

# Finishing Repetitive Regions Automatically with Dupfinisher

Cliff S. Han<sup>1</sup>, Patrick Chain<sup>2</sup>

<sup>1</sup> University of California, Los Alamos National Laboratory, DOE Joint Genome Institute, Biosciences Division, M888, Los Alamos, NM 87545. <sup>2</sup> University of California, Lawrence Livermore National Laboratory, DOE Joint Genome Institute, 7000 East Ave. L-441, Livermore, CA 94550

*Corresponding Author:* Dr. Cliff S. Han, Los Alamos National Laboratory, Mail Stop: M888, Los Alamos, NM 87545. Phone: (505) 665-1724; FAX: (505) 665-3024; Email: han\_cliff@lanl.gov

*Key words:* Genome finishing, repetitive sequence.

## Abstract

Currently, the genome sequencing community is producing shotgun sequence data at a very high rate, but genome finishing is not keeping pace, even with the help from several automated finishing tools, such as autoFinish. One reason for the slow progress in finishing is that repetitive regions longer than the length of a sequencing read cannot be assembled correctly with many current assembly tools. Therefore, most repeat regions have to be checked manually. If finishing rates are to increase further, most repetitive regions must be assembled correctly and be finished in an automated fashion. The Dupfinisher computer program is designed to finish repeats with minimal human interaction. It can automatically detect repetitive regions, assemble each repeat individually using paired draft reads and primer walk reads, check the quality of these subassemblies, create artificial joins for finished and properly assembled repeats and run automated gap closure scripts on unfinished subassemblies. Dupfinisher is able to solve the majority of repeats in a microbial genome automatically, thus greatly reducing the amount of human attention needed. Dupfinisher has now been used in finishing more than 60 genomes and can be adapted to aid in finishing processes for whole bacterial genome and large insert clone projects.

## INTRODUCTION

Shotgun sequencing has become the standard strategy in most genome sequencing centers, whether they involve small bacterial genomes or bigger eukaryotic ones. The rapid pace with which genome sequencing has progressed, particularly for microbial genomes,

has prompted us to consider faster automated ways to obtain a properly assembled product that represents the genome in question. For example, there are currently 273 published complete genomes and another 724 ongoing prokaryotic genome sequencing projects (<http://www.genomesonline.org>), despite the first published genome being released only a decade ago. Shotgun sequencing is usually carried out from one or both ends of clones derived from several libraries that contain inserts of different sizes (e.g. the Joint Genome Institute uses 3 kb, 8 kb, and 40 kb libraries). Data from one end (a sequencing run) is called a read and paired (or sister) reads are derived from the same clone. Thousands of reads are assembled together with specialized computer software tools (genome assemblers). Theoretically 10 fold coverage of a genome will cover more than 99.99%, however there are almost always gaps and low quality regions that remain in assemblies. These gaps may be due to biases in clonability of particular DNA fragments, misassemblies across repetitive regions, or may simply be random. The purpose of finishing is to close all gaps, to enhance the quality and to resolve misassemblies in an assembly.

Assembly programs such as Phrap (P. Green, U. Washington) or Cap3 (1) use Smith-Waterman-like dynamic programming algorithms (2) to determine overlaps of sequencing reads and generate contiguous sequences (contigs). As these assembly algorithms are solely based on sequence information and individual base quality scores, the programs discard relevant information such as clone size, read orientation, and read pair information. Several modern assemblers such as ARACHNE (3,4) Euler

assembler (5) CELERA assembler (6) PHUSION (7), RePS (8), JAZZ (9) and PCAP (10) make use of paired read information. However many of these assemblers were developed for specific types of computer platform. Phrap (including parallel version from Southwest Parallel Software) and PGA are the two assemblers available to many different platforms. Phrap remains the most commonly used assembly program for bacterial genome assembly for its close coupling with finishing tool of Consed package. While Phrap has demonstrated its robustness and its ability to assemble data in the least number of contigs(11,12), it is particularly susceptible to assembly mistakes that arise from repetitive sequences in a genome. Therefore, a manual inspection of misplaced reads originating from artificial algorithmic effects of these assemblers is necessary. Several software packages are available that allow users to visualize assemblies, often stored as ace files which is the standard output file of Phrap and other assemblers. Most of these packages are limited to the display of sequence read positions and do not allow taking advantage of the information arising from read pairs, resulting in a requirement for auxiliary software. The two most commonly used packages that do allow display of read pair information are Gap4 (13) and Consed (14), the latter of which is the only package that both allows the display of read pair information on a genomic level and interfaces directly with the Phrap assembler. In the Consed reads alignment window, reads in potentially repetitive regions are highlighted by default. A recently developed program, ReDiT (15), could add tags to an ace file for potential repeats as well. In the Consed assembly view, misplaced read pairs can be semi-manually sequestered into a mini-assembly. Some repeats (or duplications) can be corrected in this fashion. However most duplications must still be resolved by sequencing from repeat-flanking clones or by PCR confirmation across each duplication. This manual process of resolving duplications is incredibly time consuming and expensive.

