# Response to Reviewer #1:

1) *In Contigs Binning step, by using kmer coverages to determine the contigs membership, is the kmer size sensitive to this?*

The multiplicity can vary by different choices of kmer. While we only use the SPAdes optimized value of kmer for our pipeline, the multiplicity estimation only apply for contigs with significant length. In addition, topology information from the assembly graph is also considered to reduce the number of mis-classified cases.

2) *npGraph is a hybrid scaffolder based on streaming data. In table 1, Unicycler has high accurate assemblies than other algorithms, even it takes long time. Though with unique merits by itself, it would be good to show the performances and benchmarks among hybrid scaffolders with increasing coverage of long reads (Such as 1x -> 5x -> 10x -> 20x).*

Even though this analysis is of interested and would illustrate a more exhaustive benchmarking point of view, we do not want to set a case-study of runs on partitioned data (which has been conducted by Wick et al 2018) but focusing on a truely real-time operation instead.

Regarding this, npGraph and npScarf are the only tools to be able to run real-time hybrid assembly in which the results can be continuosly observed as the data coverage increased over time.

3) *Another two potentially missing software used in scaffolding would be FastSG (https://github.com/adigenova/fast-sg) and LRScaf (https://github.com/shingocat/lrscaf), which are two differences strategies on hybrid scaffolding. FastSG does the scaffolding step by simulating mate-pair information from long reads, whereas LRScaf is fast hybrid scaffolders on low coverage long reads by using full length information.*

FastSG and LRScaf are novel methods for hybrid scaffolding and are both designed to work with larger genomes than microgenomics data. While appreciate the limitations, authors want to claim that npGraph, similar to npScarf and Unicycler, is originally dedicated and optimized for microbial assembly. As a consequence, our comparison in the main text would only focus on a handful of tools explicitly nominated for this task.

Follow Reviewer #1 suggestion for additional benchmarking practice, we have attempted to install and run FastSG workflow on our HPC system in an fully automated way without success.

LRScaf is more straight-forward to use and fast to run. It requires a setup of parameters to operate with highest performance that varies between the wide-range of its supported genomes. We included the results of LRScaf with default parameters using SPAdes assembly and minimap2 alignments in the Supplementary Table S1.

4) *As mentioned by authors, npScarf is a greedy scaffolder with only 1 spanning long-read, whereas npGraph requires 3 reads. If improving the required number of long reads for npScarf, would the misassemblies and contiguous of assemblies be decreased? It is reasonable to benchmark the performance between npScarf and npGraph on similar circumstances.*

Even though require only 1 supported read to form the initial bridge, npScarf implemented a self-correction mechanism in which an existing bridge could be parted and reformed using different pair of anchors. Increasing this parameter can affect immediate results but would not much in long term. The parameter has been set as it is as a nature of methods themselves.

5) *Line 79, “In order to define a customised metric which is sample and fast to calculate”, is this a typo for simple?*

Typo corrected.

6) *In supporting figure S2, the figure is not consistent with the description.*

Typos corrected.

# Response to Reviewer #2:

0) *My only doubt about the proposed approach is the significance of having a streaming algorithm for hybrid assembler. There are important real-time analysis of the ONT reads, but I am not so sure about hybrid assembly.*

Having a streaming ONT hybrid assembly would facillitate the sequencing optimization mechanism, i.e. to preven over-sequencing or under-sequencing using the Oxford nanopore platforms when a complete assembly is of interest.

In fact, we are having a version of npGraph that enable ReadUntil protocol for selective sequencing oriented toward complete microbial genomes but it is not included here in the scope of this study.

1) *How the important parameters (the eps and min samples ) for DBSCAN are determined? What's their impact on the performance of npGraph.*

The default parameter for DBSCAN is set empirically on a set of significant contigs to identify the main replicon groups (equivalent to different chromosome, plasmids, taxa). It worked well with isolates or simple mixture of microbial genomes. In more complicated cases, additional curation might need to be done and/or using external binning algorithms.

2) *I assume that the binning results are used to constraint the bride candidates -- bridges are only considered between contigs in the same bin. This needs to be clarified in the methods.*

Bridges can be considered between contigs from different bins given enough evidence from long reads supporting such connections. Binning results thus can be calibrated accordingly over time as well.

3) *The paper mentioned that external binning algorithms including MetaBAT and maxBin can be utilized. Are they already implemented in the pipeline? If not, will that be straightforward to implement?*

Using of external binning algorithms has been implemented in npGraph. However in this research, only isolate data sets were subjected to study and the built-in DBSCAN algorithm was good enough.

4) *Is it a typo in algorithm 2: set of candidate paths connecting v0 to v2 -> v0 to vn?*

Typo corrected.

5) *The relationship between Algorithm 1 & 2 is not well explained.*

Edit second paragraph from the “Path finding algorithm” Section (line 163-177) for better explaination and connection between 2 algorithms. Basically Algorithm 1 is a sub-routine invoked inside Algorithm 2.

6) *It is unclear how the estimated multiplicity is used for path finding (aren't paths candidates ranked according to the likelihood computed based on long reads to contig alignment?).*

The multiplicity provides an estimation of the number of times a contig should appear in the final assembly thus it is important to assign uniqueness/repetitiveness of a contig. The multiplicity of shorter, repetitive contigs are less confident and only used to augment the likelihood score of a path in addition to the alignment score.

7) *Line 79, sample and fast -> simple and fast?*

Typo corrected.