International Journal of Multimedia and Ubiquitous Engineering
Vol.12, No.1 (2017), pp.191-202
http://dx.doi.org/10.14257/ijmue.2017.12.1.16

Study on a Multi-Hierarchy Topological Sort Algorithm for Automatic Calculation

Zhengguang Li[1,2], Huimin Zhao[1,2,3*], Wu Deng[1,3,4,5], Junwei Li[1], Botao Li[1] and Hao Tian[1]

[1]Software Institute, Dalian Jiaotong University, Dalian 116028 China
[2]Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 530006 China
[3]Key Laboratory of Guangxi High Schools for Complex System & Computational Intelligence, Guangxi University for Nationalities, Nanning 530006, China
[4]The State Key Laboratory of Mechanical Transmissions, Chongqing University, Chongqing 400044 China
[5]Traction Power State Key Laboratory of Southwest Jiaotong University, Chengdu 610031 China
**Corresponding author: Huimin Zhao, E-mail:hm_zhao1977@126.com*

Abstract

Topological sort algorithm (Toposort) is widely used for practical problems. However, standard toposort don't solve the automatic calculation of formula problem (ACFP) directly. To gain the feasible topological order of ACFP, this paper puts forward a modified algorithm based toposort ,which models for ACFP using directed acyclic graph (DAG) firstly, and then puts the 0 in-degree formula nodes into a level list in execution process, lastly decreases the in-degree of other formula nodes ,whose parameters is correlate to the 0 in-degree nodes and in-degree is not equal to 0.Application of the modified algorithm in ACFP with 183 formulas, our system achieves a feasible topological order and uses the order and reflection mechanism to gain the final correct result. Compared to the customer's manual calculation, our algorithm spends a little time but achieves the same results.

Keywords: *Topological sort algorithm; Automatic calculation; Formula; DAG; Reflection mechanism*

1. Introduction

Many practical tasks such as reporting system, engineering calculation, includes lots of formulas and these formulas depend on each other. If the calculation sequence is not correct, huge loss will be caused. However, the calculation sequence is mainly depended on manual generation, which spends the massive time and manpower. Some scholars have carried out a lot of research and put forward the corresponding solution to the dependency relationships among formulas and the performance optimization.

In [1], the sequence of calculation formulas in reporting system was gained by directed graph algorithm. Juta Pichitlamken presented a high performance spreadsheet simulation system which added power of parallel computing on Windows-based desktop grid into popular Excel models by using standard Web Services and Service-Oriented Architecture (SOA), and the proposed system obtained more than 7 times speedup for some test application on a 8-PC system [2]. MA *et al.* [3] gave the solution for the directed cycle graph. Semi-supervised quadratic partitional cluster algorithm ,which created directed maximum acyclic graph and gained the better calculation order by cooperative computation, was proposed in [4-5].

Topological sort algorithm is mature and widely used graph algorithm, the research mainly focuses on two aspects, namely, the improvement of efficiency and the application for different problems. Er [6] proposed a new topological sorting algorithm using the parallel computation approach. David [7] presented a new algorithm for the problem of maintaining the topological order of a DAG [8] in the presence of edge insertions and deletions. In [9], Deepak *et al.* [10] presented a simple algorithm, which maintained the topological order of a DAG under an online edge insertion sequence in O(n2.75).For dense DAGs, this was an improvement over the previous best result. An I/O-efficient algorithm for topologically sorting directed acyclic graphs was proposed in [11] and the algorithm was extremely inefficient and performs O(n• sort(m)) I/Os in the worst case but achieved good performance in practice. David [12] improved the previous algorithm in [7], by only recomputing those region(s) of order affected by the inserted edges to maintain the topological order. Katriel and Bodlaender studied online algorithms to maintain a topological ordering of a directed acyclic graph, it is optimal that the algorithm was implemented to run in O(n log n) time on trees [13]. Haeupler *et al.* [14] proposed the two algorithms for maintaining topological order, detecting the cycle when an arc was added and maintaining the strong component by extending the new algorithms. Liu [15] found a sorting algorithm, which was different from Kahn or DFS algorithm and owned low complexity. Pang [16] studied the topological sorts of two directed acyclic graphs (DAGs) and the associated properties, and the results showed these problems are solvable either in linear time cost or in the same time cost as to compute the transitive closure.

