**Gelsa: GPU-Accelerated Local Similarity Analysis Toolkit**

Yang Li1, Shuaishuai Xu2,#, Shengwei Hou2,#, Yinhui Xiong1, Li Charlie Xia1,\*

*1School of Mathematics, South China University of Technology, Guangzhou 510641, China*

*2Department of Ocean Science & Engineering, Southern University of Science and Technology, Shenzhen 518055, China*

# Equal contribution.

\* Corresponding author(s).

1. mail: [lcxia@scut.edu.cn](mailto:lcxia@scut.edu.cn) (Xia L).

标准图表尺寸为85毫米（单栏）和175毫米（双栏），宽度为238毫米，纵向或横向。请调整字体大小以确保清晰度。图表格式可以是ppt、ai（Illustrator）、jpeg。请在手稿中添加行号，并确保提交的PDF文件具有相应的行号。

1. 在首次提及时，请将术语完整拼写，并在括号中标注缩写。

**Abstract**

We present a novel parallel computing method called GeLSA (GPU-accelerated local similarity analysis), designed to accelerate the analysis of time series data, particularly in the fields of biological and environmental sciences. GeLSA enhances computational efficiency by ~144x acceleration on a GPU machine and enables large-scale data processing in a short time scale, thanks to its parallelisation of the alignment and dispatch algorithms. GeLSA integrates a range of algorithms designed for local correlation analysis, including theo and perm, MBBLSA, DDLSA, and STLTA, so that GeLSA can analyze non-independent and identically distributed time series data. GeLSA maintains the same level of accuracy as traditional methods while significantly outperforms them in terms of efficiency and speed when benchmarking on simulated datasets. The research on real datasets underscores the efficiency of GeLSA in analyzing local correlations. Due to the computation efficiency and algorithm optimization of GeLSA, we expect it will be widely used in ecological data analysis as its predecessor eLSA does.

**KEYWORDS:** Local similarity analysis; Accelerated local correlation computations; Time series data

**Introduction**

In fields such as biological and environmental sciences, analyzing the interactions among multiple factors and their impacts on ecosystems or biological populations is often essential. Time series analysis serves as an effective method to understand the interactions of various factors over time. Traditionally, interactions were mainly detected using approaches developed based on the overall correlation of pairwise factors over the entire time interval. However, the real biological or environmental data often exhibit more complex relationships and dynamic changes, including localized and lagged correlations, as observed in various fields such as microbiology [1-4], molecular biology [5,6], and functional neuroscience [7,8]. Thus, methods based on global similarity analysis may fail to detect these relationships. To address the limitations of global correlation, other alternative methods have been introduced, among which local similarity analysis (LSA) [10-13] has been widely adopted. LSA is a locally aligned method that identifies optimized alignment configurations between any given two time series, thereby detecting paired local or delayed correlations. This local similarity-based analysis has been adapted by other methods, such as local trend analysis (LTA) [14-16], to identify local patterns in time series datasets. Through these analyses of local similarity, we can gain deeper insights into the relationships among factors. This is crucial for predicting trends in biological population changes, community dynamics, and more.

LSA is widely used in many fields and has undergone significant theoretical and practical improvements. Qian et al. initially proposed the LSA method for analysing gene expression profiles [10], which was later introduced to analyze microbiome data by Ruan et al [13] and Xia et al. [12]. Methodological advances in LSA also include support for replicated series data (eLSA) [12], statistical significance approximation (*p*-values) [11,9], moving block bootstrap LSA (MBBLSA) [17], and data-driven LSA (DDLSA) [18]. Xia et al. [16] first proposed a theoretical approximation of the statistical significance of LTA, which was further improved by Shan et al. as steady-state theory local trend analysis (STLTA) [19]. MBBLSA, DDLSA, and STLTA enable LSA to perform well on autocorrelated data, which addresses the limitation of traditional LSA,

With the continuous deepening of research in this field and the ongoing development of data mining techniques, traditional LSA computing softwares need some changes. On the one hand, previous computational methods for LSA incur significant computational costs, requiring quadratic comparisons for n data points. On the other hand, novel theoretical algorithms for LSA, (MBBLSA, DDLSA and STLTA) [17-19], have expanded the scope of LSA and LTA research to analyze autocorrelated time series data. Here, we propose a parallel computing method to accelerate the local similarity analysis of time series data. This method speeds up the computation process via parallelization and optimization of the LSA computing core, significantly reducing the time and space complexity of computations. We compare the efficiency of GeLSA with the LSA computing core on a single core. We compare the overall software performance of GeLSA and eLSA on simulated datasets, found that GeLSA greatly enhanced the efficiency and accuracy of time series data analysis by ~144x acceleration on a GPU machine. Additionally, we integrate the expanded algorithms for LSA (MBBLSA, DDLSA and STLTA) [17-19], which enables effective analysis for autocorrelated time series data.