In order to relieve much of the problems currently associated with the finishing process, including resolution of repetitive regions, Dupfinisher has been designed to achieve the following objectives: automatically detecting repeats using the BLAST program (16); pairing repeat-flanking unique sequences such that the two unique regions share the largest number of common templates mutually; assembling paired unique regions together; checking the quality of those subassemblies; making artificial reads for finished duplications (which are used to guide future assemblies of the main project); and

identifying and tracking standard finishing reactions to complete those subassemblies with unfinished repeats. Dupfinisher can be configured to perform selected or all combinations of these tasks. Dupfinisher has been used in our production finishing line, which utilizes the Phred/Phrap/Consed suite (14,17), since autumn of 2003. We have found that Dupfinisher resolves 70 - 90% of the duplications in a prokaryotic genome without any human interaction, thereby significantly reducing the time and cost typically associated with finishing microbial genomes.

## Results and Discussion

We have used Dupfinisher in our microbial genome finishing pipeline since 2003 in finishing 60 bacterial or archaeal genomes. The automated finishing process that engages Dupfinisher includes two individual cycles of repeat resolution with Dupfinisher, and two cycles that combine Consed autoFinish and repeat resolution with Dupfinisher.

The number of repeats in each of the 60 genomes varies from fewer than ten to several hundred copies. We note that the final number of repeats for many of the genomes have not yet been fully resolved, however the sizes of repeats are very similar, 2 – 3 kb on average with 5 – 8 kb as maximal length. Most of the shorter ones are insertion sequences or transposons and the longest ones are usually ribosomal DNA sequences. The consensus sequences of repeats are stored in a file which is used along with a modified tagRepeats.perl from the Consed (14) package to tag the repeats in a project.

Dupfinisher resolves repeats individually in subprojects. Each of the subprojects is verified and categorized as either: 1) finished, which means that the repeat flanked by the two unique anchors resides in a single contig of high quality bases throughout the repeat; 2) low quality, which means that the repeat flanked by the two unique anchors resides in a single contig but there are low quality bases within the repeat region; 3) unfinished, which means that the two unique anchors are not in one contig and that there must be at least one gap between the two anchors; or 4) misassembled, which means that the two unique anchors reside in a single contig with some misassembled paired reads. Approximately half of the repeats within a genome project were resolved with only the original shotgun reads. Most duplications in the unfinished category, or those of low quality are then resolved by primer walking, and generally, 90% of unfinished repeats are finished correctly after two rounds of autoFinish, where

primer walking is used because it is the cheapest finishing method. The automatic and semi-automatic resolution of such a large fraction of the repetitive regions within a genome dramatically reduces the

**Table 1.** Comparison of contig numbers of using of Dupfinisher to PGA assembly.

Genome	Phrap	Dupfin and phrap	PGA	Number of duplications in the genome
<i>Methanospirillum hungateii</i>	39	15	36	70
<i>Rhodofereus ferrireducens</i>	79	79	70	29
<i>Shewanella baltica</i> OS1155	182	155	164	144

Misassembled repeats in subprojects are detected but cannot be corrected by the current version of Dupfinisher and have to be corrected manually. Identical repeat clusters that are only separated by short unique sequences are likely misassembled and long repeats (> 3 kb) tend to result in subprojects of the “low quality” and “unfinished” categories.

