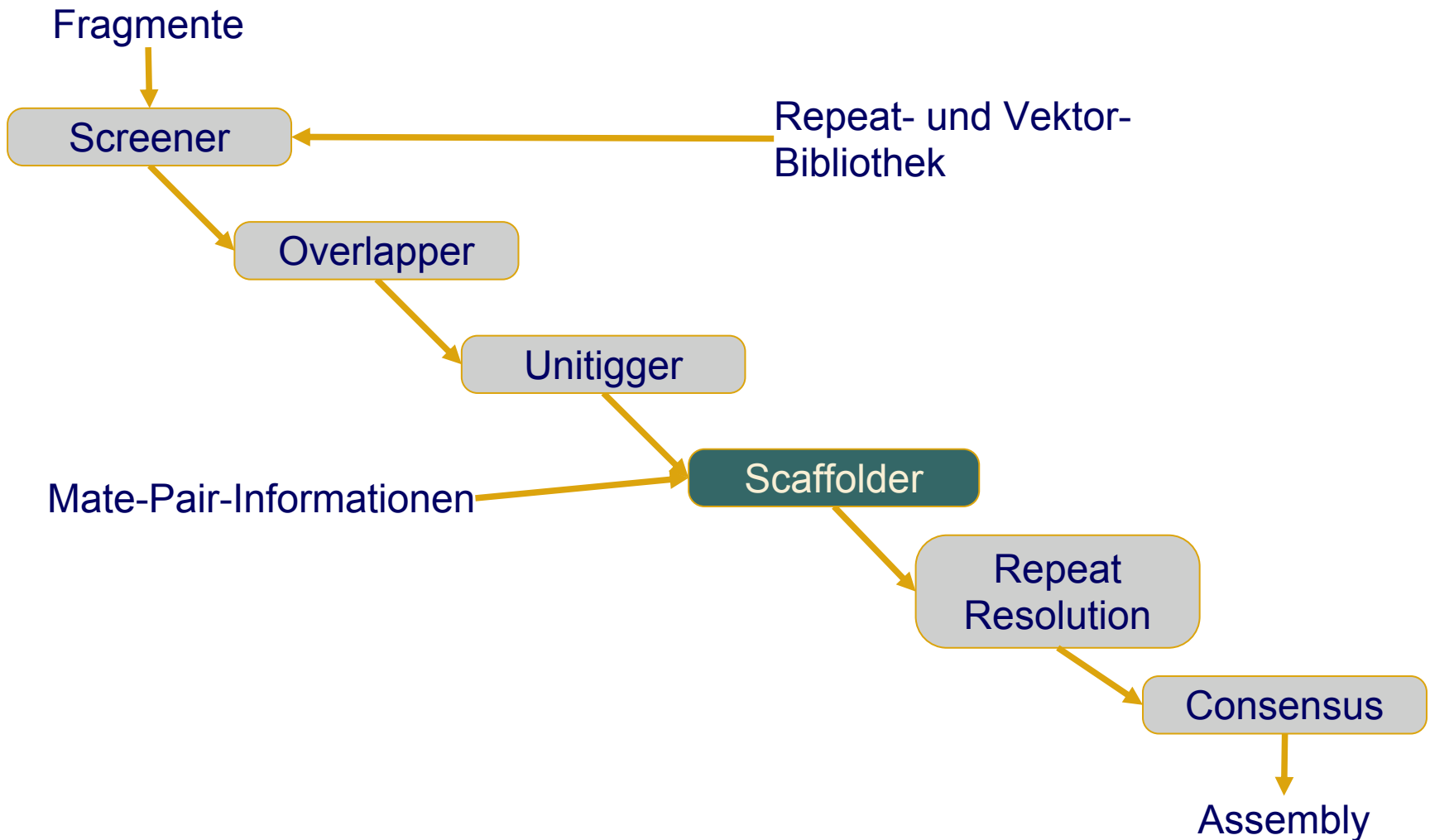
Unique Unitigs

- Unitigs (unique contigs)
 - Mit Overlap Graph konsistente Contigs
- U-unitigs (unique unitigs)
 - Einzigartige Sequenz (kein Repeat)
 - Nicht oversampled (Hinweis auf Repeat)



