Gradus Vol 13, No 1 (2026)
ISSN 2064-8014

ALGORITHMIC SEARCH OF DEBT CYCLES IN AN ECONOMIC ENVIRONMENT

Zsombor Baranyi [1], Marcell György Gyurászik [2] and Koppány István Körmöczi [3]*

[1] Department of Information Technology, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary, https://orcid.org/0009-0009-5016-4517

[2] Department of Information Technology, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary, https://orcid.org/0009-0003-3211-3341

[3] Department of Basic Sciences, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary, https://orcid.org/0009-0008-8326-3573

https://doi.org/10.47833/2026.1.CSC.007

Keywords:
Circular debt,
debt elimination,
economic efficiency,
algorithmic search,
directed graphs.

Article history:
Received 12 Dec 2025
Revised 02 Apr 2026
Accepted 10 Apr 2026

Abstract

This study's focus is on finding a proper algorithm that can identify cycles in an economic environment. Such environments can be represented as weighed directed graphs. In these graphs the vertices are the participants of the economy, and the arcs are the pending payments between them. If a graph includes a debt circle that circle can be eliminated or more precisely every arc's weight can be reduced with the cycle's smallest value arc. This is why we must find all these circles because removing them increases economic efficiency.

For testing the algorithm in our perspective, we had to make randomized testing environments. First, we declared a static case when the environment does not change overtime. And then a dynamic one which is changing like a real-world economic system. This dynamic case has parameters that can be modified depending on what kind of environment we want to create and how it will represent a real-world case.

1 Economic concepts

There are certain economic concepts that must be cleared up to perfectly understand our point of view. First, what do we take as a debt? A debt can be inserted into an economic system by creating an invoice that will later become a transaction between two participants. Different structures can be formed by debt. In this study we researched chain (or linear) debt and circular debt.

By circular debt we do not mean the occurrence when an economic participant tries to pay for its original debt with another debt. That would be spiral debt, and these terms are often confused with one another. But the major difference is that circular debt is not created because of a shortfall between cash inflows and outflows, it is created naturally, invisibly and unintentionally.

Circular debt is such an event, when several participants of the economy owe each other money, in a way that it forms a circle. When participants owe each other money in a sequence, we call that a chain debt. And circular debt is a special case of chain debt, where the starting and ending participant is the same.

2 The problem

It is undoubtable that cycles form in the economy. Given its size and the number of transactions between the participants, after a certain density circles start to form. These circles are harmful for

* Corresponding author. Email: baranyizsomborl2@gmail.com

the economy's wellbeing because the participants are unnecessarily waiting for their money. They can be eliminated or notedly each debt's value can be reduced with the cycle's smallest valued arc thus breaking the cycle. [3]

The problem of circular debt frequently occurs in construction. In this field the main problem is chain debt between the contractors, subcontractors and the commissioning company. Our final product would also give us a solution for this industry too, because it would reduce time and transaction fees [2]

3 Goals

Our goal is to create a system that can find and eliminate all circles in an economic system. We believe that it would greatly increase its efficiency and would make a huge impact. This system would be beneficial for all participants. Most of the time participants don't even realize that they are part of a circle. That is the source of the problem and the solution to that is transparency.

If a state would take the necessary steps to implement a system like this into the real-world economy, it could also acquire several other benefits. Given that there would be more transparency it would most certainly make the economy more legal. Because only lawfully working participants, companies could benefit from the circular debt cancellation system.

Ultimately if such a system gets implemented, the economy may become more efficient. This could cause a major development among the participants. Probably it would allow more liquidity for them that they can use to increase their productivity. But ultimately it would lessen the cycle time of the money. [A Framework for Examining and Improving the Accounts Payable Process in Construction]]

4 An analogy to physics – cyclic debt and friction

Through our research we have discovered that an analogy to physics easily explains the problem to everyone even without any practical knowledge in economics and its complex systems.

If we look at the economy as system of physics, we could say that cyclic debts are occurring like friction. They take away money (the force of the economic system) unnecessarily creating inefficiency.

Just like in physics when we reduce friction and the forces can be used more efficiently towards the task in question, here in economics it also applies that if we reduce the number of cyclic debts the economy could work more effectively and could improve more quickly.