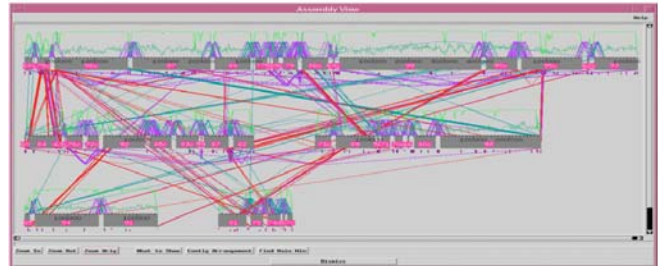
As shown in Table 1, after running Dupfinisher with JGI drafted genome sequencing projects (only shotgun reads), assemblies were improved with reduced numbers of contigs compare to assemblies with Phrap only. And in two of the three projects tested with PGA, the assemblies from combination of Phrap and Dupfinisher were better than those from PGA. For example, in *Methanospirillum hungateii*, a significant improvement was observed; the number of contigs was reduced from 39 to 15 and the number of scaffolds from five to one after the initial two runs of Dupfinisher (Figure 1). In two other projects, *Psychrobacter cryopegella* and *Sphingopyxis alaskensis* RB2256, the scaffold numbers increased after two runs of Dupfinisher, however this is due to breaking numerous false joins between repeats (misassemblies).

Dupfinisher is sometimes unable to completely finish all repeats in a project correctly, thus it is typically used immediately after the shotgun draft stage. After the Dupfinisher + autoFinish stage, the repeat regions should be checked manually. Repeat consensus sequences should be removed if there is any concern of misassembly. Dupfinisher can be run again after a new assembly for the project has been done. All subprojects should be checked manually before moving any artificial reads for finished repeats into the main project.

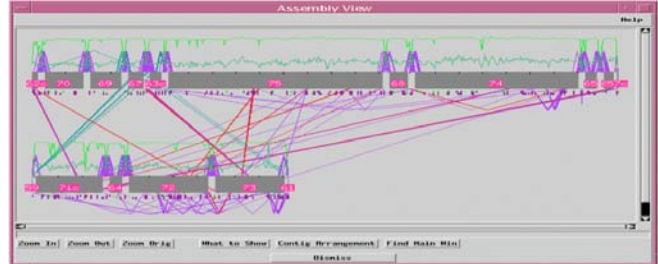
Due the the method of detecting methods used by Dupfinisher, tandem repeats is going to be likely missed. This type of repeats is also likely not being able to be resolved with paired end information. We intend to leave this type repeats for human finisher to call transposon bombing to resolve them.

amount of manual time that is normally spent on these issues, and thereby allows individuals to focus more of their attention and time on the remaining problematic regions.

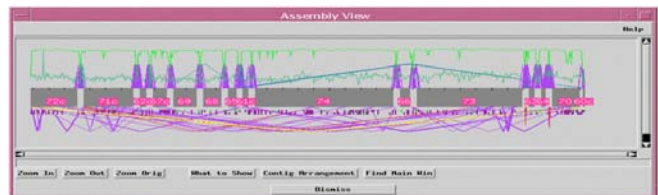
The success of Dupfinisher and this technique depends on several items: the size of the clones (which themselves are required to be sequenced from



A.



B.



C.

**Figure 1.** Assembly views of the genome of *Methanospirillum hungateii* before and after runs of Dupfinisher. **A.** View of assembly with shotgun draft reads before running Dupfinisher. **B.** View of assembly after the first run of Dupfinisher. **C.** View of assembly after the second run of Dupfinisher. Gray horizontal bars in the middle of each row represent contigs that are joined in scaffolds by lines above the bars. Lines between or within contigs indicate possible misassembled paired reads.

both ends, to have read pairs), the shotgun draft coverage of the genome in question, as well as the size and complexity of the duplications that cause assembly problems for the genome in question. The JGI microbial sequencing process is standardized and eliminates the first two concerns; while the latter concern is genome-specific. A limitation of this program is that the name of draft and finishing reads must include information for the template used and the reaction type. Briefly, the template name for drafting reads is the letter and numbers before the first dot, and the reaction type is represented by the first letter after the first dot. An avenue for improvement might be to use the information for templates and reaction types stored in the phd files. Furthermore Dupfinisher does not track repeats and repeat families from run to run. It would be highly useful to track a particular repetitive region across different assemblies, therefore another avenue for improvement would be if the program could name the same repetitive sequences consistently so that the repeat will be tagged with the same name.