In [17], Abo-Tabl proposed new definitions of lower and upper multiset approximations, basic concepts of the rough multiset theory, using concept of multiset topology and ambiguity. Reininghaus *et al.* [18] designed a stable multi-scale kernel using topological machine learning, and the proposed approach gained considerable performance in 3D shape classification/retrieval and texture recognition. Barnat *et al.* [19] presented a new technique that guaranteed correct construction of the reduced state space graph w.r.t. the cycle proviso using topological sort proviso. The proposed technique has been implemented within the parallel and distributed-memory LTL model checker DIVINE and indicated the similar performance, comparing to the traditional approaches. High efficiency algorithm for vertex with a level (denoted by LAOV) was researched and the novel algorithm for LAOV was proposed [20].

For automatic calculation of our formulas, the algorithms mentioned above, do gain the feasible toplogical order, however, they don't hierarchize the formulas in order to calculate them in different cycles. So we add the level list to store the topological order which includes all the current nodes, calculated in the current cycle. Meanwhile, for decreasing the cycle times, we add a flag to distinguish between creating a new list and using the last created list. By testing the customer's practical dataset with 183 formulas, our algorithm gain the feasible topological order and the last result. Compared to the result provided by the customer, we achieve the same results but spend a little time.

2. Problem Description and Modeling

Our customer provides a list of formulas, and uses the all formulas to calculate about 2000 lines data. Some of formulas depend on other formulas' calculation results. Meanwhile, some of formulas also need to adjust its parameters because of the calculation result inaccuracy. So we must be very careful with the order of their execution, which is generated automatically. If the relationship between these formulas are simple enough we could represent them as a linked list or trees, which would be great and we will know the exact order of their execution. The problem is that sometimes the relations between the different formulas are more complex and some formulas depend on two or more other formulas even other lines' results such as previous, next line.

Thus we can't model this problem using linked lists or trees. The only rational solution is to model the problem using a directed acyclic graph (DAG) .Our DAG is defined as following rules.

(1) each formula ,needing to be calculated ,is viewed as a vertex.

(2) if formula A depends on calculation result of formula B, we add an arc, B ->A.

(3) if formula A depends on next line calculation result of formula B, we add a flag, called next flag, to B ,denoting formula B and formula A can't be calculated in same level cycle.

For example, we have formulas: A, B, C, D, E, F, G, each formula's parameter lists as Table 1.According to the Table 1 and rules mentioned above, the modeled DAG shows in Figure 1.

Table 1. One Sample of Formulas

Formula	Parmeters	Formula	Parmeters
A	B,B|NEXT,C	E	F,G
B	C	F	-
C	D,E	G	-
D	F,G		

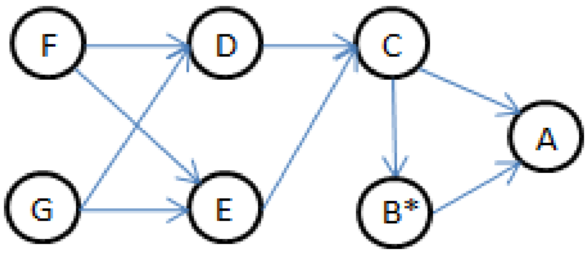


Figure 1. Modeled DAG of Sample

*representing node with next flag.

Our target is search the order of the DAG created by the above rules. Especially, the formula with next flag, must be calculated before the other formula and not be calculated in same level cycle.

