FINISHING FOSMID XBAA-40F24
(2nd Revision)

Michelle Itano

Abstract:

Analysis is being done to better understand how chromatin structure influences the function of DNA. Two *Drosophila* species, *D. melanogaster* and *D. virilis*, are convenient organisms to investigate this matter using genomic comparisons. Data suggest that the small dot chromosome of *D. virilis* is euchromatic, while the dot chromosome of *D. melanogaster* is known to be heterochromatic. Therefore, continuation of the comparative study of the *D. virilis* and *D. melanogaster* dot chromosomes may identify critical differences in genome organization that direct packaging, and provide information as to the effect of packaging on function of DNA. In order for this comparison to occur, the *D. virilis* genome must be finished to high quality, including sequencing of the repeat areas.

Fosmid XBAA-40F24 of the *D. virilis* dot chromosome was found to contain a tandem repeat of ~5kb. The presence of this long repeat presented many problems during assembly. The clone ends were found to lie within this tandem repeat. This suggests a layout of ~18kb unique sequence attached to the same tandem repeat sequence at both ends. Due to problems in assembly, this fosmid was not finished, and still contains many problem areas. The assembly must be checked for consistency and the clone ends must be properly added to the unique segment. While this project was very difficult, it was a good exercise for learning creative assembly techniques that can be used when troubleshooting strategies are necessary, often needed in the presence of large repeat sequences.

FINISHING FOSMID XBAA-40F24
(2nd Revision)
Michelle Itano

Initially, I examined the 171 reads from one plate using **QSTAT**, which can analyze and give a histogram of the data. The reads from the XBAA-aac54 plate were found to generate 2.4x coverage. The reads had an average length of 730 bp (see Figure 1 below). The first spike, around 25bp, represents failed reads. The initial assembly with only the 171 reads resulted in 8 contigs (Figure 2). To get more thorough coverage for analysis, we added more reads before beginning the finishing of this fosmid.

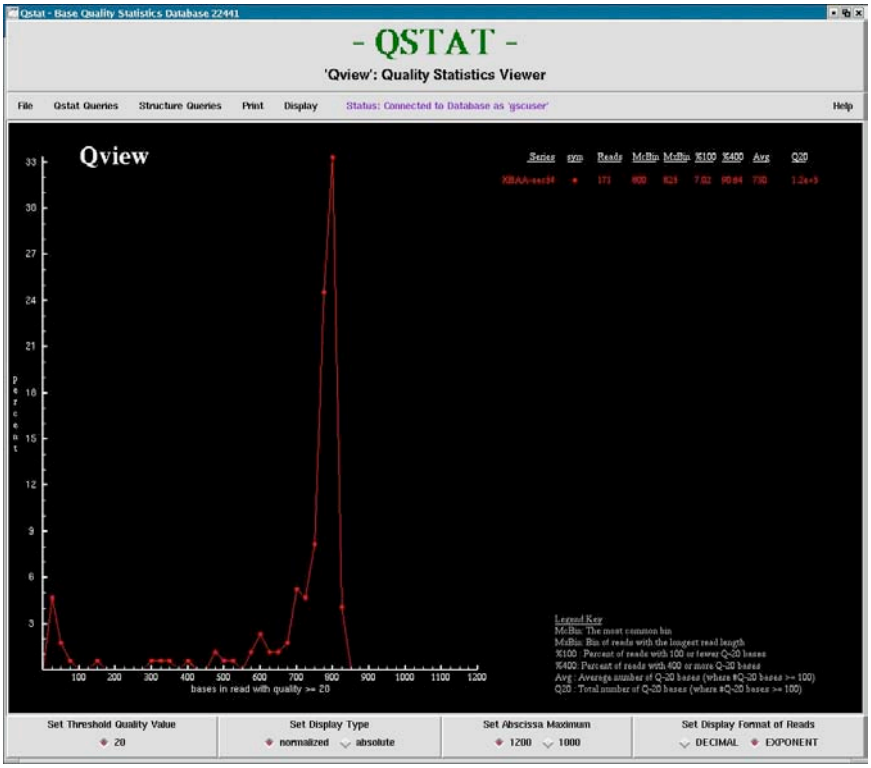

Figure 1. QSTAT of 171 reads

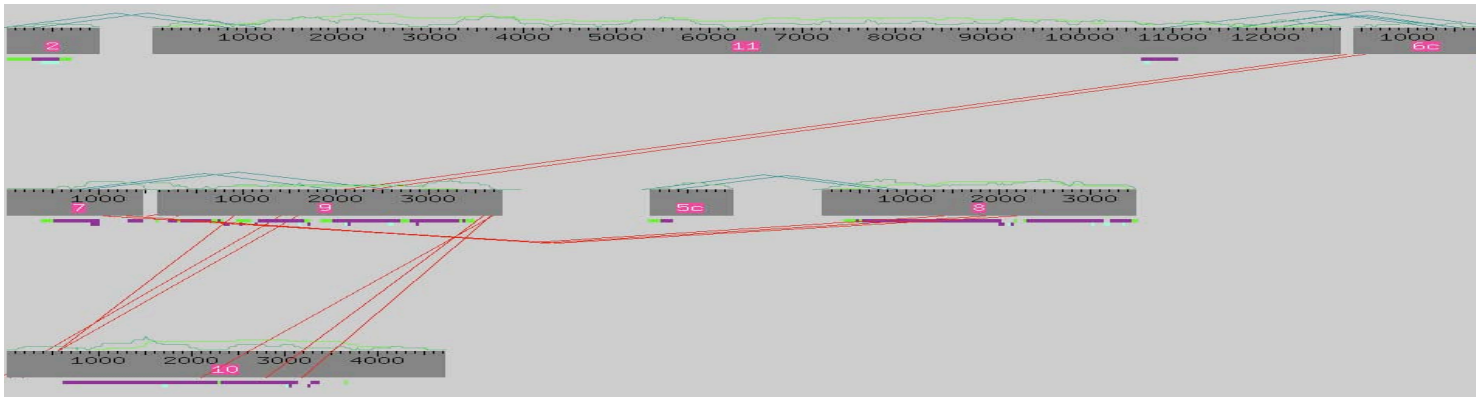

Figure 2. Initial assembly based on 171 reads

When all available reads were added to the assembly using the program **phrap** and visualized with **Consed**, several of the contigs from the earlier assembly were joined (Figure 3a). However, the assembly still consisted of 6 contigs with several gaps. The red lines in the assembly view represent inconsistent forward/reverse pairs, which can be an indication of misassembly. To further investigate whether or not my fosmid had been misassembled, I ran an initial **crossmatch** at 95% (Figure 3b). This program identifies sequences that are at least 95% identical to other sequences in the assembly and marks those sequences in the assembly view with bars and lines to indicate where the matches occur (note: % identity can be varied). The orange bars and lines indicate sequence matches that are uncomplemented, while the black bars and lines indicate sequence matches that are complemented. The repeats observed are long and common to many different areas of the assembly, indicating the presence of tandem repeats. Also, note that the assembly shown in Figure 3 has a total length of ~30 kb. Because I know that I sequenced a fosmid of ~40 kb, this is further evidence for the presence of misassembled repeats. In sum, the initial assembly view with **crossmatch** suggests that the fosmid consists largely of tandem repeats and a unique sequence of approximately 18kb.

