Author's response to reviews

Title: SOAPdenovo2: Empirically improved memory-efficient short-read de novo assembler

Authors:

Ruibang Luo (luoruibang@genomics.org.cn)
Yinlong Xie (xieyinlong@genomics.org.cn)
Zhenyu Li (lizhenyu@genomics.org.cn)
Binghang Liu (liubinghang@genomics.org.cn)
Weihua Huang (huangweihua@genomics.org.cn)
Jianying Yuan (yuanjianying@genomics.org.cn)
Guangzhu He (heguangzhu@genomics.org.cn)
Yanxiang Chen (chenyanxiang@genomics.org.cn)
Qi Pan (1578499973@qq.com)
Yunjie Liu (applefury@gmail.com)
Jingbo Tang (tangjingbo@genomics.org.cn)
Gengxiong Wu (wugengxiong@genomics.org.cn)
Hao Zhang (zhanghao2@genomics.org.cn)
Yujian Shi (shiyujian@genomics.org.cn)
Yong Liu (liuyong2@genomics.org.cn)
Chang Yiu (yuchang@genomics.org.cn)
Bo Wang (wangbo@genomics.org.cn)
Yao Lu (luyao@genomics.org.cn)
Changlei Han (hanchanglei@genomics.org.cn)
David Cheung (dcheung@cs.hku.hk)
Siu-Ming Yiu (smyiu@cs.hku.hk)
Yingrui Li (liyr@genomics.org.cn)
Huanming Yang (yanghm@genomics.org.cn)
Jian Wang (wangjian@genomics.org.cn)
Tak-wah Lam (twlam@cs.hku.hk)
Jun Wang (wangj@genomics.org.cn)

Version: 2 Date: 4 November 2012

Author's response to reviews: see over
Responses to Comments of reviewers

SOAPdenovo2: Empirically improved memory-efficient short-read de novo assembler
Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, Jingbo Tang, Gongxiong Wu, Hao Zhang, Yujian Shi, Yong Liu, Chang Yu, Bo Wang, Yao Lu, Changlei Han, David Cheung, Siu-Ming Yu, Guangming Liu, Yingrui Li, Huanming Yang, Jian Wang, Tak-Wah Lam, Jun Wang

MS: 1553022259770272

Thanks again for the comments from the reviewers. In the followings, we try to address the issues raised by the reviewers based on the revised manuscript.

Reviewer: 1
Version: 1
Date: 28 August 2012
Reviewer: Mario Caccamo
Reviewer's report:
General comments:
The manuscript presents SOAPdenovo2, an upgrade for the popular short-read sequence assembly tool. In order to demonstrate the novel features and improvements of this new version the authors have applied SOAPdenovo2 to two different datasets: the raw data used for the Assemblathon 1 project and the YH human whole-genome sequencing data. These datasets are distributed as part of the manuscript’s submission in the spirit of the GigaScience journal. This is very valuable data as allows the reviewers and readers to reproduce the result and evaluate other aspects of the software interface. We welcome this approach and hope this manuscript will set the standard for similar bioinformatics software publications in the future. We have evaluated this manuscript both based on the quality of the presentation of the article and the assembled sequences.
The new developments in SOAPdenovo2 are focused on four areas:
• more efficient algorithms and data structures to reduce the memory requirements for the step of the de Bruijn graph construction;
• use of a multiple k-mer approach to improve the handling of errors and low-coverage regions;
• better handling of heterozygosity with a reduction of misassemblies and chimeric sequences at the scaffold level; and
• an improved method to implement gap closure.
Although we have seen significant improvements in the past couple of years the assembly problem for short reads remains an important challenge. The main conclusion from some of the recent competitions and comparative studies such as the Assemblathon and GAGE is that there is no “one-fits-all” solution, rather every study and datasets seem to require a tailored and customised tool. In this context only few tools have been successfully applied across a wide range of genomes and SOAPdenovo is one example. We believe that the authors decision to focus on the improvement of the genome structure rather than the contig level is the correct direction as the weaknesses in the current tools are indeed in the scaffolding stage. We therefore would like to recommend the publication of this manuscript and associated datasets in
Gigascience. In order to ensure the best quality for the final version, however, we ask the author to address some major points before the article is accepted for publications.

RE: We do really appreciate Dr. Caccamo for his careful reading and professional comments for us to improve the manuscript. We have carefully revised the manuscript and made following major changes:

1. Added the description of new design concepts and algorithm details of each module to the supplementary methods.
2. Evaluated SOAPdenovo2 on GAGE datasets.
3. Assembled the YH genome using both the old and new sequencing dataset with SOAPdenovo version 1 and the combinations of optional modules in version 2. The results showed that the improvements are due to algorithms instead of new data.
4. The initial release of SOAPdenovo is a great contribution to the community (cited 417 times since 2010 according to google scholar). We illustrated that compare with the first version published two years ago, SOAPdenovo2 now provides assembly with much high quality.
5. The binary of SOAPdenovo2 has already been downloaded for over 500 times (in SourceForge). Thus we target the paper a technical note as a reference to the new algorithms and improvements for the readers.
6. We have packed up each of the experiment into a one-for-all command shell to facilitate the readers for validating the results.

