Journal of Computer Science

Original Research Paper

An Empirical Criterion for Transitive Closure

Marius Orlowski

Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, USA

Article history
Received: 23-03-2022
Revised: 07-09-2022
Accepted: 02-12-2022

Email: m.orlowski@vt.edu

Abstract: The paper establishes a simple and computationally economical criterion for determining whether a given adjacency matrix represents a transitive closure or not. The criterion is based on a new iterative method for finding the transitive closure adjacency matrix. This method is equivalent to but computationally more efficient than the traditional Boolean sum of successive powers of the original adjacency matrix to determine the transitive closure matrix. However, it is computationally less efficient than the Warshall algorithm and the recent more advanced algorithms. Nevertheless, in many cases, in which a given matrix is different from the transitive closure, but may not differ much, a few of the proposed looping steps may suffice to find the transitive closure, avoiding the computational burden of the best-in-class algorithms. Thus, the criterion and the recursive relation are apt in complementing the extant methods to establish an efficient evaluation engine for the determination of the transitive closure.

Keywords: Transitive Closure, Graph Theory, Warshall Algorithm, Search Algorithms

Introduction

Many scientific and technical problems can be represented as directed or undirected graphs. As a result, developing methods for tackling graph problems is both practically and theoretically interesting. In many applications of the graph theory, the determination of the transitive closure is a common technique. On a sequential computer architecture, the well-known Floyd algorithm (Warshall, 1962; Floyd, 1962) solves the problem in O (n^3) steps, where n is the dimension of the corresponding adjacency matrix.

Many computational tasks require an efficient determination of the transitive closure of a given graph. In computer science, for example, the concept of transitive closure amounts to the construction of a data structure that renders it possible to find an answer to reachability inquiries.

The transitive closure algorithms are also used in the construction of parser automata in compilers. Since, practically, all actual recursive searches are transitive, effective transitive closure computation has recently been identified as an important subclass of problems in analyzing recursive database queries (Cormen *et al*., 2009). Compiler construction for parsing automata also uses transitive closure operations (Aho *et al*., 2008). All-pairs shortest routes problems, in which the shortest paths between arbitrary vertex pairs must be discovered (Aho *et al*., 2008), are, too, intimately connected with transitive closure. Today, the fundamental challenge of transitive closure algorithms is the numerical efficiency of their algorithms.

A directed graph is an ordered pair G (V, E) where V is the set of vertices and E is the set of edges between pairs of vertices. The transitive closure G* connects vertices u and v if and only if there is a path in G from u to v.

The Warshall method (Warshall, 1962; Floyd, 1962) or repeated breadth-first search (Cormen *et al*., 2009) or depth-first search (Even, 2011) can be used to solve the problem.

Frequently, one may encounter an adjacency matrix that is assumed to be already the transitive closure or to be extremely close to transitive closure in several scenarios. In the case of existing algorithms, the full algorithm must run its course, regardless of its computational efficiency, to establish the transitive closure, as the final result. This is a computationally intensive process, particularly, for a large adjacency matrix.

However, the adjacency matrix at hand may already represent a transitive closure. Nevertheless, to confirm it, one would have to apply one of the transitive closure algorithms. For a large matrix this is very time-consuming.

Therefore, it would be useful to avail oneself of a criterion that provides a simple, computationally efficient calculation that allows one to determine upfront whether a particular adjacency matrix already reflects the transitive closure. The paper provides such, numerically very efficient, criterion which for a large class of cases obviates the need to use unreduced and time-consuming algorithms to determine a transitive closure for a given graph, cutting, thus, the numerical burden drastically.

The Powering of Graph Methods and Warshalll Algorithms

The basic method for determination of the transitive closure consists in by taking the adjacency matrix A of a given graph and raising it to the nth power, where n is the number of vertices in G. Raising the adjacency matrix A to the kth power, is equivalent to adding exactly the edges which represent paths of length k in the original graph. Since the longest path may be only of length n, the addition of all power matrixes is equivalent to adding paths of all lengths that are possible in a given graph.

