Dear Mrs. Murakami:

Thank you for your letter and for the reviewer’s comments concerning our manuscript entitled “MASC: A Linear Method for Multiple Nucleotide Sequence Alignment on Spark Parallel Framework” (ID: JCB-2017-0040). Those comments are all valuable and very helpful for revising and improving our paper, as well as the important guiding significance to our researches. We have studied comments carefully and have made correction which we hope meet with approval. Revised portion are marked in red in the paper. The main corrections in the paper and the responds to the reviewer’s comments are listed below this letter.

We hope that the revised version of the manuscript is now acceptable for publication in your journal.

I look forward to hearing from you soon.

With best wishes,

Yours sincerely,

Shaoliang Peng

We would like to express our sincere thanks to the reviewer for the constructive and positive comments.

Replies to Reviewer

Specific Comments

**Major issue Comment 1:**

” I would suggest the manuscript be read carefully by someone not familiar with sequence alignment. The writing is poor in places and some terms are unexplained (see many minor comments below). Also the grammar is pretty poor in many places (most notably the results).”

**Answer:**

Thanks for your suggestion and it helps a lot. We have sent the manuscript to some computer science undergraduate students who don’t have any background knowledge about bioinformatics and collect the feedback message from them to improve our paper. And we go through the whole paper to correct the grammar mistakes. We also find out many terms which are unexplained, then we add some citation or give the explicit explanation of these terms.

**Major issue Comment 2:**

” The description of the algorithm is disorganized and ambiguous in locations. And parts are very confusing (see minor comments below).”

**Answer:**

We have carefully read your minor comments below and they are really helpful. The description of the algorithm is truly not good enough and we have re-organized this part in our paper in order to show our method and algorithms more clearly.

**Major issue Comment 3:**

” Pg 7. Why does star alignment on homologous sequence reduce the running cost? You previously said the complexity is O(m^2t+mt). What are the contributions of finding the center sequence and constructing the similarity matrix?”

**Answer:**

Thanks for your question and the speedup by star alignment is one of the key of our method. The complexity of star alignment strategy is O(t\*m^2+mt) which is no less than progressive strategy in general. The time consuming by calculating the similarity between every two sequences is O(t\*m^2) and the pairwise alignments between center and other sequence consume O(m\*t) time. However, when we align a set of homologous sequences, we firstly calculate the similarity score of every two sequences in the set. And we find the that the scores are almost the same which means we don’t need to calculate the scores between every two sequences to find the center sequence. Every sequence could be a center sequence. So in our method we skip the process of calculating the similarity matrix and just select a sequence at average length randomly as the center and do the pairwise alignments between center and other sequences. In this way, we reduce that time to O(m\*t), where m is the number of sequences and t is the time consumption of a pairwise alignment. This method sounds like quite simple but with many experiment on homologous sequence sets, we find it really makes sense. With the development of pharmacogenomics and other techniques such as next generation sequencing technology, more and more homologous sequence sets are excavated, more and more researches are focusing on these highly similar sequences. Nevertheless, a lot of mainstream software takes too much time to pre-align the sequence or build a guide tree which does not bring better results than our method when aligning highly similar sequences like homologous sequences.

**Major issue Comment 4:**

” Pg 10. The use of length 20 as a ‘too short’ substring alignment is arbitrary. How do you decide if two sequences align due to homology or due to chance? Surely this is a function of the two sequences and what you are trying to align (DNA vs. RNA. Vs. amino acids). This should be some probabilistic argument justifying this.”