Below, please find our replies.

Major Compulsory Revisions:
- **A major contribution of Gigascience is to provide a framework to publish “data-driven” scientific studies alongside the datasets following open-access open-data principles.** As mentioned above we believe this is an important advance in the way algorithms and software are published in bioinformatics. With this in mind we would like to request the authors to provide a detailed flow-chart with the steps to follow to reproduce (at least partially) some of the results reported in the manuscript. The parameters and software used should also be described to allow the reviewers and readers to run the tools as done by the authors. We understand there will be certain points that might require manual interventions or the use of unpublished software but those steps should be clearly indicated. We tried to follow the steps described in the “readme” document obtaining the following results using the Assemblathon 1 data and running just up to the assembly step:
  - Before scaffolding: there are 77,636 contigs longer than 100, Total sum 114,997,317 bps, with average contig length 1,481. Contig N50 is 3,146 bp, contig N90 is 818 bp. 460,293 contig(s) longer than 32bps
  - After scaffolding: longest scaffold 21,794,522 Scaffold and singleton number 4,093 Total sum 118,860,825 average length 29,040 Scaffold N50 8,018,180 N90 3,409,497

Although we have tried different parameters and effectively done four different assemblies, no combination of parameters actually allowed us to get similar statistics to the figures mentioned on Table 1, so this
is why we think reproducibility will be greatly improved by inclusion of the requested flow-chart. In a similar way, a list of all the software or tools used should be included, detailing whether it is publicly available, being published in this work, soon to be released or to remain closed.

RE: We totally agree with the idea to provide clear flow-charts for readers to reoccur the results shown in the paper. We’ve packed up each of the experiment into a one-for-all command shell to facilitate the readers for validating the results. Please download the shells from ftp://public.genomics.org.cn/BGI/SOAPdenovo2.

- It is not clear whether the impressive improvement in the YH assembly is a result of the new libraries added to the dataset or the novel algorithms in SOAPdenovo2. There are two new libraries, one of them with insert size 4-times larger than anything used before. Was the previous version of SOAPdenovo able to cope with this type of data effectively?

RE: We are sorry to include this apparent flaw in the paper. We reassembled the YH genome using both the old and new sequencing dataset with SOAPdenovo version 1 and version 2. The results were shown in Table 2. By aligning the assembly of SOAPdenovo1 and SOAPdenovo2 (both using new data) to the reference genome, we obtained a reference coverage increase from 81.2% to 93.91%, and we found that ~95.9% of the newly assembled regions were repetitive sequences. The result illustrated that the improvements are due to algorithms instead of new data.

- From the readme document distributed with the data it is clear that a consensus step (step 6) was conducted after the assembly. Was this step used to modify the assembly result? Are the metrics (i.e. substitution rates) affected by this step?

RE: The consensus step is trivial and optional. It targets to provide heterozygous bases according to the alignment results thus do not affect the metrics including substitution rates. Since SOAPdenovo2 targets to maintain the maximum flexibility for users, the consensus step was set as an example while users can modify the step with faster aligners and more accurate variation callers.

- Although the authors indicated SOAPdenovo2 outperforms SOAPdenovo in memory usage, it isn’t clear from Table 1 what is the level of improvement. The usage of extensive read filtering and correction should be taken into account and be mentioned either as an equal or different scenario.

RE: Sorry for the confusion. To illustrate the memory consumption improvement, we’ve evaluated SOAPdenovo1, SOAPdenovo2 and SOAPdenovo2 with sparse DBG enabled. The peak memory consumptions of graph construction step were shown in the Table 2. SOAPdenovo2 using full graph module consumed 155GB memory while sparse DBG only consumed 35GB memory, which is about 4 times lower. However, the results of sparse DBG are not as good as full graph and we’ve explained the reasons in maintext and supplementary method 2. In addition, the new algorithm of read correction was
introduced in supplementary methods and it consumes less than 16GB of peak memory. Thus the largest memory consumption component remains the graph construction step.

Minor Essential Revisions:
• More detail is needed to understand some of the new algorithms and improvements. This is a list of points that we suggest the authors should address to improve the quality of the final version of the manuscript:

RE: In the new version, we’ve described the algorithms in detail in supplementary methods.