Figure 3. Initial Assembly

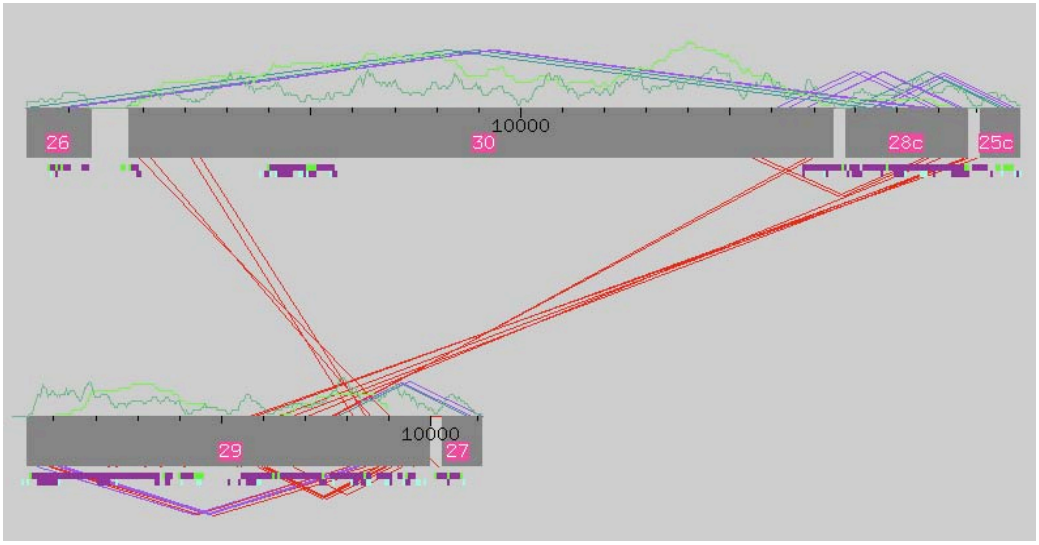

Figure 3a. Initial assembly with all available reads

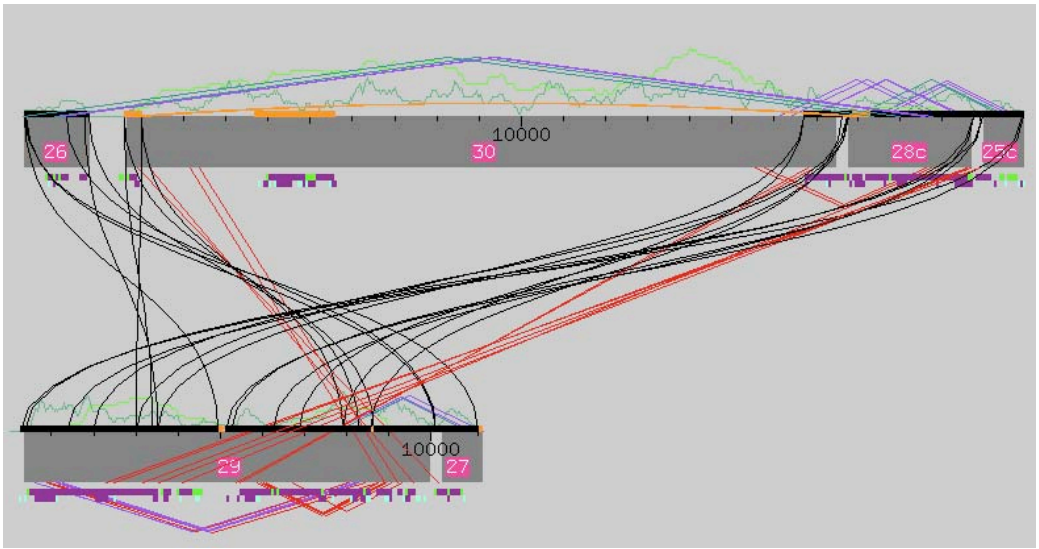

Figure 3b. Initial assembly with all available reads after running **crossmatch** at 95%

Strategy 1: Guide Phrap

Because my initial assembly suggested that my fosmid had been misassembled, I decided to look for areas where **phrap** may have aligned sequences that actually belonged to another location within the fosmid. My fosmid was particularly likely to have an error of this nature due to the presence of tandem repeats. **Phrap** has a difficult time identifying tandem repeats because they consist of sequences that are nearly identical to one another, which is **phrap**'s main indication for alignment. Initially, I navigated through my fosmid looking for high quality discrepancies (Figure 4a and Figure 4b). First I would decide whether or not the base had been called properly by examining the traces in the area (Figure 4c). If the trace showed a high peak that was consistently spaced, with no other abnormalities, I would be confident that the base was high quality. I would also note if there were several reads that called the same high quality discrepancy at that location, indicating that this discrepancy represented a misalignment and not a miscalled base. If I decided that the area had been misassembled, I would tag the area containing the discrepancy and "Tell phrap not to overlap reads discrepant at this location." I continued to navigate through the high quality discrepant areas until there were zero high quality discrepancies. Then I re-ran **phrap** and continued this process three more times until re-running **phrap** resulted in no high quality discrepancies in the entire assembly (Figure 5a).

Figure 4. Guiding Phrap

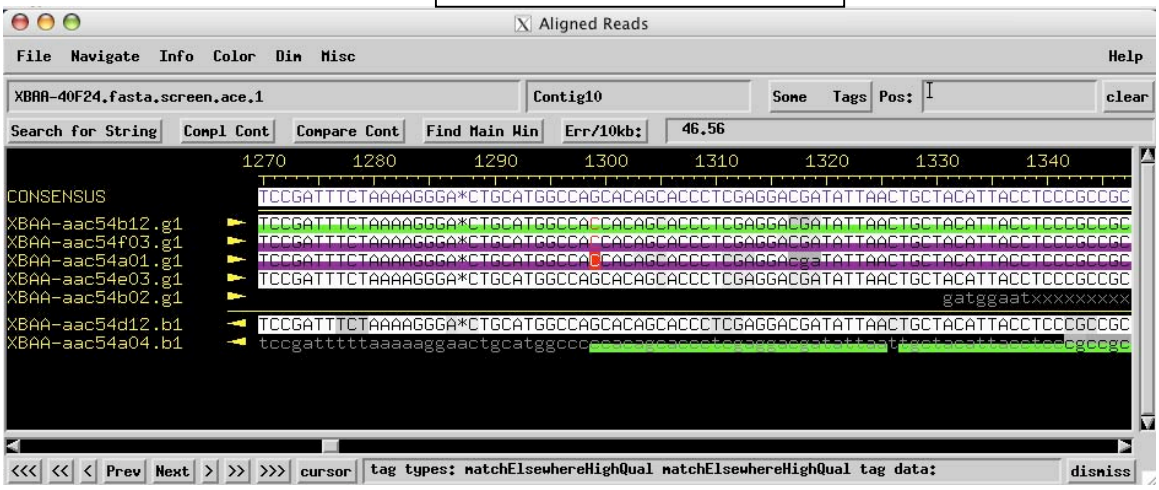

Figure 4a. Misassembly indicated by a high quality discrepancy

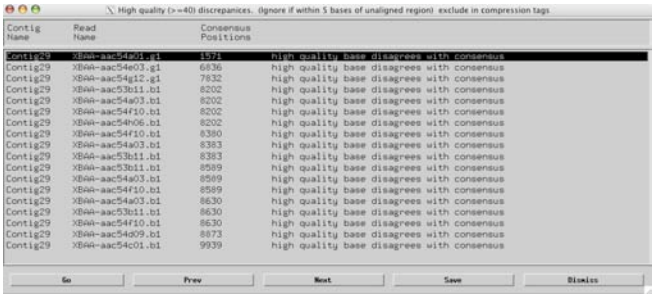

Figure 4b. Navigation by high quality discrepancies

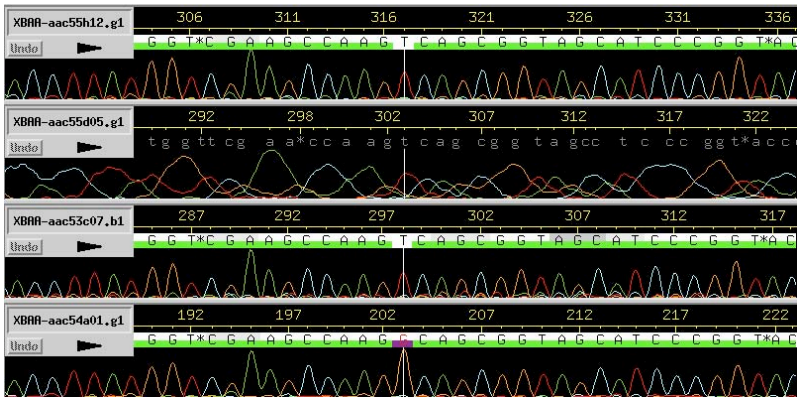

Figure 4c. Traces showing misassembly indicated by a high quality discrepancy

Strategy 2: Resolve Forward/Reverse Pairs

After all of the high quality discrepancies had been resolved by telling **phrap** not to overlap unique repeats, I identified misassembled reads that were still in my assembly. The assembly view at this point (Figure 5a) consists of 7 contigs. While the assembly has become more segmented, I have resolved some of the misassemblies in the tandem repeats. There are many indications of inconsistent forward/reverse pairs (shown by red lines). This suggests that one or both of the reads have been misassembled and belong at another location within the fosmid. Also, much information can be obtained from the consistent forward/reverse pairs (shown with blue legs). An area with many consistent read pairs that "walk" across a contig, or overlap each other across the entire contig, suggests that the assembly in that area is probably correct. Areas with very few consistant read pairs indicate areas where misassembly may have occured. Therefore, assembly view, showing both inconsistent and consistent forward/reverse pairs, was very useful for identifying areas that may have incorporated reads that belong in other locations.

Misassembled areas identified in assembly view can be resolved by pulling out inconsistant reads and attempting to place each read in a different location more consistant with its forward/reverse pair. I could navigate to these areas either by clicking on a red line in the assembly view (Figure 5b), or by choosing a read within the aligned reads window (Figure 5c), and copying it into the read list in the main window to locate its corresponding forward/revese read (Figure 5d). I have indicated inconsistent reads by highlighting them so that they appear purple in the aligned reads window (Figure 5c); note that in many areas there are many inconsistent reads. This is also observed in Figure 5d, which shows that only one read pair out of four has both reads currently assembled into the same contig. In this manner many reads were analyzed individually and reassembled, resolving many previously misassembled areas.