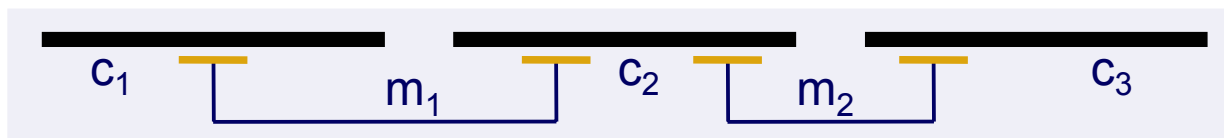
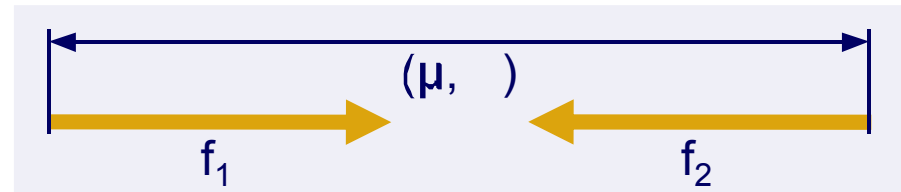
- Vergleich erwarteter Überlagerung mit beobachteter Überlagerung

Überblick



Scaffolder

- Verbindet Unitigs zu Scaffolds
 - Gerüst für ein komplettes Chromosom
- Verwendet Unitigs und **Mate-Pairs**
 - "Double Barrel" Shotgun:
Paare von Fragmenten mit bekannter Orientierung, Distanz μ und Standardabweichung σ
- Scaffold
 - Durch Mate-Pairs verbundene Contigs



Sich bestätigende Mate-Pairs

- Abschätzung der Abstände zwischen Contigs
 - Berechnung über Distanzen der Mate-Pairs
 - Mit kombinierter Standardabweichung
- Mate-Pairs **bestätigen sich**, wenn sie
 - Gleiche Orientierung und
 - ähnlichen Abstand (3σ) der Contigs zur Folge haben
- Signifikanz von Mate-Pairs
 - Einzelnes Mate-Pair zwischen 2 Contigs: unerheblich
 - Rauschen, Zufall, Fehler, etc.
 - **Mehrere sich bestätigende Mate-Pairs zwischen 2 U-Unitigs**: verlässliches Scaffolding

Contig-Mate Graph

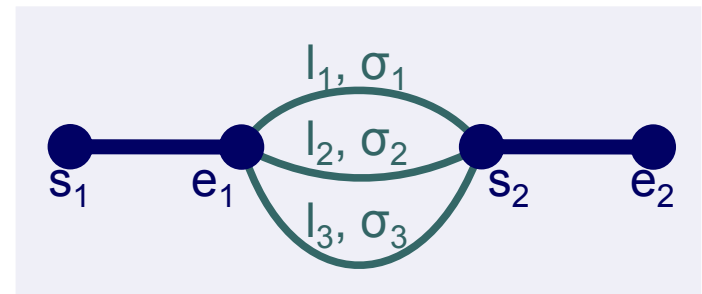
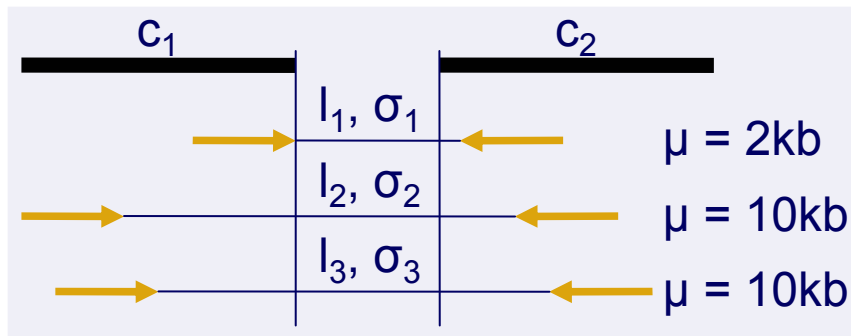
- Contig-Mate-Graph

- Knoten:

- s_i und e_i (Start- und Endknoten von Contig c_i)