- Sparse graph. It is not clear which sparse graph approach has been used for SOAPdenovo2. From the reference in the text the indication is that a "sparse de Bruijn graph" as implemented by sparseassembler1. A more recent version of this approach, sparseassembler 2, has tackled some of the problems of the original development based on a "sparse k-mer graph". There seems to be a mismatch between the description in the text and the citation. If this was the first version, could this stage be improved by the use of the new approach? More concretely: at which stage is the sparse graph used? From running SOAPdenovo2 ourselves we believe this new data structure is only used at the initial stage of building the graph resorting to a traditional graph for later stages. The two example assemblies have apparently not been generated with the sparse graph, is this because of some trade-off or is the sparse-graph now the default and has been effectively used in all the assemblies? On a technical note will the sparse graph construction directly impact the ability to preserve important meta-information such as sequence coverage? How did this information loss affect the final assemblies?

RE: We’ve changed the citation to sparseassembler. The underlying concept of sparseassembler 1 and 2 are the same, while sparseassembler 2 mainly improved in error filtering, bubble merging and etc., which makes it more suitable for real data processing. We’ve adopted only the concept of sparse DBG from sparseassembler and combined the data structure with SOAPdenovo2’s sophisticated error filtering, bubble merging and graph traversal algorithms. The problems sparseassembler 2 solved have already been considered and solved at the very beginning of implementing SOAPdenovo2. The major difference between the algorithm used by SOAPdenovo2 and sparseassembler 2 is that SOAPdenovo2 traverse the sparse DBG in parallel while sparseassembler do the job in serial. The sparse DBG was only used in initial stage while after the initial stage, both the full graph and sparse DBG will be simplified to edge graph, which consumes much less memory compared to the initial full graph. Another reason we did not use the sparse DBG concept in all stages is that it does have trade-off. To preserve the same level of k-mer depth, sparse DBG requires shorter k-mer length, which makes the repetitive sequences that could be solved by longer k-mer length now unable to be solved (explained in sparseassembler’s paper). In our experiments, the YH genome assembly using full graph is better in both contig and scaffold N50 than the assembly using sparse DBG. We also explained the trade-off in maintext and supplementary method 2. The sparse DBG module is an option in
SOAPdenovo2 and users can select the module if the memory available is limited or the genome is too large to be fitted in memory if using the full graph.

- **Multi k-mer approach.** The authors recommend to start with a small k-mer, how small? How many k-mers are tried and what are the criteria to preserve information (if anything at all) from one k-mer level to the next one? The relevant `-m` option is apparently only used on the contig step of the assembly, so we assume that the k value is still used as before, but on traversing the graph on the contig step larger k could be used on ambiguous regions? Is this a contig step only modification on is there some kind of support from the other steps? This parameter forces the use of the configuration file as parameter too, is this because it is effectively re-evaluating k-mer content from the input files?

RE: We adopted the multi k-mer approach from IDBA-UD assembler (Peng et al., 2012). We recommend k-mer as short as 31-mer while the sequencing depth is over 30-fold with 100bp paired-end reads. The multi k-mer approach was implemented in the contig step only. K-mers in the initial graph were merged into edges before multi k-mers iteration. K-mers were exercised from the smallest (k-mer+1) to the longest set by parameters, 1bp a step and if a branch (an ambiguity) in graph could be solved (linearize) by a k-mer 1bp longer, the branch will be duplicated into two copies of edge with identical suffix, which effectively solved the problem of paralogous repetitive sequences no longer than the k-mer size. Configuration file is now required in contig step if multi k-mer was enabled because we need to shear longer k-mers from the raw reads. Notably, reads already aligned to linear edges would not be scanned in long k-mer iterations.

- **Heterozygosity.** The authors mention that a topology-based method has been developed to work around heterezygous sites. Which kind of structure/topology this method detects in the graph?

RE: We have explained the scaffolding algorithms in detail in supplementary method 4.

- **Read correction.** The main goal is to reduce memory consumption and remove the source of spurious contigs. How is the procedure used in SOAPdenovo2 comparable the other assemblers, particularly with the SOAPdenovo run on YH? Is there any evidence that this help SOAPdenovo2 getting longer and more reliable contigs? Should read correction always be used as a previous step on SOAPdevono2? We would also like to suggest the authors to describe any rules and principles to set the parameters for read correction. From the data it seems that the only reads corrected are from PE libraries, is this linked to the chimeric contigs introduced by these libraries?

RE: We have explained the error correction algorithms in detail in supplementary method 1. Notably, we’ve designed a new method called space k-mer, to increase the accuracy and sensitivity of error correction while requires only a small amount of memory. However, we are not going to justify the necessity of error correction since it is a genome dependent problem. The error correction is an optional module of
SOAPdenovo2 thus users can decide whether to include error correction by comparing the outcomes. In YH genome specifically, the error corrections were only performed on fragment libraries, and not on large insert-size mate-pairs. This is also the protocol of SOAPdenovo1, because the libraries of large insert-size mate-pairs were not included in contig construction (only used in scaffold procedures), limited by it's sequencing quality (PCR duplication, relatively shorter read length, mixture of chimeric reads). Thus the per base accuracy of large insert-size mate-pairs won't affect the per base accuracy of assembly.