5 The algorithm

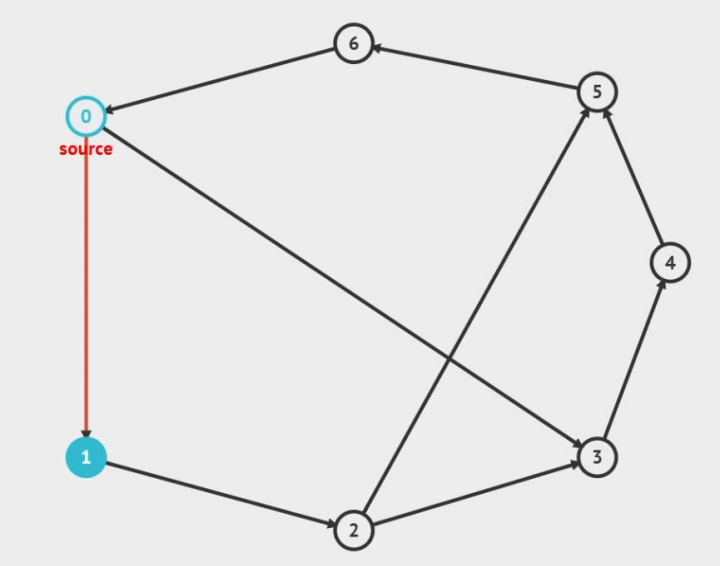
We used Johnson algorithm as a base for our program in which we get a KEY that is the starting vertex of the graph. In our case this could be any of the vertexes. Then we create an adjacency list from which the program will select the next vertex to visit. If in the adjacency list there is the target vertex which is the starting one, we save it as a cycle. Then without priority we move on to the next vertex which we add to the path stack. If a vertex is already in the stack, we copy the exact contents of the stack that forms the cycle, into a set called cycles and save it (although we rotate the cycle around before copying and compare with the already saved cycles so that we do not save a cycle twice).

If the program runs into a dead end, it will move a vertex back and declare the vertex as blocked so it will not visit that vertex again. When the graph that is accessible from the starting vertex is fully mapped out and there are vertexes that have not been reached, the program dumps the blocked list and sets the starting vertex to one of the not visited vertexes. Then it maps out the graph from that one as well. [1]

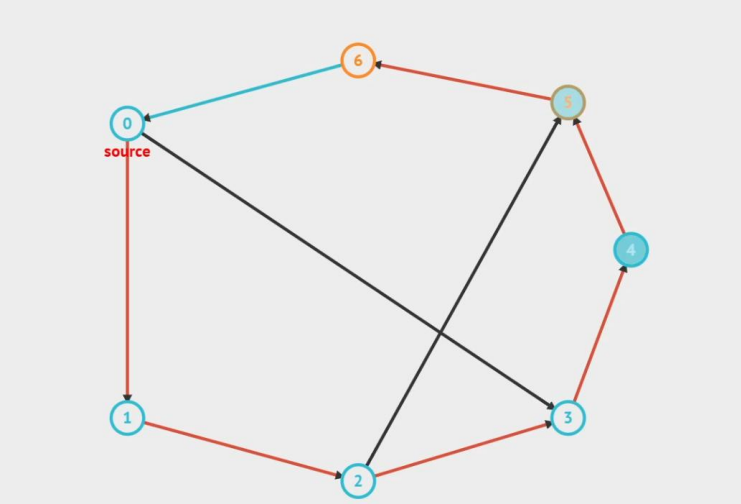
5.1 Limitations

We can say with certainty that the program finds all cycles. But because it's recursive it doesn't run in dense graphs. To fix this issue we applied a limitation to how many times the recursive part can rerun itself, therefore creating an iterative version of the program. This makes it possible to run in dense graphs at the cost of not being able to find the cycles greater than the limitation. In our case

it is not crucial that we don't get information on bigger cycles because we can still eliminate the smaller ones then run the program again. Eventually after enough attempts we will get an acyclic graph.



Step 1 when we start from the KEY (source) and move to the next vertex



Step 2 we arrive back at the KEY (vertex indicated with blue) after going through multiple vertexes and this is the moment when the program checks if it has ever found this circle if not it saves it for elimination.
Step 3 trace our steps back from the path stack and when a not checked edge is found the code checks that path. Once visited edges will be labeled blocked, so the code does not run into an infinite loop.
The basic data structure that the algorithm uses is an adjacency list. For the quick conversion from arcs to adjacency list the program uses an effective method which is described in detail this source bellow. [5]

5.2 Different cases and use cases

We have declared two models that our study will focus on. The Static and the Dynamic, as we have named them. The Static one is the more robust method. Its focus was to make an estimate of how many circular debts can occur in an economic system the size of a country. Unfortunately, we could not make this estimate regarding that there are not enough studies and valid data to create a realistic simulated system yet, but we are still working on it. Although failing its original purpose, the

Static method was very useful to test our modified algorithms so they would work like we intended them to. Dynamic is the method that will be the base of the real model. It simplifies the problem and makes it more optimizable.

5.3 Static

The Static model creates a fully randomized data set and uses Johnson algorithm to detect the circles in the graph. The data of the circles are then stored, and we would get notified of how many circles were found and would list them out in an ascending order. This simulates looking at an economic environment and then deciding to stop time and see how many cycles there are.

5.4 Dynamic

The Dynamic model, however, is based on the analogy that when you are looking at debts in a graph in real time, then you stop adding edges and wait for a period. That's why it is the case that is closer to representing a real-world scenario. What you will notice is that the graph becomes uncircular or even empty. That means we do not have to worry about already existing circles; we just must prevent the creation of new ones. So, in this state for testing purposes, we created uncircular graphs to test the limitations of the algorithm we have modified.

