Progressive enumeration of bitriangles in large bipartite networks

Juan Pablo Royo Sales
Universitat Politècnica de Catalunya
Barcelona, Spain
juan.pablo.royo@upc.edu

Edelmira Pasarella
Universitat Politècnica de Catalunya
Barcelona, Spain
edelmira.pasarella@upc.edu

Cristina Zoltan
Universitat Politècnica de Catalunya
Barcelona, Spain
zoltan@cs.upc.edu

Maria Esther Vidal
TIB-Leibniz Information Centre of Science and Technology
Leibniz University of Hannover
L3S Research Centre
Hannover, Germany
maria.vidal@tib.eu

ABSTRACT

In this work, we tackle the problem of providing an algorithm for progresively enumerating bitriangles in large bipartite networks. In bipartite graphs, the bitriangle is considered the smallest unit of cohesion so that counting bitriangles is extremely useful for conducting social analysis and computing metrics such as clustering coefficient. Thus, there has been efforts in developing efficient algorithms for counting bitriangles. Besides, structural information about bitriangles is useful for any problem requiring linking data in bipartite networks, e.g. in disorder/disease gene association networks where it is important to know the gene causing the disease. Depending on the size of the graph, enumerating or listing bitriangles could be a high-resource consuming task and an "all-or-nothing" computation approach might not deliver any result at all. In this work we propose an algorithm for enumerating bitriangles progressively based on a concurrent/parallel computational model called the Dynamic Pipeline Approach. This computational model is oriented to stream processing and nicely supports the *pay-as-you-go* approach. In addition, we conduct, analyze and report experiments against large bipartite networks having up to 350 millions of bitriangles. Obtained results satisfy our expectations.

CCS CONCEPTS

• **Theory of computation → Parallel computing models**; **Concurrent algorithms**; **Streaming models**; • **Computing methodologies → Concurrent computing methodologies**.

KEYWORDS

Bipartite Graphs, Bitriangles, Concurrency, Parallelism, Stream Processing, Algorithm

womENcourage'24, June 26–28, 2024, Madrid, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Juan Pablo Royo Sales, Edelmira Pasarella, Cristina Zoltan, and Maria Esther Vidal. 2024. Progressive enumeration of bitriangles in large bipartite networks. In *Proceedings of womENcourage™ 2024 Responsible Computing for Gender Equality ACM Celebration of Women in Computing (womENcourage'24).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In many real-world applications, the relationships among two different types of entities can be modeled by means of bipartite graphs. This is a graph in which the set of vertices is formed by the union of two disjoint sets of vertices corresponding to the two different types of entities in the relationship. In this kind of bipartite graph or network, edges only connect vertices from each one of the different entities or vertices. In the literature, we can find many examples of bipartite networks in different domains. Just for mentioning some few examples we have phenotype-disease gene associations network (*diseasome* bipartite network) [2], drugs-side effects network [11], customer–product network, author-paper network, the Netflix subscribers-TV shows etc. In general, the majority of metrics used for analyzing unipartite graphs, for instance, clustering coefficient, social analysis, or triangle-based community computation [6, 7, 10] are based on computing the number of triangles in the network. Since computing and using triangles do not fit well for the case of bipartite graphs, Opsahl [8] proposes to use another locality graph pattern or motif, the bitriangle or 6-cycle, i.e. a cycle with three vertices from one type of vertices and three vertices from the other type of vertices, as the smallest unit of cohesion of a bipartite graph. In his work, Opsahl argues that bitriangles in bipartite graphs capture the idea of the triadic closure in unipartite graphs. Figure 1 depicts an example of a bitriangle. Yang et al. [12] study the problem of counting bitriangles and, propose and analyze, different algorithms to do it. In particular, they propose an algorithm for local counting bitriangles. This is, counting bitriangles in a bipartite graph in which a given vertex/edge occurs, i.e. according to a given criterium.

Depending on the nature of the addressed problem, counting bitriangles in a (possibly persistent) bipartite graph is not enough to establish underlying relationships among its vertices. This happens because establishing these relationships could require knowing

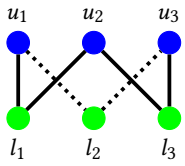

Figure 1: A bitriangle formed by the 6-cycle $\langle u_1\, l_1\, u_2\, l_3\, u_3\, l_2\, u_1\rangle$. The considered graph is $G = (L \cup U, E), L \cap U = \emptyset$, $l_1, l_2, l_3 \in L$ and $u_1, u_2, u_3 \in U$. The bitriangle can be seen as a convenient composition of three *wedges*: $\langle u_1\, l_1\, u_2\rangle$, $\langle u_2\, l_3\, u_3\rangle$ and $\langle u_1\, l_2\, u_3\rangle$, where l_1, l_2 and l_3 are the middle vertex, respectively.

specific structural details. We mean that not only the number of bitriangles is important but which are these bitriangles. For example, in the *diseasome* bipartite network presented in [2] two disorders are connected if there is a gene that is implicated in both. In order to make decisions, a scientist not only could need to know how many bitriangles are in that graph but –an even more specific question– given a disorder which genes are involved (connected) to it. Link prediction problem considers evolving networks. Effective streaming processing of large amounts of data has been studied for several years [1, 3, 5] as a key factor providing fast and progressive results in big data algorithmic problems. One of the most explored techniques, regardless of the approach, is the exploitation of parallel techniques to take advantage of the available computational power as much as possible. In this regard, the *Dynamic Pipeline Approach* (DPA) [9] has lately emerged as one of the models that exploit data streaming processing using a dynamic pipeline parallelism approach [5]. In this computational model a dynamic pipeline (DP) is deployed, i.e. the DP increases and decreases depending on incoming data. The stages of the DP are stateful functions. The implementation of an algorithm according to the DPA is suitable to generate progressive results. Thus, the dynamic pipeline approach is a proper computational model for solving the problem of enumerating (progressively) bitriangles.

