A System For Detecting Cascading Fuzzy Cycles

James P. Buckley, Jennifer M. Seitzer, Nagini Addagatlan, Gandhi Babu Chilaka
Computer Science Department
University of Dayton
300 College Park
Dayton, Ohio 45469-2160

Abstract

This paper defines the concept of a cycle and how it is constructed using mined rules. Cascading cycles, those that have a common vertex, are defined and discussed. These concepts are formulated into a fuzzy cycle paradigm. A system that uses RDF files as input and detects cascading cycles was developed and tested.

Keywords: data mining, cycle mining, machine learning, fuzzy cycles.

1. Introduction

This paper initially defines what traditional data mining is and the concepts of cycles, cycle mining, cascaded and cycles. A system implementation is then described and defined.

2 Traditional Data Mining and Cycles

With the present state of technology, we have the capability to store extremely large amounts of data in organized and automated systems. The preponderance of data warehouses and datamarts [3], [5] are concrete evidence that this is not only possible, but of great interest to researchers, government agencies, and large corporations. But what is the meaning and usefulness of these large repositories of data?

The classical definition tells us that "Data mining is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [4]". We are no longer looking for tabular answers or aggregations of the data; rather we are looking for *patterns* within the data that reveal knowledge previously unknown. One of the most common applications of data mining is to generate all significant association rules between items in a data set. We can employ an efficient algorithm to mine a large collection of basket data type transactions for association rules between sets of items with some minimum specified confidence [1], thus providing both meaning and usefulness to the data.

The patterns we discover in our data sets through data mining may not be necessarily isolated. There may be chains of rules forming patterns of patterns, or *meta-patterns*, where the head of one rule is the body of another rule. In particular, the chain of rules may form a *cycle*. This form of meta-pattern is the focus of our work in this paper.

Traditional data mining algorithms identify associations in data that are not explicit. Cycle mining algorithms identify *meta-patterns* of these associations depicting inferences forming chains of positive and negative rule dependencies. We define some cycles as *cascading cycles* because they are not independent, and directly affect the behavior of the overarching cycle. By modifying one participating rule of a cascading cycle, we can enable or disable it. This in turn, for this particular application, will have an effect on the overall cycle. In other words, the effect is *cascaded* to the primary chain. In particular, we identify cascaded cycles as those having one or more vertices that are shared by the overarching cycle. Thus, changes made to the ancillary cycle will "cascade" down to the main cycle.

In some cases there are cycles that have not yet manifested themselves. For instance, there may be a cascaded cycle that has one vertex with too low a value to make the cycle whole. If the cycle is incomplete, a formal paradigm for cycle +mining these partial cycles using fuzzy techniques is defined. In order to differentiate between those cycles that we want to perpetuate and those that we want to break, this research will use the α-*cycle* and β-*cycle* as the underlying formalism of the paradigm. An α-*cycle* is a cycle that is good and should be perpetuated. A β-*cycle* is a cycle that has negative consequences and should be broken. Specifically, α-*cycles*, desirable cycles, should be reinforced such that complete positive cycles are created, and β-*cycles* can be weakened to keep a negative incomplete cycle from forming.

3. Cascading Cycles

Of primary interest in this paper are both complete and partial cascading cycles that are discovered in our data. Intuitively, two cycles are *cascading* if they share at least one vertex.

Definition 3.1 (cascading cycles)
Let cycle X= $<x_1, ..., x_k, x_1>$ *and let cycle Y =*$<y_1, ..., y_l, y_1>$. *Cycles X and Y are* cascading *if* $x_i = y_j$ *for some vertex* x_i *in X and some vertex* y_j *in Y.*

Example 3.1 (cascading cycles)
Let X = <a1, b1, c1,d1, a1> and Y = < h1, i1, c1, h1>. Y is a *cascading* cycle of X because it has a node in common with X (c1). (Likewise, X is a cascading cycle of Y.) Figure 1 illustrates this situation.

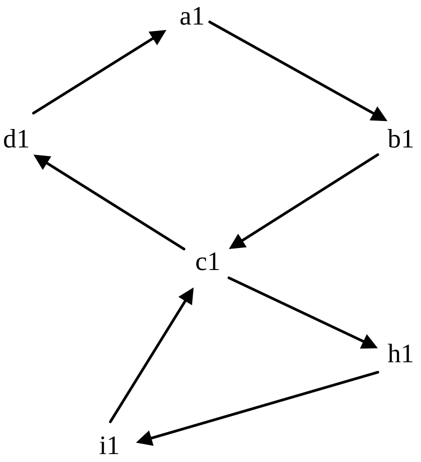


Figure 1 Cascading Cycle

Imagine for a moment that we have another cycle W that is a cascading cycle of Y. If we were to break cycle W, then cycle Y would be broken or at least weakened thereby also breaking or weakening cycle X. If we were to strengthen cycle W, this would have a similar effect on cycles Y and X. Therefore, changes to cascading cycles will have a direct and linear impact on the associated cycles "downstream".