**Answer:**

**You are right, the use of length 20 as a “too short” substring threshold is arbitrary. We realize that this is a great mistake that we have made. And we do some experiments on different datasets. The results show’s that the threshold varies with the changing of datasets. Thanks for pointing it out. So we improve our method by changing the standard for eligible substrings. We keep the start positions and lengths of a matching pair of substrings. The standard is that the absolute value of the difference between two start position of matching pair should less than their length. If it is the matching pair of common substrings are eligible. With the new criterion, the SP value of the result of MSA doesn’t increase which means that the result doesn’t get worse. Apart from this, the performance of our program remain the same. So it is a good idea to change the standard for eligible common substring, and thanks a lot for telling us that the use of length 20 as “too short” is wrong.**

**In the newest submitted manuscript 没有改完需要先改好论文再修改**

**Major issue Comment 5:重要问题**

” Pg 13. ‘When aligning similar sequences, every sequence can be regarded as an average one. This means that each sequence can be selected as a center, so that selecting a random one is a good choice, which reduces the time complexity to O(mn).’ This needs to be justified mathematically.”

**Answer:**

Thanks for your questioning. This is the problem we need

**Major issue Comment 6:**

” RESULTS. You switch between present and past tense in the results. Pick one and be consistent. Why only one method comparison in Figure 7.?”

**Answer:**

I am so sorry for my poor English standard and my carelessness. And thank you for pointing them out. I have tried my best to revise the grammar. We have gone through the paper, particularly the part of Result. We have changed all past tense sentences to present tense and keep them consistent. Thanks a lot again.

**Major issue Comment 7:**

” ‘MASC is more accurate than MAFFT when the mutation rate is no more than 54%’ From Figure 7 it looks like MASC Is always more accurate than MAFFT. And how does the statement that ‘MASC works when there are more than 56.8% residues …’ follow from the text? I do not follow…”

**Answer:**

**Major issue Comment 8:**

” Why is your method more accurate than the other methods? Is it a function of the data you simulated? The only arguments I saw in the manuscript is that it should be faster. This doesn’t seem to be justified anywhere. Also, are the results statistically significant?”

**Answer:**

**Major issue Comment 9:**

” The conclusion offers a summary but not very much conclusion, or interpretation, or description of future work, or pitfalls/caveats.”

**Answer:**

**Minor issue Comment 1:**

” Pg 2. Correct ‘Imformation’”

**Answer:**

I am sorry for my carelessness, and we have revised it as ‘Information’ in the latest vision. Thanks for your correction.

**Minor issue Comment 2:**

” Pg 3. Be more specific than ‘deduce the biological facts’. What specifically is MSA useful for? What biological workflows does it enable? The concluding sentence of the abstract is relatively weak.”

**Answer:**

Thanks for your suggestion. The writing of the first sentence of abstract is too general and the readers cannot get what MSA is used for. So we rewrite it as “Multiple sequence alignment (MSA) is an essential prerequisite and dominant method to deduce the biological facts from a set of molecular biological sequences, including homology modeling, secondary structure prediction, phylogenetic reconstruction, and protein structure and function prediction.” We add several application of MSA behind the original sentence to illustrate what is MSA useful for and what kind of biological workflows MSA can enable.

Thanks f

首先删去那句话，然后加入现有的实验成果。

**Minor issue Comment 3:**

” Pg 4. Consider switching the positions of your first two sentences. If someone doesn’t know what MSA is, then they will not understand the first sentence until reading the second.”

**Answer:**

Thanks for your suggestion, that is so wise. So in the revise version manuscript, we switch the position of first two sentences in the Introduction, in this way, people with no background knowledge of sequence alignment can have a better understanding of what MSA is.

**Minor issue Comment 4:**

” Consider breaking up the sentence beginning with ‘An MSA is visualized…’. The final clause isn’t very well explained either. Is the MSA optimizing the ‘biological relationship of the sequences’? It doesn’t seem so. For instance, if your similarity/distance matrix is poorly calibrated, you could get relationships that are not biologically meaningful. Even if it is well calibrated, the alignment may be due to randomness. It’s hard to say anything about the biological relevance of an alignment without some additional analysis (e.g. hypothesis testing). That being said, I do agree with the next sentence in the paragraph.”

**Answer:**

Thanks for your advice, you are right, I agree with you. We could not just say that MSA optimize the biological relationship of sequences. The relation couldn’t be found if the MSA gives a bad result. Apart from this, MSA is just a basic tool. The relevance of the biological sequence couldn’t be found without any additional analysis. So in the revise version manuscript, we break up the sentence and delete the final clause which convey the wrong point of view.