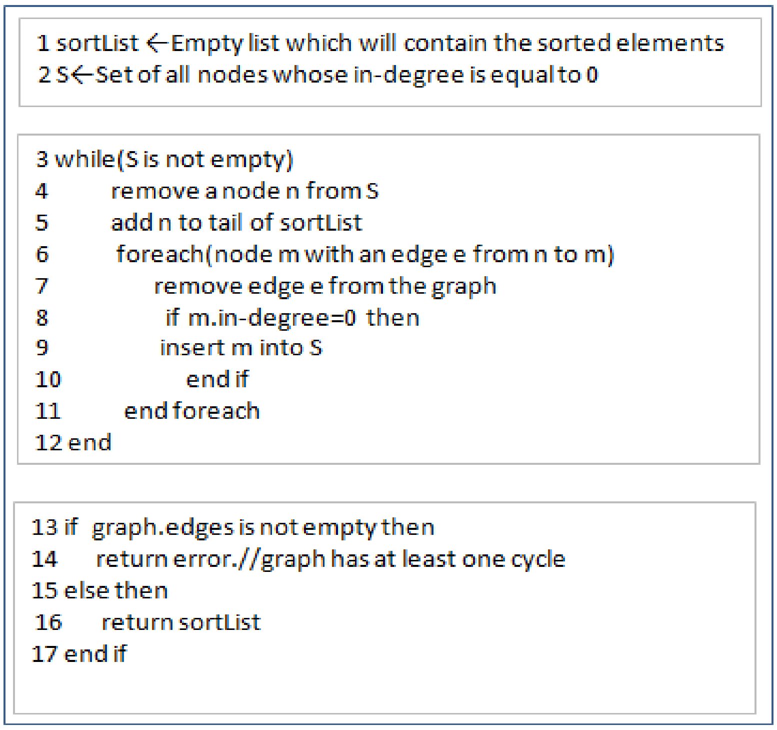
3. Standard Topological Sort Algorithm

The Topological sort algorithm (Toposort [10]) first described by Kahn (1962) [21], works by choosing vertices in the same order as the eventual topological sort. First, find a list of "start nodes" whose in-degree equal 0 and insert them into a set S; at least one such node must exist in a non-empty acyclic graph. Then, remove the edges whose start node in S but end node not in S from the graph. If the graph is a DAG, a solution will be contained in the list L (the solution is not necessarily unique). Otherwise, the graph must have at least one cycle and therefore a topological sorting is impossible. The procedure is showed in the pseudo code in Figure 2.

Based mentioned above algorithm, the example of Table 1,the toposort order may be F, G, D, E, C, B, A.

4. Multi-Hierarchy Topological Sort Algorithm

Directly using standard topological sort algorithm, we can't gain the solution to ACFP for the next node in the DAG created by rules for practical calculation formulas. The constraint condition requires the topological order is not only order but also hierarchy calculation sequence. However there is not the hierarchy calculation mechanism in the standard toposort, and so we have to modify the standard topsort from following two aspects.



```
sortList ←Empty list which will contain the sorted elements
S←Set of all nodes whose in-degree is equal to 0

while(S is not empty)
    remove a node n from S
    add n to tail of sortList
    foreach(node m with an edge e from n to m)
        remove edge e from the graph
        if m.in-degree=0 then
            insert m into S
        end if
    end foreach
end

if graph.edges is not empty then
    return error.//graph has at least one cycle
else then
    return sortList
end if
```

Figure 2. Pseudo Code of the Algorithm Described by Kahn

(1) adds a list to store each level topological order and the ordering nodes of each level also be stored another list.

For gaining the hierarchy calculation order and the order in every level, a level list, whose elements are also list, needs to be created. The list represents the hierarchy information and each element in the list is used to store the order list of current level.

(2) for enhancing the calculation efficiency, needs to add a flag to distinguish between creating a new list and using the last created list.

Each element of the level list all needs to be calculated in one loop. The number of elements in the level list affects the number of loop which determines the execution efficiency. So We use the next flag to reduce the number of loop iterations .

The procedure of Modified algorithm is indicated in the pseudo code in Figure 3.

According to modified algorithm, the execution processes of sample Table 1 are listed as follows.

Initialization:Formula_list={A:2,B*:1,C:2,D:2,E:2,F:0,G:0},S={},LevelList={}.A:num representing formula A node's in-degree is equal to num. For example,A:2 means node A's in-degree is 2.

Step1:next_flag=true,LevelList[0]={F,G},S={F,G},
after removing nodes in S, Formula_list= {A:2,B*:1,C:2,D:0,E:0};
Step2:next_flag=false, LevelList[0]={F,G,D,E}, S={D,E}
after removing nodes in S, Formula_list= {A:2,B*:1,C:0 };