The results presented in this study show that Dupfinisher provides a powerful and useful genome finishing tool for microbial genomics projects. Dupfinisher is a tool that automatically detects and resolves misassemblies due to repeat regions, and has significantly contributed to increasing the efficiency of the microbial genome closure process. After microbial genomes are processed with Dupfinisher, the end product is an assembly with fewer gaps, few if any misassembled regions, and output directing the attention toward the remaining problematic regions. This program is freely available (please contact [han\\_cliff@lanl.gov](mailto:han_cliff@lanl.gov)) and has been tested with more than 50 microbial genomes, including one fungal genome, on Sparc machines with Solaris versions 5.8, and 5.9. The results of the implementation of Dupfinisher in the LANL genomic finishing pipeline suggest that it will serve as a valuable finishing tool that can considerably enhance the efficiency of genome finishing.

## System and Methods

### Dupfinisher Components

Dupfinisher consists of two components: a parameter file and a program written in the PERL programming language. PERL was chosen because of its superiority as a parsing language, its widespread availability, and ease of development. The parameter file contains parameters that can be modified according to user-specific needs. The parameter file should be in at least one of the three locations: a copy

in the project edit\_dir directory where the program is run, a personal copy in the user's home directory (defined by \$HOME), and/or a master copy in the Consed home directory defined by \$CONSED\_HOME/bin. Parameter values in the copy residing in the specific project directory supercedes the personal copy, and the personal copy supercedes the master copy.

The Dupfinisher program schema is shown in Figure 2 and was designed to be used with the Consed package. The Dupfinisher program requires one input file from the command line: an ace file which contains most of the information needed by Dupfinisher. The ace file can be generated with Phrap or compatible assemblers, such as parallel Phrap (High Performance Software, G. Montry, unpublished). Dupfinisher will collect information about contigs and reads per contig from the ace file. Other inputs are from the parameter file. Dupfinisher calls several programs that must be installed and available from the unix system's command line. They include blastall and formatdb from the National Center for Biotechnology Information's (NCBI) blast package (16), and several scripts from the Phred/Phrap/Consed package: phredPhrap, and Consed autoFinish, and ace2Oligo.perl. The NCBI Blast can be replaced by other versions of Blast as long as the two outputs are compatible. The Boulder Perl library package (Lincoln Stein, <http://stein.cshl.org/software/boulder/>) must also be installed to parse the Blast results.

The major output from Dupfinisher includes individual assemblies for each of the repeat copies or for unpaired repetitive ends. These subassemblies are collected in a single directory named dfRun#, where the # represents the cycle run for Dupfinisher within the project. Together with these individual subassemblies, there are three text files that summarize the results, a directory containing artificial reads representing finished repeats and a directory containing files of finishing reactions for unfinished repeats.

### Repeat detection

Repetitive regions in an assembly are determined by blast search. Consensus sequences of "good" contigs, defined by parameters in the parameter file, are extracted from the ace file and saved in a fasta file. A self Blast search is performed with this fasta file (ie. used both as query and database) and is done with "-F F -W 12 -e 1e-10" parameters, which act to speed up the process while keeping the sensitivity high enough to find all repetitive regions of interest (those that cause misassemblies). Blast results are parsed with scripts from the Boulder Perl package. Non-self

(mapped by contig and position) blast hits that are above the criteria set in the parameter file will be collected as repetitive regions. By default, a hit that is longer than 500 bases and with greater than 95 percent identity will be collected. The repetitive regions are sorted according to contigs and locations within a contig. The repeats are then compared with each other and grouped according to sequence similarity (> 95%). The longest sequence in a repeat family is used as the consensus of the group. Consensus sequences of repeats are saved, and can be used to tag repetitive regions in future assemblies.