- Library insert size. They appear to be very exact for some of the libraries (the ones coming from v1 YH, including the LMP) and just an estimation for other libraries. Is this parameter checked/calibrated at the scaffolding stage? Is there any particular restriction on the distribution of insert sizes that could be taken into account for the scaffold to work optimally? The GapCloser configuration file specifies a min and a max parameters for insert sizes: how are these values calculated or set? In particularly we would like to know why those same limits aren’t the same as the ones used in the previous steps.

RE: The real insert size of libraries was real time estimated by SOAPdenovo2. The insert size set in the configuration file is a limitation to the length of contig being included into the insert size estimation. There is no particular restriction on the setting of the distribution of insert sizes for the scaffolding module to work optimally since all parameters were inferred in real time. The min and max parameters in GapCloser configuration file can be set according to the estimated insert size and deviation by SOAPdenovo2’s scaffolding module.

The insert size used in the configuration file for SOAPdenovo2 is from experiment estimation and the one in GapCloser’s configuration file is estimated by SOAPdenovo2’s scaffolding module. The GapCloser works better when using the insert size and deviation estimated by SOAPdenovo2’s scaffolding module.

Discretionary revisions:

- The results from the Assemblathon 1 datasets suggest a modest enhancement in the contigs but a remarkable improvement on the scaffolds (even if not on the range of 44-fold). This scenario is consistent with the described features of SOAPdenovo2 and should be emphasized.

RE: We’ve emphasized the improvement of both accuracy and length in the maintext.

- Although the N50 statistic is a widely used metric for assembly quality, it is important when evaluating assembly algorithms to look at the full set of contigs/scaffolds and their lengths. A plot of accumulated sequence length vs. contig count could provide better insights on the improvements of the algorithms than simply N50, which is effectively a single first-derivative value of this curve on the point where it reaches half of its final value.
RE: This is a very good suggestion. We’ve plotted the continuous change from N90 to N10 for Assemblathon 1 and YH genome assemblies respectively as Figure 1 and Figure 2 in maintext.

- The authors suggest that chimeric scaffolds are mainly introduced by smaller PE libraries. Is this just an effect of the sequence depth as usually shorter inserts are generated with more coverage? In that case: wouldn’t it be just a simple step to deal with chimeras through coverage?

RE: Sorry for the confusion, the chimeric contigs are repetitive contigs with exceedingly large depth. We’ve further explained in supplementary method 4.

- The term "coverage" is used for two different concepts across the text when referring to:
  - how much sequencing data has been generated with respect to the estimated target genome size, and
  - how much of the target genome/sequence is assembled into contigs.
Although in most cases this can be resolved from the context it is potentially confusing for the less experience readers.

RE: Sorry for the confusion, we have changed the word to represent “how sequencing data has been generated with respect to the estimated target genome size” to ‘depth’.

- The solution to resolve heterozygosity by taking a "majority rules" strategy is in principle fine but will inevitably lead to mosaic consensi where two or more haplotypes will be represented within a single scaffold/contig. This is not per se an important issue but we suggest this is clarified in the text.

RE: We have clarified this point in supplementary method 4.

- What is the error profile of the sequence added by the gap closer? Is it clearly different from the rest of the assembly in this aspect? Is there some situation in which the use of such a tool is highly inconvenient or this reason? We believe this is an important issue that hasn’t been properly discussed in recent publications. Hopefully as assembly tools improve this point will be raised as a priority.

RE: The algorithm details of GapCloser were described in supplementary method 5. However, the error profile of the sequence added by GapCloser or other available gap filling utilities is genome dependent. ‘Repetitive patterns similarity’, ‘length distribution’, ‘evenness in genome’ and many other metrics would change the error profile and the discussion would require another paper for systematic analyses on multi-samples, which is beyond the target of our paper.

- We suggest the authors name files in the submission with a schema that is consistent with the procedure used to generate the data
• Running scripts depend on some pre-existing tools as SGE, this could be stated for clarity, and although there are options not to use those features it is not necessarily straightforward to figure them out.

RE: We’ve packed the procedures (with a one-for-all command shells and all requiring binaries) into a single file for each experiment to facilitate the readers to reoccur the experiments.

• Some manual tuning is obviously in place (mapping reads with different k sizes, manual creation of the peGrads file). What are the criteria behind the parameters used for these manual steps?
• Mapping for scaffolding on the v1 YH data has been done on k-mer=45 but gap closing is done on overlap=31:
  - this means that gap closing is done in a more permissive way than scaffolding, which makes sense, but then read mapping on the new reads is done with k=31, why is this?
  - Is there any rule to set values for this parameters?
  - Are the tools used for combination of the mapping being distributed as part of SOAP2.