**Minor issue Comment 5:**

” Make sure you add a space (can do this with ~ in LaTeX) between citations and the test (see bottom of page 4).”

**Answer:**

I am sorry for my carelessness. And in the latest version of manuscript, we go through the paper to make sure it would not happen.

**Minor issue Comment 6:**

” You really should have citations for the claim beginning with ‘MSA remains under continuous development...’”

**Answer:**

Thanks for pointing it out. It is really helpful. So in the revised manuscript, we add a citation for the claim of “MSA remains under continuous development …”. The citation we find is a review named *Multiple Sequence Alignment* published on *Bioinformatics* in early 2017, which elaborates the development of multiple sequence alignment in last decade. The detail of this review is shown in revised manuscript.

**Minor issue Comment 7:**

” Correct ‘Futhermore, COMPUTING optimal MSA…’ ”

**Answer:**

Thanks for your advice and we changed it to “computing optimal” in the latest version of manuscript.

**Minor issue Comment 8:**

” Pg 5. Add citations for line “Therefore, most of the algorithms…”. Also, some combinatorial optimization algorithms produce exact solutions right? Or are all of these comb opt methods inherently heuristic? (this would be easier to figure out with citations).”

**Answer:**

Thanks for your correction. You are right and I agree with you. So we add a citation in the latest version manuscript, which is a review named *Multiple Sequence Alignment* published on *Bioinformatics* in early 2017, the same one in the answer of Minor issue Comment 6. This review also lists a number of MSA methods and gives the comments “…wide diversity MSA methods all share a major key property: their reliance on approximate and usually greedy heuristics…”. In the review, the author also describes some comb opt methods that produce exact solutions. So in the latest version of our paper, we change “most of the algorithms…” to “many of the algorithms…”.

**Minor issue Comment 9:**

” Where does this 56.8% number come from? Can you provide some intuition? ”

**Answer:**

I am sorry to mention a number (56.8 percent) in Introduction suddenly. This number comes from our experiment on the sequence similarity analysis in the Result. The reason we put it here is to illustrate the applicable scope of our method and define what kind of sequence is regarded as highly similar. So in order to avoid confusion. We add the following sentence ‘…56.8 percent, which is calculated in the experiment sector…’ in paper.

In our definition, two sequences which have at least 56.8 percent nucleotides in common, after pairwise alignment, as highly similar sequences. And our method MASC is designed primarily for highly similar sequences.

**Minor issue Comment 10:**

” Pg 6. I would wait to talk about specific algorithms in their algorithm type section. The first paragraph of Related Work should be more general.”

**Answer:**

Thanks for your advice. The Related work sector should give the information of different algorithms and strategies that currently used in mainstream software, so that the first paragraph of this part should be more general, and then give more detail information and background knowledge on what we want to state.

So we add the paragraph “Multiple Sequence Alignment to be one of the most widely used modeling methods in biology, which is indeed an important modeling tool whose development has required addressing a very complex combination of computational and biological problems.”. The sentences are summarized from two reviews named *Multiple Sequence Alignment* from *Bioinformatics* 2017 and *Multiple sequence alignment modeling: methods and applications Multiple Sequence Alignment Modeling: Methods and Applications* from *Briefing in Bioinformatics* 2015. And we think this paragraph can generally generalized the function and status of multiple sequence alignment method in bioinformatics.

**Minor issue Comment 11:**

” Rephrase better: ‘method suffers from greediness’”

**Answer:**

Thanks for your suggestion. I realize that this sentence doesn’t state the main caveat of progressive alignment strategy directly and accurately. The main caveat of the progressive alignment approach is the existence of local minima. For instance, if the guide tree induces the alignment of two distantly related sequences, it often happens that the optimal alignment of these two sequences will not correspond to the pairwise projection one would get from the optimal MSA of the entire dataset. This situation is very common when dealing with low identity or low complexity sequences. When this occurs, the early computation of the first pairwise alignment may prevent the computation of a globally optimal MSA.