- Kanten für

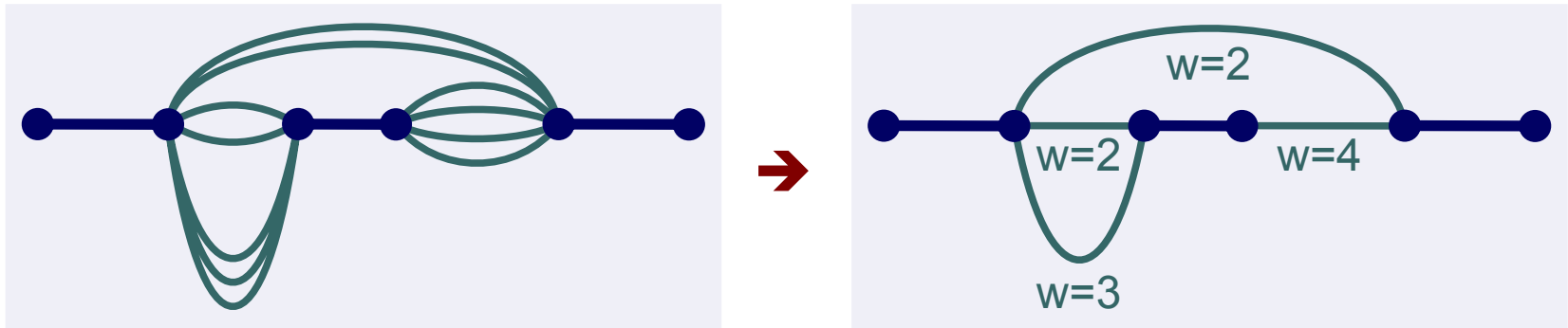
- jedes Contig c_i (zwischen Start- und Endknoten)
- jedes Mate-Pair