RE: The use of different k-mer lengths in modules is a standard procedure introduced in http://soap.genomics.org.cn (using the ‘prepare’ utility). We allow using longer k-mers for contigs construction and smaller k-mer in scaffolding due to the reason that the mate-pair reads are usually shorter than the fragment reads, sometime even shorter than the k-mer size used in the contig module. Using k-mer length longer than the read length is prohibited in de Bruijn graph assembly. For YH genome assembly specifically, the read length of the 40k insert size library is only 44bp, which is longer than the k-mer length used by contig module. So we changed the k-mer length for scaffolding to a size smaller than 44bp.

To avoid confusion, we’ve reassembled the YH genome (both new data and old data) with SOAPdenovo’s mapping module only (without external mapping and results combination procedures) in the revision. The YH assemblies shown in maintext Table 2 are without external alignment.

The reason why GapCloser is using smaller k-mer size than SOAPdenovo’s mapping module is that it’s targeting repetitive sequences that could not be solved by SOAPdenovo itself. Using shorter k-mer length allows more reads to be aligned, albeit the incorrectly aligned reads will increase too.


• Are the structural errors to the lower "contig/scaffold path NG50" vs. "contig/scaffold N50" ratio compared with the results obtained with ALL-PATHS?

RE: The metrics contig/scaffold path NG50, N50 and structural errors were generated by Assemblathons1’s official evaluation pipeline. The structural errors generate by SOAPdenovo2 and ALLPATHS-LG are 1,414 and 1,244 respectively.
Reviewer: 2  
Version: 1  
Date: 22 August 2012  
Reviewer: Aleksey Zimin  
Reviewer's report:  
This paper describes improvements to the original SOAPdenovo genome assembly software by BGI. The improvements are rather minor and I am not convinced that they will result in significant improvements when the assembler is used on real-life data sets.  
Based on my comments listed below, the paper in its current shape requires much more than a major revision. The authors need to present more and different kinds of results to merit the publication, as well as clarify the presentation. The paper needs to be rewritten and resubmitted. Therefore I recommend that the paper is rejected at this time.

RE: We thank Dr. Zimin for his careful reading and comments for us to improve the manuscript. We have carefully revised the manuscript and made following major changes:

1. Added the description of new design concepts and algorithm details of each module to the supplementary methods.
2. Evaluated SOAPdenovo2 on GAGE datasets.
3. Assembled the YH genome using both the old and new sequencing dataset with SOAPdenovo version 1 and the combinations of optional modules in version 2. The results showed that the improvements are due to algorithms instead of new data.
4. The initial release of SOAPdenovo is a great contribution to the community (cited 417 times since 2010 according to google scholar). We illustrated that compare with the first version published two years ago, SOAPdenovo2 now provides assembly with much high quality.
5. The binary of SOAPdenovo2 has already been downloaded for over 500 times (in SourceForge). Thus we target the paper a technical note as a reference to the new algorithms and improvements for the readers.
6. We have packed up each of the experiment into a one-for-all command shell to facilitate the readers for validating the results.

Below, please find our replies.

Major compulsory revisions.
1. Authors compare the performance of the SOAPdenovo2 to SOAPdenovo1 and Allpaths-LG on faux data set from Assemblathon 1. Faux data usually is much easier to assemble than the real data and the evaluation presented in the subsequent works (e.g. GAGE assembly competition by Salzberg et al., 2011) painted completely different picture. The comparisons of assemblers using faux data have no practical value for determining real-life usability and performance in de novo genome assembly projects. Papers that present genome assembly software must demonstrate the performance of the software on real-life data sets for which finished sequence exists. For example one can use data sets from
the GAGE project (Salzberg et al., 2011). The data sets are available at http://gage.ccb.cb.umd.edu.

RE: We appreciate Dr. Zimin’s great effort on the GAGE paper (Salzberg et al., 2011). We have tested SOAPdenovo2 one the three GAGE datasets, which include two small genomes (Staphylococcus aureus: 2,872,915bp; Rhodobacter sphaeroides: 4,603,060bp) and an intermediate size genome (Bombus impatiens: 250Mbp). We didn’t included Human Chromosome 14 (88Mbp) since its sequencing data were down sampled form a human whole genome sequencing project, and we have already shown the whole genome assembly of YH genome, the size of which is 34 times larger than the Chromosome 14. The results illustrated that SOAPdenovo2 outperformed SOAPdenovo1 in all genomes (Supplementary Method 6, Supplementary Tables 3,4). For the correct scaffold N50 length metric, SOAPdenovo2 is 3.15, 36.28 and 63.84-time longer than SOAPdenovo1 for the three genomes respectively. ALLPATHS-LG performs better in small genome assemblies, but is unable to assemble the Bombus impatiens dataset due to the lack of overlapping fragment libraries. However, SOAPdenovo2 does not force to use specific dataset settings, thus provides users with higher flexibility.