4. Fuzzy Cycle Paradigm

In previous work [2], the authors presented a methodology that uses the individual rule supports and confidences to detect and categorize different types of cycles, as well as presenting an enhanced cycle detection algorithm that uses a metric τ computed from constituent rule support and confidence factors. This metric is used to characterize the strength of the encompassing cycle. The definition of support and confidence was used and presented in [7].

We defineτ, the average of the above two metrics, to be our threshold measurement for any specific rule. No rule with τ less than a user-specified threshold *U* will be considered meaningful enough to be placed in the system knowledge base. Hence, it will not be detected as part of any cycle.

Definition 4.1 (τ)

τ = (support + confidence) / 2

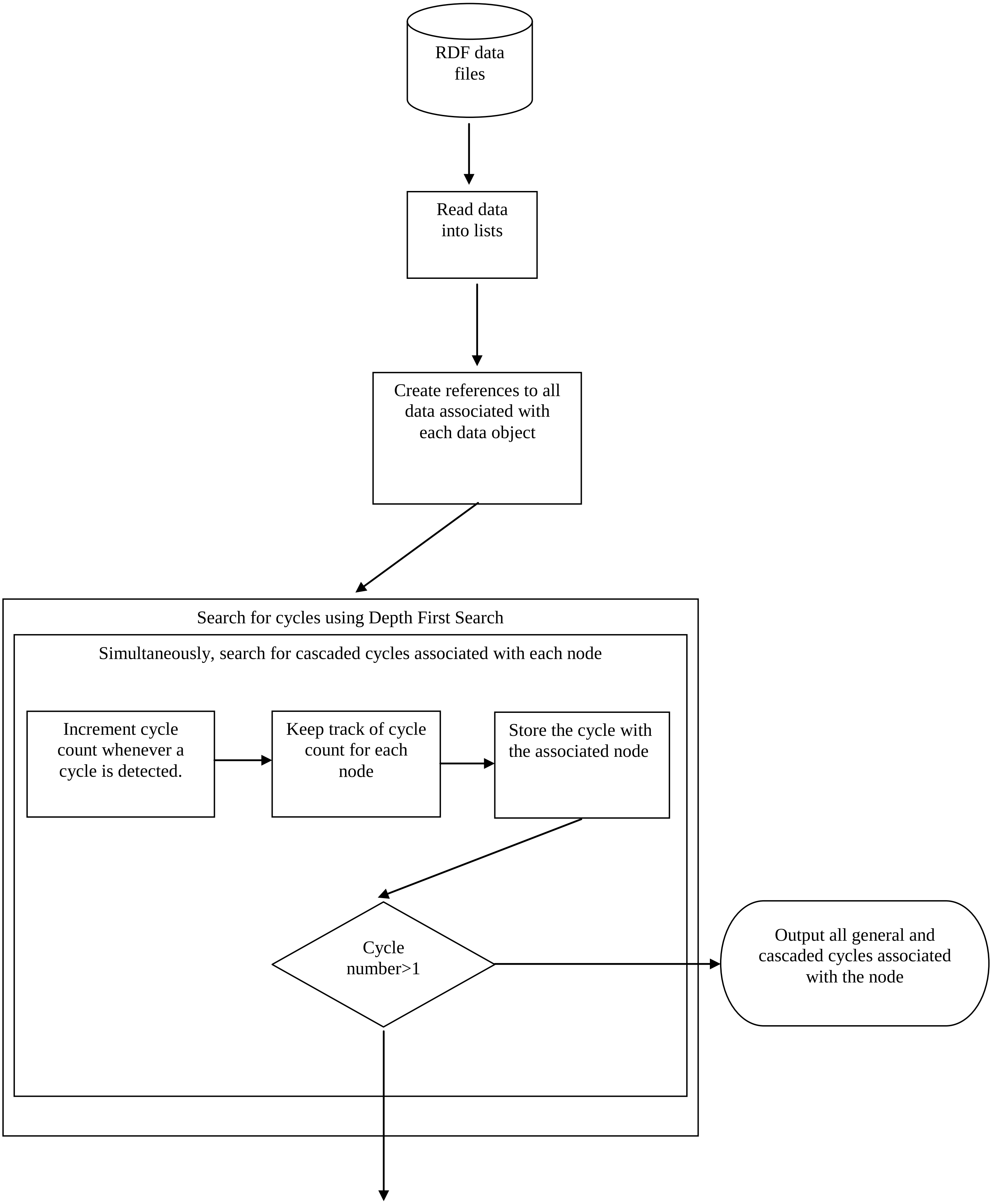
Cycles are composed of n individual rules, so we define the strength metric τ applied to cycles as

$T = \min(\tau_1, \ldots, \tau_n)$, where τ_i is the strength measurement of rule *i*.