So we add the sentences: “the main caveat of the progressive alignment approach is the existence of local minima, e.g. the early computation of the first pairwise alignment may prevent the computation of a globally optimal MSA” in our paper to elaborate the caveat of progressive alignment more accurate and delete the sentence “method suffers from greediness”.

**Minor issue Comment 12:**

” Why is progressive alignment time consuming? Justify in text.”

**Answer:**

Apologize for my carelessness. The expression is not correct and I was wrong when I wrote the paper at the first time. You are right and I do agree that the complexity of progressive alignment strategy is no more than other strategies. However, the non-linear complexity can be a caveat of progressive strategy and this limit is rather severe in a context where the explosion of genomic sequence availability has resulted in unprecedented large homologous families that can require aligning up to 1.5 million members and soon many more. And when we need to run MSA on large-scale datasets, most strategy consumes too much time. So in our paper, we change the sentence to “progressive alignment is time-consuming when dealing with large-scale datasets, due to its nonlinear time complexity.”

**Minor issue Comment 13:**

” You never explain what an ‘iterative strategy is’.”

**Answer:**

Thanks for pointing it out. In the latest version of our paper, we add a citation on the algorithm and give the explanation of iterative strategy clearly. The paper we cite is named *Evaluation of iterative alignment algorithms for multiple alignment* published in Bioinformatics. It evaluates the how iterative alignment strategy improve the accuracy of MSA. And we add the sentence “…which is based on tree-based progressive strategy and involves re-estimating trees and alignments until both converge.” in our paper to summarize the primary idea of iterative strategy.

**Minor issue Comment 14:**

” You talk about the similarity matrix before defining it.”

**Answer:**

I am sorry for my carelessness and the mistake I made. We correct it in the latest version of our paper. We add the definition of what similarity matrix is in our paper. The definition is “an upper triangular matrix that stores the similarity scores of each of the two sequences. The similarity score is calculated by pairwise alignment.”

**Minor issue Comment 15:**

” Pg 7. What is m?”

**Answer:**

I am sorry to use an undefined term here. We correct the error in the latest version of our paper. We add the sentence “where m is the number of sequences.” to explain what it is clearly.

**Minor issue Comment 16:**

” Why does MAFFT fail on long sequences?”

**Answer:**

I was so wrong to make the mistake. We do some experiment after your pointing it out, and we find that the MAFFT doesn’t fail when dealing with sequences short than 100 thousand bps. I have state a wrong conclusion.

So in the latest version of our paper, we delete the sentence “but in most cases are ineffective for aligning very long sequences.”

**Minor issue Comment 17:**

” Pg 8. Correct: “based on a powerful data structure suffix tree” You might want to stick with the description of the data structure as either a trie (explain keys and values) or the tree (you do a good job explaining the nodes and edges). ”

**Answer:没看懂问题**

**Minor issue Comment 18:**

” Line 35: remove ‘as a tree’ You should probably define what the indexing operation S[] means. Also, should the definition include n+1 leaves to accommodate the empty string?”

**Answer:**

I am so sorry for I have made a mistake when I give the definition of suffix tree. The tree truly has n+1 leaves to represent n+1 suffixes of a string including a leave for empty string. Thanks a lot for pointing it out. And the operation S[i…n] is getting the substring of string S from ith character to nth character. But we think it is too complex and it doesn’t help to understand what the suffix is. So we delete the sentence with S[] operation and give the definition of suffix.

**Minor issue Comment 19:**

” Pg 10. ‘Then the process of picking out common strings…’ Describe the process.”

**Answer:**

I am sorry for neglecting the most important part of this paper. “the process of picking out common strings” is the key of our pairwise alignment algorithm. Thanks for pointing it out.

So in the latest version of our paper, we add a paragraph to describe the process. Apart from this, we realize that we just describe how to construct a suffix tree in O(n) time, here n is the length of a string, and we don’t show how to use a suffix tree to search for a show-up of a string’s prefix in another string which is the algorithms to select the prefix of a biological sequence in another sequence.