The "powering of graph" method states therefore that to determine the transitive closure Z* it is sufficient to fulfill Eq. (1) (Nuutila, 1998):

$$Z^* = A + A^2 ... + A^n \qquad (1)$$

using Boolean operations OR (+) and AND (.) and *A*. $A^j = A^{j+1}$. The computation time for Eq. (1) is O (n^4) bit operations. The well-known algorithms (Warshall, 1962, Floyd 1962) reduce the number of operations from O (n^4) to O (n^3). Given an adjacency n × n matrix with matrix elements a(i,j), the Warshall algorithm, given in Eq. (2), determines the adjacency matrix Z* by the following operations on the matrix elements (Warshall, 1962):

$$a^{p+1}(s,r) := a^p(s,r) + a^p(s,I).a^p(I,r) \; 1 \le s,r,l \le n \qquad (2)$$

For convenience, the sign "·"between the matrix elements will be suppressed in the following. The algorithm constructs a sequence of adjacency matrices $W_o, \ldots W_n$, where W_o=A and each ¸W_k represents all paths of G containing no intermediate vertices of greater length than k. Consequently, $W_n = Z^*$. Recently, because of increased interest in hierarchical and recursive queries in databases transitive closure algorithms attracted more research and resulted in more advanced algorithms based on accelerated algorithms for matrix multiplication, (Alman and Williams, 2021; Duan *et al*, 2022).

Materials and Methods

In this purely mathematical investigation, no materials have been used and the standard methods of linear algebra have been employed.

Results and Discussion

Proposed Recursive Matrix Relation for Transitive Closure

This study proposes a novel recursive matrix relation to calculate the transitive closure for a given directed graph, represented by the adjacency matrix A by generating a sequence of matrices Z_k:

$$Z = A \quad Z_{k+1} = Z_k + Z_k \cdot Z_k \;\; k = 0, \lceil In_2 n \rceil \qquad (3)$$

where, for a real number x $\varepsilon \mathbb{R}$, the notation of ⌈x⌉, the so-called ceiling function, is implemented to determine the nearest larger integer. It can be readily proven that $Z_{\lceil \ln 2n \rceil}$ must represent the transitive closure matrix Z*.

Proof: Any matrix Z_k, from Eq. (3) computed after k iterations for *k*>1, can be written in the following form:

$$Z_k = Z_0 + \sum_{k=1}^{m_2} Z_0^2 + ... + \sum_{k=1}^{m_{k^2-1}} Z_0^{k^2-1} + Z_0^{k^2} \qquad (4)$$

There m_j are integers resulting from Eq. (3) By way of illustration, for three explicit iterations one obtains the following three equations:

$$Z_1 = Z_0 + Z_0 \cdot Z_0$$

$$Z_2 = Z_1 + Z_1 \cdot Z_1$$

$$Z_3 = Z_2 + Z_2 \cdot Z_2$$

Resulting in:

$$Z_3 = W_0 + 3Z_0^2 + 6Z_0^3 + 9Z_0^4 + 10Z_0^5 + 8Z_0^6 + 4Z_0^7 + Z_0^8$$

Thus, the coefficients are m_2=3, m_3=6, m_4=9, m_5=10, m_6=8, m_7=4, and m_1=m_8=1, where the subscript of parameter m denotes the power of the matrix $Z_o^{j\cdot}$

It should be noted that under the logic OR represented symbolically here by "+" the operation A+A must return A and by extension $\sum_i^n A = A$ or any integer n or more generally:

$$\sum_{i=1}^n Z_0^k = NZ_0^k = Z_0^k \qquad (5)$$

for any k, ≥ 2 N, and therefore the exact values of the coefficients m_j do not matter. Therefore, Eq. (4) is equivalent to:

$$Z_k = \sum_{l=1}^{k^2} Z_0^l \qquad (6)$$

If $n \le k^2$ then according to the powering of graph method the transitive closure must have been reached. Thus, the highest order of the iterations is given by k = ⌈$\ln_2$n⌉. Consequently, the matrix for transitive closure can be expressed as:

$$Z^* = \sum_{l=1}^{\lceil In_2 n \rceil} Z_0^l \qquad (7)$$

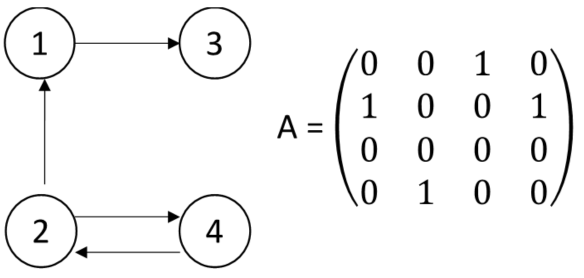

Fig.1: Directed graph with four directed edges and four vertices represented by the corresponding matrix A