The tandem repeats within my fosmid caused many inconsistent forward/reverse pairs to occur in the **phrap** assemblies. The tandem repeats within this fosmid are very large, later determined to be ~5 kb repeats (see Strategy 7). Therefore, a read of approximately 730 bp may lie within this repeat and could be assembled by **phrap** in many different locations. I placed a large emphasis on making the forward/reverse pairs consistent, because I knew that these reads were from the same subclone and therefore must lie within a given distance of one another in the overall assembly. Overall, this strategy was very helpful to assemble my fosmid and locate areas where misassembly had occured.

Figure 5. Resolving Forward/Reverse Pairs Continued

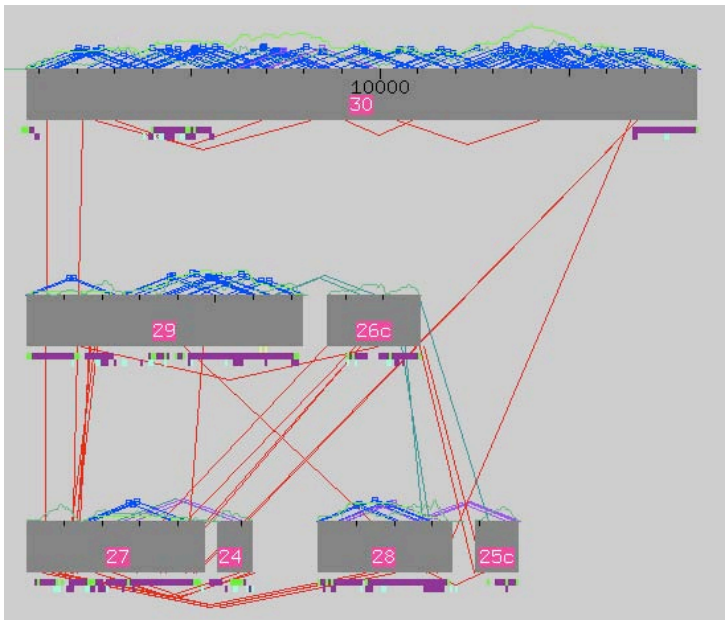

Figure 5a. Assembly view with consistent and inconsistent forward/reverse pairs

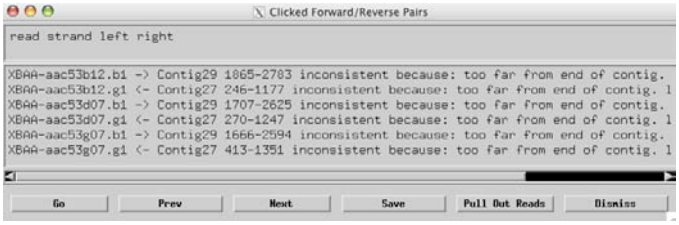

Figure 5b. Navigation by clicking on forward/reverse pairs in assembly view

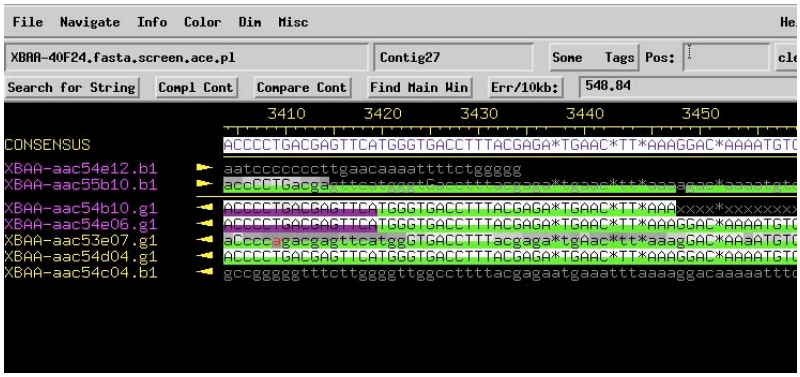

Figure 5c. Aligned reads window with inconsistent forward/reverse pairs highlighted

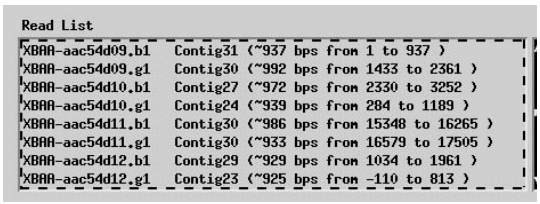

Figure 5d. Navigation by read list in main window

Strategy 3: Perform Tears and Forced Joins

Another strategy to resolve misassembled areas is to identify locations where tears or forced joins could occur. These strategies can aid assembly where **phrap** was either too conservative or too liberal when making an alignment. As tears and joins are made within the assembly, it becomes possible to make a more complete assembly. Tears create new contig ends which may pull up new reads which were previously unimportant and will add more data to the final assembly.

I looked for possible tear locations with two main strategies. First I located areas where there was an absence of consistent forward/reverse pairs, for example near the beginning of contig 43 (Figure 6a). Second, I looked for areas with very low coverage. I examined these areas in the aligned reads window. I would look for areas where all of the reads ended and other groups of reads began, with a notable lack of spanning reads in between (Figures 6b and 6c). A tear would break the current contig into two new contigs at the point indicated. In order for this to occur, I would have to specify which direction (to which contig) I wanted each read to go. In this manner I would dictate which reads were to belong to which new contig. In the area described above, I would send all of the reads that ended to the left (with all of the reads lying to the left of the current contig position). I would send all of the reads that began in this location to the right (with all of the reads following the current contig position). I would make a tear if there did not appear to be any reads spanning this area (where all reads either ended or began). A tear could be made at this point because the overlap is not very strong, and if the area were broken into two separate contigs other areas might be found with a more significant overlap. If it looked like a tear could be made, I would tear the contig at that point and check the results in terms of the impact on forward/reverse pair consistency and number of high quality discrepancies. After making a tear I would look at the new ends and see if any new joins could be made, or if any new contigs had been pulled into the assembly.

I identified possible join locations by first looking at contig ends that are spanned by subclones (green lines) in assembly view (Figure 6a). This means that the forward read from the sublcone lies at the end of one contig, while the reverse read on the subclone lies at the beginning of the other contig. If several subclones span two contigs, there is a high probability that the contigs should be joined together in the order suggested by assembly view. To make the join, I searched for string sequences near the end of the contigs that were to be joined (Figure 6d). Ideally, the string would match to the appropriate end of the contig to which I was attempting to make the join. Then I would compare contigs at the location where the string matched to both contigs (Figure 6e). If there were no high quality discrepancies, I would make the join. If the compare contigs window located a high quality discrepancy I looked at the contig to see if any of the reads could be moved (Figure 6f). In Figure 6f, note how there are two bases being called at that position. In this case, the discrepancy could be resolved by removing discrepant reads (those with a G) and the join could be made if the other contig has a C at that position. After each join I re-examined the forward/reverse pairs for consistency and examined the assembly as a whole to ensure that the join was properly made.

However, many times making the join was not as easy as described above. Sometimes no string could be found which matched between the two contigs, the compare contigs window would show many high quality discrepancies that could not be resolved, or there were no spanning subclones between contigs. However, following the

rest of the procedure outlined above, joins could still be made. In the cases where joins could not be made, I used Strategy 6 (to be described later). Throughout my finishing, I made as many joins as possible, while continuing to minimize high quality discrepancies and maintain a high level of forward/reverse pair consistency.