### Unique anchor determination

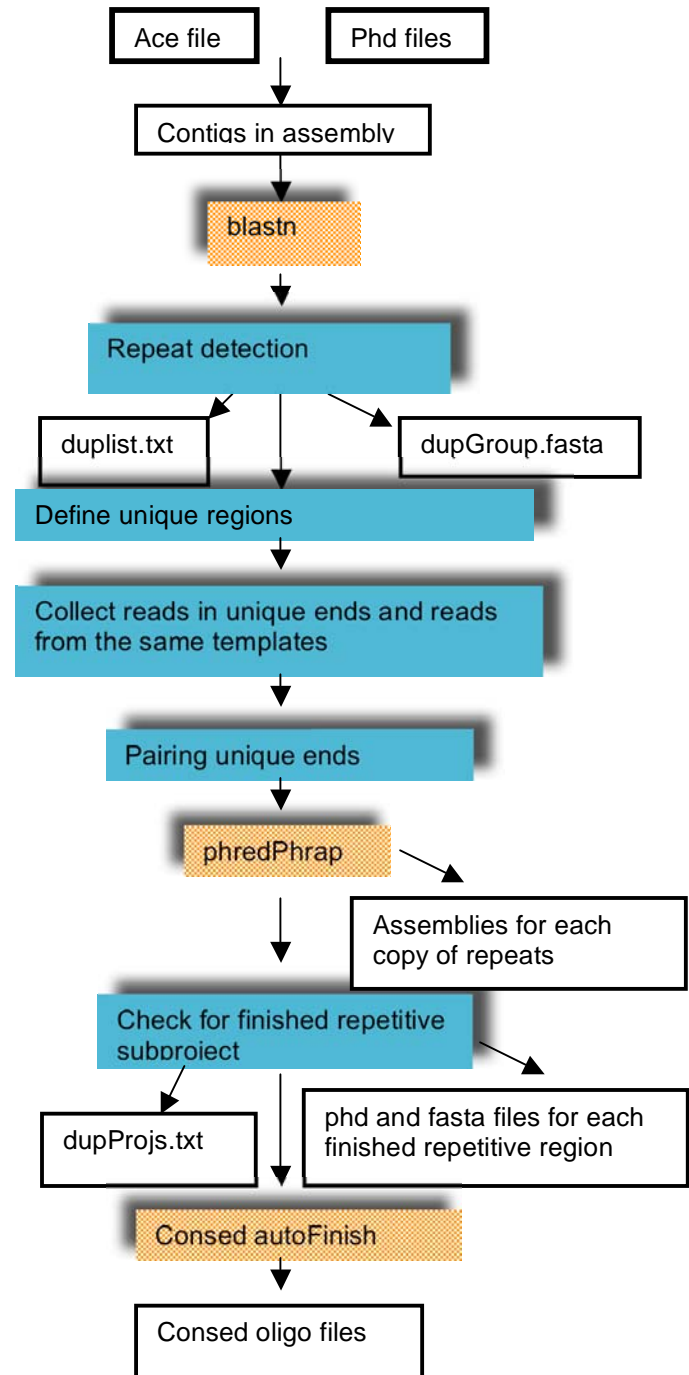
Repeats can only be resolved when there is enough unique sequence (called anchors) adjacent to them. Paired shotgun and primer walking sequences on the same template as the reads in the anchor are key to resolving repetitive regions. Depending on the average insert size of the shotgun library and the distribution of repeats, unique regions used to resolve a repeat can be large or small. By default, we use anchors of 10 kb if possible as most of our drafting reads come from 3 kb and 8 kb insert libraries. An anchor that bridges two repeats will be served twice, targeting a repeat at each of its ends. Reads whose sequences align to the consensus sequence of an anchor, will be collected. If a read goes in the direction where the target repeat is located, its paired shotgun read and any primer walk reads generated from the same template will be collected as well to make individual assemblies for each repeat copy. A unique sequence of 50 bases adjacent to the target repeat is stored in the edit\_dir directory of the subproject for the purpose of possible quality assessment of the assembly by human finishers.

### Assembling subprojects for each repeat

Before an assembly for a repeat can be performed, the two anchors flanking the repeat have to be determined. The unique anchors will be paired together according to common reads shared between them. The two anchors sharing the greatest number of common reads can be paired together to assemble the repeat they flank. Another condition to be considered in the first round of pairing anchors is that the repeat regions to be solved by the two unique anchors have to be classified into the same repeat family. Reads within the two collections will be used to make a subassembly for the target repeat using phredPhrap from the Consed package. Repeats that have been resolved by Dupfinisher in previous cycles will not be assembled again.

The subassemblies generated for repeats are checked automatically for quality and to determine if the target repeats have been assembled into one contig. Paired

reads are checked for correct orientation. Dupfinisher makes an artificial read for each finished repeat and adds a doNotFinish tag to the proper regions of contigs in those unfinished assemblies and subsequently runs autoFinish on them.