The structure of this paper is as follows: Materials and methods, this article introduces the GeLSA algorithm, introduces the simulation and real datasets. Results, this article gives the efficiency of GeLSA computing core, demonstrates the overall efficiency of GeLSA, performes analysis on the continuous 72-hour dataset from Daya Bay, draws the analysis results into correlation networks. Discussion, the paper gives the evaluation of GeLSA.

**Materials and methods**

**Algorithm optimization**

Firstly, we redesigned the LSA algorithm(Fig.1).In theory, for a pair of time series data, the new LSA algorithm, compared to the traditional LSA algorithm, reduces the spatial complexity from 3 \* n^2 to (2D+1) \* n, where D is the time series offset, n is the length of the time series, and D is much smaller than n.

To take advantage of GPU computing cores, we redesign the traditional LSA algorithm based on GPU architecture.The time complexity of GeLSA is O(T), while the time complexity of LSA is O(t\*m^2), where T is the time complexity of new LSA, and t is the time complexity of LSA, m is the number of time series.

**Simulation Studies and Real Datasets**

To evaluate the runtime efficiency of the GeLSA software, we conducted analysis computations by using both GeLSA and the eLSA software on the same simulated dataset. We compared the runtime of these two software packages and generated 8 performance comparison graphs.

Firstly, for given (n, m) parameters, we generated n pairs of independent and identically distributed standard normal random variable pairs (Xi, Yi), where each pair represents observations on the time series, and each time series has a length of m. Xi and Yi are mutually independent.

Next, on one hand, keeping the number of data points (m) fixed, we varied the number of data points (n) and conducted experiments to obtain the runtime of both software packages as the length of the time series data changed; on the other hand, keeping the number of data points (n) fixed, we varied the number of data points (m) and conducted experiments to obtain the runtime of both software packages as the number of the time series data changed. Each experiment generated a simulated dataset, and the corresponding runtime was got. Under this comparison method, we obtained the runtime of both software packages theoretically (GeLSA theo, DDLSA, STLTA, and eLSA theo) and experimentally (GeLSA perm, MBBLSA, and eLSA perm). Each algorithm was run 5 times on the experimental dataset, and the mean runtime was calculated to obtain a sufficient and reliable amount of data for evaluating the performance of GeLSA and eLSA software.

Finally, based on the runtime of the above software, we plotted line graphs of the runtime to visually demonstrate the performance differences between the two software packages.

We applied the GeLSA method to analyse the Daya Bay dataset. We performed GeLSA calculations on this dataset and then plotted correlation networks based on it.The Daya Bay dataset consists of ASV (Amplicon Sequence Variants) abundance data sampled at high frequency over 72 hours. We selected 400 abundant ASVs from the Daya Bay dataset, with a minimum relative abundance of 1%, covering up to 97.0% of all ASVs. For the time series, we examined time-lagged correlations with delays of (0 hours, 6 hours, 12 hours, 18 hours, 24 hours, and 48 hours). P-value comparisons for local similarity analysis were conducted in supplementary Table S1.

**Results**

**Performance of Computation Core**

The effectiveness of the new designed algorithm are clearly demonstrated through the comparison of runtime graphs(Fig.2).The new designed algorithm achieved better performance both on a single CPU core and on a GPU.

On the one hand, (Fig.2 A), we compared the computational performance of the new LSA algorithm and the traditional LSA on a single CPU core. The experiments show that the new LSA computing core has improved performance compared to the traditional LSA computing core. For example, when m = 1000, new LSA = 6775000s, LSA = 10023300s, the runtime ratio = 1.47; when m = 5000, new LSA = 171716081s, LSA = 253681000s, the runtime ratio = 1.48; and when m = 10000, new LSA = 693567567s, LSA = 1026480000s, the runtime ratio = 1.48.