Figure 6. Performing Tears and Forced Joins

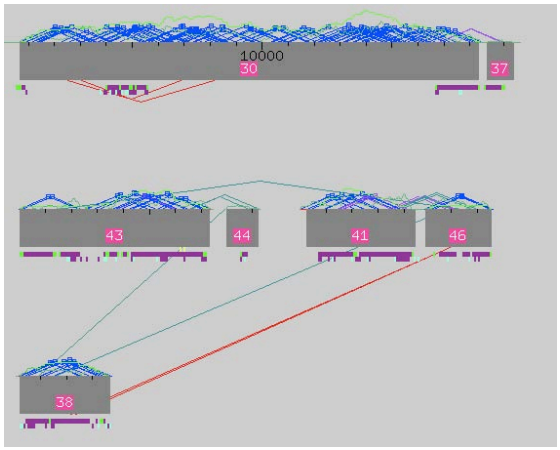

Figure 6a. Assembly view looking for tear and join locations

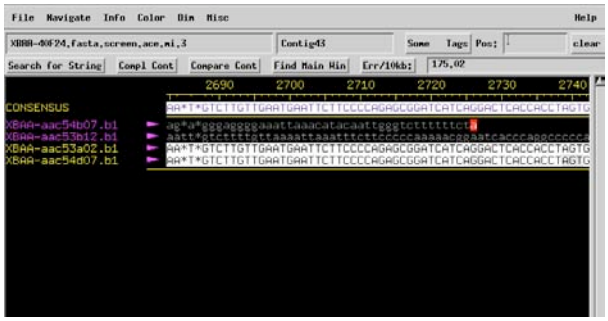

Figure 6b. Aligned reads window containing a possible tear site

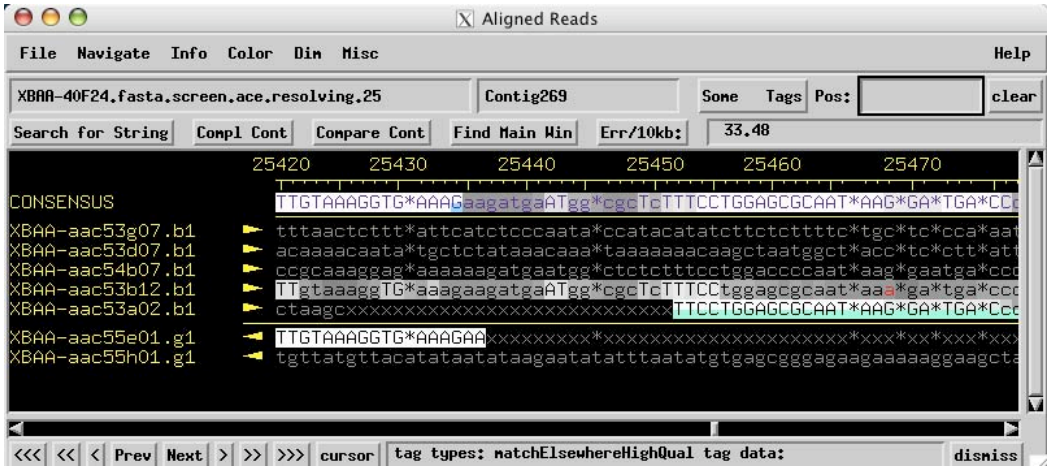

Figure 6c. Traces of possible tear site

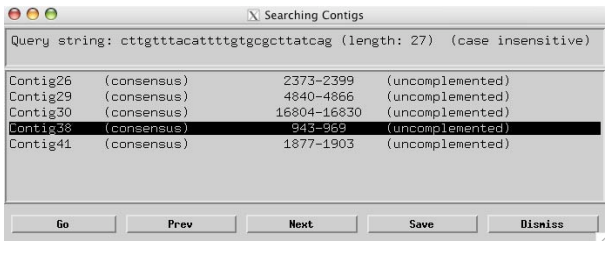

Figure 6d. Search for string to identify possible join site

Figure 6. Performing Tears and Forced Joins Continued

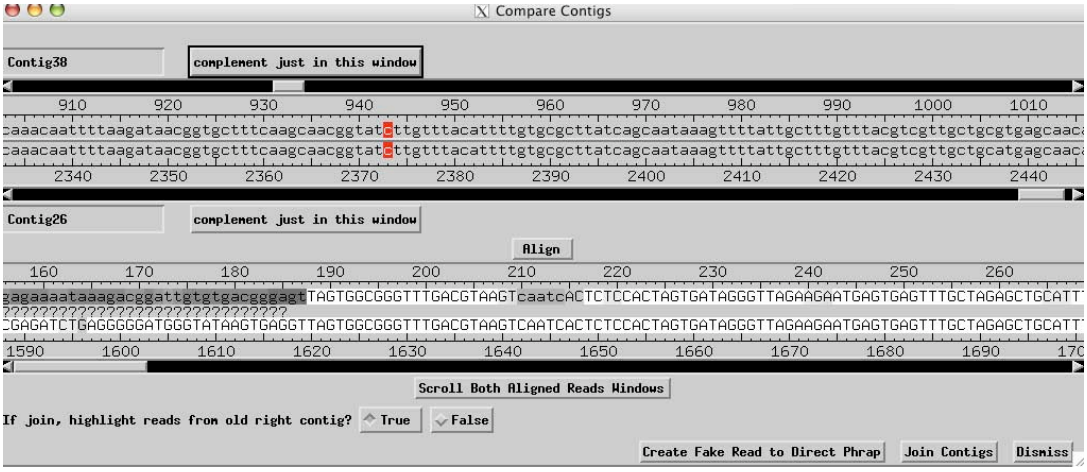

Figure 6e. Compare contigs window for possible join

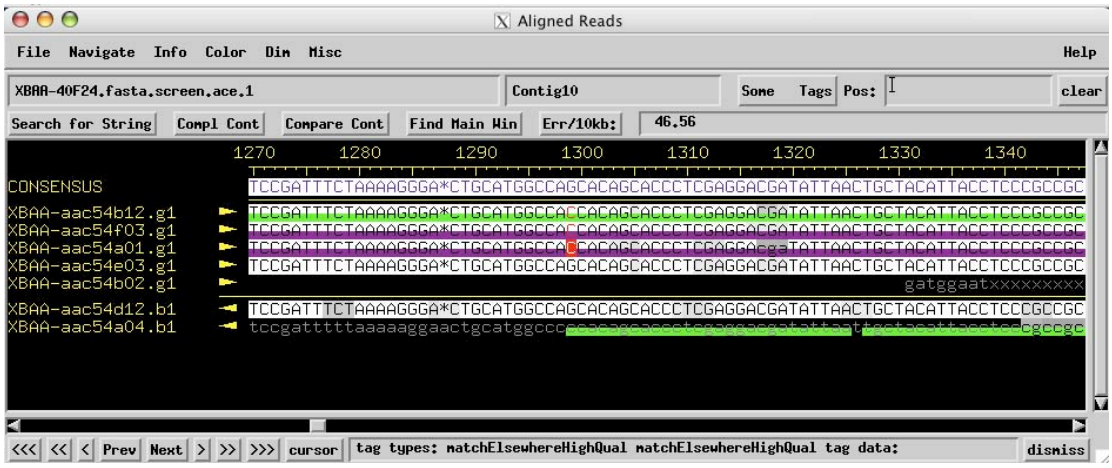

Figure 6f. High quality discrepancy where reads could be moved to make a join

Strategy 4: Locate Clone Ends

Another step in my assembly process was to locate the clone ends of my fosmid. This aided in my assembly process because I knew that any reads that continued past my ends were misassembled and belonged at another position within my fosmid. I knew that the cut sites for the clone were 'GATC' and that sequences matching to the *E. coli* vector were marked with an 'x'. Therefore, to identify the cloning ends of my fosmid, I searched for 'xxxxxxGATC' and 'GATCxxxxxx'. Because these reads had been incorporated into my assembly at various points, I had to go through my assembly manually looking at the aligned reads window and pulling out reads that contained the clone ends. I then placed these reads together and tried to make joins off of the appropriate ends of the contig (not the clone end) to attach the ends to the rest of the assembly (Figures 7a and 7b).

During this process, I discovered that both of my cloning ends were within the tandem repeat found in my fosmid. This contributed to my difficulty in identifying the clone ends. Also, I also ran into difficulty adding my ends to the assembly, as the reads all seem to have terminated at a given point in the repeat. At this point, therefore, it is impossible to join the clone ends to my assembly. However, many experiments are possible to finish this part of my fosmid; I will talk about these in the 'Remaining Problems' area of this paper. While the ends are not joined, I can say that the general layout of my fosmid consists of a segment of unique sequence sandwiched on both ends by a tandem repeat.