2 ALGORITHM FOR ENUMERATING BITRIANGLES

Depending on the size of the considered bipartite graph, enumerating all the possible bitriangles is computationally hard and a high resource-consuming process. Hence, an "all-or-nothing" computation approach can lead to a timeout or memory stuck. For overcoming this problem, instead of considering a classical enumerating algorithm, we follow an iterative "pay-as-you-go" approach [4]. This means that bitriangles can be progressively emitted as results so that users continuously receive answers from the algorithm as long as the provided resources support the computation. In that sense, we provide a query-oriented algorithm to search and list the bitriangles that matches a given query criteria and that incrementally deliver results to the user.

The algorithm for enumerating incrementally bitriangles in a large bipartite graph consists in two main phases. During the first phase, the bipartite network is received by the dynamic pipeline (DP) as a stream of edges and a graph index structure is created. This graph index is represented by the different structures stored along the stages of the DP. In effect, the first phase decomposes the graph into compressed structures: The algorithm aggregates results, i.e. the set of wedges having the same middle vertex is represented as a pair $\langle l, W_l\rangle$ where l is the middle vertex and W_l is the set of adjacent vertices of l called *aggregated wedge*. The algorithm first collects aggregated wedges for the vertices in L. Afterwards it constructs *aggregated double wedges* for every pair of distinct vertices in L. Finally, constructs *aggregated bitriangles* for selected triples of vertices. The second phase is a querying phase. During the querying phase local queries can be submitted to the DP. When a query arrives to the pipeline, the process of progressive enumeration of bitriangles –according to the criteria in the submitted queries– is launched. In this phase, enumerated bitriangles are extracted from the different aggregated bitriangles occurring in the graph index, i.e. the aggregated bitriangles stored in the stages of the dynamic pipeline. See [13] for details of the algorithm and an empirical study about it.

ACKNOWLEDGMENTS

Edelmira Pasarella is partially supported by MCIN/ AEI /10.13039/501 100011033 under grant PID2020- 112581GB-C21. Maria-Esther Vidal is partially supported by Leibniz Best Minds: Programme for Women Professors project TrustKG-Transforming Data in Trustable Insights with grant P99/2020.

REFERENCES

[1] Foto N. Afrati, Dimitris Fotakis, and Jeffrey D. Ullman. 2013. Enumerating subgraph instances using map-reduce. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*. 62–73. https://doi.org/10.1109/ICDE.2013.6544814

[2] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. 2007. The human disease network. *Proceedings of the National Academy of Sciences* 104, 21 (2007), 8685–8690.

[3] Michael I Gordon, William Thies, and Saman Amarasinghe. 2006. Exploiting coarse-grained task, data, and pipeline parallelism in stream programs. *ACM SIGOPS Operating Systems Review* 40, 5 (2006), 151–162.

[4] Yike Guo, Moustafa Ghanem, and Rui Han. 2012. Does the cloud need new algorithms? an introduction to elastic algorithms. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*. IEEE, 66–73.

[5] I Lee, Ting Angelina, Charles E Leiserson, Tao B Schardl, Zhunping Zhang, and Jim Sukha. 2015. On-the-fly pipeline parallelism. *ACM Transactions on Parallel Computing* 2, 3 (2015), 17.

[6] Jesper Nederlof. 2019. Detecting and Counting Small Patterns in Planar Graphs in Subexponential Parameterized Time. arXiv:1904.11285 [cs.DS]

[7] M. E. J. Newman. 2003. The Structure and Function of Complex Networks. *SIAM Rev.* 45, 2 (Jan 2003), 167–256. https://doi.org/10.1137/s003614450342480

[8] Tore Opsahl. 2011. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. arXiv:1006.0887 [physics.soc-ph]

[9] Edelmira Pasarella, Maria-Esther Vidal, Cristina Zoltan, and Juan Pablo Royo Sales. 2024. A computational framework based on the dynamic pipeline approach. *Journal of Logical and Algebraic Methods in Programming* (2024), 100966.

[10] Thomas Schank and Dorothea Wagner. 2005. Approximating Clustering Coefficient and Transitivity. *Journal of Graph Algorithms and Applications* 9 (01 2005). https://doi.org/10.7155/jgaa.00108

[11] Mohan Timilsina, Meera Tandan, Mathieu d'Aquin, and Haixuan Yang. 2019. Discovering Links Between Side Effects and Drugs Using a Diffusion Based Method. *Scientific Reports* 9 (07 2019), 10436. https://doi.org/10.1038/s41598-019-46939-6

[12] Yixing Yang, Yixiang Fang, Maria E. Orlowska, Wenjie Zhang, and Xuemin Lin. 2021. Efficient Bi-Triangle Counting for Large Bipartite Networks. *Proc. VLDB Endow.* 14, 6 (Feb. 2021), 984–996. https://doi.org/10.14778/3447689.3447702

[13] J. P. Royo Sales, E. Pasarella, M. E. Vidal, C. Zoltan,. [n. d.]. An algorithm for incrementally enumerating bitriangles in large bipartite networks. https://upcommons.upc.edu/handle/2117/361615. Accessed: 2024-02-20.