We first write a transition to change topic, the paragraph is shown below:

*“…For a pattern, for which we want to search, can be taken as a prefix of a suffix. A prefix is simply a subsequence that begins at the beginning of the sequence and extend to any position of the sequence. A pattern can be found in linear time by traversing a unique path in the tree from the root node to an inner node or a leaf node. In our work, suffix tree is used to find out the common substrings of two biological sequences in a pairwise alignment.****”***

Then we give the pseudocode of the process picking common substring in two biological sequences.

|  |
| --- |
| *Input: S1，S2: two Strings;tree-S1: the suffix tree of S1;*  *Output: result\_list: a list of substrings' information, whose element is composed of (a substring's start position in S1, a substring's start position in S2, the length of substring)*  *index=0;*  *while(index<S2's length){*  */\*select\_prefix is a function to find the S2's longest common prefix with S2 using suffix tree tree-S1. If a prefix is found the function will return the start postion of the substring in S1 and the length of the prefix，otherwise a tuple of [-1,0] will be returned\*/*  *st,len=tree-S1.select\_prefix(S2,index);*  */\*A common prefix(substring) is found\*/*  *if(st!=-1){*  *Record the start postion of the prefix in both S1(st) and S2(index) in result-list;*  *Record the length of the prefix in result-list;*  *index+=len;*  *}*  */\*Common prefix is not found\*/*  *else{*  *index++;*  *}*  *return result-list;*  *}* |

Then, we explain the pseudocode and how the process works:

*“A function named select\_prefix, which is a member function of suffix tree data structure, is applied to find the longest common prefix between a given input string and a suffix in the tree. When we search common substrings between a string and a suffix tree, the select\_prefix function is used to find a common prefix, then the common prefix is skipped and reused select\_prefix to find another common prefix. The process would be repeated until all character in S2 in scanned. Thus, in a single scan of string S2, all common substrings can be identified.” In this way, we use* pseudocode and its description to show how to pick all the common substrings out.

**Minor issue Comment 20:**

” Why define “subI” if you only use it in that one sentence? And this sentence is extremely confusing. subI is an integer, and then you say “subI’s first character’s index”. An integer’s first character’s index? There is another manual threshold, how is this set?”

**Answer:**

I am sorry for make the description redundant and confusing. Apart from this, there are some mistakes in this sentence. So we give up using these redundant terms and change the description of our standard for dropping out ineligible substrings. In the latest version of our paper, we rewrite three paragraphs to describe how we filter the substrings. The paragraphs are shown below:

*“The first standard is that common substrings cannot be too short. We set the length threshold at 20bp for nucleotide sequences. This is because very short strings are extremely common in nucleotide sequences. Very short common substrings between sub-S1 and sub-S2 would not correspond with one-to-one homology, which would lead to misalignment.*

*The second standard is that the starting positions of two matching substrings cannot be too far away. The absolute value of the difference between the starting positions of the two matching substrings must be less than their length. For highly similar sequences, most part of the sequences are the same. The matching of remote substrings would enlarge the areas which need to be aligned by dynamic programming algorithm so that the time of alignment is extended.*

*After finding all common substrings and discard the substrings that do not meet the first two criteria. These substrings are sorted according to their position in S1. In some cases, the order of matching substrings in S2 may be reversed. The position of substrings are not in increasing order. This situation is diagramed below in Figure 2. Although this situation does not occur often in highly similar sequences, we still have to deal with it. The LIS algorithm (DanGusfield, 1997) is employed to find the largest set of substrings which are in ascending order in both S1 and S2. As shown in Figure 2.”*

**Minor issue Comment 21:**

” Pg 11. The methods seem to suggest you don’t handle complex variations like inversions. You should explicitely state this. And Figure 2 isn’t very informative. Maybe a complete diagram of every variation you do not support should be shown instead.”

**Answer:**

You are right. Our method is aiming at dealing with highly similar homologous sequences at an extreme high speed. We will state the truth in the conclusion and state the caveat of our method.