We all agree that the world of genomes is diversified. For small genomes, Escherichia coli is easy to be sequenced and assembled, while Bordetella pertussis (67.7% of GC content) and Plasmodium falciparum (19.3% of GC content) are hard to be assembled (Quail et al., 2012). For large genome, human genome contains over 40% of the repetitive sequences, but it’s relatively easier to be assembled because most of the repetitive sequences are long, patterned and diversed (HGP, 2001). Oyster genome contains only 30% of repetitive sequences, but is rather hard to be assembled because its high heterozygosity and interspersed short repetitive elements (Zhang et al., 2012). It’s not possible for any assembler contest to cover all the situations of genome assembly. In addition, real data assembly does not provide a clear ground of the correctness of assembly while using simulated data defines the word ‘correctness’ clearly. Simulated data are far not ‘faux data’ since they were simulated based on the parameters studied from the real data. In the sense, we don’t agree with reviewer’s comment: “using faux data have no practical value for determining real-life usability and performance in de novo genome assembly projects”, which strongly denied the contribution of Assemblathon 1 and other published assembly studies based on simulated data.

Alternatively, one can download data for mouse B6 genome available at SRA. Then authors can create an assembly, compare it to the finished sequence and clearly comment on the contiguity, connectivity and correctness of the assembly. For example one can split the assembly at the locations of all misassemblies and compare the resulting N50 size to the N50 size of the original assembly for both contigs and scaffolds. A comparison to another major software package (such as Allpaths-LG) on the same data set would be a big plus.

RE: Mouse B6 is a mammal genome as large as YH human genome. As we target a technical note, we do not try to cover as much genomes as possible. In each of the experiment, we have included ALLPATHS-LG, which is one of the best assemblers we have got at this time. Especially, for Assemblathon 1 dataset, which provides a definite answer for justification, we compared the results of SOAPdenovo1, SOAPdenovo2 and ALLPATHS-LG (maintext Table 1). The results shows that, comparing SOAPdenovo2 to
SOAPdenovo1, the new scaffold N50 was nearly an order of magnitude longer and the accuracy was higher due to the reduction of structural error by 90.12%, substitution error by 92.13%, and copy number error by 69.47%. Comparing to ALLPATHS-LG, SOAPdenovo2 produced contig N50 and scaffold N50 that were about 1.53 and 1.84-time longer. The SOAPdenovo2 assembly also had a much lower amount of copy number errors, but did have more substitution errors.

Therefore I ask that authors demonstrate the performance of the SOAPdenovo2 on a real mammalian (one chromosome is sufficient) genome data set and compare the resulting assembly to the finished sequence to evaluate the performance of the assembler.

RE: Thanks for reviewer’s suggestion. We evaluated both the YH genome assembled by SOAPdenovo1 (Li et al., 2010) and SOAPdenovo2 respectively by aligning them to the NCBI human reference genome hg19 (http://genome.ucsc.edu/). We obtained a reference coverage increase from 81.2% to 93.91%, and we found that ~95.9% of the newly assembled regions were repetitive sequences.

2. The performance of SOAPdenovo2 was evaluated against SOAPdenovo1 using YH chomorome data. However, the data used for the two assemblies was different, the new assembly with SOAPdenovo2 has more data in it. Yes, the assembly is better, but would SOAPdenovo1 generate the same improvements with the additional data?

RE: We are sorry to include this apparent flaw in the paper. We reassembled the YH genome using both the old and new sequencing dataset with SOAPdenovo version 1 and version 2. The results were shown in maintext Table 2. By aligning the assembly of SOAPdenovo1 and SOAPdenovo2 (both using new data) to the reference genome, we obtained a reference coverage increase from 81.2% to 93.91%, and we found that ~95.9% of the newly assembled regions were repetitive sequences. The result illustrated that the improvements are due to algorithms instead of new data.

Minor essential revisions
1. This paper is written in a style of more of a technical report than a structured scientific publication. Material is presented in haphazard fashion. I would call this paper an initial draft that would require more work to be worthy of a scientific publication. I suggest that the authors divide their paper into more common Introduction -- Results -- Methods -- Discussion sections, or alike.

RE: We target for technical note of GigaScience journal. We arranged the paper in the sequence of “introduction”, “brief methods”, “simulation results” and “real data results”. Details methods were moved to supplementary methods due to the limitation of technical note.

2. Authors should clarify what improvements were made to the GapCloser module described on page 4. Are the improvements described implemented in the main assembler code of in the GapCloser? It is not clear from the text.
RE: Thanks for reviewer’s suggestion. We’ve described the algorithm of GapCloser in detail in Supplementary Method 5.
Reviewer: Alexander J. Nederbragt
Reviewer's report:

This paper introduces an update to the SOAPdenovo program, describes its major improvements and shows improved results on two datasets. We reviewed this paper with a group of four people from the same research group.