**Figure 2.** Processing scheme used by Dupfinisher. Input and output files are in white boxes. Functions performed by other programs are in gray boxes.

Functions performed by Dupfinisher are marked in dark gray boxes.

Oligonucleotide files that contain information for primer walking reactions generated by autoFinish will be copied into a common location for the convenience of human finishers.

### Finishing procedures with Dupfinisher

Here is the LANL finishing procedure involving Dupfinisher: 1) run Dupfinisher on the assembly ace file; 2) put the artificial reads generated by Dupfinisher into the main project; 3) assemble with parallel Phrap; 4) repeat steps 1-3 with the new ace file; 5) run Consed autoFinish on the main project and do only primer walks from the main project and those from subprojects of unfinished repeats; 6) repeat step 4; 7) run autoFinish using primer walks for the main project and those from subprojects of unfinished repeats and use PCR to close gaps between scaffolds in main project; 8) repeat step 4; 9) perform manual finishing including closing gaps, resolving low quality and single clone coverage regions and checking repeat resolutions from Dupfinisher.

### Acknowledgements

We thank David Sims for reading and commenting this manuscript. This program is supported by the U. S. Department of Energy under the contract No. W-7405-ENG-36 and under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

### References:

1. Huang, X. and Madan, A. (1999) CAP3: A DNA sequence assembly program. *Genome Res*, **9**, 868-877.
2. Waterman, M.S. and Eggert, M. (1987) A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *J Mol Biol*, **197**, 723-728.
3. Batzoglu, S., Jaffe, D.B., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J.P. and Lander, E.S. (2002) ARACHNE: a whole-genome shotgun assembler. *Genome Res*, **12**, 177-189.
4. Jaffe, D.B., Butler, J., Gnerre, S., Mauceli, E., Lindblad-Toh, K., Mesirov, J.P., Zody, M.C. and Lander, E.S. (2003) Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome Res*, **13**, 91-96.
5. Pevzner, P.A., Tang, H. and Waterman, M.S. (2001) An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci U S A*, **98**, 9748-9753.
6. Huson, D.H., Reinert, K., Kravitz, S.A., Remington, K.A., Delcher, A.L., Dew, I.M., Flanigan, M., Halpern, A.L., Lai, Z., Mobarry, C.M. *et al.* (2001) Design of a compartmentalized shotgun assembler for the human genome. *Bioinformatics*, **17 Suppl 1**, S132-139.
7. Mullikin, J.C. and Ning, Z. (2003) The phusion assembler. *Genome Res*, **13**, 81-90.
8. Wang, J., Wong, G.K., Ni, P., Han, Y., Huang, X., Zhang, J., Ye, C., Zhang, Y., Hu, J., Zhang, K. *et al.* (2002) RePS: a sequence assembler that masks exact repeats identified from the shotgun data. *Genome Res*, **12**, 824-831.
9. Aparicio, S., Chapman, J., Stupka, E., Putnam, N., Chia, J.M., Dehal, P., Christoffels, A., Rash, S., Hoon, S., Smit, A. *et al.* (2002) Whole-genome shotgun assembly and analysis of the genome of *Fugu rubripes*. *Science*, **297**, 1301-1310.
10. Huang, X., Wang, J., Aluru, S., Yang, S.P. and Hillier, L. (2003) PCAP: a whole-genome assembly program. *Genome Res*, **13**, 2164-2170.
11. Chen, T. and Skiena, S.S. (2000) A case study in genome-level fragment assembly. *Bioinformatics*, **16**, 494-500.
12. Pevzner, P.A. and Tang, H. (2001) Fragment assembly with double-barreled data. *Bioinformatics*, **17 Suppl 1**, S225-233.
13. Staden, R., Beal, K.F. and Bonfield, J.K. (2000) The Staden package, 1998. *Methods Mol Biol*, **132**, 115-130.
14. Gordon, D., Abajian, C. and Green, P. (1998) Consed: a graphical tool for sequence finishing. *Genome Res*, **8**, 195-202.
15. Tammi, M.T., Arner, E., Kindlund, E. and Andersson, B. (2004) ReDiT: Repeat Discrepancy Tagger--a shotgun assembly finishing aid. *Bioinformatics*, **20**, 803-804.
16. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, **25**, 3389-3402.
17. Ewing, B. and Green, P. (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res*, **8**, 186-194.