Step3:next_flag=false, LevelList[0]={F,G,D,E,C}, S={C}
after removing nodes in S, Formula_list= {A:1,B*:0};
Step4:next_flag=true, LevelList[1]={B* }, S={B*}
after removing nodes in S,Formula_list={A:0 };
Step5:next_flag=false, LevelList[1]={B*,A },S={A}
after removing nodes in S,Formula_list={};
So, the sample Table 1 will be splited 2 levels, one level is {F,G,D,E,C} and another is {B*,A}.

```
LevelList←a list which will contain all the sorted and hirachical
elements for storing list
Formula_list←a list which contains all the formulas,waiting to
be sorted and hierarchized.
S←a Map for storing topological ordering of current leve .
```

```
next_flag = true
while(Formula_list is not empty)
        if next_flag = true then
           LevelList.add(new List())
        end if
        next_flag = false
        foreach(m in Formula_list )
           if m.in-degree =0 then
              LevelList[LevelList.count-1].add(m)
              S.put(m.name,m)
              if m.next_flag = true then
                 next_flag = true
              end if
              remove m from Formula_list
           end if
        end foreach
        foreach(F in Formula_list )
            foreach (p in F.parameters)
              if (S.contains(p)) then
                 F.in-degree--
              end if
            end foreach
        end foreach
        S.clear()
end
```

Figure 3. Pseudo Code of the Modified Algorithm

5. Results and Discussion

5.1. Dataset

We use dataset with 183 formulas provided by our customer to test and verify validation of our algorithm. The parameters and dependencies among part formulas are showed in Table 2.To facilitate our calculation and representation, we ignore the parameter or formula which is the formula itself (for example, parameter NXT|ST5_Q is the formula ST5_Q), and the known parameters such as "ranges", although they are listed in Table 2. Converting these data into network graph with 183 vertexes and more than 700 edges, showed in Figure 4, we analyze the distribution of the degree in the graph using graph package [22], the Figure 5 shows that the node degree mainly concentrated on 4-10,namely, lots of formulas depend on 4-10 other formulas.

Table 2. Part Practical Data Provided by Customer

Formula	Parameters	Formula	Parameters	Formula	Parameters	Formula	Parameters
ST3_AB	ST3_AE,ST3_AM,para,ST3_AA	ST5_B	ST3_P	ST5_BE	ST5_Z,ST5_B,NXT|ST5_B,ST5_AM,ST5_AL,ST5_AH,ST5_AO,ST5_S,ST5_Y,ST5_AE,ST5_N,PRE|ST5_N,PRE|PRE|ST5_N,NXT|ST5_N,NXT|NXT|ST5_N	ST5_BQ	PRE|ST5_Z, ST5_BM, ST5_BE, PRE|ST5_AB, NXT|ST5_AB, PRE|ST5_N, NXT|ST5_N, ST5_N
ST3_AC	ST5_F,ST3_Q, ST5_E,ranges	ST5_C	ST3_Q	ST5_BF	ST5_Z,ST5_AL,ST5_AY,ST5_AH,PRE|ST5_N,NXT|ST5_N,ST5_N,ST5_I,ST5_K,ST5_L,ST5_AV	ST5_BR	ST5_BN,ST5_BF,ST5_AB,ST5_N,ST5_K
ST3_AD	ST3_W,ST3_S	ST5_D	ST3_Y	ST5_BG	ST5_Z,ST5_AL,ST5_AZ,ST5_AH,PRE|ST5_AH,ST5_AW, ST5_N,ST5_P,ST5_V,ST5_T, ST5_AE,ST5_R	ST5_BS	ST5_BN,PRE|ST5_BN,ST5_AB,PRE|ST5_AB,PRE|ST5_N,ST5_N,ST5_K,ST5_J,ST5_AA,para,ST5_B, ST5_BM,ST5_BJ,ST5_BR,PRE|ST5_BI, PRE|ST5_BQ
ST3_AE	ST3_S,ST3_W,ST3_T	ST5_E	ST5_N, ST5_AM, ST5_M, ST5_I, ST5_K, ST5_J	ST5_BH	ST5_Z,NXT|ST5_AL,ST5_BA,ST5_AH,NXT|ST5_AH,ST5_AX,ST5_Q,ST5_N,ST5_S, ST5_W,ST5_U,ST5_AF	ST5_BT	ST5_AA,ST5_N,NXT|ST5_N,para,ST5_AB,NXT|ST5_B,ST5_K,ST5_J,ST5_BM,NXT|ST5_BM,ST5_BN,NXT|ST5_BN,NXT|ST5_BQ,NXT|ST5_BI,ST5_BR,ST5_BQ,ST5_BJ
ST3_AF	ST3_X,ST3_S	ST5_F	PRE|ST5_F, ST5_B	ST5_BI	ST5_B,ST5_AM,ST5_BE,ST5_AL,NXT|ST5_B,NXT|ST5_N,PRE|ST5_N,ST5_N	ST5_BU	ST5_BS,ST5_BT

Note: if a parameter don't be included in column "Formula", it means the parameter is known. PRE means using previous line data and NXT represents using next line data.