Vereinfachung

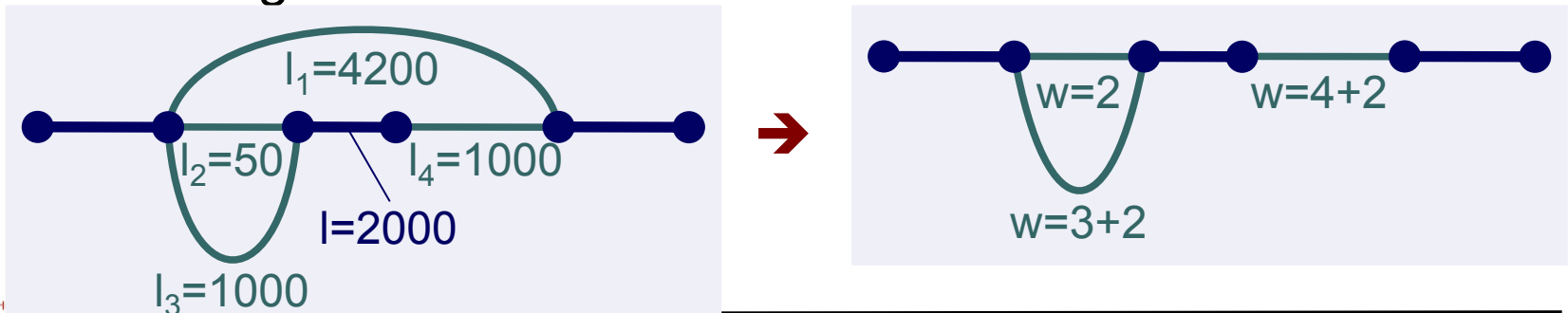
- Edge-Bundling

- Heuristisch sich bestätigende Mate-Kanten verschmelzen



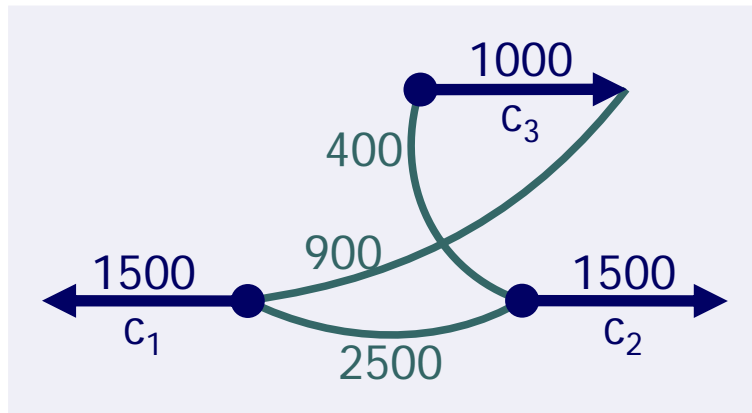
- Transitive **Kantenreduktion**

- Lange Mate-Kanten auf Pfade reduzieren

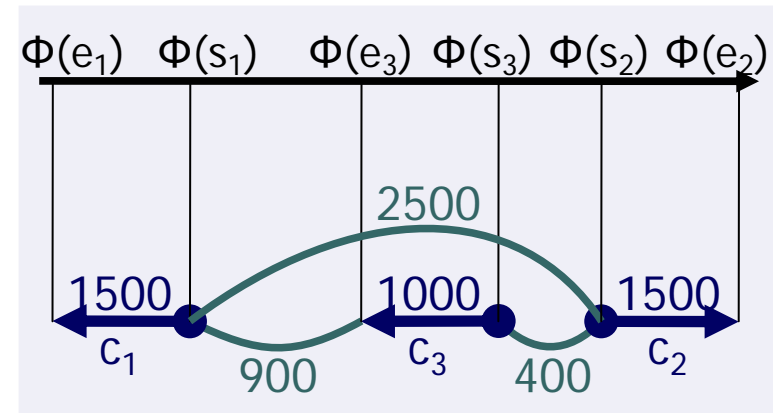


Happy Mate-pairs

- Koordinatenberechnung für Knoten
 - Reihenfolge und Orientierung der Contigs im Graph
 - Erhaltung der Längen und Abstände



è

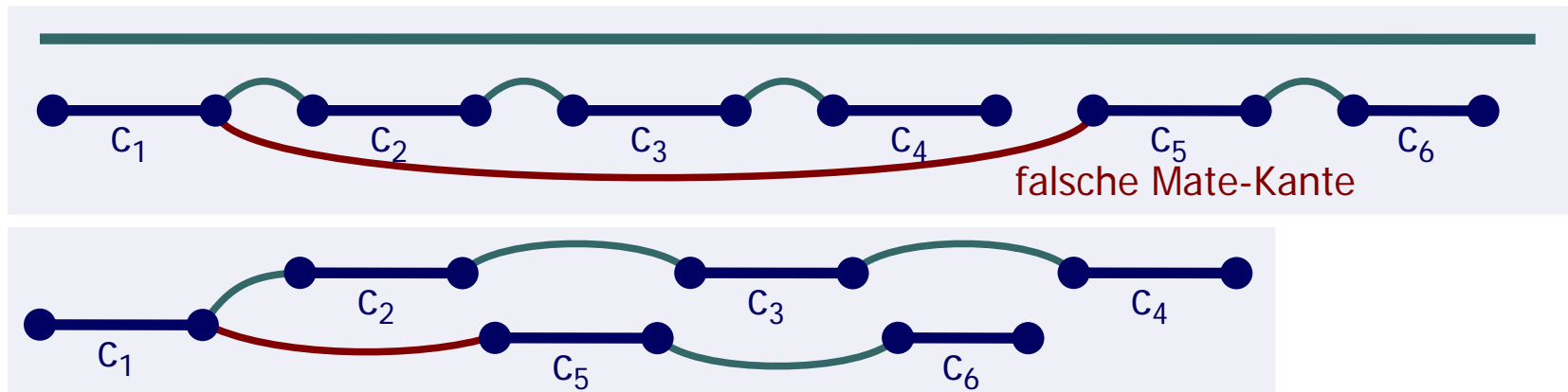


Happy Mate-Pairs

- Richtige Orientierung, Länge, Abstand bezogen auf Φ
- Wir suchen Anordnung Φ so, dass **möglichst viele „wichtige“ Mate-pairs happy sind**

