

Bioinformatik

Der Celera Assembler

Ulf Leser

Wissensmanagement in der
Bioinformatik



Genome Browser 1

Genomes Blat Tables Gene Sorter PCR DNA Convert Ensembl NCBI PDF/P

UCSC Genome Browser on Human May 2004 Assembly

move <<< << < > >> >>> zoom in 1.5x 3x 10x base zoom out 1.5x 3x 10x

position/search chr7:127,115,583-127,851,332 jump clear size 735,750 bp. configure

chr7 (q32.1)

chr7: 127300000 | 127400000 | 127500000 | 127600000 | 127700000 | 127800000

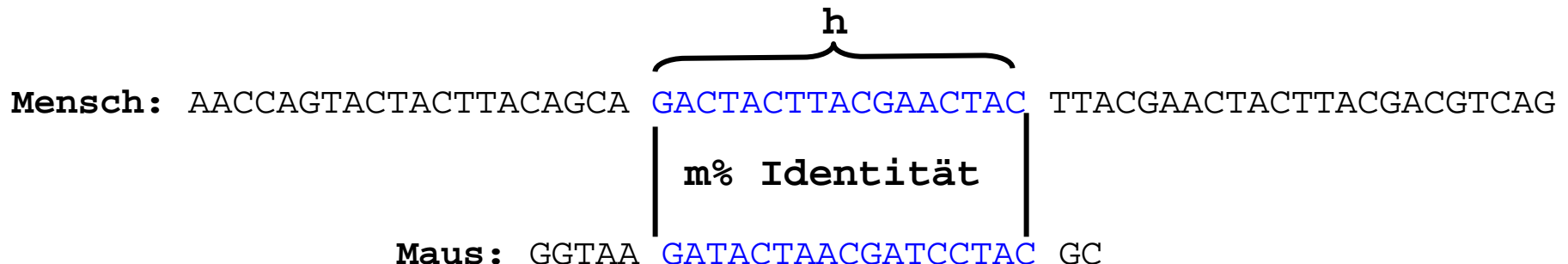
STS Markers
Gap
UCSC Known Genes (June, 05) Based on UniProt, RefSeq, and GenBank mRNA
SND1
BC017180
AF194537 | LRRC4 |
NRG8 |
LRRC4 |
LEP
BC024231 | IMPDH1
RBM28
IMPDH1
AK131294
IMPDH1
BC064929
HIG2 |
RefSeq Genes
ExonWalk
U22055
BC017180
AK096563
AB209510
BT009785
CR594791
G65703
AK130615
AK123730 | G30882 |
AF191492 | BC020423 |
AK024176 | G12061 |
BC112141 | CR614491 |
BC093748 | G30803 |
AF095667 | AY927474 |
BC036685 |
AK172751 |
AJ297858 |
AY358307 |
LEP
BC024231 | IMPDH1
RBM28
IMPDH1
AK131294
IMPDH1
BC064929
HIG2 |
Human mRNAs from GenBank
U43653 | BC063092 | AF144755 |
BC060630 | CR601604 | G30697 |
BC069323 | CR603273 | CR624903 |
BC069452 | CR591535 | CR624882 |
BC069527 | AK054640 | G30729 |
AF008123 | AK092452 | AY358237 |
D49487 | AK122994 | AF066194 |
U18915 | AK131294 |
CR936803 | J05272 |
AK001384 | DQ067456 |
BC024231 | CR933672 |
CR604729 | AK024729 |
AK001239 | AK222583 |
CR594156 | BC008573 |
AK222716 | BC001863 |
BC013689 | CR623904 |
CR596407 | BC112183 |
CR603937 | AK125539 |

Motivation

- Warum sehr schnelle Alignments? Viel gegen Viel
 - Transcript Mapping
 - $2 \cdot 10^9$ humane EST-Basen gegen das humane Genom ($2.9 \cdot 10^9$ Basen)
 - Comparative Genomics
 - $1.7 \cdot 10^6$ Maus-Reads gegen das humane Genom
 - Alle Maus-Human ESTs miteinander verglichen
- BLAST ist nicht schnell genug
 - Neue Daten kommen schneller, als Alignments berechnet werden
- BLAT ist **500-mal schneller als BLAST** und gleich sensitiv für **sehr ähnliche Sequenzen**
 - Kann nur hoch konservierte Sequenzen finden
 - Das erlaubt deutlich höhere Anforderungen an Seeds und Gaps

BLAT Szenario

- Vergleich einer Maus-cDNA Q mit einer humanen cDNA P
- Hintergrundwissen über Menschen und Mäuse
 - Wenn es eine homologe Sequenz in der Maus zu einer Sequenz im Menschen gibt, wird diese zu $\sim m\%$ 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 mit P finden, wenn Q und P homolog sind?



BLAT – Statistik

- Ziel: **Maximales k so, dass sehr wahrscheinlich nichts verloren geht**
 - 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 einer 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

Falsch-positive Treffer

- Wie viele k-mere **matchen zufällig**?
 - Abhängig von g , q und a
 - $F = (q-k+1) * (g/k) * (1/a)^k$
 - $(1/a)^k$: Alle Zeichen eines k-mers matchen per Zufall
 - (g/k) : Anzahl nicht-überlappender k-mere in DB
 - $(q-k+1)$: Anzahl (überlappender) k-mere in Query

B. K	7	8	9	10	11	12	13	14
F	1.3e+07	2.9e+06	635783	143051	32512	7451	1719	399

$$a=4, g=3*10^9, q=500$$

- Falsch-positive werden in **Extensionsphase** ausgesiebt
- **Trade-Off**
 - Hohe k-Werte: Wenig falsch-positive, aber eventuell fehlende echte Hits
 - Niedrige k-Werte: Viele falsch-positive, aber weniger falsch-negative

Variante 1 – Hits mit Mismatches

- Suche nach Hits mit **höchstens einem Mismatch**
 - Wahrscheinlichkeit, dass ein gegebenes k-mer in einer homologen Region 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

Suchvariante 2 – mehrere Hits

- Wahrscheinlichkeit, dass man beim Vergleich **n oder mehr perfekte Matches** (der Länge k) findet
 - $p = p_n + p_{n+1} + \dots + p_z$
- Vorteil: Drastische Verringerung der erwarteten Anzahl falsch-positiver Hits

Table 7. Sensitivity and Specificity of Multiple (2 and 3) Perfect Nucleotide K-mer Matches as a Search Criterion

	2,8	2,9	2,10	2,11	2,12	3,8	3,9	3,10	3,11	3,12
A. 81%	0.681	0.508	0.348	0.220	0.129	0.389	0.221	0.112	0.051	0.021
83%	0.790	0.638	0.475	0.326	0.208	0.529	0.339	0.193	0.099	0.045
85%	0.879	0.762	0.615	0.460	0.318	0.676	0.487	0.313	0.180	0.093
87%	0.942	0.866	0.752	0.611	0.461	0.809	0.649	0.470	0.305	0.177
89%	0.978	0.940	0.868	0.761	0.625	0.910	0.801	0.648	0.476	0.314
91%	0.994	0.980	0.947	0.884	0.787	0.969	0.914	0.815	0.673	0.505
93%	0.999	0.996	0.986	0.962	0.912	0.993	0.976	0.933	0.851	0.722
95%	1.000	1.000	0.998	0.993	0.979	0.999	0.997	0.987	0.961	0.902
97%	1.000	1.000	1.000	1.000	0.999	1.000	1.000	0.999	0.997	0.987
B. N,K	2,8	2,9	2,10	2,11	2,12	3,8	3,9	3,10	3,11	3,12
F	524	27	1.4	0.1	0.0	0.1	0.0	0.0	0.0	0.0

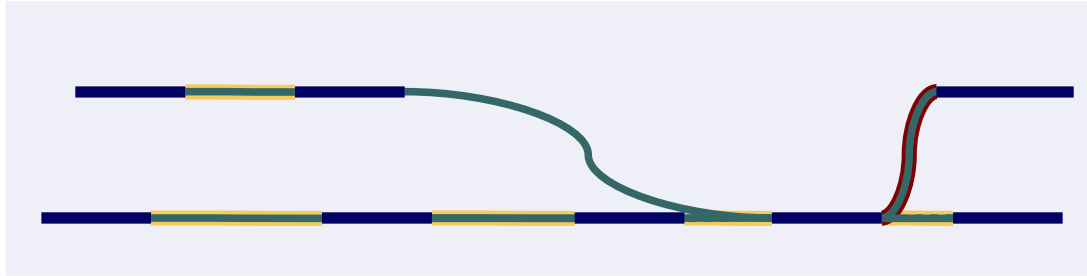
Zwischenstand

- Wo sind wir
 - Wir haben Proto-Clumps gefunden
 - Kern davon sind einer oder mehrere (fast) perfekte Hits
 - Diese Hits liegen auf einer Diagonalen und haben maximal Abstand w
 - Um die Hits herum wurde noch einiges an Sequenz hinzugefügt
- Was fehlt?
 - **Alignments** der Query mit den Sequenzen, in denen die Proto-Clumps gefunden wurden

Inhalt der Vorlesung

- Der Celera Assembler
 - Alignment & Assembly in the real world

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 H. Huson, Knut Reinert, Eugene W. Myers (2002) . „The greedy path-merging algorithm for contig scaffolding“. *Journal of the ACM* 49(5): 603-615
- Daniel Huson, Uni Tübingen. Vorlesungsskript "Algorithms in Bioinformatics: Genome Assembly", 2004