On the other hand,(Fig.2 B), we compared the computational performance of the traditional LSA with the GeLSA computing core. Theoretically, the GeLSA algorithm has a significant improvement in time complexity compared to the LSA algorithm. The time complexity of GeLSA is O(T), while the time complexity of LSA is O(t\*m^2), where T is the time complexity of new LSA, and t is the time complexity of LSA, m is the number of time series. Meanwhile, the experiments show that the GeLSA computing core has improved performance compared to the traditional LSA computing core. For example, when m = 1000, GeLSA = 69606.25s, LSA = 10023300s, the runtime ratio = 144; when m = 5000, GeLSA = 1761673.61s, LSA = 253681000s, the runtime ratio = 143; when m = 10000, GeLSA = 7128333s, LSA = 1026480000s, the runtime ratio = 144.145.

**Performance Comparation of Softwares**

Overall, it can be clearly seen that each algorithm in GeLSA (MBBLSA, perm, DDLSA and STLTA) exhibits significantly improved efficiency compared to eLSA.

It is evident that the parallel algorithms, which designed based on theoretical formulas, can fully utilize the hardware resources of the computer, resulting in a considerable improvement in runtime efficiency of GeLSA relative to eLSA (Fig.3 A1). For theoretical calculations, there is a substantial reduction in runtime compared to eLSA theo in the same dataset. For example, when m=100, GeLSA theo=13.37s, GeLSA DDLSA=15.54s, while eLSA theo=188.49s, indicating a speedup of 14.09 times. Similarly, conclusive results can be derived from (Fig.3 A2) regarding the LTA theoretical algorithm.

For permutation-based experimental computations, GeLSA also exhibits significantly improved runtime efficiency compared to eLSA, (Fig.3 B1). The computational performance improvement is especially valuable for analyses where theoretical calculations (n<20) are not feasible. Both for autocorrelated time series data (GeLSA MBBLSA) independent and identically distributed time series data (GeLSA perm), there is a substantial reduction in runtime compared to eLSA perm under similar computational loads. For example, when m=10, GeLSA perm=51.67s, GeLSA MBBLSA=125.33s, while eLSA perm=704.16s, indicating a speedup of 13.89 times. Similar conclusions can be drawn from (Fig.3 B2) regarding the LTA permutation algorithm.

Although only the length of the time series data is varied, in (Fig.3 C1), it can be observed that GeLSA significantly outperforms eLSA in theoretical computations, similar to the findings in (Fig.3 A1). For example, when n=20, GeLSA theo=10.75s, GeLSA DDLSA=11.42s, while eLSA theo=164.04s, indicating a speedup of 16.09 times. Similar conclusions can be drawn from subplot C2 regarding the LTA theoretical algorithm. Additionally, it can be observed that the runtime of GeLSA DDLSA is almost identical to GeLSA theo, which is consistent with the findings of Zhang F et al. [18], indicating a negligible difference of 0.12% on average for n=20, 60, and 100.

It is evident that GeLSA significantly outperforms eLSA in permutation algorithm, (Fig.3 D1), similar to the findings in (Fig.3 B1). For example, when n=20, GeLSA perm=384.13s, GeLSA MBBLSA=1957.38s, while eLSA perm=25669.11s, indicating a speedup of 65 times. Similar conclusions can be drawn from subplot D2 regarding the LTA theoretical algorithm. Additionally, it's evident that the runtime of GeLSA MBBLSA is significantly slower than GeLSA perm, consistent with the findings of Zhang F et al. [17], indicating a 1.5 times slower runtime for GeLSA MBBLSA compared to GeLSA perm for n=20, 60, and 100.

**Algorithm Tidely**

We compared the results obtained from running GeLSA and eLSA software using the simulated dataset generated in section 2.2 "Simulation Studies and Real Datasets". The outcomes are presented in (Fig.4). Overall, it is evident from the plots that there is no deviation between the GeLSA's "theo" algorithm and the traditional eLSA software's "theo" algorithm when analyzing a dataset comprising 100 pairs of sequences which generated 4950 pieces of data.

It is clear that, (Fig.4 A), the results of GeLSA(theo) and eLSA(theo) for LS (local similarity) calculations are consistently distributed along the diagonal line, forming a straight line composed of closely aligned points. This suggests that GeLSA performs LS calculations accurately. In (Fig.4 B, C, D), the results of GeLSA(theo) and eLSA(theo) for the starting positions (xs, ys) of local maxima, as well as for the subsequence lengths (len), are consistently distributed along the diagonal lines, forming straight lines composed of closely aligned points. Similarly, in (Fig.4 E), the results of GeLSA(theo) and eLSA(theo) for p-values are consistently distributed along the diagonal line, indicating accurate calculations. This observation is consistent with the conclusion drawn from (Fig.4 A), further corroborating the software's accuracy.In (Fig.4 F), it is evident that this comparative experiment of software performance was conducted with a delay of 0.