Figure 7. Locating Clone Ends

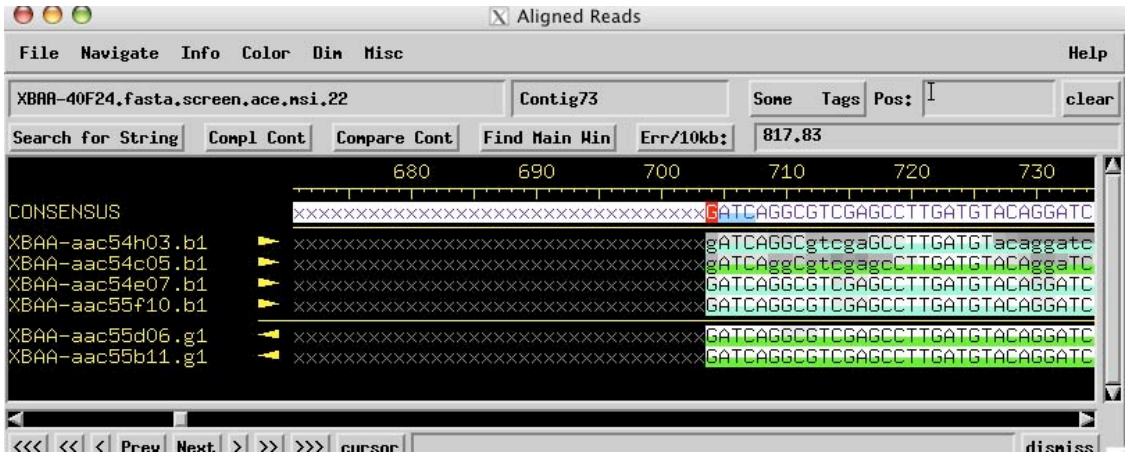

Figure 7a. Aligned reads window showing clone end

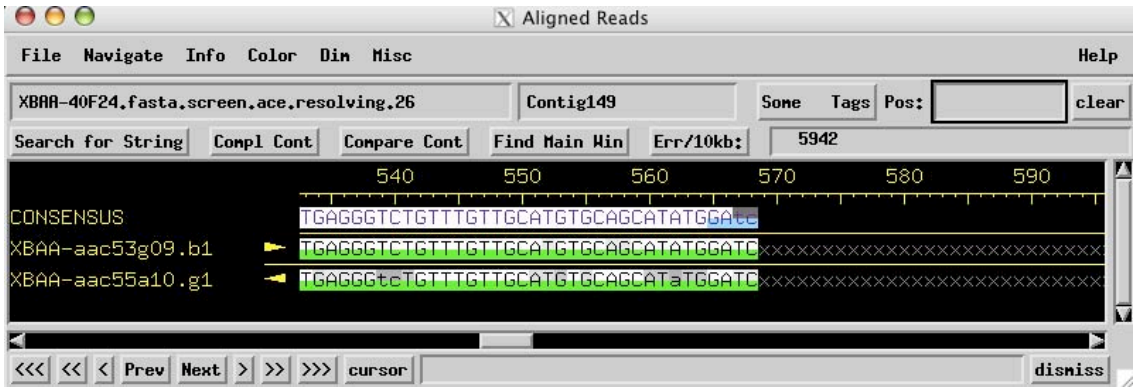

Figure 7b. Aligned reads window showing other clone end

Strategy 5: Call Reads To Resolve Gaps

At one point, I had an assembly that included several gaps spanned by multiple subclones (Figure 8a). Therefore, I called reads at the arrow points indicated to resolve these gaps, hoping to join the contigs to create a more finished assembly. The reads I called are listed in Table 1. I did ensure that the oligos were unique by using 'search for string' and finding no matches; this ensured that I would be sequencing the desired portion of the subclone without contamination due to multiple sites for primer annealing. When compared to **Autofinish**, the reads that I called were redundant in that they spanned that same gaps, while I ordered more reactions for each gap to try different chemistries and different subclones to guard against failed reads. **Autofinish** called 364 reactions to resolve low quality and single stranded areas, which I have not even begun to resolve yet in my assembly. At the time that I ordered these reactions, I was still unsure of the assembly, and knew that portions were still misassembled. Therefore, I was not sure that the reads I was calling actually spanned gaps. Due to this, calling reads was more of an exercise for me than an actual strategy. Had I been a professional finisher, I would not have called reads at this time, but would have waited until my assembly was more complete. However, as a student, I decided to call the reads as best I could to gain the experience of calling reads, even if ultimately they ended up not being necessary. Because of the unique nature of my fosmid, I have not included a table detailing the **Autofinish** reactions, as they did not affect my assembly or finishing of my fosmid. As I continued to work on finishing the assembly while waiting for reads, I did find that several of the subclones I had called reads on had been misassembled, so in these areas, the new data added coverage, but did not resolve a gap.

When I received the data, all of my reads were successful. However, all of the reads were misassembled when they were added as indicated by high quality discrepancies (Figures 8b and 8c). This occurred because the gaps all lay within tandem repeats; therefore **phrap** had been confused and put the reads in an area where they didn't belong. To resolve this, I pulled out each read that I had called, and individually force joined the read to that from the subclone read template used, which it must either follow or precede. After repeating this procedure for each read, I looked for the predicted joins among the resulting contigs. Ultimately, the assembly consisted of larger contigs and pulled up several other contigs that previously did not even appear in assembly view (Figure 8d).

Figure 8. Calling Reads to Resolve Gaps

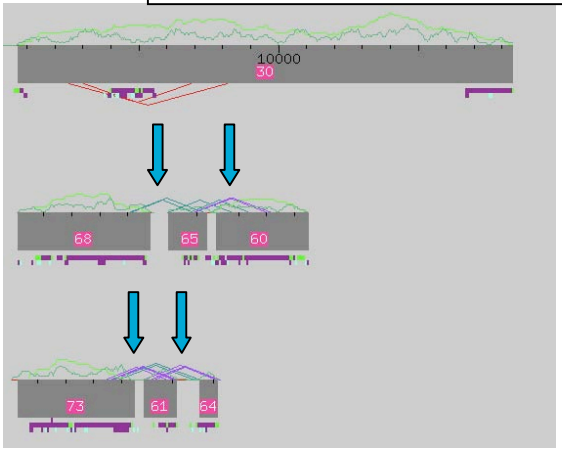

Figure 8a. Assembly view when reads were called

Figure 8. Calling Reads to Resolve Gaps Continued

Table 1. Reads called

Gap to Cover	Oligo	Sequence	Template	Chemistry
61-64	1	tgcgtcgtattttgaggtc	aac55e02, aac54g12	Big dye and dGTP
	2	caagttccgtgggctc	aac55e02, aac54g12	Big dye and dGTP
73-61	3	tgaatgggtggataactgg	aac53c08, aac55d10	Big dye and dGTP
	4	cagacacttttacacatttctcc	aac53c08, aac55d10	Big dye and dGTP
68-65	5	tgcatttaatgatttgaatgtgt	aac53b07, aac54h10	Big dye and dGTP
	6	cagacacttcgcatacatcg	aac53b07, aac54h10	Big dye and dGTP
60-68	7	tgattggctagcgggt	aac54e06, aac54b10	Big dye and dGTP
	8	aaactccaaaacagttaagggt	aac54e06, aac54b10	Big dye and dGTP

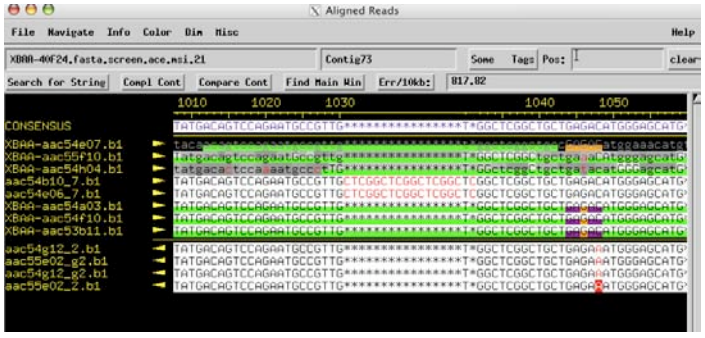

Figure 8b. Aligned reads window showing misassembled reads added

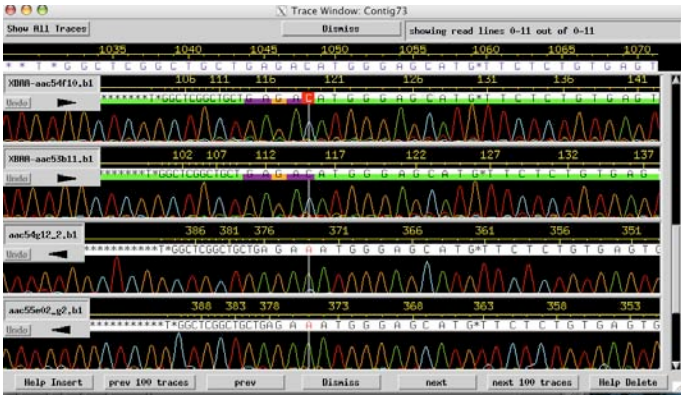

Figure 8c. Traces window showing misassembled reads added

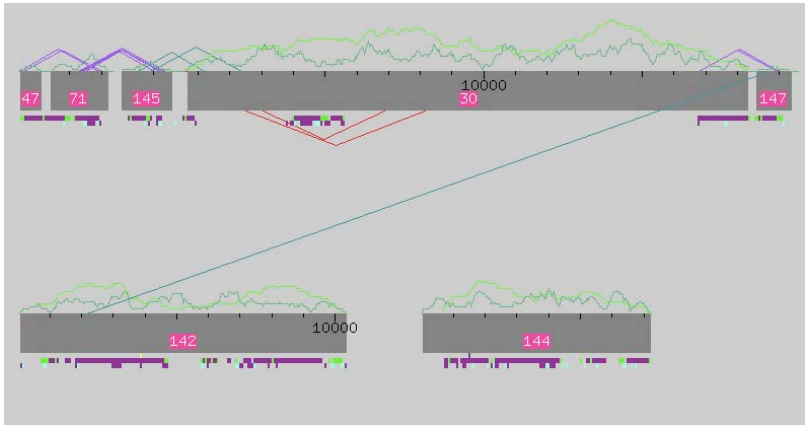

Figure 8d. Assembly view after reads added and joins made

Strategy 6: Steal Reads To Resolve Gaps

After the reads were added I continued attempting to reassemble my fosmid using the strategies outlined above as appropriate. I found that some of the gaps left were located in the tandem repeat area (Figure 9). Note that a purple tag, indicating the presence of a repeat at those locations, marks all of the contig ends. While it was impossible to join the gaps with the data I had at this point, it was possible to join contigs by stealing data from a corresponding repeat. Reads were only removed if their removal would not compromise the coverage of the actual location where they belonged. The fewest number of reads to cover the gap necessary to join the contigs in question were removed and relocated. In this manner, the gaps were resolved but new areas of low quality and high quality discrepancies resulted. I realize that the sequences in this area are not correct, but inserting sequence that is similar to what I believe is the correct sequence helps to give an overall view of my fosmid.