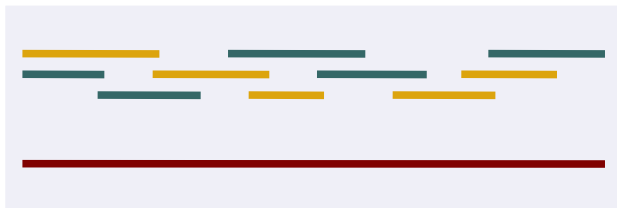
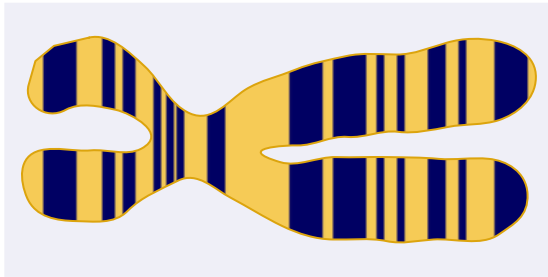
Celera

- 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)
 - Und viele weitere



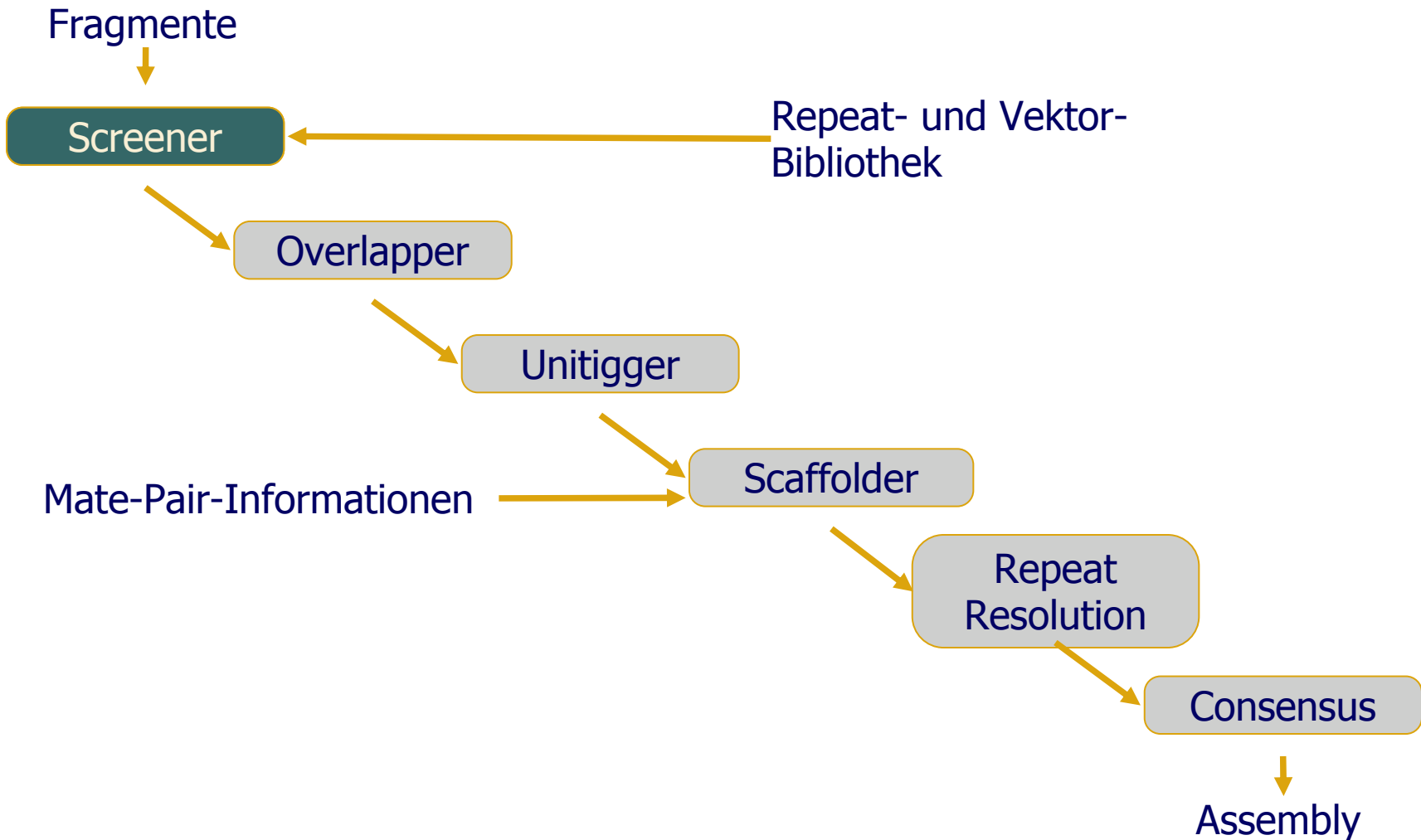
CELERA

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

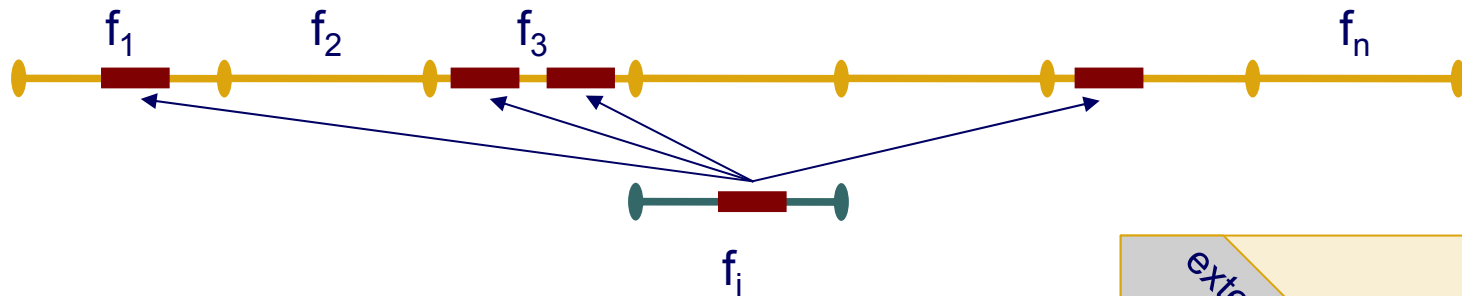
- Suche nach Überlappungen zwischen Fragmenten
- Es gibt drei (bis aus Symmetrie) Möglichkeiten für die **Überlappung** von zwei Fragmenten:



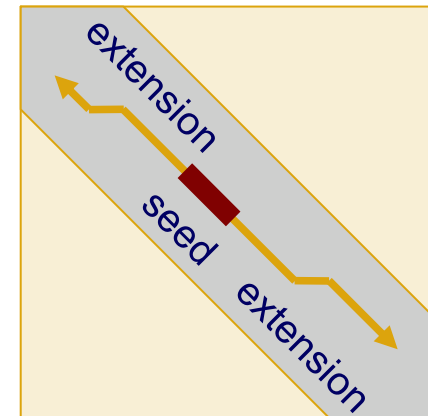
- Dynamische Programmierung?
 - Viel zu langsam
 - Es sind nur **high-scoring Overlaps** für die weitere Verarbeitung relevant

Seed and Extend

- "Seed and Extend" Ansatz (ähnlich BLAST)
 - Konkatination aller Fragmente
 - Seeds: Suche **exakte Matches der k-mere** von f_i in der konkatenierten Gesamtsequenz

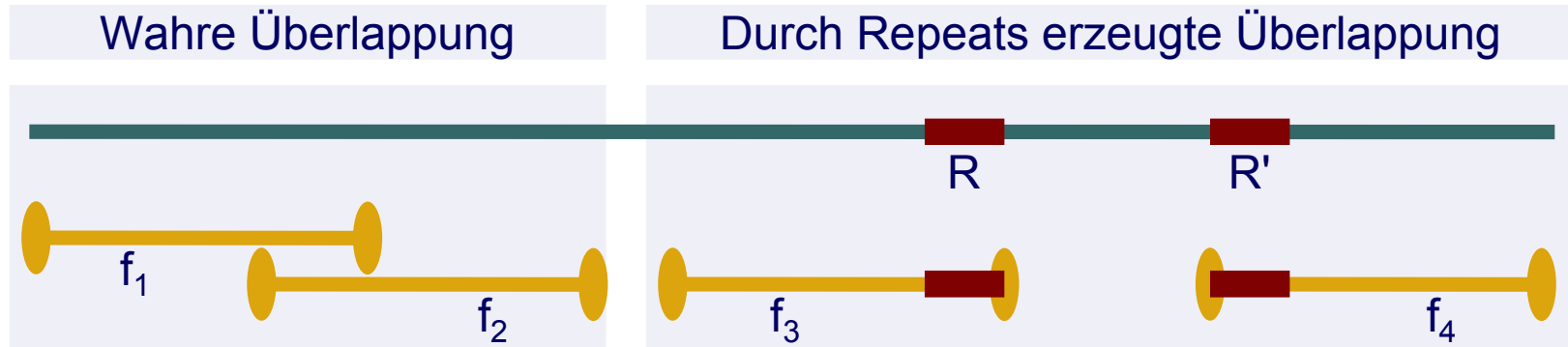


- Extension: Erweiterung der Seeds durch **Banded-Alignment**
 - Bis heuristisches Abbruchkriterium erfüllt