5.5 Elimination

Elimination is essentially the most important part of this algorithm. Because only identifying the circles does not provide any benefits to efficiency. We also must be careful with the method of elimination we choose. For example, it is not always the most efficient method to eliminate the cycles we have just found. The participating debts of this cycle could be part of another cycle which has not been found yet. And it might provide more efficiency if the second cycle were eliminated instead of the first one.

5.5.1 Circular debt elimination

To eliminate these virtual debt circles we need to clarify a few things. We want to eliminate the most amount of virtual debt in one elimination and we want to do it the fastest way. The method we are using is that we measure the length of the circle (what we mean by that is that we take how many participants are in that circle) and the maximum amount of debt value that is part of all debts in the circle. We multiply these factors, and we get the virtual debt amount that is contained by the circle. Then we select the circles that contain the most. Next, we check whether any of the selected circles has an edge that appears in more than one. If yes, we sort them out just like we did until we have no edges that are in multiple circles. Then we eliminate the selected circles. After this we will get a graph without circles.

5.5.2 Chain Debt elimination

Chain debt elimination works similarly. We measure the length of the chain and the maximum amount of debt value that is part of all debts in the circle, and then we take that amount out of every debt and we rearrange it to be between the first and last participant of the chain. This method is recursive because one edge will be removed and it creates two separate chains out of the one that we eliminated. This way we must continue eliminating chains until we are left with a graph that only contains one edge long (trivial) chains. This method might be problematic because of liability issues between participants where not all participants have the same ability to pay. This means we could only implement this system if we consider that a state does not have this problem. If a state is already in a chain, it will simply not matter to it who it pays to because it is the most reliable out of all the ones who can pay debt. This means we can eliminate chains but only the ones that begin with the state.

5.6 Optimizing

We have optimized the dynamic and static state to make them runnable on our devices. This contains limiting the modified DFS algorithm which means that we drew a line in given density what

the ongoing DFS cannot step over this is important because DFS is a recursive algorithm so by limiting how deep it can reach it will shorten the run time by a lot.

In this case we can only find circles that are maximum as long as the limitation, so we lose an amount of data by not detecting greater circles. But after elimination we can run the algorithm again and we can set the limitation higher as the density of the graph decreases overtime.

The other method we used for optimization is parallelization in the dynamic state and we used this on the inserted new edges. Each new edge gets a new parallel running DFS testing only on the edge that it was assigned to, because only the freshly inserted cycles are able to create new cycles, therefore we only need to analyze them. This way we eliminate the incoming debts faster if they create a circle.

6 Conclusion

Through our research we have concluded that making a system like this is completely feasible. The main goal of our study was to try and find methods of discovering and eliminating all circles in an economic environment. We successfully found ways to look for all circles and developed our own method of saving them without repetition. In this study our goal was also to point out this specific problem of circular debt and how it can be resolved.

7 PROSPECTS

This study that we have been working on in the past year has proven that such a system is possible to create. The fundamental methods of finding and eliminating circles in a graph like economic environment are very feasible in today's information technology. We plan to continue our research in this field, and we want to go deeper into the IT aspects. We want to create a realistic environment where we can showcase what effect it would have on the economy. Then we want to put everything that we have found into one working prototype of a circle elimination system. This could later be implemented into any economy that is interested in its benefits. And that is the final goal of our research. To provide a working system that could increase an economic environment’s efficiency and. transparency.

8 Acknowledgment

Project No. 2025-2.1.1-EKÖP-2025-00021 was implemented with support from the Ministry of Culture and Innovation's National Research, Development and Innovation Fund, financed by the 2025-2.1.1-EKÖP grant program.

We would like to thank our mentor Koppány István Körmöczi, who gave us the idea for our study and supported us throughout our research

9 References

[1] A. Gupta and T. Suzumura, “Finding All Bounded-Length Simple Cycles in a Directed Graph,” May 26, 2021, *arXiv*: arXiv:2105.10094. doi: 10.48550/arXiv.2105.10094.

[2] R. Al-Hussein, “A Framework for Examining and Improving the Accounts Payable Process in Construction,” University of Alberta Library, 2014. doi: 10.7939/R3TT4G218.

[3] Z. Hong, S. Guo, R. Zhang, P. Li, Y. Zhan, and W. Chen, “Cycle: Sustainable Off-Chain Payment Channel Network with Asynchronous Rebalancing,” in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2022, pp. 41–53. doi: 10.1109/DSN53405.2022.00017.

[4] “Dynamic cycle detection - ScienceDirect.” Accessed: Mar. 24, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020019083900388

[5] S. Arifuzzaman and M. Khan, “Fast Parallel Conversion of Edge List to Adjacency List for Large-Scale Graphs”.

[6] J. LeRoy, “A SEARC H ALGORITH M FOR FINDIN G T H E SIMPL E CYCLE S O F A DIRECTE D GRAP H.” [Online]. Available: https://unbscholar.dspace.lib.unb.ca/server/api/core/bitstreams/40439ff2-d05a-4392-b470-9ef50382374f/content