I identified possible reads to steal by looking at the contig ends that I was attempting to join. I performed searches for strings at the contig ends I wanted to join, and looked at the results at the same time (Figure 9a). I looked for places where both ends matched another area, with a reasonable distance between the two (~2000 bp). I would then look at that area in a matching contig to identify reads that could be pulled out of the assembly and reinserted into my gap (Figure 9b). Then I compared the matching contig to both contig ends I wanted to join (Figure 9c). Several high quality discrepancies resulted from these joins, as the reads do not belong here, but are my best estimate as to what the sequence should be. After the reads are stolen, the join is made following Strategy 3 (Figure 9d).

As this strategy introduces unreliable data back into the assembly, including high quality discrepancies, I only used this strategy due to the time limitations of my analysis. Ultimately, the stolen reads would be removed and replaced with new reads, which would be called to resolve the areas requiring more data. Or, given more time, this strategy would not have been used at all, and new reads would have been called initially. The purpose of this strategy was to get a general estimate for the number of repeat sequences present in my assembly, and the general layout of these repeat sequences within the overall assembly. This strategy was also used as a preemptive attempt to make the in silico digest data analysis more applicable. Because I believed that many of the remaining assembly errors were located within these repeats, and the similarities of these repeats (>95%), it was possible that the differences between the repeats would not cause different restriction cut sites to emerge. Therefore, a general repeat sequence could be inserted in an area known to lie within a repeat, but with the exact sequence currently unknown. This would give a good estimate as to the overall length of the assembly potentially without introducing or removing any restriction cut sites.

