Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

tinygarden — A java package for testing properties of spanning trees

Manuel Dubinsky [a,*], César Massri [b], Gabriel Taubin [c]

[a] Ingeniería en Informática, Departamento de Tecnología y Administración, Universidad Nacional de Avellaneda, Argentina
[b] Departamento de Matemática, Universidad de CAECE, CABA, Argentina
[c] School of Engineering, Brown University, Providence, RI, USA

ARTICLE INFO

Keywords:
Graphs
Spanning trees
Fundamental cycle bases

ABSTRACT

Spanning trees are fundamental objects in graph theory. The spanning tree set size of an arbitrary graph can be very large. This limitation discourages its analysis. However interesting patterns can emerge in small cases. In this article we introduce *tinygarden*, a java package for validating hypothesis, testing properties and discovering patterns from the spanning tree set of an arbitrary graph.

Code metadata

Current code version	1.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2021-47
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/9539109/tree/v1
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	java
Compilation requirements, operating environments & dependencies	jdk
If available Link to developer documentation/manual	https://github.com/manudubinsky/tinygarden/wiki
Support email for questions	mdubinsky@undav.edu.ar

1. Introduction

Graph theory is a longstanding and well-established area of discrete mathematics. Graphs are abstract models of pairwise relations between entities in some domain. A graph $G = (V, E)$ is composed of V -a set of *vertices* (or *nodes*)- and E -a set of *edges*-. Each node is a point representing an entity and each edge is a line connecting two nodes. For example we can model friendship relationships in a social network by associating a node to each person and connecting two nodes with an edge if the corresponding people are friends.

Let $G = (V, E)$ be the graph in Fig. 1.

A *subgraph* $G' = (V', E')$ is a graph such that $V' \subseteq V$, $E' \subseteq E$ and such that V' contains all endpoints of the edges in E'. A *path* $p = (e_1, \ldots, e_k)$ is a sequence of consecutive edges of G (see Fig. 2).

A *cycle* in G is a non-empty path in which the only repeated nodes are the first and last (see Fig. 3).

We say that G is *connected* if there is at least one path between each pair of nodes. A *tree* is a connected graph without cycles. And finally, a *spanning tree* $T \subseteq G$ is a tree that contains all the nodes of G (see Fig. 4).

Spanning trees are important in optimization [1], network design [2], VLSI interconnection [3], clustering [4], complexity theory [5], graph invariants [6], fundamental cycle bases [7] and in many more areas of applied and theoretical sciences.

The spanning tree set of an arbitrary graph can be very large [8]. This limit prevents the entire set from being explored. However, there are algorithms to generate all spanning trees of a graph [9–11] that are suitable in the case of small instances.

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: mdubinsky@undav.edu.ar (M. Dubinsky).

https://doi.org/10.1016/j.simpa.2021.100083
Received 26 April 2021; Received in revised form 6 May 2021; Accepted 10 May 2021

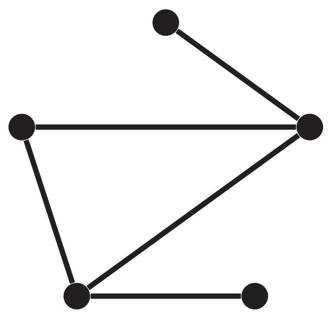
Fig. 1. Example of a graph.

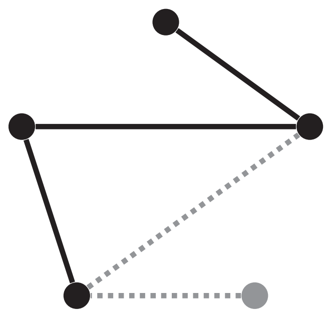
Fig. 2. Example of a Path.

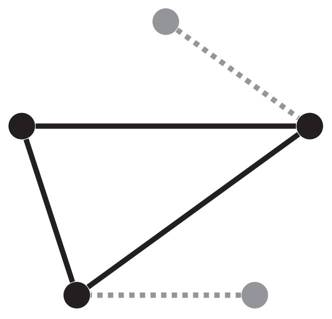
Fig. 3. Example of a cycle.