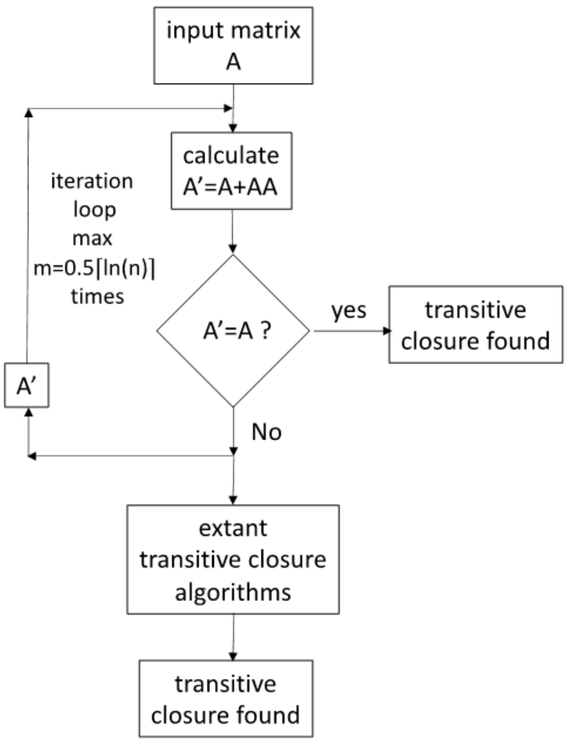

Fig. 2: A flowchart of how the proposed method can be used to determine the transitive closure in conjunction with extant algorithms

In some cases, for which the equation $n = 2^k$ holds for $n \,\varepsilon\, \mathbb{N}$, the rounding equation is substituted by the argument itself, that is $k = \lceil k \rceil$. Interestingly, the runtime number $\ln_2 n$ has been derived in an investigation of the transitive closure algorithms (Benedikt and Senellart, 2011), although in a slightly different context. In sum, the proposed recursive relation in Eq. (3) is equivalent to the powering of graph method but avoids the explicit calculation of all power matrices Ak for k = 1, n which results in greater computational efficiency. The equivalency between the recursive matrix relation in Eq. (3) and the powering of the graph method in Eq. (1) establishes a new path to establish the transitive closure. Although the algorithm is faster than the powering of graph method in Eq. (1), it is slower than the most recent matrix multiplication techniques (Alman and Williams, 2021; Duan *et al.*, 2002) and hence, it does not appear to be attractive at first glance. However, since the recursive matrix relation given in Eq. (3) is based on matrix multiplication, the proposed method can take advantage of the most advanced algorithms.

However, more significantly, the proposed method may be used to construct a simple criterion for determining whether a given adjacency matrix is transitively already closed or not.

New Transitive Closure Criterion

The recursive matrix operation in Eq. (3) can be used to determine whether a given matrix A fulfills Eq. (8):

$$A = A + A \cdot A \tag{8}$$

If Eq. (8) is satisfied, A must represent the matrix Z* of the transitive closure, i.e., A=Z*. Thus, a relatively simple operation ascertains the transitive closure condition. If condition Eq. (8) is not met, then clearly A can does not represent transitive closure. Therefore, the inequality Eq. (9) signals that the transitive closure is still to be determined:

$$A \neq A + A \cdot A \tag{9}$$

Thus Eq. (9) is equivalent to the inequality, $A \neq Z*$ However, in some cases, Eq. (9) might turn into Eq. (8) only after a few iterations. That is A' = A+ A. A or A'' = A' + A'. A' or higher order iterations may lead to Eq. (8) or $A^{s'} = A^{s'} + A^{s'}.\ A^{s'}$ for an integer s' significantly smaller that the dimension of the matrix A. In this case, the proposed method is still computationally more efficient than the explicit algorithms. This could turn very useful in cases, where there exists other information that the matrix may be very close (close-in terms of a few iterations of Eq. (9)) or in case of sparse matrices. Thus, depending on the degree of the transitive closure of the initial adjacency matrix, the above technique may effectively reduce the computing cost to O $(n^2 \ln_2 n)$ or much less in many actual applications where the transitive closure matrix is partially warranted. As a result, no application of a full-length algorithm is required and a lot of computational time is being saved.

As a result, an avoidable use of the Warshall O (n^3) brute force approach or of the more advanced O (n^s) methods with s<2.5 can be avoided:

$$A^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, A^3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}, A^4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$Z * A^1 + A^2 + A^3 + A^4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

We illustrate the proposed method in an example provided in Fig. 1, where a simple graph with four vertices and the corresponding adjacency matrix, A, is given. Above, the power matrices A^2, A^3, and A^4 have been calculated according to the powering of the graph method.