Repeats: DAS Problem

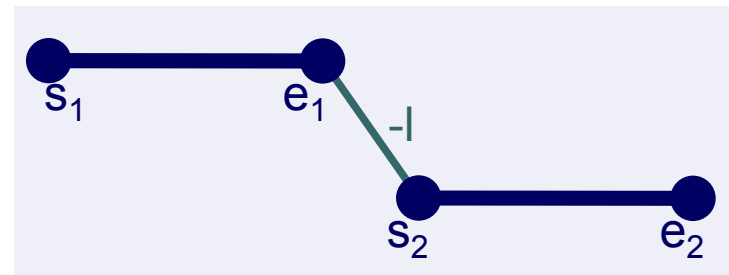
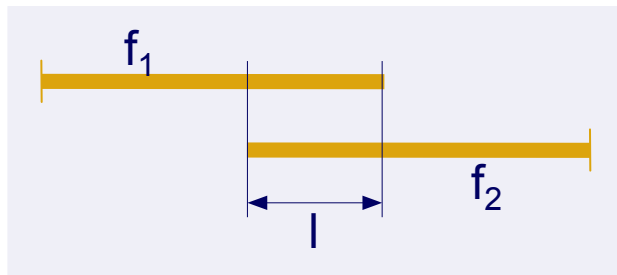
- Repeat-induced Overlaps



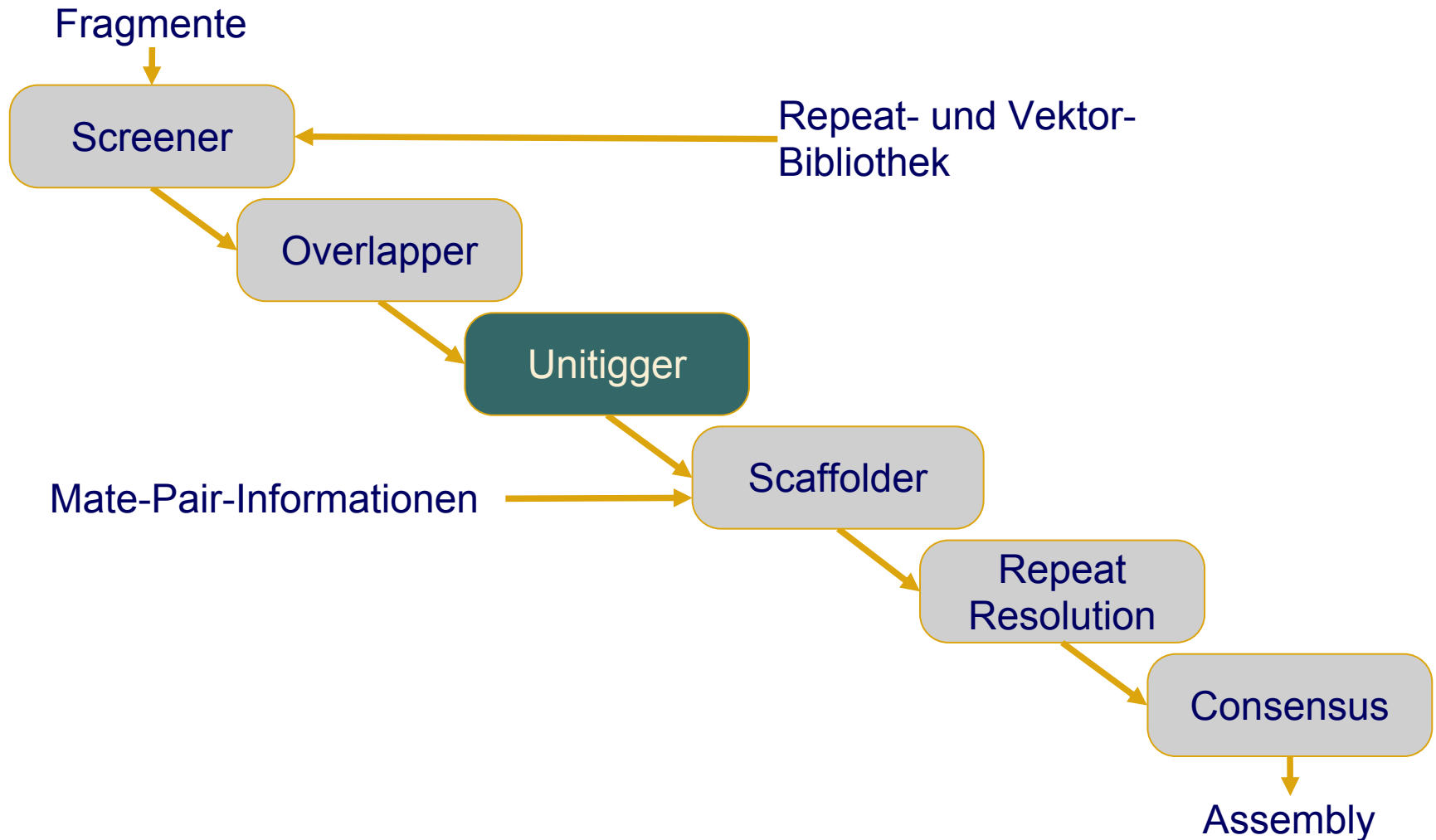
- So gut wie möglich vermeiden durch
 - Screening bekannter Repeats (ALU, SINEs, LINEs, ...)
 - Sehr häufige k-mere ignorieren
- **Repeats bleiben das Hauptproblem des Assemblies**
 - Führen zu falsch-positiven Überlappungen
 - Bilden Verbindungen über das gesamte Chromosom
 - Zerstören/Überlagern die wahre Anordnung

Overlap Graph

- Bisher: Alle Fragmente und deren Überlappungen
- **Overlap Graph**
 - Knoten
 - Zwei Knoten pro Fragment f_i : s_i und e_i (Start- und Endknoten)
 - Kanten für
 - Jedes Fragment f_i
 - Jede Überlappung
 - Werden mit dem Offset der Fragmente beschriftet (l)

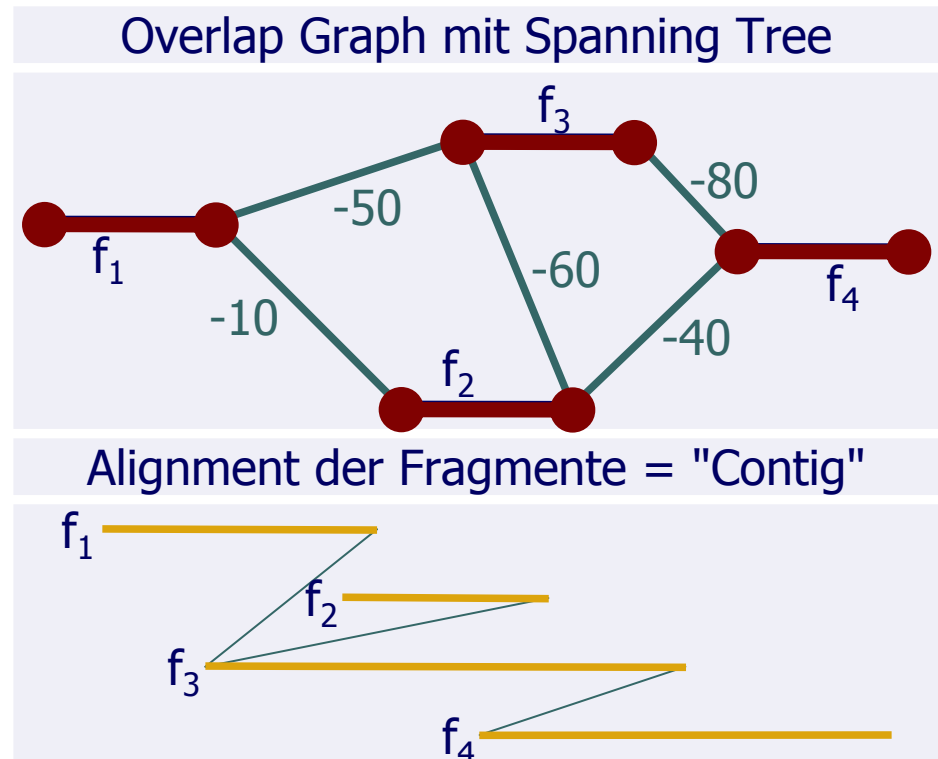


Überblick

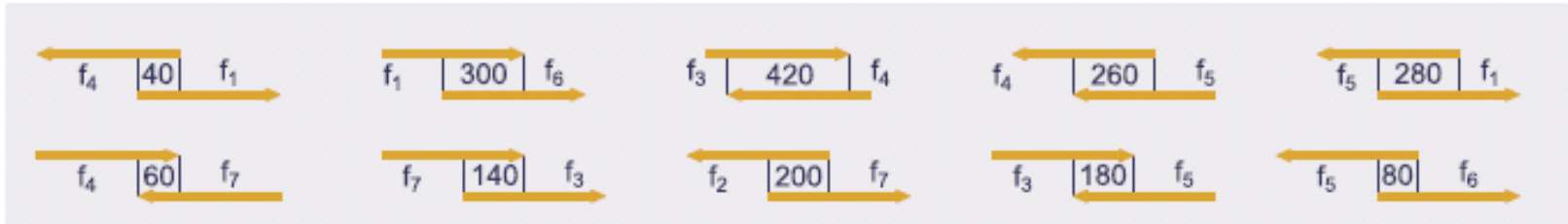


Unitigs

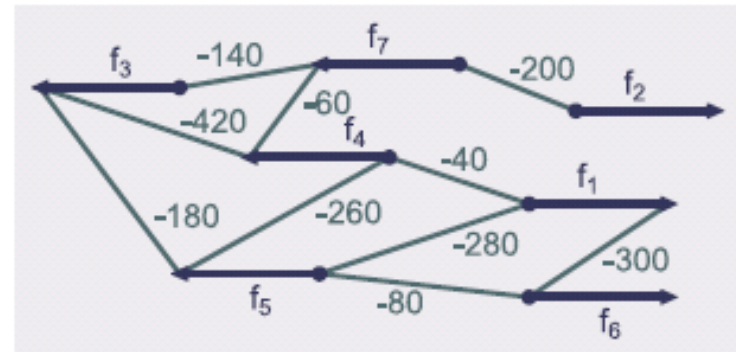
- Verbindet Fragmente zu Unitigs
 - „Unique Contigs“
- Zunächst: **Contigs** mit einer Heuristik finden:
Spannender Wald
 - Greedy
 - „Leichte“ Kanten zuerst
 - Entsprechen großen Überlappungen
 - Kanten, die **Zyklen erzeugen, ignorieren**
 - Das ist sehr greedy
 - So lange, bis jede **Zusammenhangskomponente** abgedeckt ist