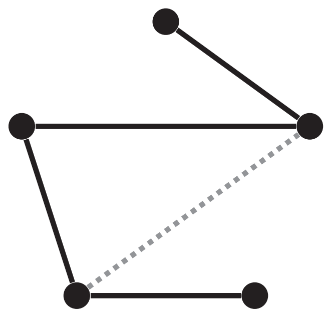
Fig. 4. Example of a spanning tree.

2. Software description

The *tinygarden* package is a self-contained group of *java* classes that enables the exploration of the set of spanning trees of a given graph. It implements *Matsui algorithm* [11] to generate it.

The architecture is simple. The class *SpanningTreesMatsui* organizes the whole process. Custom descendants of two class hierarchies: *Collectors* and *Processors* are invoked sequentially to process each spanning tree. *Collectors* are useful for global analysis of the entire set, for example, count the number of spanning trees with a certain property. Processors, on the other hand, are useful for local tasks, such as generating a pretty-print version of a specific spanning tree.

The package also implements helper classes, such as *Graph*, *SparseMatrix*, *UnionFind*, etc. Since a *Graph* can be built based on a text file containing its incidence matrix, it works well together with *nauty* [12], the well-known graph software.

3. Limitations and improvements

The scope of the *tinygarden* package is restricted to specific graph theory problems. The most important limitation is the size of the instances that can be processed; it was conceived to explore the spanning tree set of all non-isomorphic graphs of at most 9 nodes. Consequently, two important improvements would be:

- A distributed implementation of Matsui algorithm: in order to process larger graphs.
- Compute the spanning tree set size [13]: to process graphs with smaller size compared to a prescribed threshold.

The first version of the package was recently made public. At the moment is only being used by the authors.

4. Impact

The *tinygarden* package is a tool for validating hypothesis, testing properties, and discovering patterns from the spanning tree set of an arbitrary graph. To the best of our knowledge this is the only tool with these features.

In [14] we introduced the *mstci problem*, the *tinygarden* package proved to be effective in finding a pattern related to the *intersection number* of complete graphs which led to its main result, and to set a solid basis for the conjecture posed in it. It was an important tool to evaluate a metaheuristic algorithm for graph integration [15]. We are currently using it to test an algorithm in order to find a good solution to the *mstci problem*.

The package can be used to analyze general questions, i.e.: list the spanning trees with minimum diameter of a graph, as well as to carry out statistical analysis, i.e.: calculate the distribution of the graphs of a fixed number of nodes with respect to the spanning tree with shortest total path length [5].

Based on our package, several NP-hard problems can be analyzed from a statistical perspective, for example Fig. 5 shows the distribution of the *Minimum Fundamental Cycle Basis* (or *Min FCB*) [16] of 8 node graphs. This information can be used to implement approximate or heuristic algorithms in order to find good solutions for these problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Prof. Massri was supported by Instituto de Investigaciones Matemáticas “Luis A. Santaló”, UBA, CONICET, CABA, Argentina.

Prof. Taubin was partially supported by the National Science Foundation, USA grant number IIS-1717355.

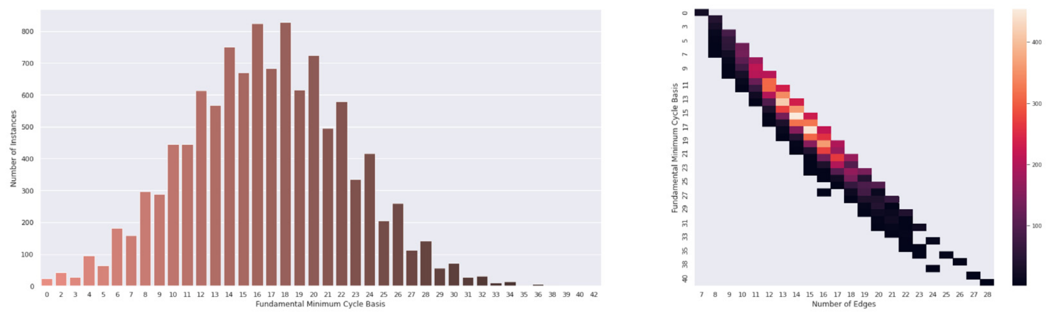

Fig. 5. Distribution of the Min FCB of the graphs of 8 nodes (left). Aggregated relation of the Min FCB with respect to the number of edges (right).