Contig-Mate-Pair Ordering Problem

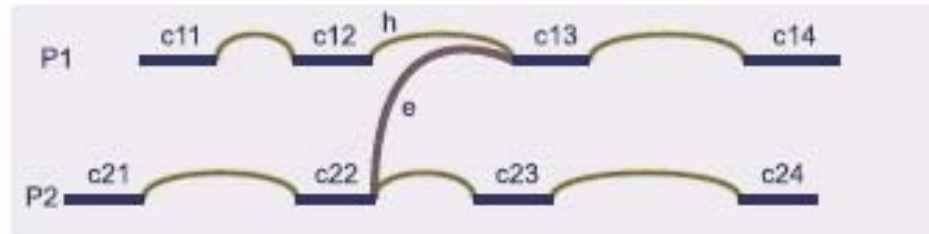
- Gesucht: Reihenfolge der Contigs, die die Summe der Mate-Gewichte maximiert
- Natürlich NP-vollständig
- Greedy Spanning-Tree Heuristik funktioniert nicht
 - Falsche Mate-Kanten führen zu fehlerhafter Verschränkung von Contigs - schlecht



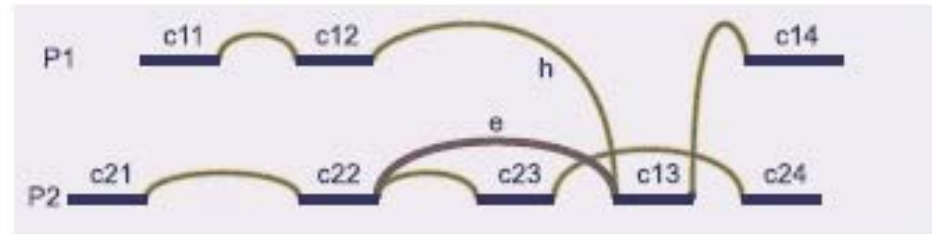
Greedy Path-Merging Algorithmus

- Verwendung einer anderen Heuristik
- Greedy Path-Merging Algorithmus
 - Markiere alle Contigkanten
 - Sortiere Mate-Kanten absteigend nach Gewicht
 - Für alle Kanten $e = (v,w)$
 - Markiere e
 - Seien P_1 / P_2 zu v / w inzidente Pfade
 - Kann man sie mergen und dabei kommen mehr Happy- als Unhappy-Edges hinzu
 - Ersetze P_1 & P_2 durch neuen Pfad P
 - Entferne alle nicht markierten Kanten
- Ergebnis: Φ
 - Enthält immer noch unhappy Kanten

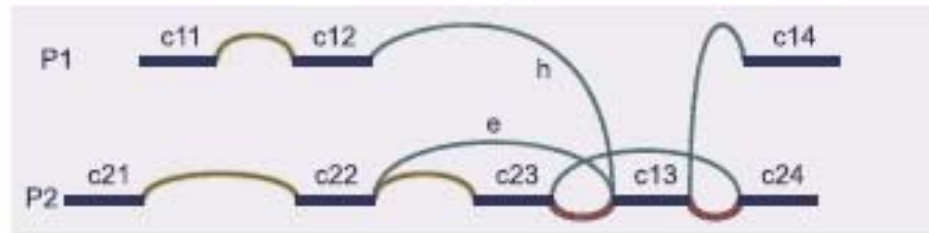
Beispiel



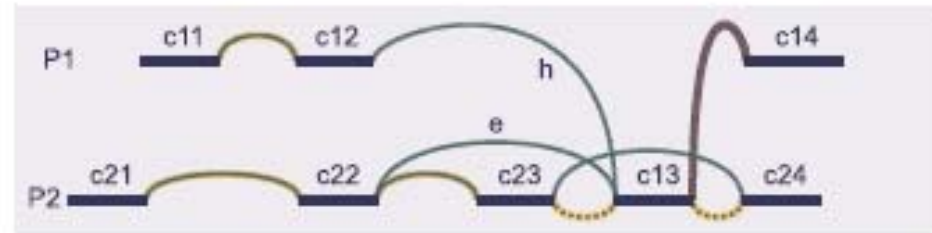
Die Länge der Kante e suggeriert, daß c_{13} zwischen c_{23} und c_{24} liegt.