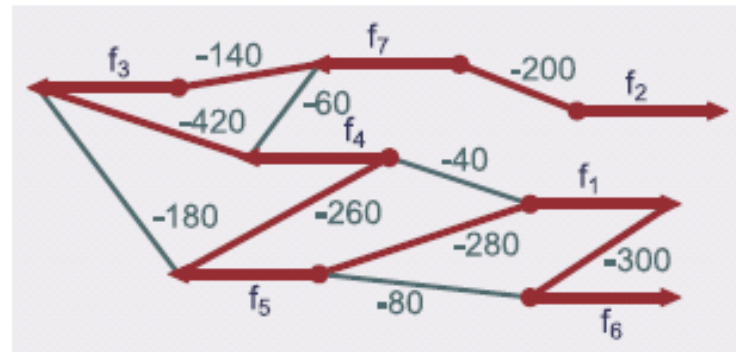
Beispiel



- Overlap Graph bilden
 - Nicht alle Kanten haben Richtung



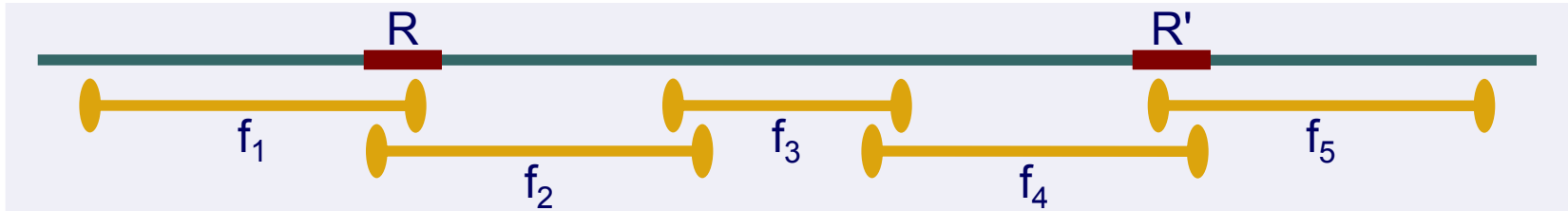
- Kanten sortieren und Greedy markieren
 - -180 würde Kreis schließen
 - Nach -140 Terminierung – danach nur noch Kreise



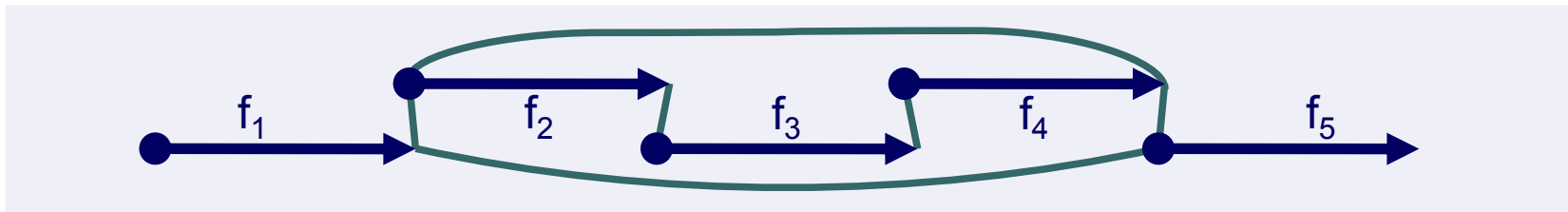
Greedy?

- Was wäre eine optimale Lösung?
 - Minimale Teilmenge der Overlap-Kanten des Overlap Graph so, dass eine konsistente Anordnung aller Fragmente möglich ist
 - Schwierig
- Warum ist der Algorithmus greedy?
 - Wahl nach absteigendem Benefit – häufig angewandte Heuristik
 - Bei auftretenden Zyklen wird nicht versucht, ob die Entfernung anderer Kanten im Zyklus nicht günstiger wäre
 - Frühe Fehler (unerkannte Repeats) haben weitreichende Folgen
 - ...
- Warum nicht möglichst viele Kanten mitnehmen?
 - Weil wir zunächst nur eine Anordnung suchen
 - Mehr Kanten würde aber die Evidenz mancher Bereiche verstärken

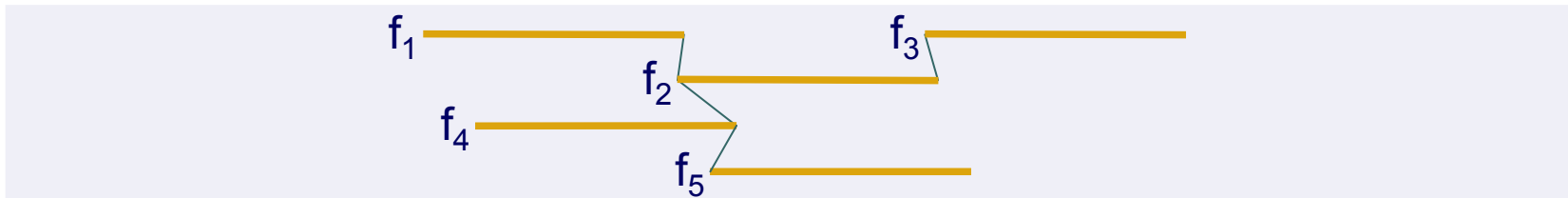
Repeats und Unitigs



- Erzeugen "falsche" Overlap-Kanten:



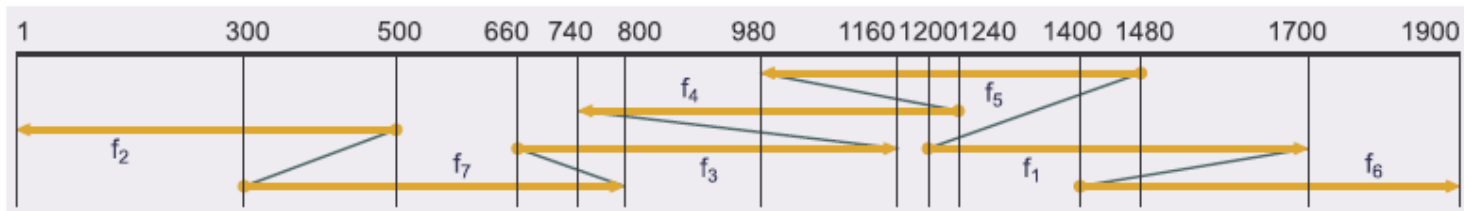
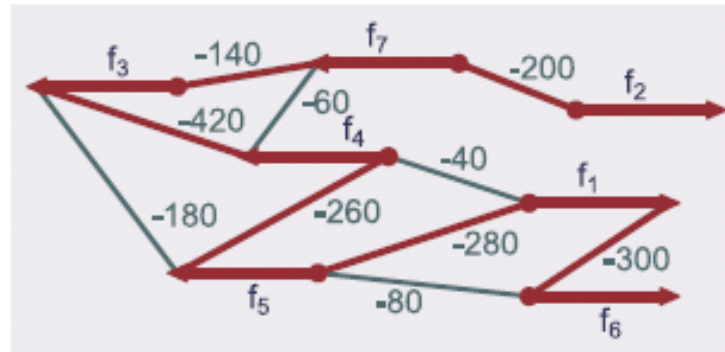
- Erzeugter Contig:



→ Kein konsistentes Layout möglich

Konsistente Layouts

- Wie sieht ein hierzu konsistentes Layout aus?