References

[1] B. Wu, K. Chao, Spanning Trees and Optimization Problems, in: Discrete Mathematics and Its Applications, CRC Press, 2004, URL https://books.google.com.ar/books?id=SI82UzzyVN8C.

[2] D. Eppstein, Spanning trees and spanners, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, North-Holland, Amsterdam, 2000, pp. 425–461, http://dx.doi.org/10.1016/B978-044482537-7/50010-3, URL https://www.sciencedirect.com/science/article/pii/B9780444825377500103 (Chapter 9).

[3] J. Cong, L. He, C.-K. Koh, P.H. Madden, Performance optimization of VLSI interconnect layout, Integration 21 (1–2) (1996) 1–94, http://dx.doi.org/10.1016/s0167-9260(96)00008-9.

[4] L. Rokach, A survey of clustering algorithms, in: Data Mining and Knowledge Discovery Handbook, Springer US, 2009, pp. 269–298, http://dx.doi.org/10.1007/978-0-387-09823-4_14.

[5] M. Garey, D. Johnson, Computers and Intractability: A Guide To the Theory of NP-Completeness, Macmillan, New York, 1979, p. 340.

[6] W. Tutte, A contribution to the theory of chromatic polynomials, Canad. J. Math. 6 (1954) 80–91, http://dx.doi.org/10.4153/cjm-1954-010-9.

[7] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, K.A. Zweig, Cycle bases in graphs characterization, algorithms, complexity, and applications, Comp. Sci. Rev. 3 (4) (2009) 199–243, http://dx.doi.org/10.1016/j.cosrev.2009.08.001.

[8] A. Cayley, A theorem on trees, Q. J. Pure Appl. Math. 23 (1889) 376–378.

[9] R. Read, R.E. Tarjan, Bounds on backtrack algorithms for listing cycles, paths, and spanning trees, Networks 5 (3) (1975) 237–252, http://dx.doi.org/10.1002/net.1975.5.3.237.

[10] H.N. Gabow, E.W. Myers, Finding all spanning trees of directed and undirected graphs, SIAM J. Comput. 7 (3) (1978) 280–287, http://dx.doi.org/10.1137/0207024.

[11] T. Matsui, Algorithms for Finding All the Spanning Trees in Undirected Graphs, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo (1993), 1993, URL https://www.keisu.t.u-tokyo.ac.jp/data/1993/METR93-08.pdf.

[12] B.D. McKay, A. Piperno, Practical graph isomorphism, II, J. Symbolic Comput. 60 (2014) 94–112, http://dx.doi.org/10.1016/j.jsc.2013.09.003.

[13] G. Kirchhoff, Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird, Ann. Phys. Chem. 148 (12) (1847) 497–508, http://dx.doi.org/10.1002/andp.18471481202, URL https://doi.org/10.1002/andp.18471481202.

[14] M. Dubinsky, C. Massri, G. Taubin, Minimum Spanning Tree Cycle Intersection problem, Discrete Appl. Math. 294 (2021) 152–166, http://dx.doi.org/10.1016/j.dam.2021.01.031.

[15] M. Dubinsky, C. Massri, F. Asteasuain, Una metaheurística GRASP para integración en grafos, in: XV Simposio Argentino de Investigación Operativa, (SIO)–JAIIO 46 (Córdoba, 2017), 2017, pp. 36–44, URL http://sedici.unlp.edu.ar/handle/10915/66445 (in Spanish).

[16] N. Deo, M.S. Krishnomoorthy, G. Prabhu, Algorithms for generating fundamental cycles in a graph, ACM Trans. Math. Software 8 (1) (1982) 26–42.