Reviews for "Move Schedules: Fast persistence computations in coarse dynamic settings"

05/25/2023: This document records the reviews we obtained from a prior submission of the manuscript to the *Discrete & Computational Geometry Journal* (DCG), after which was deemed out-of-scope for DCG and recommended transfer to a different venue that is more aligned with its content.

Though our submission was directly transfered to APCT by one of Springer's Submissions Editorial Advisor (Rajshree Mahajan, submissionsupport@springernature.com), we were told that the prior reviews are not passed on automatically in the Snapp system, but that we may include them as supplementary material in submitting to APCT (email chains included with this document).

Below records two reviewer passes. Text that prefixed with the blockquote character '>' or the star '*' is text taken verbatim from the reviewer. Text prefixed two dashes "--" indicates our (the authors) responses. The newest submission of the manuscript addresses all of the reviewers comments thus far.

Reviewer Pass Number #2 (19 May 2023)

> My apologies to the authors that I needed a lot of time for this review cycle. I have read through the revised version. The authors made a good effort to take care of my initial feedback. I have some more remarks (below) to further improve the presentation.

> Most importantly, however, I was thinking how suitable is the contribution for the journal DCG. My recommendation, after reading for a second time, is to transfer the submission to a more specialized journal, for instance, to the Journal of Applied and Computational Topology.

> Here is the justification: I still think that the idea is interesting and the question of how to compute/update persistence in dynamic setting is useful. Also, despite DCG being a more theory-focused journal, I would consider a result that is mostly of algorithm engineering/experimental flavor to be in scope. Still, I would generally expect from a DCG paper a certain amount of ``outstandingness''. For instance, reasons that would make me recommend acceptance would be

- if the paper is of broad interest in the community: I would rather say it is of interest for the sub-community of computational persistence folks. It's part of DCG target audience, but a sub-group

- if the paper is a substantial step forward in its research field: I would rather classify the contribution as a proof of concept. After all, the experiments show that the number of column operations drops (significantly) using the updates, but that is only with respect to the standard barcode computation which is not used in practice. Even using the mentioned clear optimization, it is possible that the number of column operations drops significantly and also, there is no analysis of whether the runtime of the algorithm is proportional to the number of column operations (it's a valid hypothesis but it does not seem to be backed up in the experiments)

- if the paper introduces a novel technique: I think it is a nice combination of the move operation and finding best schedules. Certainly a valuable idea but perhaps not enough to recommend acceptance solely on its basis.

> The paper does have merits in all three aspects, but perhaps below the threshold that is required for DCG. Therefore, I cannot recommend acceptance. I would consider it suitable for the aforementioned APCT journal, however.

Line-by-line feedback

> More feedback:

- p.2 line 20: When d>>m, this .. is far more efficient - isn't is always more efficient?

-- The intention of the sentence was to juxtapose the theoretical efficiency of vineyards with its inefficiency in practice (the next paragraph). The vineyards strategy is indeed very efficient from the perspective of updating the decomposition continuously---in fact, each update takes just linear time---but as we make clear throughout the paper, continuously updating the decomposition is often not of much practical use when the time domain is coarsely discretized. The sentence has been redrafted to show this.

- p.3, line 27: "While vineyards is..."

-- Sentence reworked.

- General: The citation style is questionable: When citing a paper with two authors, "et al." should not be used. Also, sometimes, just the first author is named, and author names are misspelled ("Cohen-steiner")

-- All "et al." citations involving two authors have been replaced. The 's' has now been capitalized to "Cohen-Steiner" (which is that authors family name.)

- p.8, line 17: LCS seems not to be introduced before

-- A brief note has been added. As this is the introduction section summmarizing the results, its introduction was intentionally informal

- p.12, line 51: In the formula, I somehow miss why the indexing starts at i

-- The introduction of the indexing has been made clearer.

- Section 3.1: I sort of miss what is the point of the entire subsection. I think that only the discrete case is further considered.

-- Parts of section 3.1 have been re-writen, as have both the paragraphs immediately before and after it, for clarity sake. Aside from introducing some necessary notation, this small section (< 1.5 pages) is meant to introduce some of the combinatorial intuition behind why we cannot always discretize a continuous-interpolation between filtrations into a small (i.e. ~ O(m)-sized) set of critical events. This is important, as the central theme of the paper is to find an update strategy that balances how long it takes to update the matrix decomposition vs how long it takes to plan or "schedule" a stream of update operations. Section 3.1 shows that, in the continuous setting, there essentially cannot exist an algorithm for the latter under the assumptions we state, as the output itself is quadratic in size.

- p.31, line 41: I think saying that O(m^3) is prohibitive for exploratory data analysis is misleading. Barcode computation is also cubic in the worst-case but still practical (often). You might want to say that even linear time is too slow in practice in an interactive tool.