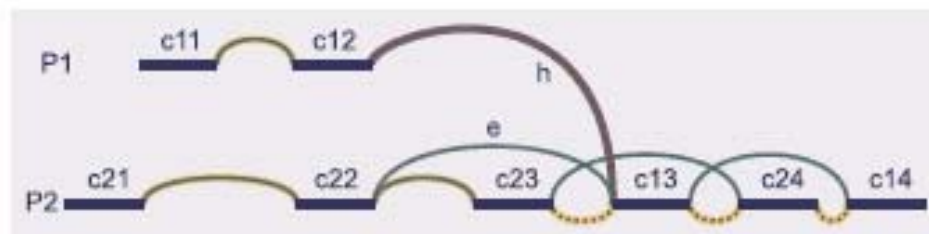
Contig c_{13} passt zwischen c_{23} und c_{24} .



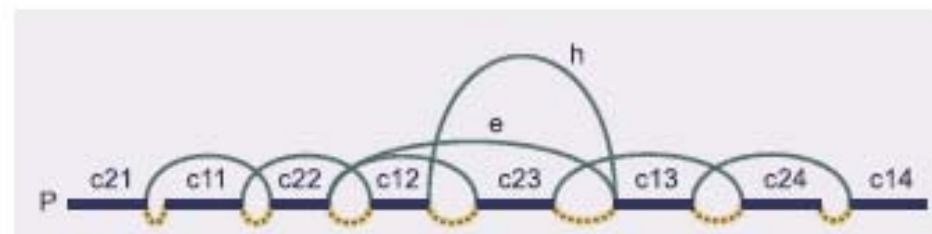
„Gefolgerte“ Kanten werden eingeführt. Sie verbinden c_{13} mit seinen neuen Nachbarn.



Den neuen Kanten wird l und σ zugewiesen. Es geht weiter mit der markierten Kante.

