Response Letter

Previous manuscript title: A general minimal perfect hash function for canonical k-mers on arbitrary alphabets with an application to DNA sequences

New manuscript title: General encoding of canonical k-mers

Roland Wittler

Response to Recommender

Comment 1: Your submission has now been reviewed by two reviewers. They found your preprint to be of interest; however, they raised a number of concerns. Two concerns in particular were raised by both reviewers. The first is the use of the term MPHF, which they found misleading. The second is the lack of convincing applications / usefuleness of your encoding.

I am therefore asking for a revision which addresses the reviewer's concerns. Though all concerns are important, I believe that demonstrating usefuleness will be critical for me to recommend the preprint.

Response: Thank you very much for the smooth and fair handling of the manuscript.

Even though the term MPHF formally fits very well of what is presented, I understand that it is confusing, because in practice it is commonly used to denote a data structure implementing a bijection for elements from a given subset and not for the whole universe of elements. This has been noted in the preprint before. But to avoid any confusion, throughout the whole manuscript, I now use the term *encoding*. To be precise, the term of an *minimal encoding* is introduced to emphasize the important property of mapping to the interval $[0, |U| - 1]$, where U is the universe of elements. I only briefly point to the term MPHF and clarify the relation. I also changed the title of the manuscript (and the git repository) accordingly.

I can also follow the second concern raised by both reviewers, i.e., the usefulness of the encoding in practice. Indeed, I am not aware of any obvious urgent need for the new encoding in practice. Sven Rahmann told me, he was happy about the encoding as it could be integrated in their bit-based compression approach to also allow handling even-length k-mers as explicitely mentioned in [Zentgraf & Rahmann, WABI 2022, `doi.org/10.4230/LIPIcs.WABI.2022.12`], Example 7. But since a restriction to odd-length k-mers is common practice anyway, there are probably more urgent things on Sven's agenda. I now reduced the

argumentation about any immediate practical use. I also removed the experiment on distributing k-mers to different buckets, because exactly this scenario is solved well in practice (usually by using randomized hashing). Instead, I now emphasize that the work at hand closes a gap in the theory of k-mers – in my opinion, a quite obvious and also interesting gap: Encoding, enumerating etc. of k-mers is straight-forward. Also the definition of being *canonical* is simple. In contrast, encoding, enumerating etc. of canonical k-mers appears to be complex. To my mind, the presented solution, in particular with its generalization (to arbitrary alphabets), presents a valuable contribution to the basic research in the field. I hope the rephrased introduction motivates this, and I would be pleased if recommender, reviewers and the audience of PCI MCB appreciate this theoretical work – without immediate use.

Response to Reviewer 1

Comment 1: The paper is written exceptionally well; the prose is precise and concise, helping to easily go thorough the technicalities. There only some minor formatting issues and typos (as noted below). I checked some of the math and found no errors. I would suggest, however, to add more examples to help the reader to better grasp the result of the various encoding procedures. It is a very elegant paper overall.

Response: First of all, thank you so much for acknowledging the style of writing/presentation. I am very happy to see than the effort to formulate a manuscript concisely pays off and is appreciated by the reader. Thank you also for pointers to smaller issues (which I corrected) and for motivating me to provide further examples. Now, every section is accompanied with an example.

Comment 2: My main criticism regards the usefulness of the proposed encoding. The application where this encoding could be used, as cited in the paper, is to implement a direct-addressing hash table where the encoded canonical k-mer is used as an index into the table. Since the proposed encoding uses 1 bit less than the simple $2k$-bit encoding, then the table can be shrunk by a factor of 2, from 4^k entries to $4^k/2$ entries. This is nice, but I guess only for relatively small values of k, e.g., $k < 16$ or so, for we expect to observe all the different canonical k-mers from the input. With larger values of k, instead, e.g. the "classic" $k = 31$, we expect to only observe a (small) subset of all possible k-mers making the direct-address table a poor choice in practice.

Response: As this comment goes into the direction of "usefullness" which I addressed in my response to the recommender, I will not go much further into detail here. In fact, as a proof of concept, I integrated the new encoding into the simple array-based k-mer counter "frigate" [Brihadiswaran & Sanath, Intern. Conf. on Bioinformatics and Biomedical Technology, 2021]. By halving the

gigantic memory consumption for medium size k (10 to 19), also the runtime was reduced significantly. But since in any case other k-mer counters such as KMC3 are faster anyway, this experiment is not mentioned in the manuscript. But there might be other applications where small values of k are of interest and direct-addressing is an option. I do not want to make anything up here and instead want to keep it as basic research.