The revised methodology and cycle detection algorithm do not consider rules with τ less than user-specified threshold *U*. Thus, any detected cycle has *T* at least U.

5. Implementation and Testing

A system was previously developed in [8] that mines the complete graph for cycles. This system has been modified so that it uses RDF files as input. Additionally, the system now detects and outputs all cascaded cycles. The diagram below describes the system components.


RDF data files
Read data into lists
Create references to all data associated with each data object
Search for cycles using Depth First Search
Simultaneously, search for cascaded cycles associated with each node
Increment cycle count whenever a cycle is detected.
Keep track of cycle count for each node
Store the cycle with the associated node
Cycle number>1
Output all general and cascaded cycles associated with the node

The steps for finding cascaded cycles are as follows:

- FOAF files are parsed and read into a list.
- The data in the list is then outputted into a text file.
- The data from this file is read into 3 different lists of vertices, outgoing and incoming edges.
- Then the search algorithm is implemented to find the cycles and from them, the cascaded cycles.

To find cascaded cycles in datasets, first it is required to keep track of all the cycles the nodes is associated with. If a node is associated with more than one cycle, then it forms a cascaded cycle.

The datasets we are considering are RDF data files. Initially, the data is stored in FOAF files. Then it is parsed and stored in a list. The data in the list is then outputted to text files from where it can be used by other methods as input. This is necessary because the data is not in RDF format anymore and is easy to read.

To detect the cycles, the data from the input text file should be read into a list of type GraphNode which is a user-defined class. Also, lists of incoming and outgoing nodes for each node are also maintained. The algorithm used in this process is Depth First Search.

In Dfs, a stack of type GraphNode is maintained to check which nodes are visited and which nodes are not. Initially, for each node a default node number (999) and cycle number (0) are set. The parent of the first node in the list is set to null and the node is pushed into the stack.

While the stack is not empty, the top node in the stack is removed and its node number is changed from 999 to an appropriate node number. For each such node, all the outgoing edges are checked to see if they are already visited. If they are not visited, their parent is set to the current node. If they are already visited (node number not equal to 999), a reverse search is started to see if it forms a cycle.

Check the parent of current node and its value. If they are not null and unequal respectively, move backwards to the parent of the current node and check for the same results. If in this reverse process a node has a parent with null value, then no cycle is formed. But, if the node value equals outgoing node value, a cycle exists. This cycle is now added to the cycles list of all those nodes that form the cycle and their cycle count is increased. This process is repeated until the stack is empty.

After all nodes are visited, all the nodes that have cycle number greater than one are said to have cascaded cycles.

5. Conclusion

In this paper, we defined the concept of a cycle and how it differs from and expands upon traditional data mining. A cascaded cycle is then defined with an example given. We define a metric,τ, that is used to define the strength of individual rules in a cycle. We also define a metric, T, that represents the strength of the entire cycle. An implementation of a system that detects cycles as well as cascaded cycles is described.

6. References

[1] Agrawal, R., Imielinski, T., and Swami, A., “Mining association rules between sets of items in large databases,” *SIGMOD Bulletin*, May 1993, pp. 207-216.

[2] Buckley, J. P., Seitzer, J., M., ,A Paradigm for Detecting Cycles in Large Data Sets via Fuzzy Mining, *Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop*, Chicago, Illinois, November 8, 1999. Pages 49-55.

[3] Buckley, J. P., Seitzer, J., M., ,A "A Framework for Detecting Vulnerable, Cascaded Fuzzy Cycles in the Carbon Chain", *Proceedings of the 2011 International Conference on Data Mining (DMIN11). July 2011*

[4] Chaudhuri, S. and Dayal, U., “An overview of data warehousing and OLAP technology,” *SIGMOD Record*, Vol.26, Num. 1, 1997, pp. 65-74.

[5] Frawley and Piatetsky-Shapiro, editors, *Knowledge Discovery in Databases*, chapter Knowledge Discovery in Databases: An Overview, AAAI Press/The MIT Press. 1991.

[6] Kimball, R., *The Data Warehouse Toolkit*, John Wiley and Sons, 1996.

[7] Seitzer, J., Buckley, J. P., Monge, A., “Meta-Pattern Extraction: Mining Cycles”, *Proceedings of the Florida Artificial Intelligence Research Society International Conference (FLAIRS-99)*, Orlando, FL; pp. 466-470.

[8] Seitzer, J., Buckley, J. P., Chilaka, G. P.; “Mining the Implicit Complete Graph of Knowledge Bases”; Proceedings of the 21st Midwest Artificial Intelligence and Cognitive Science Conference (MAICS'2010); Indiana University South Bend, Schaumburg, Illinois. *April 17-18, 2010. Pages 84-89.*

[9] Zadeh, I. A. "Fuzzy Sets," *Information And Control*, Vol 8, 1965, pp. 338-353.