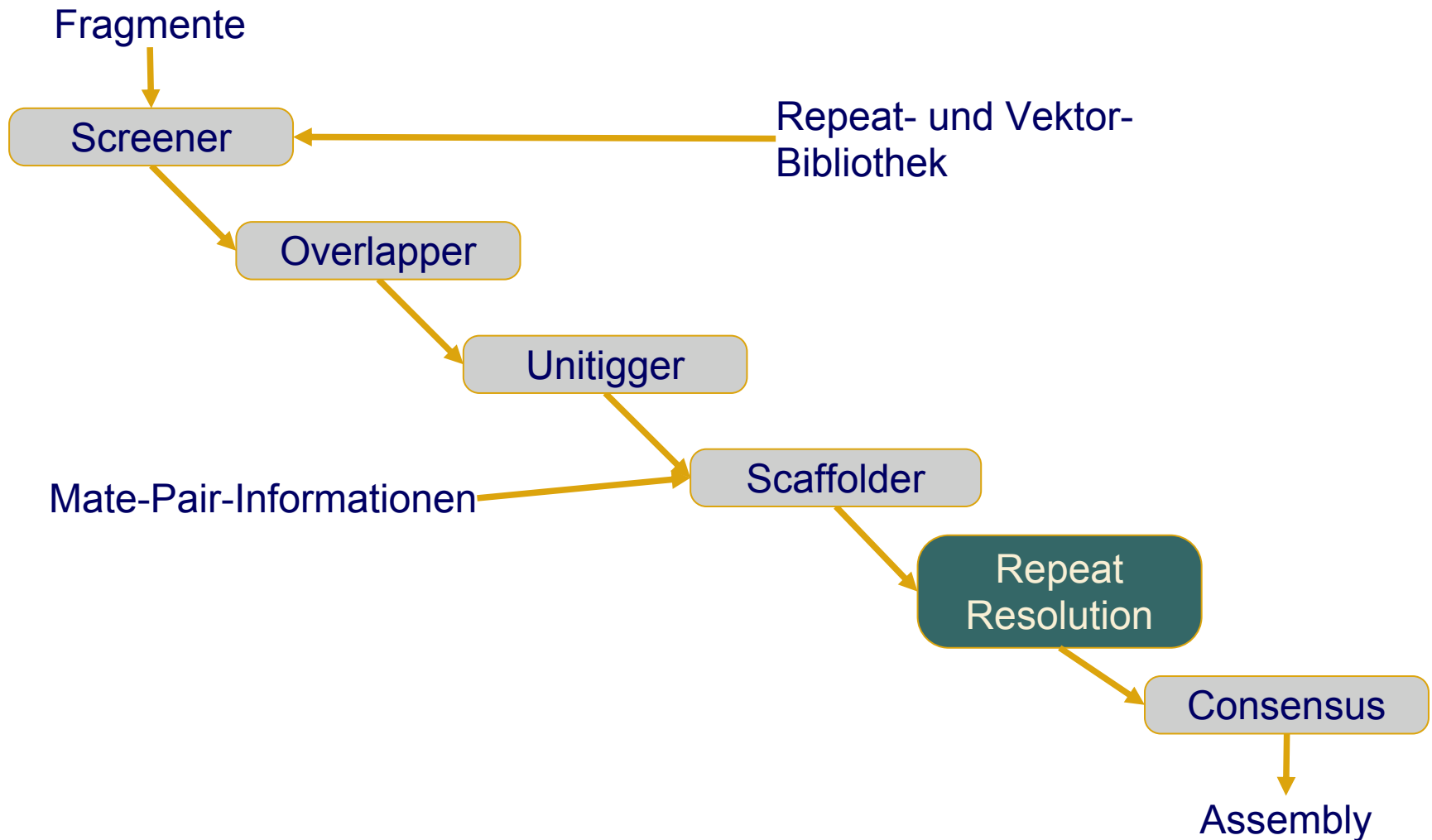


Am rechten Ende angekommen, wird jetzt versucht die Pfade mit Hilfe von Kante h nach links zu verschränken.



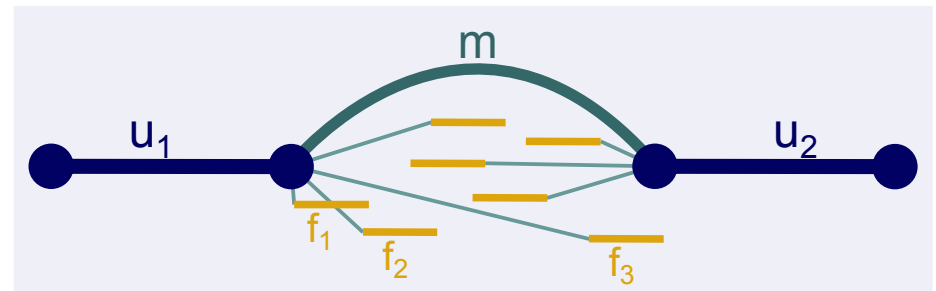
Das Reißverschlußverfahren war erfolgreich. Die Contigs aus P_1 und P_2 wurden im neuen Pfad P integriert.