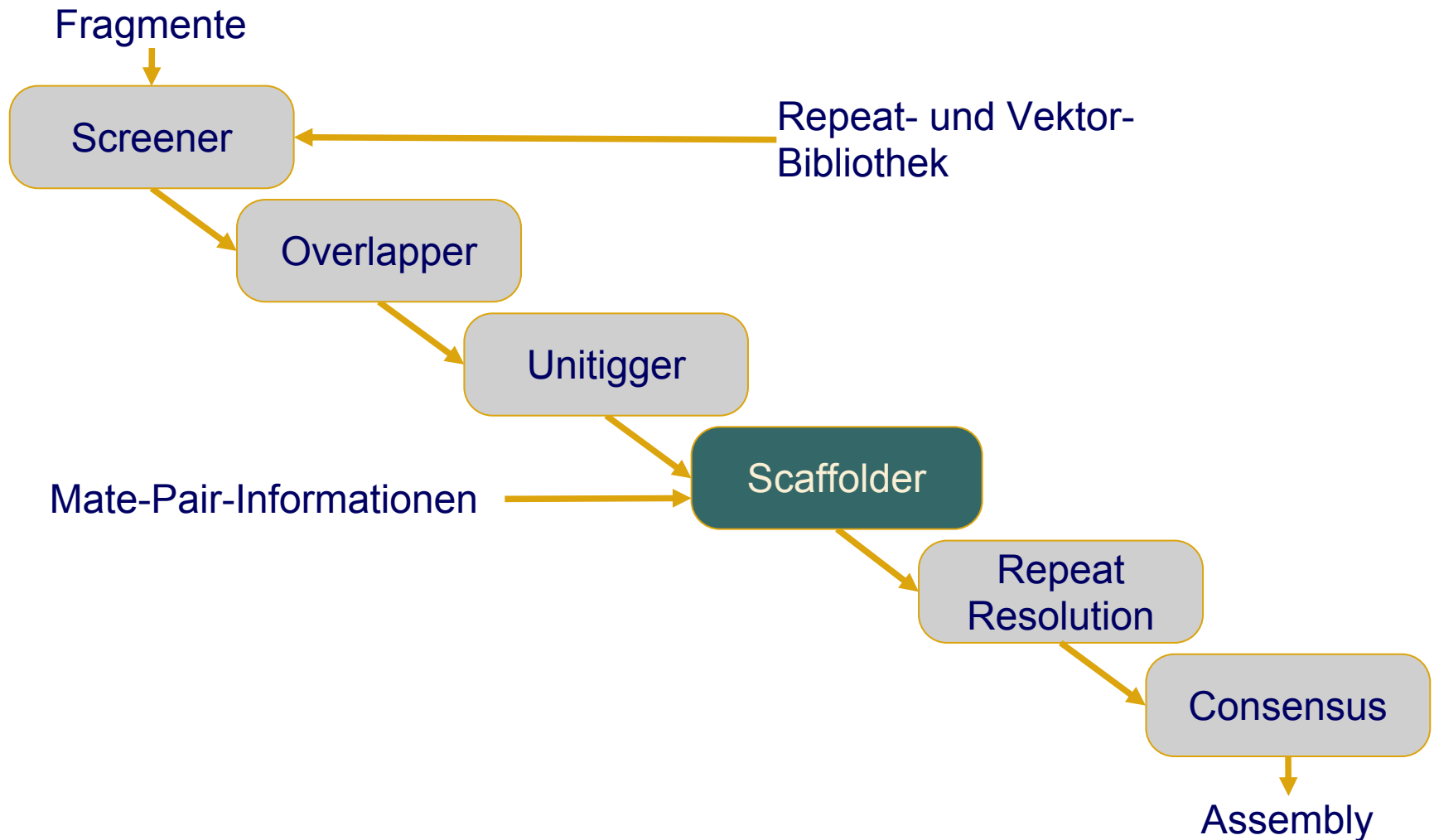
Unique Unitigs

- Unitigs (unique contigs)
 - Mit Overlap Graph konsistente Contigs
 - Konsistenztest alleine ist ein interessantes Teilproblem
 - Siehe Genome Mapping, Intervalgraphen
- U-unitigs (unique unitigs)
 - Einzigartige Sequenz (kein Repeat)
 - Nicht oversampled (Hinweis auf Repeat)



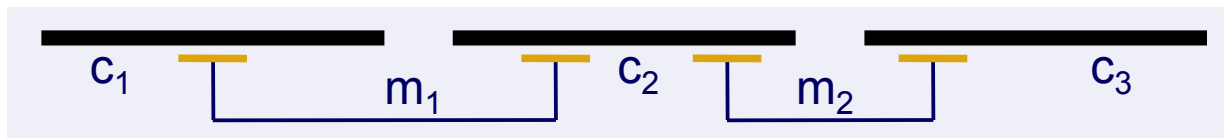
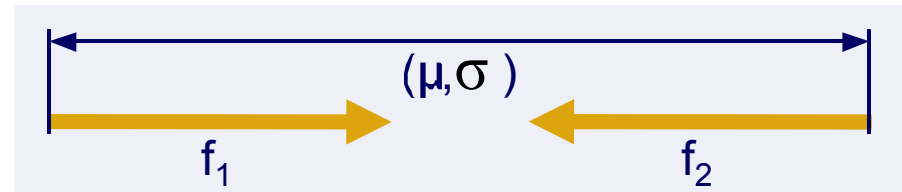
- Vergleich statistisch erwarteter Überlagerung mit beobachteter Überlagerung

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 ($<3\sigma$) 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
 - Klingt wie sehr viel verlangt
 - Aber man hat auch sehr sehr viele Daten

Contig-Mate Graph

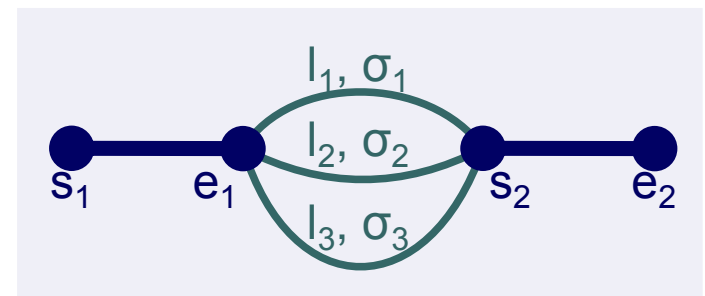
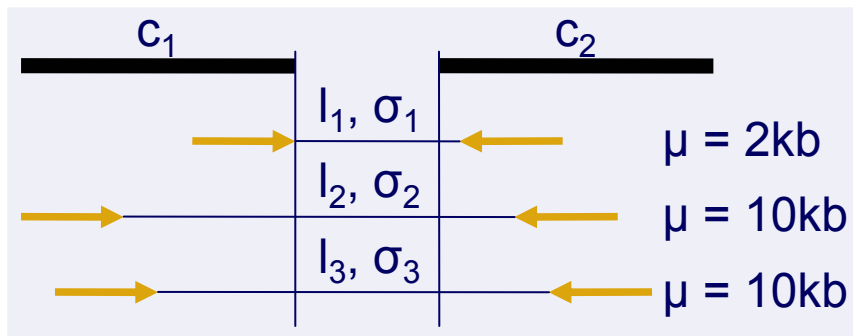
- Contig-Mate-Graph

- Knoten:

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

- Kanten für

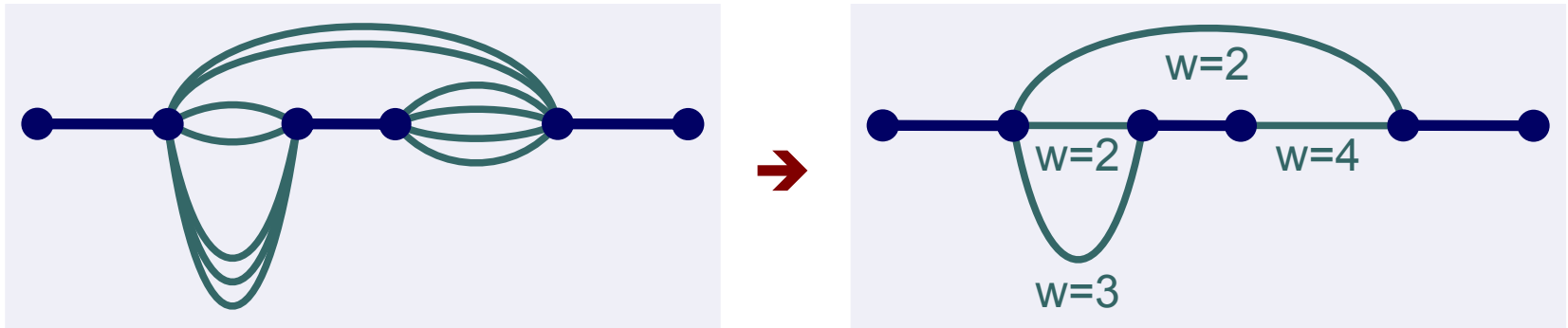
- jedes Contig c_i (zwischen Start- und Endknoten)
- jedes Mate-Pair