RE: We appreciate Dr. Nederbragt, Dr. Tørresen, Dr. Lagesen and Dr. Crawford for their careful reading and valuable comments for us to improve the manuscript. We have carefully revised the manuscript and made following major changes:

1. Added the description of new design concepts and algorithm details of each module to the supplementary methods.
2. Evaluated SOAPdenovo2 on GAGE datasets.
3. Assembled the YH genome using both the old and new sequencing dataset with SOAPdenovo version 1 and the combinations of optional modules in version 2. The results showed that the improvements are due to algorithms instead of new data.
4. The initial release of SOAPdenovo is a great contribution to the community (cited 417 times since 2010 according to google scholar). We illustrated that compare with the first version published two years ago, SOAPdenovo2 now provides assembly with much high quality.
5. The binary of SOAPdenovo2 has already been downloaded for over 500 times (in SourceForge). Thus we target the paper a technical note as a reference to the new algorithms and improvements for the readers.
6. We have packed up each of the experiment into a one-for-all command shell to facilitate the readers for validating the results.

Below, please find our replies.

Major Compulsory Revisions

We have the following major issues with this paper:

1) The explanation on the improvements in SOAPdenovo2 lack sufficient detail to be able to fully understand them. Papers of this kind usually explain approaches and algorithms used in much more detail. The authors should look at other papers describing new versions of existing software, such as the recent ALLPATHS_LG paper (Ribeiro et al, 2012, http://genome.cshlp.org/content/early/2012/07/24/gr.141515.112.abstract), or even the article describing the first version of SOAP (Li et al, 2009). Improvements of the text are needed so that the reader can understand what changes were implemented and exactly how that improved the program.

RE: In the new version, we’ve described the algorithms in detail in supplementary methods, including the improvements of the error correction, multi k-mer, sparse DBG, scaffolding and gap closure modules.
2) Even though we were given access to the underlying raw data, and obtained a pre release versions of SOAPdenovo2 from the authors, we could not replicate the results described in the paper due to a lack of detail in the section on 'Testing and Assessment': the exact commands used for the assemblies are not given.

RE: We’ve packed the procedures (with a one-for-all command shells and all requiring binaries) into a single file for each experiment to facilitate the readers to reoccur and experiments. Please download the packages from ftp://public.genomics.org.cn/BGI/SOAPdenovo2.

3) The article is very biased towards assembly of human genomes. However, SOAPdenovo can be, and often is, used for the assembly of bacterial genomes. The authors use the Assemblathon1 data for their analyses of SOAPdenovo2. In the 'Background' section, the GAGE assembly competition is mentioned, which focuses on comparing programs for assembly of bacterial-sized genomes. However, SOAPdenovo2 was not evaluated against the GAGE data, something we feel is an omission.

One of us tested SOAPdenovo2 on the Rhodobacter sphaeroides dataset from GAGE, and ran the same analysis script as was used for the GAGE publication (http://gage.cbcb.umd.edu/results/index.html). We have included a summary of this analysis as a PDF attached to this report.

From the results, we find the following:
- SOAPdenovo2, as the first version of the program, still results in many errors in contigs and scaffolds ('corrected' N50's are much lower then N50' values of the sequences generated by SOAPdenovo2)
- In our tests of the 'sparse assembly graph' approach, a better assembly was obtained by providing a larger estimated genome size then the real size. Do the authors have an explanation for this effect?
- The 'sparse assembly graph' runs improved uncorrected scaffold sizes, however they resulted in a larger number of scaffolds. Also, the corrected scaffolds N50 of these assemblies were in fact lower than reported in the GAGE article for SOAPdenovo1.
- We did see an improvement in the contigs from SOAPdenovo2 relative to the first version: fewer errors and higher corrected N50 values, but at the cost of higher contig numbers.

In conclusion, we do not see significant improvements using SOAPdenovo2 versus the first version of the program on the Rhodobacter dataset. We feel the authors should document the performance of SOAPdenovo2 on small genomes with an available reference genome, for example using the data that was the basis of the GAGE competition.

RE: We thank the reviewers for their evaluations on SOAPdenovo2. We have tested SOAPdenovo2 on the three GAGE datasets, which include two small genomes (Staphylococcus aureus: 2,872,915bp; Rhodobacter sphaeroides: 4,603,060bp) and an intermediate size genome (Bombus impatiens: 250Mbp). We didn't included Human Chromosome 14 (88Mbp) since its sequencing data were down sampled form a human whole genome sequencing project, and we've already shown the complete assembly of YH human genome. The results illustrated that SOAPdenovo2 outperformed SOAPdenovo1 in all genomes (Supplementary Method 6, Supplementary Tables 3,4).
For the correct scaffold N50 length metric, SOAPdenovo2 is 3.15, 36.28 and 63.84-time longer than SOAPdenovo1 for the three genomes respectively. ALLPATHS-LG performs better in small genome assemblies, but is unable to assemble the Bombus impatiens dataset due to the lack of overlapping fragment libraries. However, SOAPdenovo2 does not force to use specific dataset settings, thus provides users with higher flexibility. Worth mentioning, both SOAPdenovo and SOAPdenovo2 were designed and optimized for large vertebrate genomes (mentioned in the abstract). For smaller genomes, ALLPATHS-LG performs better than SOAPdenovo2 in per base accuracy.