**Minor issue Comment 22:**

” It’s hard to evaluate your claim that “picking and examining common substrings can also be accomplished in O(n) time” because an algorithm is not explicitly stated.”

**Answer:**

**I am so sorry for that we have not shown the algorithm of how to pick out common substrings clearly. We add two paragraph to explain the process and re-organized the algorithms. The** pseudocode of the process picking out the common substring is shown in the answer of minor issue comment 19. And we quote it again and list it blow:

|  |
| --- |
| *Input: S1，S2: two Strings;tree-S1: the suffix tree of S1;*  *Output: result\_list: a list of substrings' information, whose element is composed of (a substring's start position in S1, a substring's start position in S2, the length of substring)*  *index=0;*  *while(index<S2's length){*  */\*select\_prefix is a function to find the S2's longest common prefix with S2 using suffix tree tree-S1. If a prefix is found the function will return the start postion of the substring in S1 and the length of the prefix，otherwise a tuple of [-1,0] will be returned\*/*  *st,len=tree-S1.select\_prefix(S2,index);*  */\*A common prefix(substring) is found\*/*  *if(st!=-1){*  *Record the start postion of the prefix in both S1(st) and S2(index) in result-list;*  *Record the length of the prefix in result-list;*  *index+=len;*  *}*  */\*Common prefix is not found\*/*  *else{*  *index++;*  *}*  *return result-list;*  *}* |

The process is aiming at picking out all of the common substrings in two sequences with suffix tree algorithm. So the inputs are two strings S1, S2 and a suffix tree named tree-S1 constructed from S1. At the beginning of the process, we use S2’s longest suffix to traverse from root in the tree-S1 until the character doesn’t match and the process stops in a leave or an inner node of tree-S1, which means a common prefix between S2’s longest suffix and one of S1’s suffixes is found. Then process skips prefix in S2’s suffix and the remaining part of S2 is regarded as a new suffix and the process uses this suffix to restart a new traverse in tree-S2 from root. In the way, we can find all of the common substrings in two sequences in a single scan of the S2. The function named select\_prefix returns the start position of a common prefix and its length if a common prefix is found, -1 and 0 otherwise. This function takes O(m) time when a common prefix is found where m is the length of the prefix and O(1) time if no prefix is found. So the process of picking out the common substrings is O(n) where n is the length of S2.

In our paper, we have three criterions for filtering the selected common substrings. The first one is to drop out substrings short than 20 bps. The algorithm is very simple which consumes O(m) time, m is the size of substring set and m is far less than n, the length of a string. The second criterions is avoiding the transposition of substrings. Its algorithm is very simple, it just checks the start positions and lengths each pair of matching substrings and it is O(m). The third one need calculate the position of every two pairs of matching substrings so it needs O(m^2). However, m is far less than n, so that m^2 is less than n. In this way, our method for picking out substrings and checking them takes O(n) time in total.

**Minor issue Comment 23:**

” argmin over what? How is i chosen? How is this actually computed? i.e. What is the algorithm to select the center?”

**Answer:**

I am sorry for making mistakes here. The formula should be:

The purpose of this formula is to select a sequence in the set as the center sequence. So the variable we select is the sequence si which has maximum sum of scores. Here is the similarity of two sequences si and sj. A higher score means that two sequences are more similar. So the center has the least distance to all other sequences. The process of how the center is selected is explained below:

First, pairwise alignments between any two sequence in S are carried out, and a similar matrix, which is an upper triangular matrix that stores the similarity scores of each of the two sequences in set S, is calculated. Then the matrix is applied to compute the sum of scores between one sequence and others. The sequence which has maximum sum of scores is selected as the center sequence.

In the latest version of our paper, we re-edit the formula and add a paragraph to explain how the center is selected, which is marked in red color behind the formula.

**Minor issue Comment 24:**

” Pg 12. Correct: ‘two set of sequences’ I don’t know why you keep defining these long labels for quantities that you only use once.”