Figure 9. Stealing Reads to Resolve Gaps

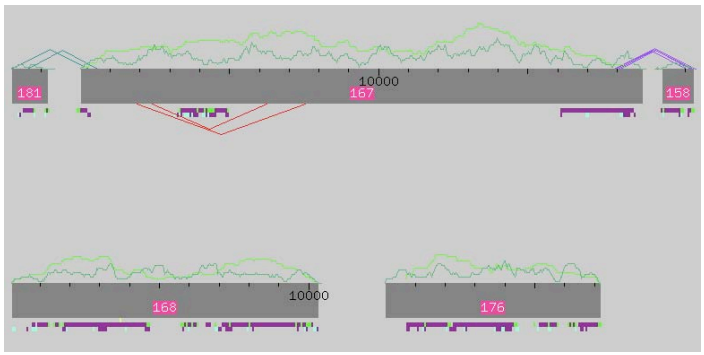

Figure 9a. Assembly before stealing reads

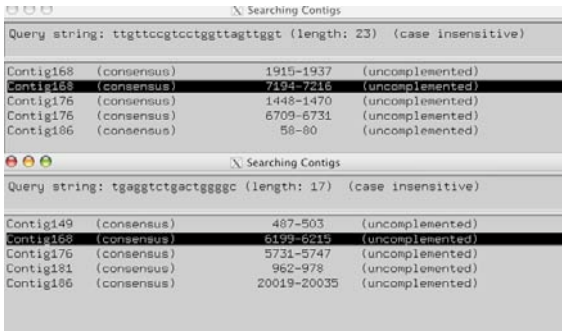

Figure 9b. Search for reads to steal

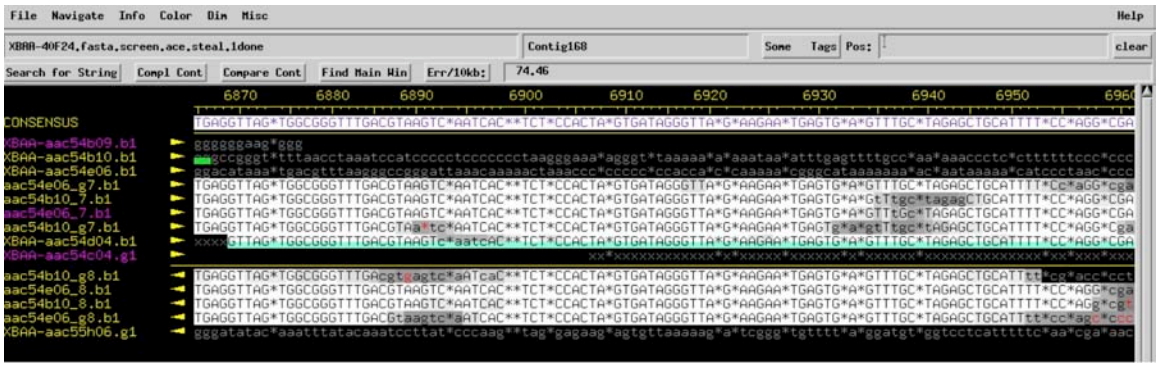

Figure 9c. Identifying reads to steal

Figure 9. Stealing Reads to Resolve Gaps Continued

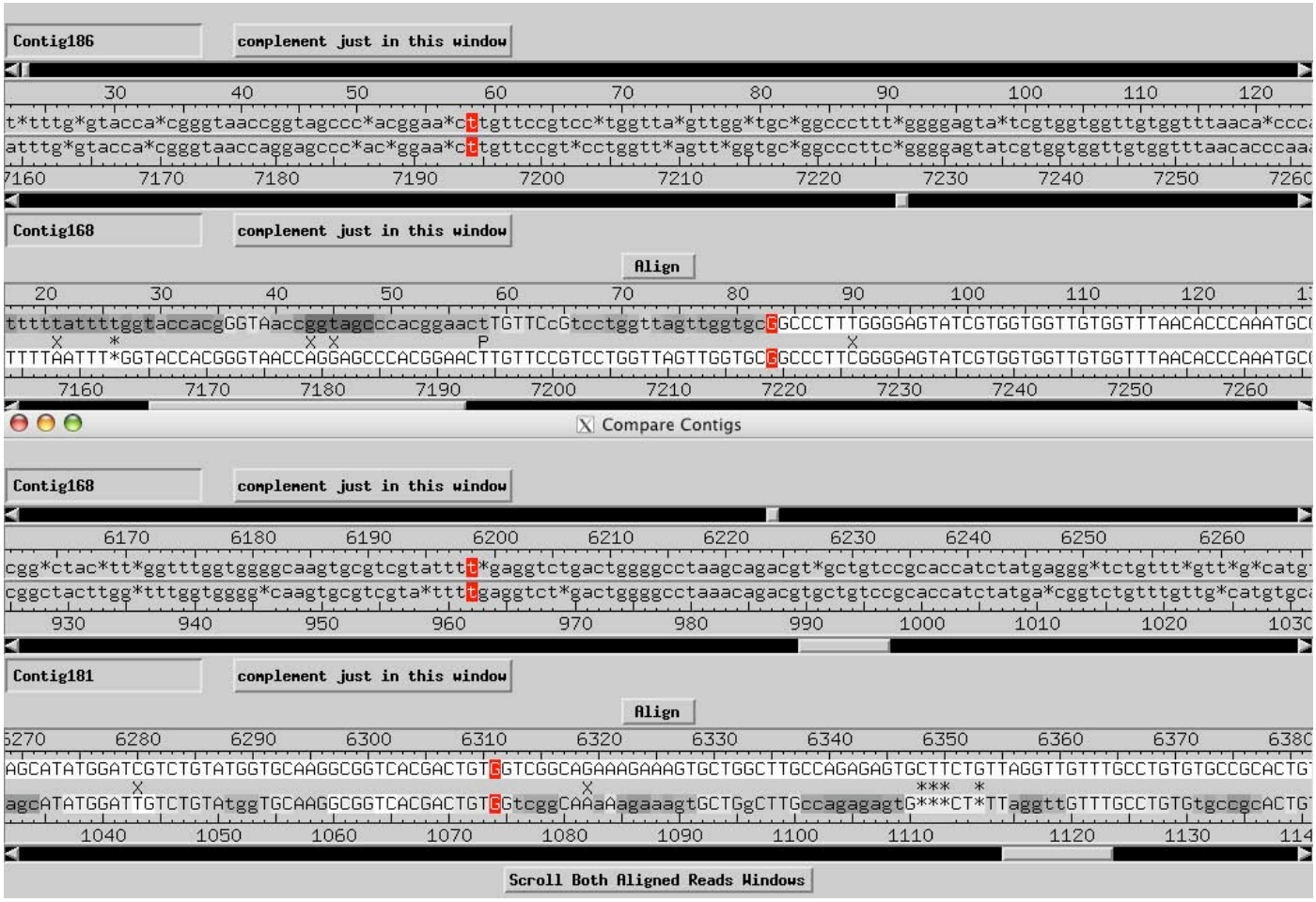

Figure 9d. Compare contigs for joining stolen reads to desired contigs

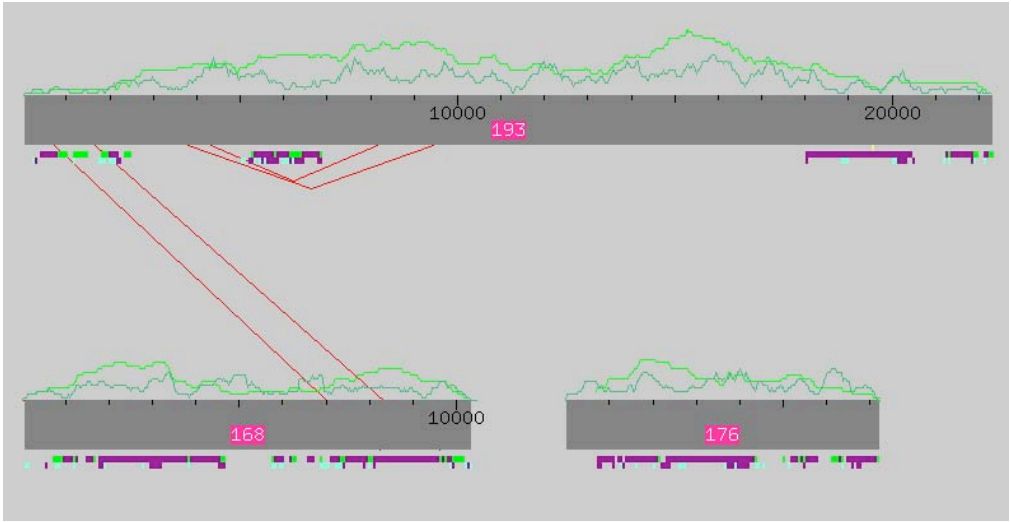

Figure 9e. Assembly after join made with stolen reads

Strategy 7: Analyze Repeats With PrintRepeats/Pal

After stealing reads where possible, I eventually had an assembly consisting of a single contig (Figure 10a). While my assembly was still not complete and included many areas in which data had been stolen from other areas to make joins (Strategy 6), I continued to investigate the repeats contained in my fosmid to the best of my ability. I ran the program **PrintRepeats**, which is a more advanced version of **Crossmatch**, that locates repeat sequences and gives a visual representation of where the repeats occur. However, for my fosmid, **PrintRepeats** identified too many repeat areas to be of use for analysis. To resolve this issue, I ran **Pal**. **Pal** is a similar program to **PrintRepeats**, but is run at a higher stringency. After several trials, I decided that **Pal100** gave the best resolution of my repeat (Figure 10b); a minimum sequence of a 100 bp match is required to be observed in the **Pal** printout. Using this analysis, a repeat of ~5 kb can be identified. According to **Pal**, there appear to be ~3 copies of this repeat in tandem at the end of my fosmid, and a little over 1 copy in the beginning of my fosmid. While this gives an idea of the length (~5 kb) of my repeat, and verifies that I probably have several copies of the same tandem repeat flanking the central unique sequence, this picture is not wholly accurate. There is also a smaller repeat in the middle of my unique segment that I was not able to analyze due to the time constraints. The total assembly is ~36 kb, which is dramatically shorter than the ~40 kb expected assembly. Therefore, it is possible that there are 1-2 more copies of the tandem repeat that is currently stacked together, producing the resulting misassembled portions of the fosmid. I have detailed approaches that could be taken to finish this fosmid to high quality in the "Remaining Problems" area of this paper.

Figure 10. Analyzing Repeats with PrintRepeats/Pal

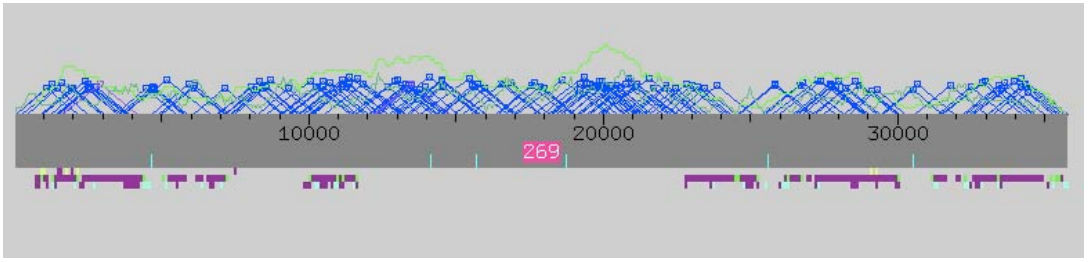

Figure 10a. Assembly into a single contig

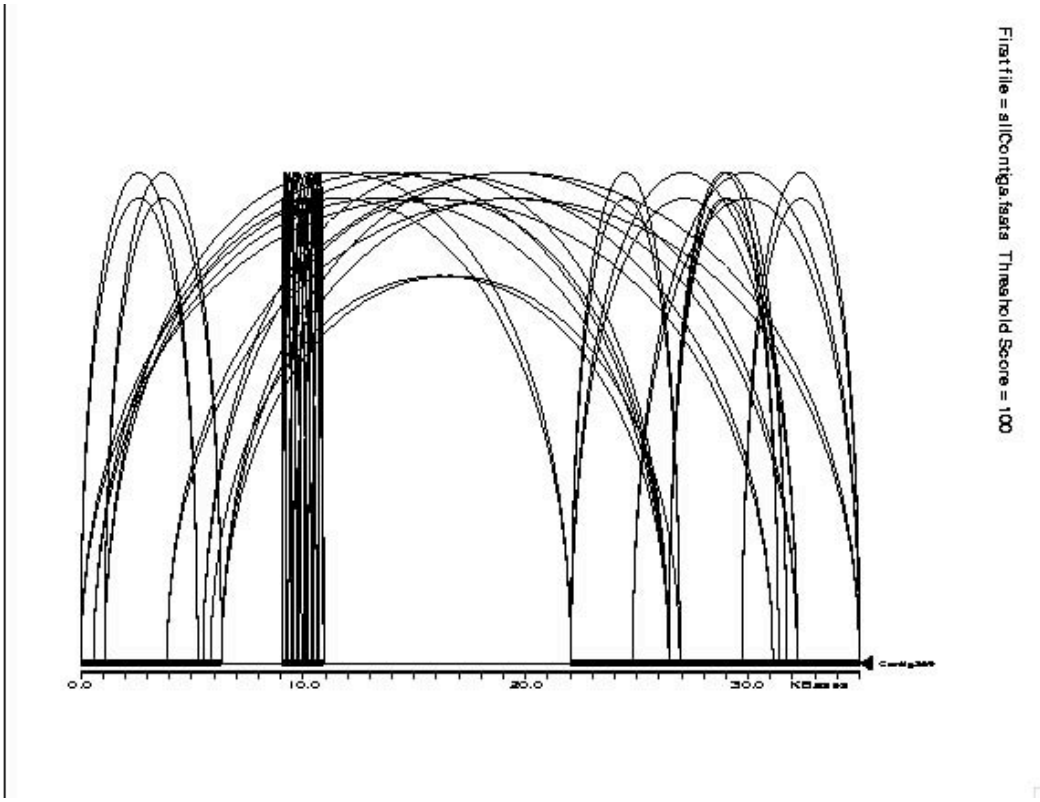

Figure 10b. Pal100 on assembly into a single contig

Strategy 8: Digest Analysis

Digest data that was run for me at the Genome Sequencing Center on the original fosmid was available for my analysis. Digests with restriction enzymes that cut at rare (not frequent) sites are useful to confirm finished sequence. The resulting bands are size separated and compared to the in-silico digest bands that are predicted from the current assembly. For the assembly to be finished at high quality, the bands in the in-silico digest should align to those of the real digest within 10% of the real band size.