The algorithmic details of sparse DBG are in supplementary methods 2. Sparse DBG uses over 4-time less memory than the full graph method in YH genome assembly (maintext Table 2). However, the resulting contig and scaffold N50 is lower. Sparse DBG is a lossy data structure compared to the full de Bruijn graph since it’s unable to maintain the depth information of each k-mer, which makes it possible to perform worse in repetitive genomes. Estimated genome size is a clue for the sparse DBG module to estimate the size of data structure it should allocate in the computer memory. Longer estimated genome size makes the assembly better should be an incident and due to the reason that the different size of the data structure altered the starting point of graph iteration. This is a limitation of sparse DBG and we’ve clarified it in supplementary method 2. We suggest using only the full graph method for small and repetitive genomes.

4) We also tried SOAPdenovo2 on data from one of our own large eukaryotic genomes. The 'default' version of the program crashed, only when we used the sparse assembly graph version did we get the program running. This may have been due to the fact that we were not able to compile the program on our system, and only could use the provided binaries. 5) GigaScience’s description of a technical note requires 'the code described be documented and tested to high standards.' We did not have access to the source code and can therefore not judge whether the code was well documented. Also, we feel the few tests reported in the paper make us uncertain whether the code can be considered 'tested to high standards' (see also above).

RE: We’ve received several feedbacks of the same problem and the problem solved after using natively compiled binary. We will release the source code of SOAPdenovo2 as soon as the paper is accepted. The binaries are now available at SourceForge with over 500 downloads. We are continuously receiving reports and feedbacks in the SOAP software news group (bgi-soap@googlegroups.com).

6) The paper makes many claims that are not referring to any articles or actual data. For example, it is written "Scaffold construction is another area that needs improvement in NGS de novo assembly programs." Can the authors point to some references to back up this claim? Similarly, when discussing the original SOAPdenovo program, the authors give three problematic areas as examples -improperly handling of heterozygous contigs, chimeric scaffolds, false contig relationships. However, no documentation of these problems is provided - real tests of
assemblies of datasets with a reference genome where these problems can be shown.

RE: Sorry for the confusion. We have cited the paper of SOPRA (Dayarian et al., 2010) when mentioning “Scaffold construction is another area that need improvement in NGS de novo assembly programs”. We’ve performed additional experiments and the problems of “improperly handling of heterozygous contigs”, “chimeric scaffolds” and “false contig relationships” have been described in detail in supplementary method 4 and supplementary figure 5.

7) The authors tested new YH 2x100 illumina data with SOAPdenovo2 but failed to show comparable analyses of the same data with the original SOAPdenovo program. To fully elucidate the improvements made from the upgrade to SOAPdenovo2, the authors should report on the analysis of these new YH data with both versions of the program.

RE: We are sorry to include this apparent flaw in the paper. We reassembled the YH genome using both the old and new sequencing dataset with SOAPdenovo version 1 and version 2. The results were shown in maintext Table 2. By aligning the assembly of SOAPdenovo1 and SOAPdenovo2 (both using new data) to the reference genome, we obtained a reference coverage increase from 81.2% to 93.91%, and we found that ~95.9% of the newly assembled regions were repetitive sequences. The result illustrated that the improvements are due to algorithms instead of new data.

8) The authors used analyses from the assemblathon1 (published February 2011) in their comparison of SOAPdenovo2 with the ALLPATHS_LG program.

However, new versions of ALLPATHS_LG have been released since February 2011. As such, we feel that the authors should test the most recent version of ALLPATHS_LG against SOAPdenovo2 (using the same data) to ensure a fair comparison between the two programs.

RE: We’ve also considered using the ALLPATHS_LG’s latest version for the Assemblathon dataset. However, as mentioned in the Assemblathon 1 paper, ALLPATHS_LG do not support dataset without 3’-end overlapping fragment library and the version ALLPATHS_LG used in Assemblathon 1 is a special version for the competition only. The version of ALLPATHS_LG that could be downloaded failed on assembling the Assemblathon 1 dataset.

Minor Essential Revisions
9) There is no reference to table 2 in main text.

RE: Sorry for the mistake. We’ve added into the main text.

10) The doi link for reference reference 11 (http://dx.doi.org/10.5524/100038) was not resolving at the time this manuscript was submitted for review.
RE: The link will be ready by GigaScience as soon as possible. As an alternative, please visit the data of YH genome by visiting http://yh.genomics.org.cn, or EBI short read archive with study accession ERP001652.