-- The sentence has been amended to more explicitly mention the interactive setting; the large amount of preprocessing RIVET does is designed to enable the interactive setting, which is what the sentence is trying to convey.

Reviewer Pass Number #1 (10 Jan 2023)

> This paper reconsiders the update algorithm for persistent homology via the Vineyard algorithm. The idea is that the Vineyard algorithm maintains a valid decomposition after every transposition of simplices, which is often not needed in applications. Instead, as shown in previous work (Ref [12] in the paper), one can proceed (practically) more efficient when performing a sequence of transpositions involving the same simplex (called "moves"). The current work builds up on this concepts and asks how to find a sequence of moves such that the updating time is minimized. For that, minimizing the number of moves seems like a natural objective, but since moves can incur very different costs, they consider a more refined, weighted strategy and solve it by a heuristic. Their experiments show a satisfying speedup of their method compared to computing persistence from scratch at all positions and to the standard vineyard algorithm.
>
> I think that this is an interesting direction of research, definitely worth of publication in a top-class venue, as the dynamic setup is of great interest for the TDA community. I think the value would be even greater if the paper would more clearly present the results as an experimental contribution - there is no shame in stating this. In the current form, after reading the abstract, I would have expected a "O-notation algorithm" paper, if I may use this slightly fuzzy term.

-- Thank you very much for the constructive feedback and thorough review. The language of the paper has been revised to focus less on the asymptotic statements and more on the experimental side of the paper. Moreover, in this revision we've made a few sections more succinct to try to streamline the presentation, though we note the core content of the paper is fundamentally unchanged. In particular, we've edited and reorganized subsections 3.1 and 3.2 to better explain the observations that lead to our methodology,

and we've moved a few things to the supplementary material (e.g. the pseudocode of reduction, move, and schedule construction algorithms). We've also included a small (new) subsection outlining the scalability of our greedy heuristic as a function of its various parameters in Section 4, and we've greatly streamlined the multi-parameter persistence section (now section 4.3). These were the largest revisions to the paper; point-by-point changes are addressed below.

Line-by-line feedback

> I would recommend to revise the paper mostly on the following points:

- The claim on page 22, line 34: "Thus, while O(m-sqrt(m)) in expectation is an improvement over O(m)" is factually wrong: m-sqrt(m) is at least m/2 and so, there is no asymptotic improvement. This might be easily fixable, but statements like these undermine the trust into the interpretation of results of the paper.

-- This was a mistake by the primary author. It has been fixed.

- The experiments seem to compare to Vineyards and to the static version. What I am missing is a comparison to `naive schedule'' (or maybe call it` vanilla [12]"): Just move the last simplex of the target filtration to the end, then the next to last one, etc. (or dually, start with the first one and move simplices backwards). In a way, that is the obvious thing to try if one knows [12], and I am wondering how much the new techniques gain over that. It is not clear to me whether the overhead to compute a (nearly) optimal move schedule is amortized by the speed-up.

-- We found this to be a valuable comparison and have added a benchmark including this strategy and others to section 4.1. We refer to the process of moving an appended simplex to its sorted position (or vice versa) as the "simple" in the figure in 4.1 (or the "selection-sort" strategy in the new text).

- As the experiments are artifically generated, it might be possible to change the parameters (m and d) and see how the performance changes in dependence of the parameter to get an idea of the asymptotic behavior in practice.

-- A figure has been added to section 4.1 addressing this point. In order to interpolate the schedule size d between m and its minimal value m - |LCS(...)|, we randomly remove elements from the LIS with a new parameter \alpha, which we discuss also in section 4.1.

- The application scenario of barcode templates is reasonable and important. It is only perhaps a bit excessive to spent around 10 pages of the paper to introduce it as it is not central to the contents of the paper (I admit that since I know this concept fairly well, I skipped over parts of this).

-- After discussing this extensively, we agree the multi-parameter persistence section in the original draft was too long (though we count 5 pages for the introduction, p31,l50 - p36,l44; we count the subsequent pages are part of the experimental results). We have edited this section to include only the essential information relevant to our use case, which is organized into 3 subsections: one to introduce the fibered barcode (4.3.1), one to describe the data set (4.3.2), and one to summarize the results (4.3.3). By our count, the fibered barcode subsection is under 2 pages, the dataset description is also 2 pages (including a figure), and the experiments are summarized in a single page. These lengths include figures. For interested readers, we include a slightly more expansive description of the 2d persistence algorithm in the supplementary section.

- Perhaps this is asking a lot, but I was wondering wehter the problem of computing the optimal weigted move schedule reduces to a problem in graph theory or optimization for which some lower bound on the complexity is known. I am thinking along the lines of 3SUM-hardness or other concepts which would make it unlikely that a subquadratic algorithm can be found to compute the optimum. This would give more justification to ressort to a different optimization scheme that is computationally tractable.