In the case of the Warshall algorithm one would have to calculate the following matrices A_1, A_2, A_3, and A_4 as given below:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

The last matrix is the sought-after solution, i.e., $A_4 = Z^*$. On the other hand, using the recursive matrix relation, proposed here and given in Eq. (3), one obtains Z_1 and Z_2 as:

$$Z_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, Z_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

where the second matrix Z_2 already represents the transitive closure, i.e., $Z_2 = Z^*$.

A not obvious but still simple case of an adjacency matrix V with five vertices that already represents the transitive closure is given below:

$$V = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Of course, using any extant transitive closure algorithm, the powering of graph method, or the proposed iterative matrix operation given in Eq. (3), it could be found readily that the matrix V represents already the transitive closure of the underlying graph. But at a considerable computational cost, even when using the current best algorithm boasting a bound of $O(n^{2.372})$ (Duan, 2022). Instead, by using the criterion given in Eq. (8), one could ascertain quickly that matrix V is already the transitive closure. The computational burden is only $O(n^2)$. Therefore, the proposed method should be combined with the best existing methods and algorithms to provide an efficient evaluation system for determining transitive closure. In Fig. 2, a flowchart is provided of how the proposed method can be used in conjunction with the extant algorithms. Of course, in evaluating Eq. (3) best matrix multiplication methods should be taken advantage of. The maximum number of iterations $m = 0.5\lceil \ln(n) \rceil$ is dictated by the circumstance that the numerical burden of using criterion in Eq. (8) should be considerably smaller than the numerical burden of using existing algorithms to determine the transitive closure. This restriction indicates that the proposed method is superior to the extant algorithms only for cases that are close to transitive closure.

Conclusion

A criterion has been derived which allows one, at a small computational expense, to determine whether an adjacency matrix at the hand of a directed graph represents the transitive closure. The criterion is based on a recursive matrix relation that by itself is another algorithm to determine the transitive closure. It is computationally faster than the powering of graph method but not as efficient as the extant algorithms if the calculation of the transitive closure is extensive. There are some cases, however, when the transitive closure can be found by a relatively small number of recursive loops compared to the number of vertices of the graph. In such cases, the proposed recursive matrix relation will be superior to the best-in-class algorithms.

If matrix A does not fulfill the criterion in Eq. (8), it is the goal of future research to determine how “near” (in terms of iterations) the given adjacency matrix is to the final matrix of transitive closure (Z*). It appears to be possible by utilizing the relation presented in Eq. (9), to derive a proximity criterion to estimate how many iterations of Eq. (3) are needed to find the ultimate transitive closure. The criterion of being close to the final solution may be established by monitoring the rate of change in the number of matrix element entries "1" from a few initial iterations. If this number of additional "1" converges quickly to zero, the use of recursive matrix relation could open a more efficient path to determine the transitive closure than the usage of full-fledged algorithms. If a given matrix is far from the transitive closure, then the criterion would indicate that a full-fledged algorithm must be applied to find the transitive closure and, in that case, criterion (8) is of no benefit. Such an additional proximity criterion is bound to be instrumental in increasing the efficiency of a software package that combines extant algorithms with the recursive matrix relation proposed here. The envisioned flowchart equipped with the proximity criterion may automatically decide which approach is more computationally efficient.

Acknowledgment

The author would like to acknowledge the reviewer's helpful comments, especially regarding in fashioning the flowchart in Fig. 2.

Funding Information

The author received no financial support for the research, authorship, and/or publication of this article.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Aho, A., Hopcroft, E., & Ullman, J. D. (2008). *The Design and Analysis of Computer Algorithms*, Addison-Wesley

Alman, J., & Williams, V. V. (2021). A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 522-539). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611976465.32

Benedikt, M., & Senellart, P. (2011). Databases. In *Computer Science* (pp. 169-229). Springer, New York, NY. https://doi.org/10.1007/978-1-4614-1168-0_10

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009) Introduction to Algorithms, 3rd ed., MIT Press, USA

Duan, R., Wu, H., & Zhou, R. (2022). Faster Matrix Multiplication via Asymmetric Hashing. *arXiv preprint arXiv:2210.10173*. https://doi.org/10.48550/arXiv.2210.10173

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, *5*(6), 345. https://doi.org/10.1145/367766.368166

Nuutila, E. (1998). Efficient transitive closure computation in large digraphs.

Even, S. (2011) *Graph Algorithms*, (2nd ed.), Cambridge University Press, pp. 46–48 ISBN-10: 1139504150

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the ACM (JACM)*, *9*(1), 11-12. https://doi.org/10.1145/321105.321107