Vereinfachung

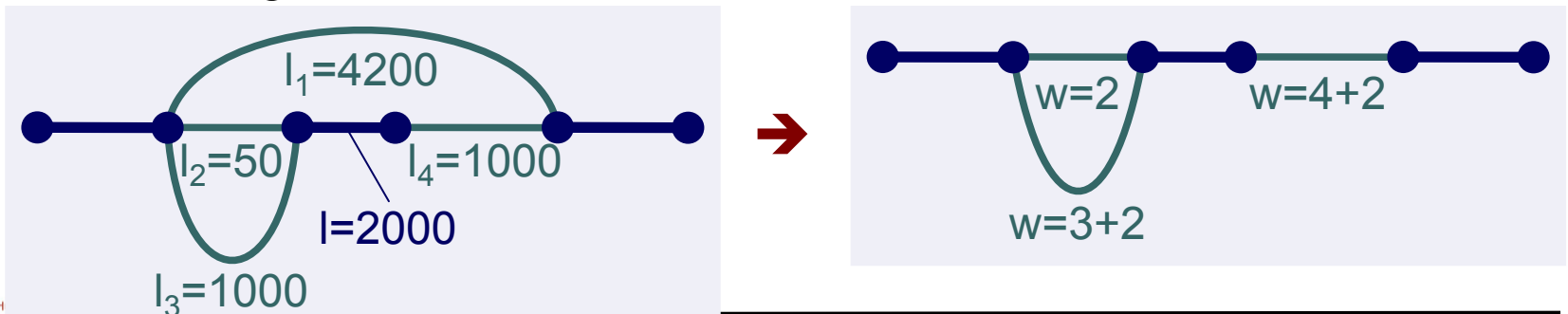
- Edge-Bundling

- Heuristisch sich bestätigende Mate-Kanten verschmelzen



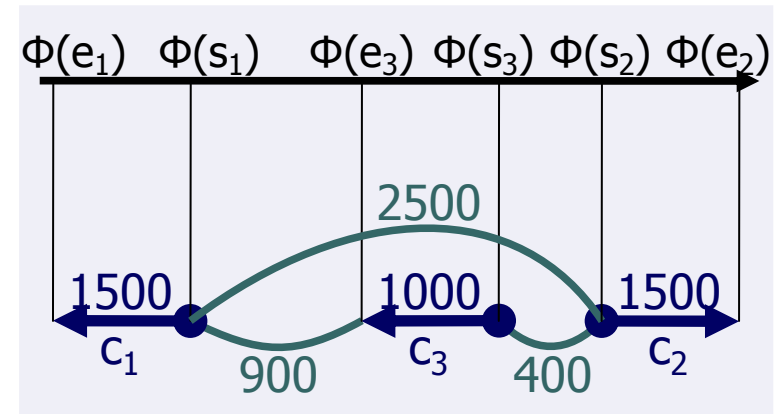
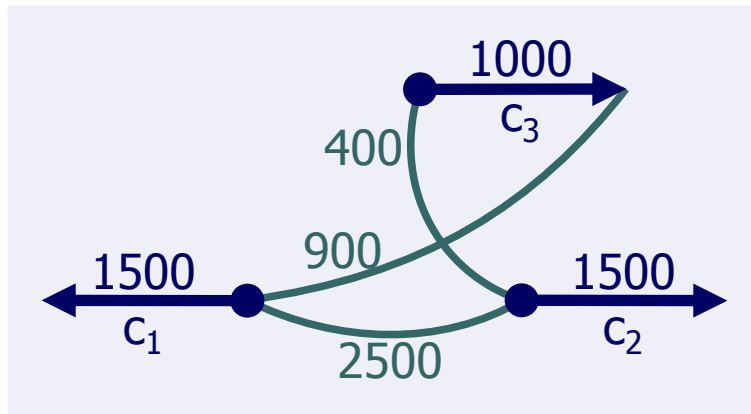
- Transitive **Kantenreduktion**

- Lange Mate-Kanten auf Pfade reduzieren



Happy Mate-Pairs

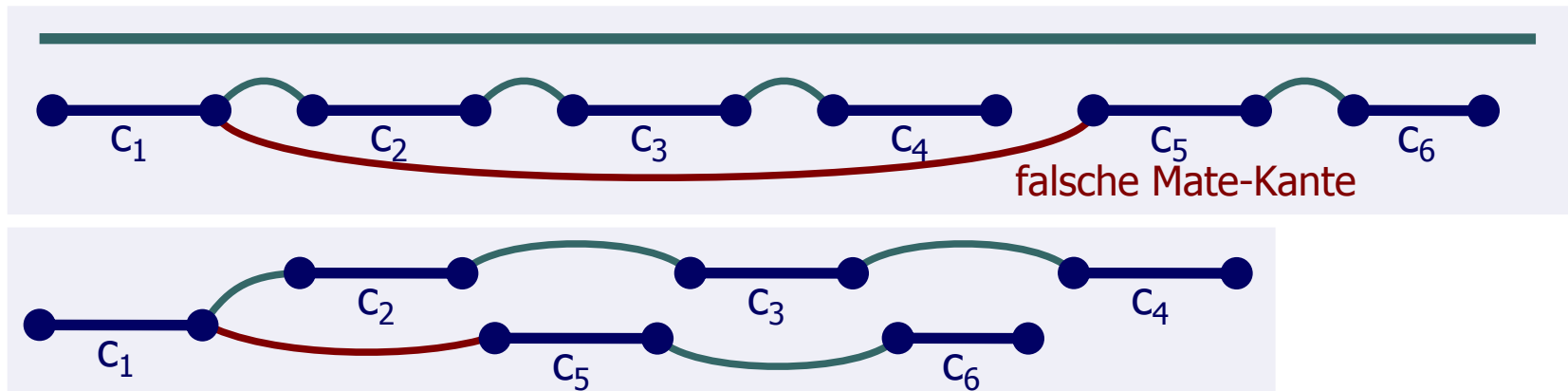
- Gesucht: **Koordinatenzuordnung** für Knoten
 - Beinhaltet Reihenfolge und Orientierung der Contigs im Graph
 - Ungefähre Erhaltung der Längen und Abstände



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

Contig-Mate-Pair Ordering Problem

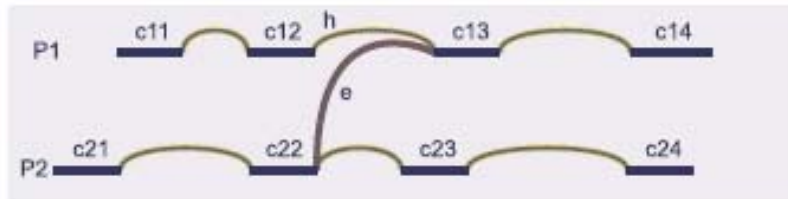
- Gesucht: Anordnung der Contigs, die die **Summe der Gewichte aller happy Mate-Pairs** maximiert
- Natürlich NP-vollständig
- Greedy Spanning-Forest Heuristik funktioniert nicht
 - Eine falsche Mate-Kanten führt zu fehlerhafter Verschränkung kompletter Contigs - schlecht



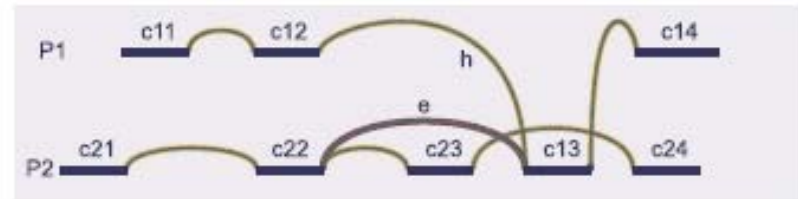
Greedy Path-Merging Algorithmus

- Andere Heuristik
- Pfad: Zusammenhänge Menge von Contigs
- Greedy Path-Merging Algorithmus
 - Sortiere **Mate-Kanten** absteigend nach Gewicht
 - Für alle Kanten $e = (v,w)$
 - Seien P_1 / P_2 die durch e neu verbundenen Pfade
 - Versuche, P_1 und P_2 zu mergen (nächste Folie)
 - Wenn dabei **mehr Happy- als Unhappy-Edges** hinzukommen
 - Ersetze P_1 & P_2 durch neuen Pfad P
 - Entferne e
- Ergebnis: Φ
 - Enthält unhappy Kanten, aber hoffentlich nicht allzu viele

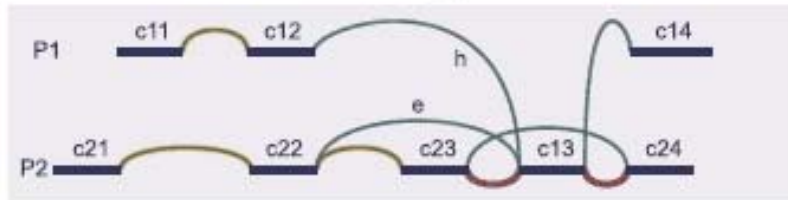
Beispiel



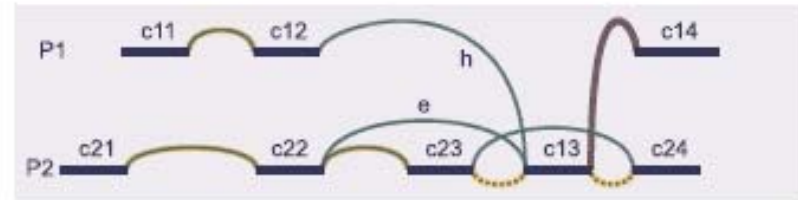
Die Länge der Kante e suggeriert, daß c_{13} zwischen c_{23} und c_{24} liegt.



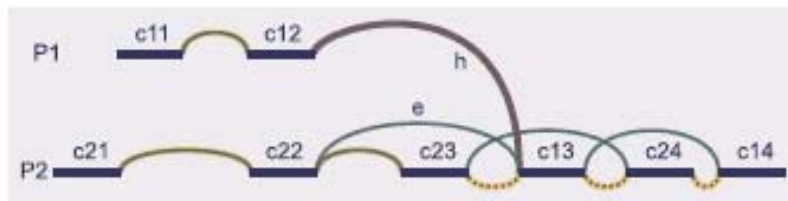
Contig c_{13} passt zwischen c_{23} und c_{24} .



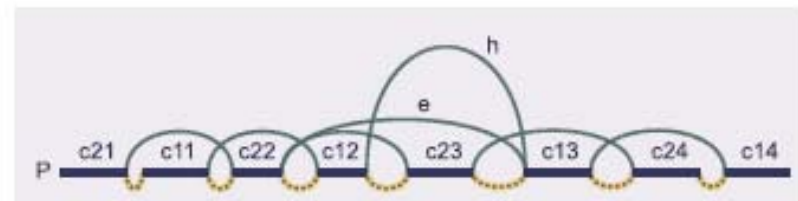
„Gefolgerete“ Kanten werden eingeführt. Sie verbinden c_{13} mit seinen neuen Nachbarn.



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

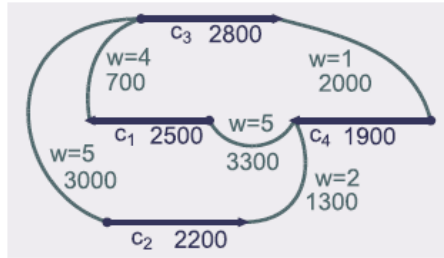


Am rechten Ende angekommen, wird jetzt versucht die Pfade mit Hilfe von Kante h nach links zu verschränken.

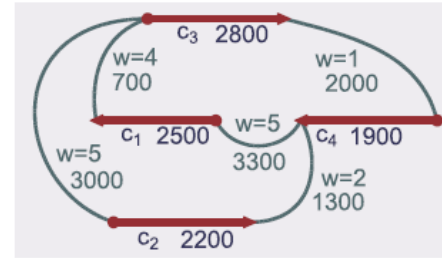


Das Reißverschlußverfahren war erfolgreich. Die Contigs aus P_1 und P_2 wurden im neuen Pfad P integriert.