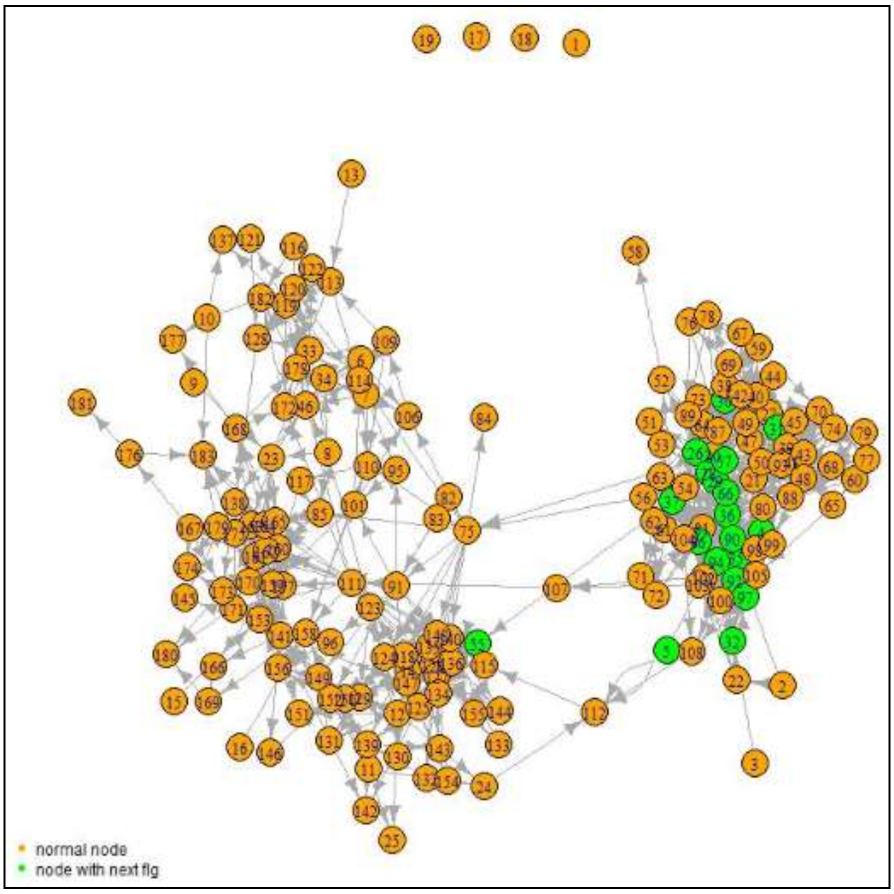


Figure 4. Network Graph of Dataset

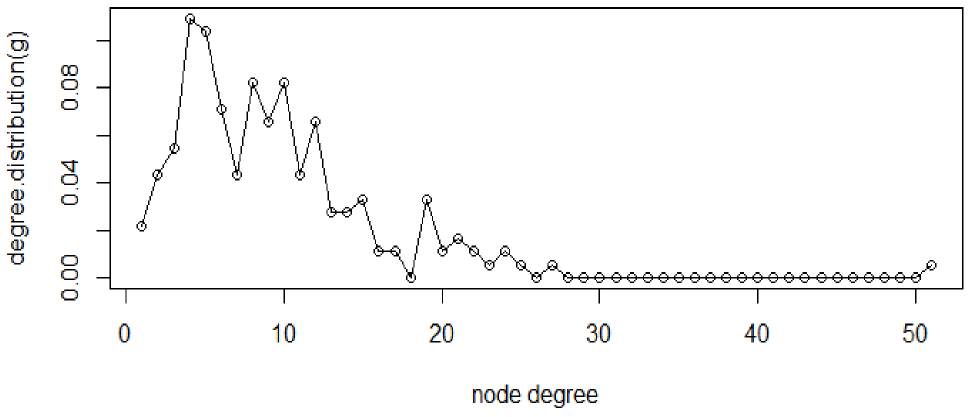


Figure 5. Distribution of Node Degree

5.2 Initialization In-degree

For above dataset, each formula's in-degree is calculated before using the modified algorithm. The procedure of initialization in-degree is illustrated as follows.