**Answer:**

**I am sorry for these confusing and redundant expressions. Thanks for pointing them out. Many of these redundant terms are deleted in the latest paper. These two set are important intermediate results. So we explain the sentence “two set of sequences” and add some explanations about this term. In this way, we don’t just define the term but use it as to make the description better. Thanks for your help.**

**Minor issue Comment 25:**

” What is a consequent sequence?”

**Answer:**

So sorry for this confusing description. The consequence sequences are the output of a pairwise alignment. The “consequence” here equals to “result”. They are the results of alignment. In the latest version of our paper, we give more detail information about the consequence sequences:

“…each pairwise alignment gives two consequent sequences, one is the result of center sequence after aligning (inserting the gap), the other one is another sequence after aligning.”

In this way, readers could have a better idea of what the consequence sequences are.

**Minor issue Comment 26:**

” What is the similarity matrix? How is it constructed?”

**Answer:**

I am sorry for using a term without definition and explanation. So we add the following sentences to explain what is the similarity matrix:

“…which is an upper triangular matrix that stores the similarity scores of each of the two sequences in set S.”

The similarity matrix is constructed by pairwise alignments, and we also add the process of the construction in our paper:

“Pairwise alignments between any two sequence in S are carried out, and a similar matrix, which is an upper triangular matrix that stores the similarity scores of each of the two sequences in set S, is calculated first.”

**Minor issue Comment 27:**

” Pg. 14. Scala collection?”

**Answer:**

I’m sorry to use a term without definition in advance or a citation. Scala collection is an abstract concept in Spark parallel framework. It just mean a collection or a set in computers’ memory, which could be distributed on different computers(nodes) in a cluster.

So in the latest version of our paper, we use “collection” instead and add sentence “The collection means a set of data which is distributed on different computers in a cluster.” to explain what the collection is clearly.

**Minor issue Comment 28:**

” Pg. 16. The logistics of moving data around might not be appropriate for the main body of the text. Consider moving to a supplement.”

**Answer: 我不太明白什么叫moving data around**

**Minor issue Comment 29:**

” Sentence beginning with ‘Concurrently’ needs to be rewritten. And, again, this isn’t justified mathematically.”

**Answer:**

Thanks for your advice and we realize that the sentence need to be corrected after you pointing it out. This first clause is quite confusing and the second clause “the parallelization of MASC seemed very promising.” need to be proved by Amdahl’s law. So in the latest version of our paper, we delete the first clause which is useless and it makes people feel confused and we change the sentence “there is no data dependency” to “there isn’t too much data dependency”. And then we add the mathematically proof of the second clause. We use Amdahl’s law and add a citation about where the formulas come from.

Amdahl’s Law:

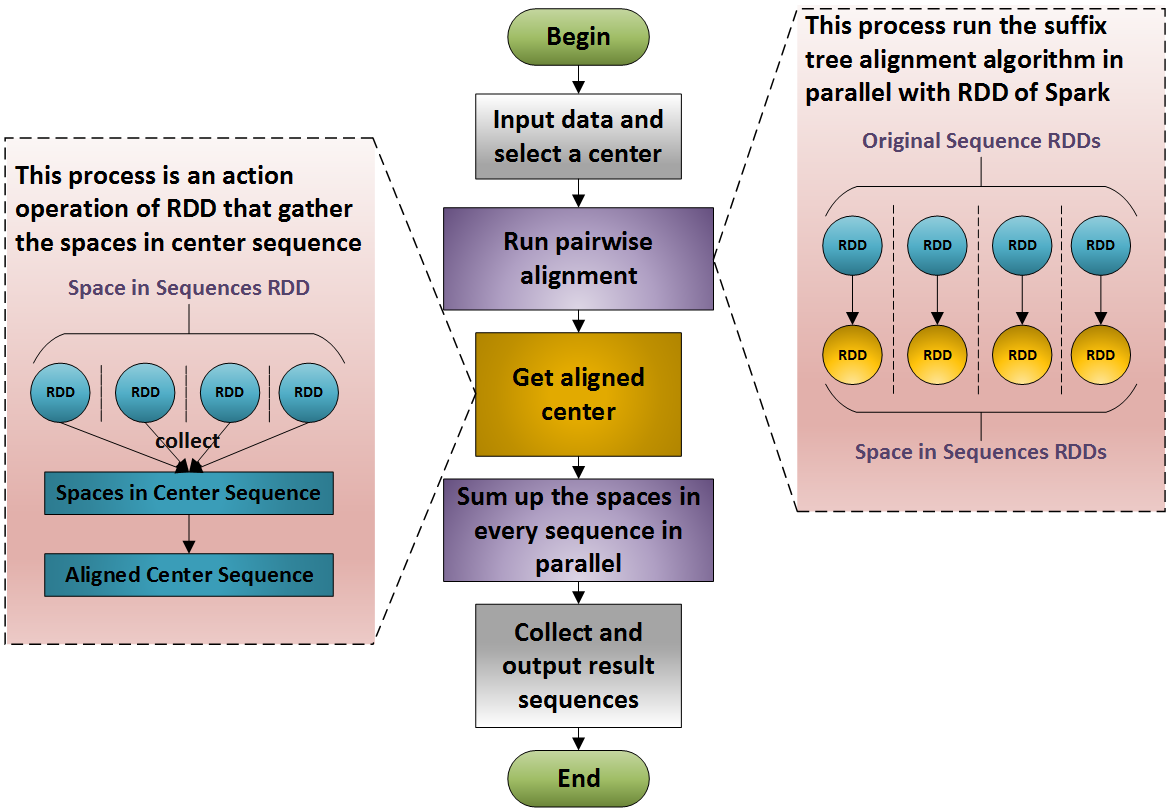
In our paper the data is that Tp=27260ms, and Ts=1741ms. So the limitation of the speedup in theory can be calculated, it is 16, which means the parallel version MASC could be 16 times faster than the original program.

**Minor issue Comment 30:**

” Fig 5. ‘an action’ ”

**Answer:**

I am sorry for my carelessness and misspell. We have correct this in the latest version of our paper. The figure is as:



**Minor issue Comment 31:**

” Pg. 18 Questionable motivation statement on line 24/25. Why do you need MSA to detect SNPs? Isn’t this usually done with sequence read pileups? You might want to state that simply duplicating a genome without introducing any variation is essentially an ideal scenario for your method.”

**Answer:**

**首先承认审稿人的说法，然后说明我们的方法的强项是什么。**

**Minor issue Comment 32:**

” Pg 19. I would suggest explaining why gaps penalized more than mismatches.”

**Answer:**

**这个要说明别也是这样做的，我也是没有什么办法。首先引用上邹权的文章还有杨晓春的文章**

**Minor issue Comment 33:**

” Figure 7. Label your axes.”

**Answer:**

**Minor issue Comment 34:**

” Pg. 21. There is probably a better label for the experiment than the file size in Table 2.”

**Answer:**

Thanks for pointing it out. In the latest version of our paper, we change use the sequence number once we align as the label in Table 2. The table is now as:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 672(1x) | 13440 (20x) | 33600 (50x) | 67200 (100x) |
| MASC-serial | 35s | 10m54s | 26m14s | 51m51s |
| MASC-spark | 7s | 1m50s | 5m11s | 8m44s |
| MAFFT | 1m59s | 3h52m14s | 21h54m18s | 3d12h41m42s |
| KAlign | 1h27m10s | --- | --- | --- |

**Minor issue Comment 35:**

” Pg 22. ’executor number’?”

**Answer:**

I am so sorry to use a term without define it in advance or add a citation. Executor is an abstract concept in Spark framework. An executor means a thread which can finish a job in parallel with other threads. We add a citation of *Spark: Cluster Computing with Working Sets* and add the sentence “Here executors mean threads that do pairwise alignments.” to explain what an executor is and what can it do. So the executor number means the thread number that run in parallel to do the MSA or the parallel scale in our experiment.