Comment 3: I am not sure what is meant by "a MPHF allows/ensures an even distribution of k-mers to buckets of equal size". This appears multiple times in the paper, so I wanted to note this down. Is it meant that all slots in the table are used, with no holes (where a slot is a bucket of size 1)? To my knowledge, MPHFs are not used to ensure load balancing, rather they are used to address an array with (almost) no space overhead. In fact, we can always hash the canonical k-mers using a pseudo random hash function (like, Murmurhash or xxhash, etc.) and take the mod of the hash code to achieve almost perfect balancing. This is also noted in the paper. I do not know what the advantage of the presented method is compared to this; better balancing and speed, perhaps?

Response: As said in my response to the recommender, I clarified the relation to MPHFs and removed the experiment on the balancing.

Comment 4: Lastly, I find it a bit misleading the choice of the term "MPHF". A MPHF is a data structure implementing a bijection for the keys of a set S, which usually is a subset of another, larger universe U. Clearly, S can be U but usually it is not and the challenge is to design a data structure implementing the bijection knowing that we will only map a subset of the keys of U. (There is a large body of research on compressed MPHFs – see, e.g. BBHash, FHC, CHD, Recsplit, and the more recent PTHash and LPHash.)

My point is that the presented method does not work for a subset of canonical k-mers, but for the whole universe C_k^σ.

Mine is not a criticism – it is stated that the work focuses on hashing the entire universe (top of page 2) – but I'm afraid most people have a different notion of minimal perfect hash function in mind and would expect to be able to build a MPHF for any wanted set of keys. Therefore, I would suggest to use the term "bijective encoding" which seems more appropriate to me.

Response: See above.

Comment 5: Minor:
[...]
Also $T_{\sigma-1} \dots T_1$ do not seem to have been introduce (but perhaps I missed the point where they have!).
[...]

Response: Thank you very much for your careful reading. In the previous version, I kept the explanation regarding the *triangular numbers* and the rank function quite short to avoid this technicality distracting from the main explanation. It is now moved into Table 1 where the relation is also demonstrated by an example. I hope, also all issues are solved satisfactorily.

Response to Reviewer 2

Comment 1: A simple implementation is available on GitLab: `https://gitlab.ub.uni-bielefeld.de/gi/mphfcan/`. The code is perfectly clear and the README file provides the necessary information.

Response: Thank you very much for acknowledging even a rather simple proof-of-concept implementation.

Comment 2: Despite the paper could be improved with more examples or figures, it is globally easy to follow and understand. Nevertheless, I'd say that formalism may appear a bit heavy.

Response: Thank you very much. As mentioned above, further examples are added.

Comment 3: Terminology. The title and the whole text use the term "MPHF". I agree that enc_c and enc_c^r are technically speaking MPHFs. However, they can apply only on a full universe of ALL words of fixed size k on a given alphabet. SO the encodable set is of size A^k. This is not possible to encode a set of say a million k-mers of size 31 on the ACGT alphabet (for instance). This does not devaluate the work, but I really think that you should modify the name that can lead disappointed readers to stop the reading at the end of the abstract. I may suggest something as "general k-mer encoding".

Response: See my response to the recommender. And thank you for the suggested title.

Comment 4: Concrete usage. Certainly I missed something. I tried to imagine practical use cases in which enc_c^r would be beneficial, but I failed. In practice (on DNA alphabet), when I want to MPHF a set of k-mers I use the following steps: 1/ transform canonical (based on lexicographic order) ASCII k-mer to binary k-mers using the *enc* function you describe in the manuscript, 2/ create an MPHF on the encoded set. Note that during step 1, the distribution of k-mers is uneven as much canonical k-mers will appear in the "smaller binary values", but this has no impact on the MPHF construction in 2.

Using enc_c, step 1 would generate uniform distribution of binary values (on a smaller space, roughly of size $1/2A^k$) but finally the final step would ends up with the same result.

Response: See my response to the recommender.

Comment 5: Evaluation. The evaluation somehow reflects the previous remark. The only result is about the distribution of encoded k-mers to 16 buckets. Why 16? And, more importantly, I'd be nice to also compare with any bijective hash function (such as reversible xorshift hash function for instance provided by Heng Li `https://github.com/lh3/bfc/blob/master/kmer.h`).

So what are the concrete advantages ? Sorry if I missed this, but the message is not clear to me.

Response: See my response to the recommender. Since the evaluation does not show any relevant results, it is now removed.

Comment 6: More minor remarks
[...]
* Some definitions are generic while the differ when applied to completed sequences or not (eg canonical k-mer definition in the preliminaries, number of palindromes, ...)
[...]

Response: Thank you very much for your careful reading. I improved all mentioned minor issues. Only regarding the comment above, I am not sure what is meant.