The digests of my assembly did not correspond well to the real digests (Figures 11a and 11b). The quality of the real digest was not very high, determined from looking at the actual gel, meaning that some of the bands on the real digest may have been miscalled or not called at all. However, the main reason my in-silico digests were not close to the real digests was the remaining misassembled areas in my current assembly. If my ends had been properly connected to the unique sequence, the in-silico digest should align more closely to the real digest. However, as I know that my current assembly is not even the correct length, even the total kb given is not helpful for analyzing my assembly. Also, it is very likely that there remain misassembled areas in the tandem repeat areas. However, the bands that do match, particularly in the *Eco*RI digest, suggest that the unique area of my fosmid has been properly finished.

Figure 11. Digest Analysis

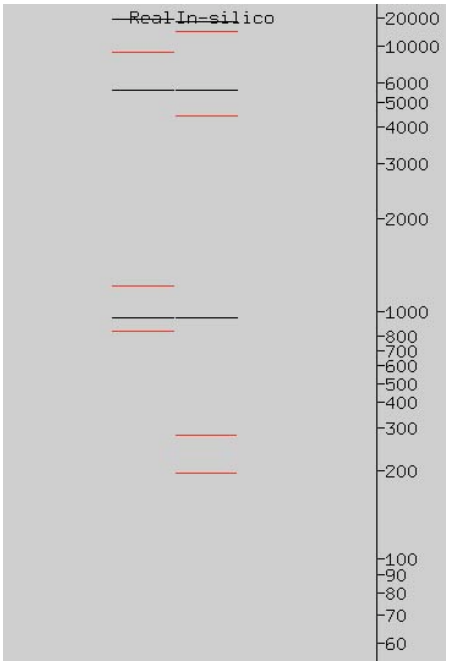

Figure 11a. *Hind*III Digest

Figure 11b. *Eco*RI Digest

Remaining Problems:

While I have worked towards completely finishing my fosmid, many problem areas are still present in the current assembly (Figure 12a). The most critical remaining problem is remaining misassembled areas. Neither end has been connected properly to the unique sequence at this time. The beginning end was force joined, to make the assembly easier to analyze, though I know that the assembly is missing a segment of sequence, as high quality discrepancies occurred after the join (Figure 12b). If I had more time, I would call reads walking off the ends using subclones that are definitely at the ends of the fosmid as templates. Hopefully, the reads would be consistent off of different subclones and the addition of ~700-800 bps of data would be enough to connect the ends to the unique segments. This is the area that needs the most work in my assembly.

The next problem that I would address would be to call reads to cover the other forced joins that I made, especially those that were resolved by stealing reads. I would again call reads off of sublclones to both increase the overall coverage, and determine whether or not the joins were appropriate. These reads are needed due to the large number of force joins that were made to resolve the assembly problems caused by such long tandem repeats.

If I had the time and money, I would then address the remaining problem areas in regards to single-stranded chemistry and low quality sequence (Figures 12c and 12d). Hopefully many of the problem areas that currently exist would be resolved by completing the assembly process. A way to reduce the cost of these experiments would be to re-use oligos that have already been ordered, though attention would have to be paid to ensure that the oligo was specific for proper annealing to the chosen subclone.

After all of these reads have been called and the data added, the assembly would be re-submitted to a digest analysis. If the in-silico results were consistent with the real digest, I would tag the assembly as containing tandem repeats, and declare the fosmid to be finished at the highest quality now possible.

Figure 12. Remaining Problems

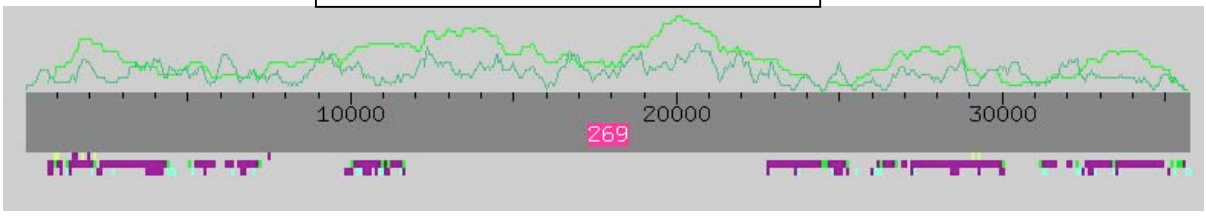

Figure 12a. Final/current assembly

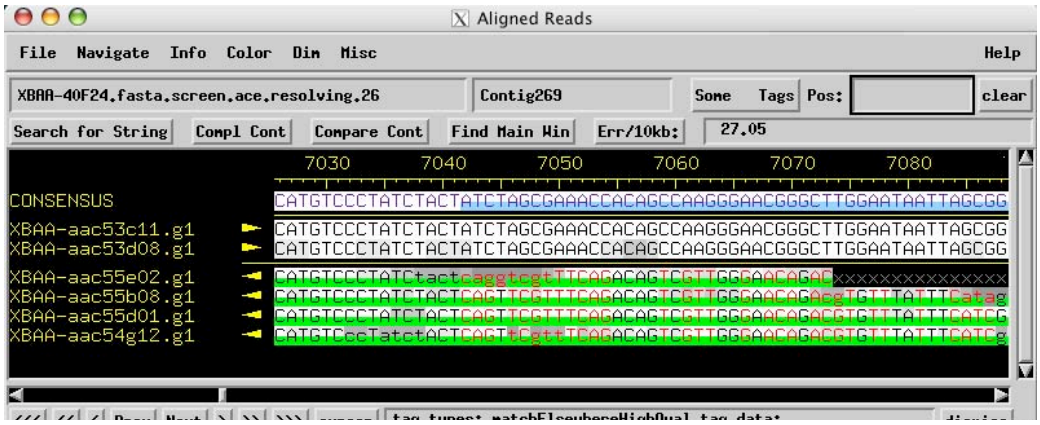

Figure 12b. Site of force join to add left end

Figure 12. Remaining Problems Continued

Regions Covered by Only 1 Strand and Only 1 Chemistry (or None at All)

Contig		Start Pos	End Pos	
Contig269	(consensus)	1	38	40 bp
Contig269	(consensus)	880	1287	446 bp
Contig269	(consensus)	2486	3478	1036 bp
Contig269	(consensus)	4786	4855	70 bp
Contig269	(consensus)	5772	5827	77 bp
Contig269	(consensus)	6939	6952	16 bp
Contig269	(consensus)	7042	7363	322 bp
Contig269	(consensus)	16344	16368	27 bp
Contig269	(consensus)	17006	17035	30 bp
Contig269	(consensus)	23625	24620	1057 bp
Contig269	(consensus)	25437	25771	366 bp
Contig269	(consensus)	26095	26305	221 bp
Contig269	(consensus)	29210	29475	308 bp
Contig269	(consensus)	30683	30808	128 bp
Contig269	(consensus)	34251	35717	1583 bp

Go | Prev | Next | Save | Dismiss

Figure 12c. Remaining single-stranded areas

Low consensus quality (<=25 or 98)

Contig Name	Read Name	Consensus Positions	
Contig269	(consensus)	1-27	base quality below threshold
Contig269	(consensus)	323-325	base quality below threshold
Contig269	(consensus)	386-388	base quality below threshold
Contig269	(consensus)	624-626	base quality below threshold
Contig269	(consensus)	684-686	base quality below threshold
Contig269	(consensus)	702-703	base quality below threshold
Contig269	(consensus)	24419-24504	base quality below threshold
Contig269	(consensus)	24512-24516	base quality below threshold
Contig269	(consensus)	24523-24534	base quality below threshold
Contig269	(consensus)	24536-24542	base quality below threshold
Contig269	(consensus)	24556-24567	base quality below threshold
Contig269	(consensus)	24582-24584	base quality below threshold
Contig269	(consensus)	25435-25441	base quality below threshold
Contig269	(consensus)	25444-25448	base quality below threshold
Contig269	(consensus)	25450	base quality below threshold
Contig269	(consensus)	25477-25482	base quality below threshold
Contig269	(consensus)	25484	base quality below threshold
Contig269	(consensus)	26354-26355	base quality below threshold
Contig269	(consensus)	26389-26390	base quality below threshold
Contig269	(consensus)	26410-26412	base quality below threshold
Contig269	(consensus)	26459-26463	base quality below threshold
Contig269	(consensus)	26475-26479	base quality below threshold
Contig269	(consensus)	34577	base quality below threshold
Contig269	(consensus)	34588-34590	base quality below threshold
Contig269	(consensus)	34592-34595	base quality below threshold
Contig269	(consensus)	34637-34639	base quality below threshold
Contig269	(consensus)	35648	base quality below threshold
Contig269	(consensus)	35684-35685	base quality below threshold
Contig269	(consensus)	35716	base quality below threshold

Go | Prev | Next | Save | Dismiss

Figure 12d. Remaining low quality areas