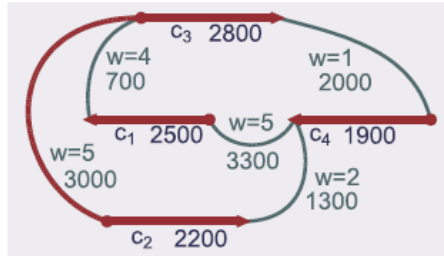
Beispiel 2



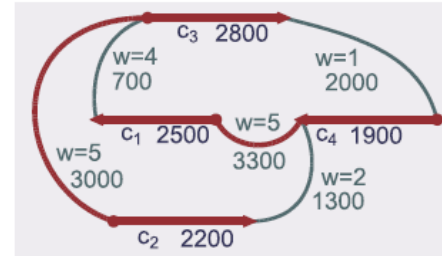
Contig-Mate-Graph nach Edge-Bundling und transitiver Kantenreduktion.



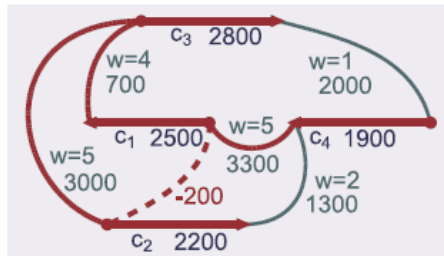
Alle Contig-Kanten werden markiert.



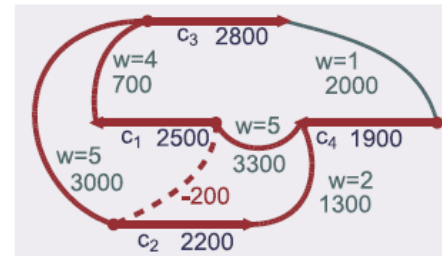
Die Kante mit dem höchsten Gewicht ($w = 5$) verbindet c_2 und c_3 .



Die nächste Kante mit $w = 5$ verbindet c_1 und c_4 . Es gibt jetzt zwei Pfade.

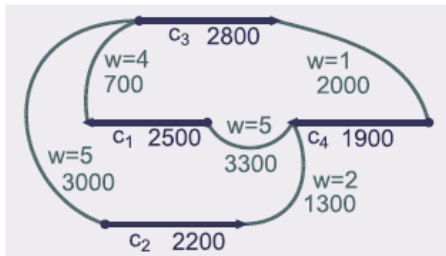


Die Kante mit $w = 4$ ordnet c_1 zwischen c_3 und c_2 an. Eine neue Kante der Länge -200 wird eingefügt.

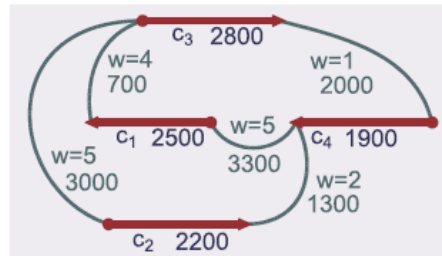


Es geht mit der Kante des Gewichtes $w = 2$ weiter. Sie bestätigt die bisherige Anordnung.

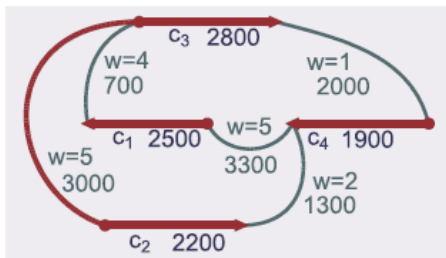
Folgende Anordnung



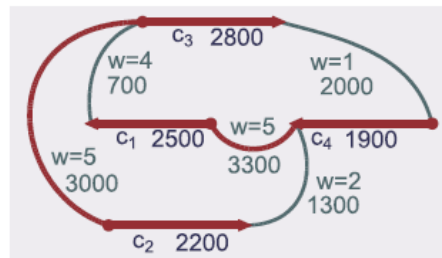
Contig-Mate-Graph nach Edge-Bundling und transitiver Kantenreduktion.



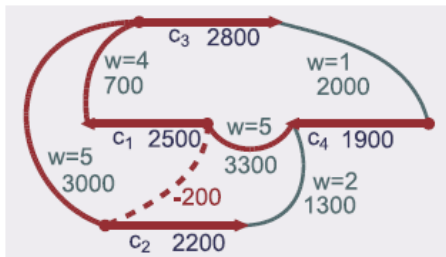
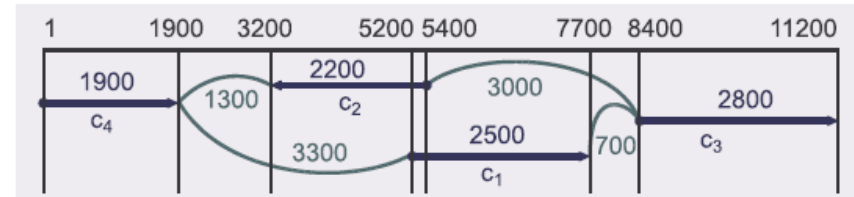
Alle Contig-Kanten werden markiert.



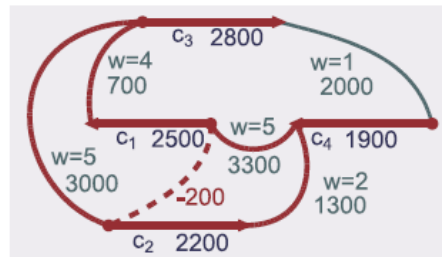
Die Kante mit dem höchsten Gewicht ($w = 5$) verbindet c_2 und c_3 .



Die nächste Kante mit $w = 5$ verbindet c_1 und c_4 . Es gibt jetzt zwei Pfade.

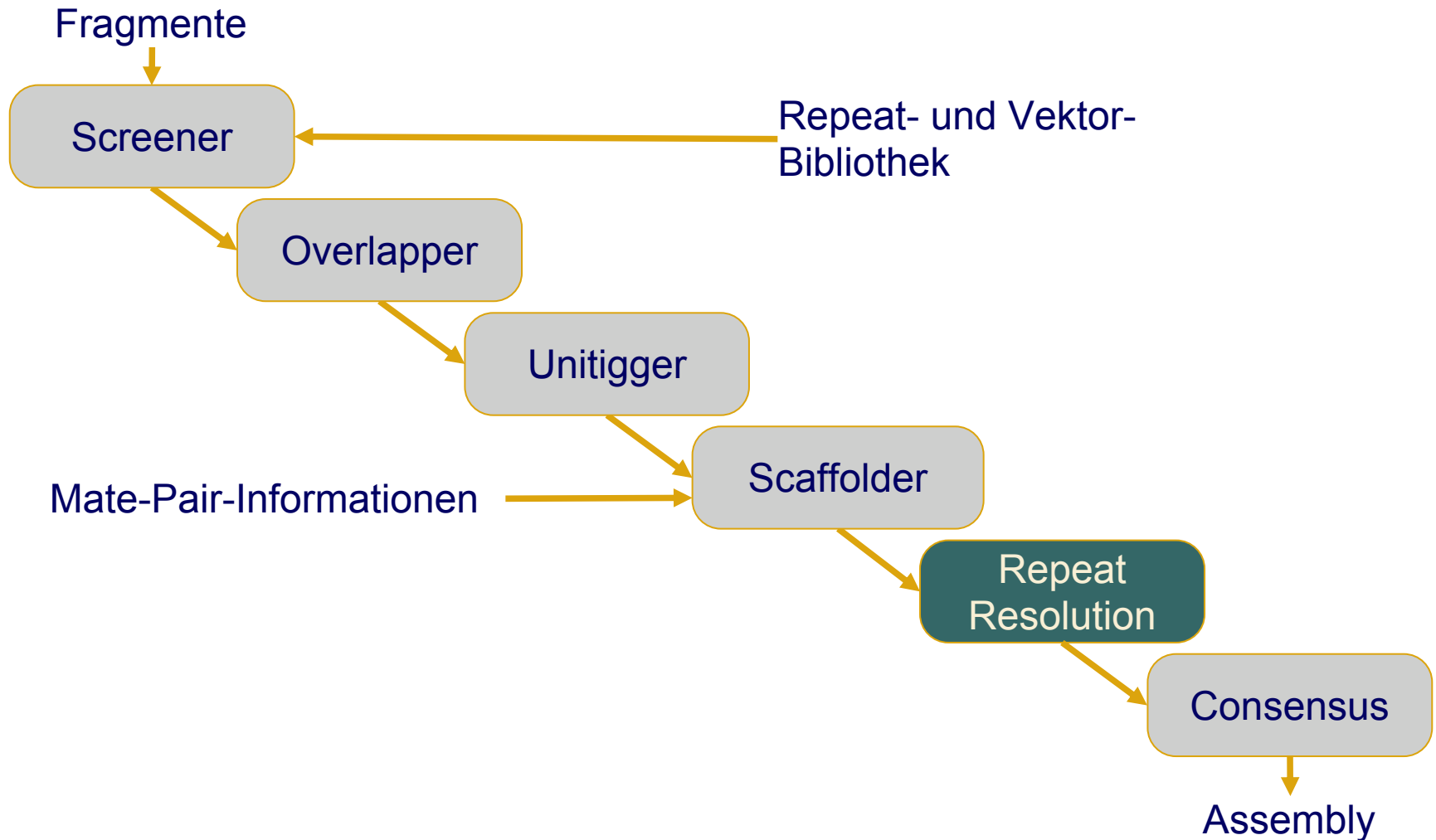


Die Kante mit $w = 4$ ordnet c_1 zwischen c_3 und c_2 an. Eine neue Kante der Länge -200 wird gefolgt.