**The Daya Bay Time Series Dataset**

In the 72-hour time series data from Daya Bay, potential microbial interactions between phytoplankton and prokaryotes were identified using GeLSA. These interactions include symbiosis, cross-nutrition, competition, parasitism, predation, and allelopathy. It was found that several significant time-lagged correlations (Spearman's |R| > 0.70, P < 0.01) exist between major phytoplankton taxa and specific prokaryotes in the Daya Bay time series. Notably, significant correlations were observed between certain diatoms and members of the Alphaproteobacteria, Gammaproteobacteria and Bacteroidota.

Furthermore, time-lagged correlations were also observed between dominant MGII (Marine Group II) archaea and diatoms such as *Chaetoceros* (Bacillariophyta) and *Gyrodinium* (Dinophyta). These findings provide insights into the interactions between traditional phytoplankton and prokaryotes, offering a higher level of phylogenetic and temporal resolution.

(Fig. 5) illustrates the microbial association network in the 72-hour time series of Daya Bay, showing that the local similarity correlations with time-lags associated with major phytoplankton taxa often involve bacteria and archaea, indicating temporal delays.

**Discussion**

The computational workload, which previously took nearly a month using the traditional eLSA theoretical method, can be completed in just one day by fully leveraging hardware resources with the GeLSA software. It's important to note that the perfromence of software acceleration mainly involve two aspects: the number of data points (m) and the method of calculating p-values. The number of data points determines whether the computer CPU hardware is fully utilized, while the p-value calculation method determines whether the computer GPU hardware is fully utilized. Therefore, when analyzing and optimizing software acceleration effects, it is necessary to consider the influence of both factors.

This advancement significantly enhances the efficiency of data analysis, allowing us to process massive datasets rapidly. This breakthrough accelerates workflow and aids in a deeper understanding and utilization of data. The Python-C++ implementation of GeLSA can be found at <https://github.com/labxscut/GeLSA>.

**Authors’ contributions**

YL revamped the LSA algorithm and applied it for parallel computation on both CPU and GPU, further parallelized the software by circumventing Python's GIL. YL participated in software performance and correctness comparisons, contributed to drafting the manuscript. SSX analyzed the network graph of the Daya Bay dataset computation results and translated this analysis into text. SWH, YHX, and LX conceived the research, participated in its design and coordination, and assisted in drafting the manuscript. All authors read and approved the final manuscript.

**Competing interests**

The authors have declared no competing interests.

**Acknowledgements**

The National Natural Science Foundation of China (42276163), Shenzhen Science, Technology and Innovation Commission Program (JCYJ20220530115401003), and Shanghai Frontiers Science Center of Polar Research (SOO2004-03) granted to S.H. (Shengwei Hou) were also acknowledged.**References**

[1] Caporaso JG, Lauber CL, Costello EK, *et al.* Moving pictures of the human microbiome. *Genome Biol* 2011;**12**:R50.

[2] Cram JA, Xia LC, Needham DM, *et al.* Cross-depth analysis of marine bacterial networks suggests downward propagation of temporal changes. *ISME J* 2015;**9**:2573–86.

[3] Steele JA, Countway PD, Xia L, *et al.* Marine bacterial, archaeal and protistan association networks reveal ecological linkages. *ISME J* 2011;**5**:1414–25.

[4] Shade A, McManus PS, Handelsman J. Unexpected diversity during community succession in the apple f lower microbiome. *MBio* 2013;**4**:e00602–12.

[5] Cho RJ, Campbell MJ, Winzeler EA, *et al.* A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell* 1998;**2**: 65–73.

[6] Spellman PT, Sherlock G, Zhang MQ, *et al.* Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*

1998;**9**:3273–97.

[7] Amar D, Yekutieli D, Maron-Katz A, *et al.* A hierarchical Bayesian model for f lexible module discovery in three-way time-series data. *Bioinformatics* 2015;**31**:i17–26.

[8] Vaisvaser S, Lin T, Admon R, *et al.* Neural traces of stress: cortisol related sustained enhancement of amygdala-hippocampal functional connectivity. *Front Hum Neurosci* 2013;**7**:313.