-- Thank you for the constructive feedback on this. We agree that a more structured reduction to a known NP-hard problem would be ideal, though we note we do discuss related NP-hard problems, such as the k-layer bipartite crossing minimization problem and rank aggregation problem, in section (3.4.2) [which introduces the proxy objective]. However, despite our effort, we do not know of a more structured reduction. In the spirit of transparency, we pose the objectives we seek to minimize as open problems in the conclusion and future work section. We also slightly reorganize sectioon 3.4.2 to better convey the intuition behind the proxy objective and its connections to existing work.

> Here is some additional line-by-line feedback:

- p.3, l 33: "dramtically" - I would not qualify the improvement as "dramatic" (but there is surely no common definition of "dramatic")

-- Removed the word.

- p.5, Fig 1: The figure is way too small. I cannot read the labels on the left

-- The figure has been made substantially bigger, its labels take up a larger proportion of space, and it's format has been re-exported via vector graphics for effectively infinite "zoomability." The same applies to the new figures in section 4.1.

- p.7, l 43: "to obtain 15" -> "to obtain (15)" (?)

-- All "\ref{}"s were refactored to "\eqref{}"s in the newest version. We previously used \ref{} to align more with the springer template for DC&G.

- p.8, l 45: a bit nitpicking, but "simplexwise" implies "essential"

-- Removed essential.

- p.13, l 14-27: I am not sure I fully understand the discussion on the data structure, but the author might find this remark useful: if all what is needed is to lookup whether an entry in a column at index i is not zero, one can just store a hash table that contains all indices of non-zero entries. Then, the lookup operation is supported in constant time (as well as insertion and deletion). Note that a hash table is *not* an appropriate data structure for matrix reduction itself because the entries are not ordered, so one does not have fast access to the pivot. But storing such a hash table additionally would only cost a constant factor in memory and constant-time lookup.

-- We realize the aforementioned paragraph did not communicate the subtleties we were trying to convey, and we've amended the paragraph accordingly. In keeping with the experimental nature of the paper, the take-a-way is that vineyards spends too much time "scanning" the status non-zero entries of the R and V matrices, due to the aforementioned quadratic blowup in permutations. Though the low entries may be accessed in constant time with an associative array implementation, vineyards additionally requires access to the non-zero status of arbitrary entries in the sparse matrices, which is typically not constant time using the ``standard'' sparse matrix representations (including the one in the original vineyards paper). We realize that none of these operations change the O(m) complexity a transposition in vineyards triggers---our only point is that stopping to evaluate these conditionals has a non-trivial impact at runtime due to how many transpositions are needed (for comparison, move operations group these queries into a single scan).

- p.14, l 55: The concept of "donor columns" is not explained. That is a bit of a pity

because it seems central to the approach (even though the description does not seem to require the concept)

-- The donor concept, including its motivation and derivation, has now been re-introduced in 2.3. We also include Busaryev's original proposition in the same section, as it is key to our results.

- p.17, l 47: The definition of S_F does not say that i,j are adjacent, but this seems to be meant, as it is written 2 lines before.

-- This is fixed in the newest revision.

- p.17, l 57: Perhaps a reference to the Kendall distance would be good here

-- Added reference to Diaconis' definiton.

- p.18, l 41: I am a a bit surprised to see moves discussed here, as the section is called "The Vineyard setting"

-- In the effort to re-organize section 3.1 and 3.2, we've re-named the sections the "Discrete" and "Continuous" settings to better illustrate to benefit of move scheduling.

- p.18, l 54: Why is the font size of the proof smaller?

-- We are using the D&CG recommended LaTeX Springer Template, which by default makes the proof font size smaller than the body font size (see line 1631 in the class file 'sn-jnl.cls' supplied by Springer). However, we agree that the change in font size seems off-putting, and the only requirements stated in the template/manual is that the amsthm environment should be used. We have changed the proofs to be normal size.

- p.20, l 32: The term "schedules" was used already before.

-- This has fixed in the newest revision.

- p.22, l 30: "from 3" -> "from Proposition 3"?

-- This is fixed in the newest revision.

- p.22, l 60: Where were i* *and j** defined?

-- This is fixed in the newest revision.

- p.23, l 28: "In the previous section" - which one do you mean? We are still in section

3.2.2 here. Do you mean Section 3.2.1 or 3.1 or 2?

-- This has been made more explicit, and the language "in the previous section" has been toned down throughout the paper.

- p.40, l 55: Put references for CSC and CSR

-- As there is no standard sparse matrix format (or reference for them), a note has been added.

- p.43, Ref [31]: The author name seems corrupted

-- The reference has a newer Springer version of the same approximation algorithm by the same author has been re-published this year, which we are now using.