Es geht mit der Kante des Gewichtes $w = 2$ weiter. Sie bestätigt die bisherige Anordnung.

Überblick

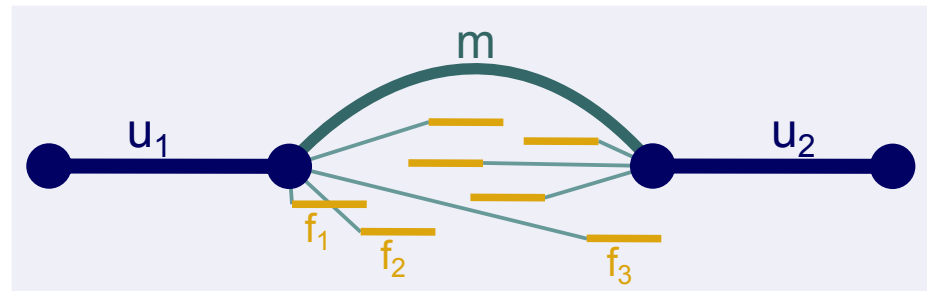


Repeat Resolution

- Aufgabe: Gaps schließen

- Input:

- Scaffolds
- Fragmente
- Mate-Pairs
- Überlappungen



- Output:

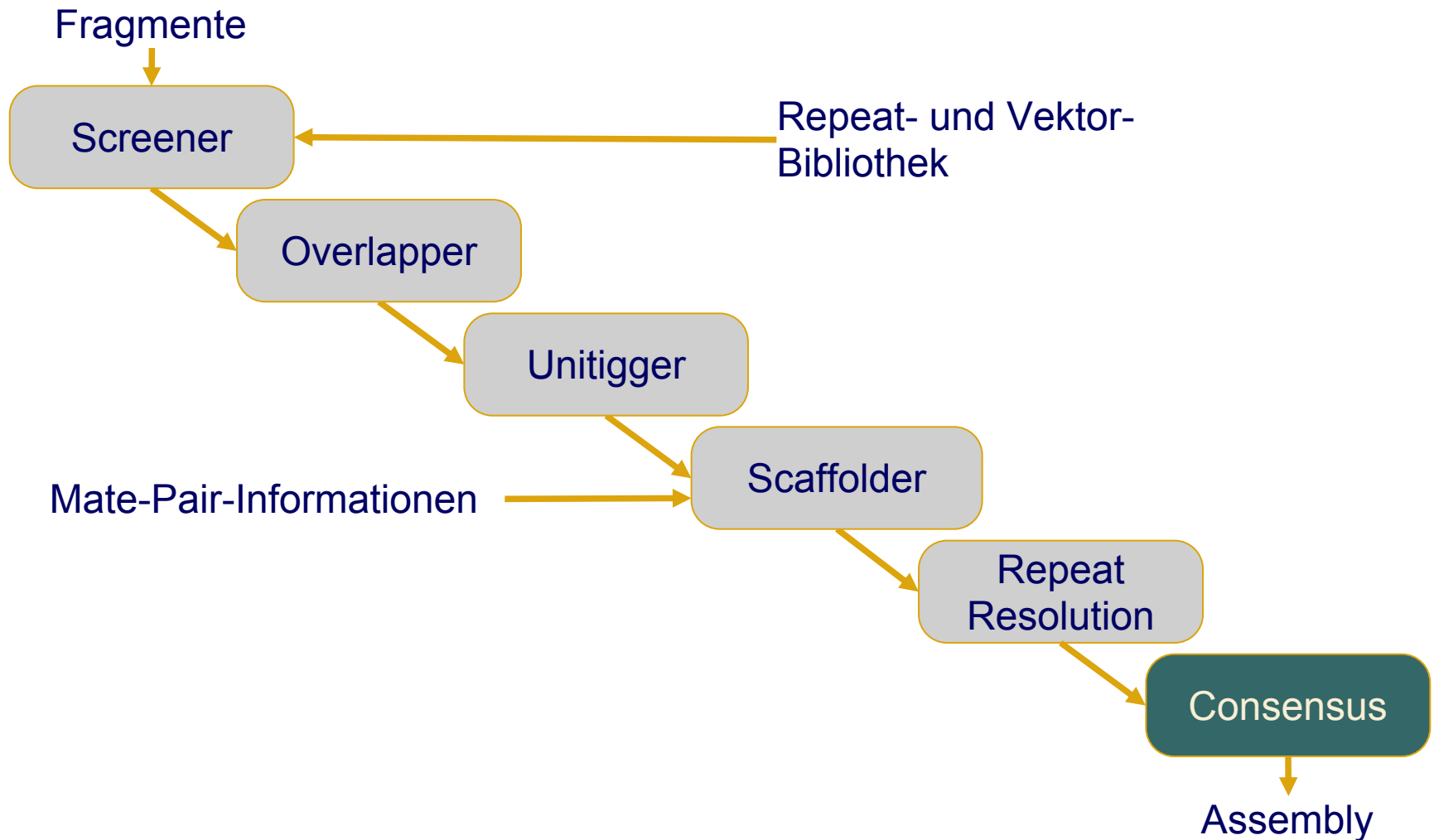
- Vollständigere Scaffolds

- Heuristisches Füllen der Lücke zwischen 2 Unitigs mit bisher nicht zugeordneten Fragmenten

- Weil durch zu wenig Mate-Pairs verknüpft
- Unitig war oversampled
- ...

- Bei Erfolg: Verschmelzen der Unitigs

Überblick

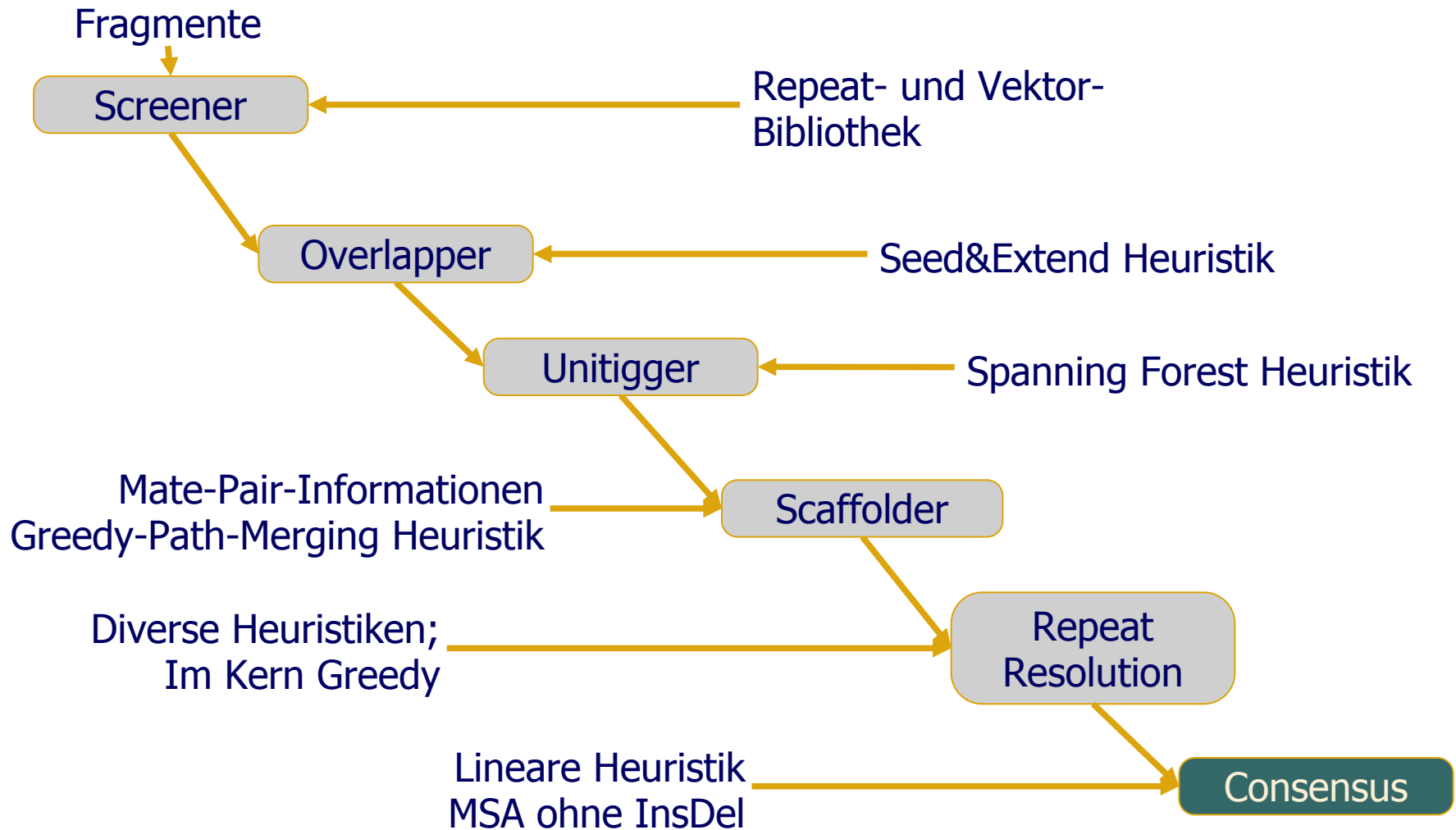


Consensus

- 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

Überblick



Menschliches Genom

- 27 Millionen Fragmente, Ø-Länge: 550b, 70% mit Mate-Pair Informationen

	CPU-Stunden		Max. Speicher
Screeener	4800	2- 3 Tage auf 10- 20 Computern	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 (2003) aus 6500 Scaffolds**
 - Umfasst 2,8 Gb Sequenz
 - **150.000 Gaps** (insgesamt 148 Mb)