Colossal Trajectory Mining: A Unifying Approach to Mine Behavioral Mobility Patterns

The authors have done a good job improving the paper compared to the previous version.

> We thank the Editor for recognizing our effort in improving the paper.
> Following the reviewers' concerns, we mainly reworked the evaluation section.
> Updates are highlighted in blue.

Reviewer #1

The manuscript is addressing the most important comments well. However, there are still some issues regarding the experiments that need to be addressed.

Section 6.5, a comparison of the proposal with benchmark algorithms from the literature was performed, please address the following concerns:

- I missed the evaluation of the robustness of the presented results, an example is to present the evaluation with dataset with different characteristics of patterns and add others benchmark algorithms.

> To improve the robustness testing we extended CTM evaluation, aiming at proving that its performance remains stable and it is not overfitted to a specific test setting. In particular, our tests now cover the following settings.

Approach/Dataset	**Oldernburg**	**Hermoupolis**
CTM	**✓ (time + #patterns)**	**✓ (time + #patterns)**
SPARE	**✓ (time + #patterns)**	**✓ (time + #patterns)**
PFPGrowth	**✓ (time + #patterns)**	**✓ (time + #patterns)**

> While *before* the review it was

Approach/Dataset	Oldernburg	Hermoupolis

CTM	✓ (time + #patterns)	
SPARE	✓ (time)	
PFPGrowth	✓ (#patterns)	

- Datasets. We tested SPARE against the whole Oldenburg, Hermoupolis, and Milan datasets. However, SPARE fails to compute whole datasets due to its Apriori-like (exponential) space exploration. Hence, we added tests on a reduced version of Hermoupolis with 200 trajectories. Milan is too sparse to build a reduced version in which meaningful co-movement patterns can be extracted.
- Algorithms. SPARE is the only big data approach to the generic extraction of co-movement patterns (as written in the paper). We added more tests with PFPGrowth, an efficient algorithm for the extraction of Frequent Itemset Mining. However, we did not add further algorithms for mining FIs since such approaches are not meant to extract co-movement patterns.

This proves that the performance of CTM is stable/robust in different settings: it always performs better for lower support thresholds and it always produces fewer patterns than PFPGrowth.

Following these points, we reworked Section 6.5, and the updates are reported in blue.

Note that

- We tested only against the swarm pattern since FI algorithms cannot fulfill shape and semantic constraints (see Table 3; so only swarm or co-location patterns can be directly post-processed from FIs). We extract swarm patterns since they require a bigger spatio-temporal tesselation rather than a spatial one (i.e., a worse setting for CTM).
- CTM's complete tests with different pattern types and scalability configurations can be found in Sections 6.3 and 6.4.

- Does the author do a time analysis with SPARE and the amount of FIs with PFPGrowth, could you include the three algorithms in the two comparisons? even needing to increase the mSup range.

We extended the tests and the result can be observed in Figure 9.

The paper reads as follows

\Cref{fig:comparison} depicts the comparison of the three approaches in terms of computational times and number of retrieved patterns (with the same hardware and software configurations).

\begin{itemize}

\item CTM and SPARE retrieve the same patterns (i.e., their $|P|$ lines are overlapping).

However, following an Apriori-like strategy, SPARE performance highly depends on the minimum support: the lower it is, the higher the likelihood of extracting patterns with sufficient cardinality and the wider the portion of search space to be enumerated.

This results in memory faults when $mSup$ is below 25 even in the modified $\sf{Oldenburg}$ dataset and the computation is stopped after three hours in the modified $\sf{Hermoupolis}$ dataset already for $mSup=30$ (there is no green line in the top-right chart).

Conversely, in CTM the computational time is less affected by the minimum support. \item As to PFPGrowth, although both FIs and co-movement patterns decrease by increasing the $mSup$ threshold (as expected, the higher $mSup$ the lower the valid patterns), the number of FIs to post-process remains orders of magnitude higher than the co-movement patterns, and with lower supports their extraction becomes orders of magnitude slower (as confirmed in \cite{DBLP:journals/tkde/LuccheseOP06}).

This makes FIM approaches strongly inefficient.

\end{itemize}

We close this section with a remark on the robustness of the algorithm. The tests shown above for CTM on different datasets and its comparison against different algorithms show that its performance is stable and is not the result of overfitting to a specific dataset or configuration.

- Furthermore, when comparing PFPGrowth with the proposed algorithm, why not explore the mSup interval used in Figure 9? Does the proportion of FIs is hold?

We extended the mSup interval, so that both datasets are tested with same range mSup \in [5, 30] (see Figure 9)