[9] Durno WE, Hanson NW, Konwar KM, Hallam SJ. Expanding the boundaries of local similarity analysis. *BMC Genomics* 2013;**14**(Suppl 1):S3.

[10] Qian J, Dolled-Filhart M, Lin J, *et al.* Beyond synexpression relationships: local clustering of time-shifted and inverted gene expression profiles identifies new. Biologically relevant interactions. *J Mol Biol* 2001;**314**:1053–66.

[11] Xia LC, Ai D, Cram J, *et al.* Efficient statistical significance approximation for local similarity analysis of high-throughput time series data. *Bioinformatics* 2013;**29**:230–7.

[12] Xia LC, Steele JA, Cram JA, *et al.* Extended local similarity analysis (eLSA) of microbial community and other time series data with replicates. *BMC Syst Biol* 2011;**5**:S15.

[13] Ruan Q, Dutta D, Schwalbach MS, *et al.* Local similarity analysis reveals unique associations among marine bacterioplankton species and environmental factors. *Bioinformatics* 2006;**22**: 2532–8.

[14] He F, Zeng AP. In search of functional association from timeseries microarray data based on the change trend and level of gene expression. *BMC Bioinformatics* 2006;**7**:69.

[15] Ji L, Tan KL. Identifying time-lagged gene clusters using gene expression data. *Bioinformatics* 2005;**21**:509–16.

[16] Xia LC, Ai D,Cram JA, *et al.* Statistical significance approximation in localtrend analysis of high-throughputtime-series data using the theory of Markov chains. *BMC Bioinformatics* 2015;**16**:301.

[17] Zhang F, Shan A, Luan Y. A novel method to accurately calculate statistical significance of local similarity analysis for high-throughput time series. *Stat Appl Genet Mol Biol* 2018; **17**:20180019.

[18] Zhang F, Sun F, Luan Y. Statistical significance approximation for local similarity analysis of dependent time series data. *BMC Bioinformatics* 2019;**20**:53.

[19] Shan A, Zhang F, Luan Y. Efficient approximation of statistical significance in local trend analysis of dependent time series. *Front Genet* 2022;**13**:729011

**Figure legends**

**Figure 1 Algorithm Demonstration of GeLSA**

**Figure 2 Comparison of Performance on Computational Cores Between GeLSA and eLSA**

**A**. Comparison of computational performance between the new LSA algorithm and traditional LSA on a single CPU core. **B**. Comparison of computational performance between traditional LSA and the GeLSA computing core.

**Figure 3 Overall Software Performance Comparison between GeLSA and eLSA**

There are a total of 8 subgraphs in Figure 3, and the line graph data in each subplot are obtained by averaging the results of 5 experimental tests.

**Figure 4 Comparison of Correctness in Running Results between GeLSA and eLSA**

With the settings were configured as follows: m=100, n=50 and d=0, this figure vividly illustrates that both the local similarity score LS (**A**) and other statistical measures (p\_value, xs, ys, len, delay) (**B**, **C**, **D**, **E**, **F**) show remarkably consistent computational results between GeLSA and eLSA.

**Figure 5 Network Visualization of GeLSA Computation Results on the Daya Bay Dataset**

**Supplementary material**

**Figure S1 Comparison of Correctness in Running Results between GeLSA and eLSA**

With the settings were configured as follows: m=100, n=50 and d=5, this figure vividly illustrates that both the local similarity score LS (**A**) and other statistical measures (p\_value, xs, ys, len, delay) (**B**, **C**, **D**, **E**, **F**) show remarkably consistent computational results between GeLSA and eLSA, with delay = 5.

**Figure S2 Comparison of Correctness in Running Results between GeLSA and eLSA**

With the settings were configured as follows: m=100, n=50 and d=10, this figure vividly illustrates that both the local similarity score LS (**A**) and other statistical measures (p\_value, xs, ys, len, delay) (**B**, **C**, **D**, **E**, **F**) show remarkably consistent computational results between GeLSA and eLSA, with delay = 10.

**Table S1 The result of P-value comparisons for local similarity analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **source** | **target** | **LS** | **p\_value** |
| ASV1 | ASV7 | 0.763482392 | 0 |
| ASV1 | ASV13 | 0.821513355 | 5.44E-15 |

*Note*: For the Daya Bay dataset with 400 entries, GeLSA was employed for LSA calculations.