Überblick

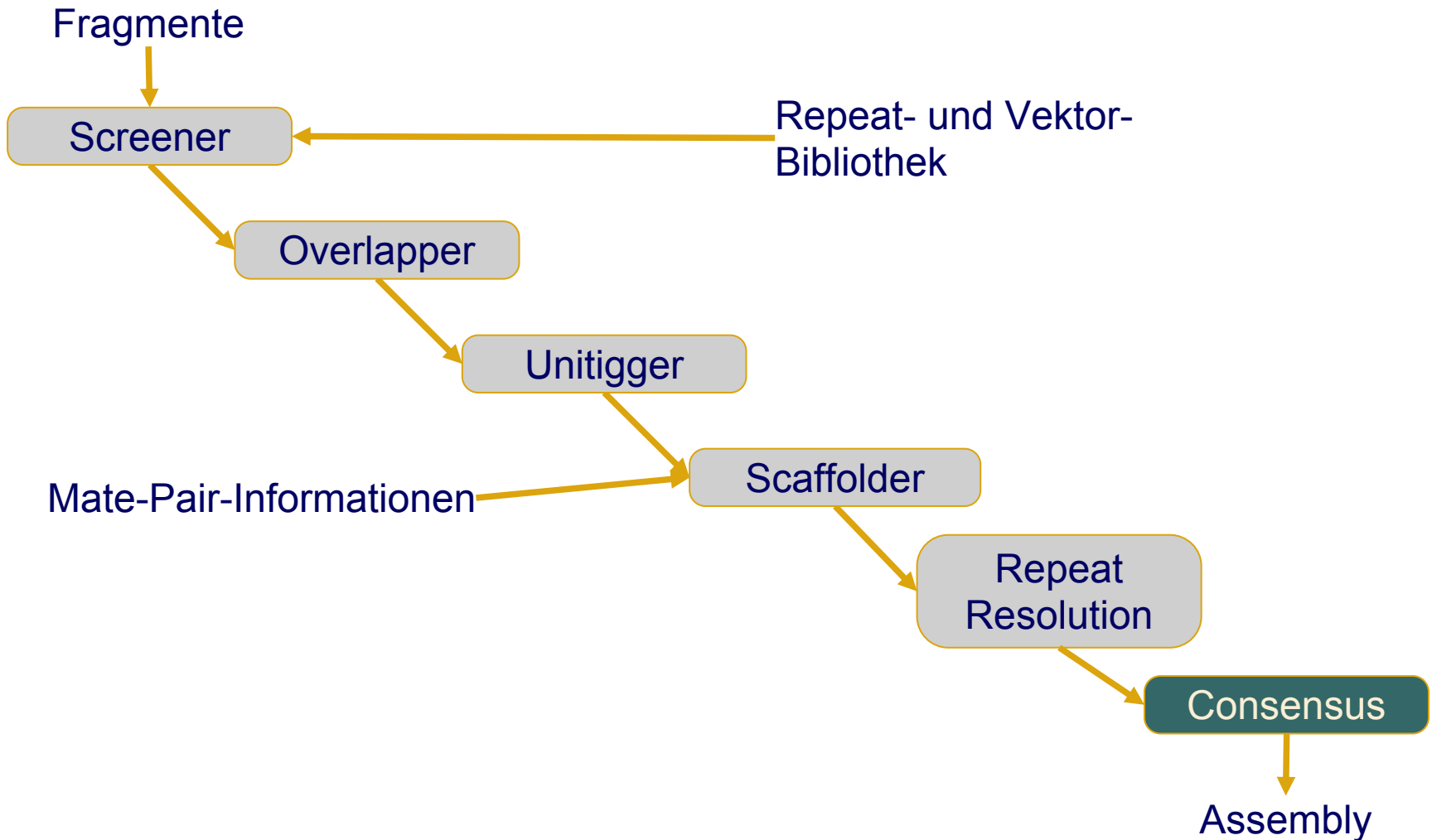


Repeat Resolution

- Versucht Gaps zu schließen
- Input:
 - Scaffolds
 - Fragmente
 - Mate-Pairs
 - Überlappungen
- Output:
 - "Bessere" Scaffolds
- Heuristisches Füllen der Lücke zwischen 2 Unitigs mit bisher nicht zugeordneten Fragmenten
- Bei Erfolg: Verschmelzen der Unitigs



Überblick



Consensus

- Ermittelt Konsensussequenz für einzelne Contigs
- Input:
 - Scaffolds
- Output:
 - Konsensussequenz
- Multialignment (Heuristik)

f_1	CAACTACAAGA-CTTCATGGC
f_2	TCCAAGAACTTAATGGCAAAT-TTC
f_3	ATTC-ACTAC
f_4	TAATAGCAAATCTTCGAAAAACC
Consensus	ATTCAACTACAAGAACTTAATGGCAAATCTTCGAAAAACC

Zusammenfassung

- Overlap Graph ([Seed-and-Extend](#))
- Konstruiere Unitigs (Spanning-Forest [Heuristik](#))
- Scaffold Unitigs ([Greedy](#) Path-Merging)
- Repeat Resolution (diverse [Heuristiken](#))
- Gesamtsequenz (Multialignment)

Menschliches Genom

- 27 Mio Fragmente, Ø-Länge: 550b, 70% Mate-Pairs

	CPU-Stunden		Max. Speicher
Screeener	4800	2- 3 Tage auf 10- 20 Computer	2 GB
Overlapper	12000	10 Tage auf 10- 20 Computern	4 GB
Unitigger	120	4- 5 Tage auf 1 Computer	32 GB
Scaffolder	120	4- 5 Tage auf 1 Computer	32 GB
Repeat Res	50	2 Tage auf 1 Computer	32 GB
Consensus	160	1 Tag auf 10- 20 Computern	2 GB
Total	18000		

- Ergebnis: **Assembly besteht aus 6500 Scaffolds**
 - Umfasst 2,8 Gb Sequenz
 - **150.000 Gaps** (insgesamt 148 Mb)