Step1: scan any one parameter in the formula.

Step2: if the parameter is equal to the formula or not included in formula list (known parameter), ignore it; else, the in-degree of the formula increase by 1.

Step3: repeat step1-2 to scan over all parameters of the formula.

5.3 Role of Next Flag

For analyzing the role of next flag, we run the algorithm without flag and with flag respectively. The former splits the formulas into 28 lists, including different formulas. However, the formulas are only divided into 10 lists when executing the latter. The Figure 6 and Figure 7 show the comparison results of the algorithm using or not using next flag. The results in Figure 6 shows that the maximum node number is about 80 in the 10th level gained by the latter algorithm. Meanwhile, this level merges 80% nodes, that is, from the 10th to 28th divided by the former ,showed by the Figure 7. The Figure 7 also shows that the second and 7th level merge the 60% and 20% nodes, respectively.

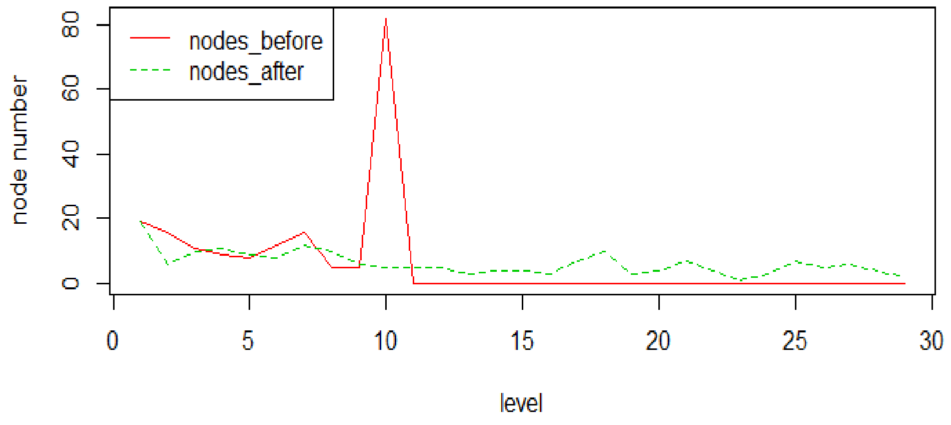


Figure 6. Node Number in Different Level, Gained by Algorithm Using and not Using Next Flag

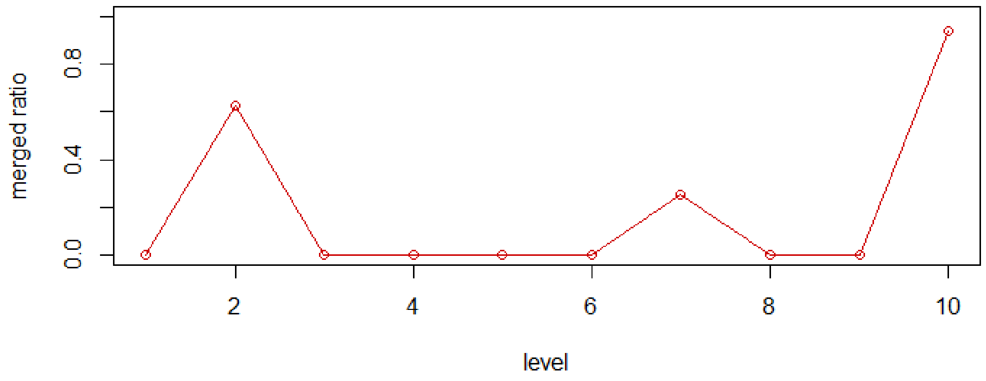


Figure 7. Merge Ratio of the Algorithm Using Next Flag

5.4. Execution Result and Comparison

By using the modified algorithm, we gain a feasible topological order and topological levels, showed in Table 3.The Figure 8 shows the results of data visualization for better understanding the final data. The results satisfying not only the topological order but also topological levels such as ST5_B. The formula, ST5_B, as parameter with next flag, emerges in ST5_BE, ST5_BI and ST5_BT. The level of ST5_B is 1 but their levels are 4, 7, 10 respectively. For gaining the last result expected by our customer, we adopt the reflection mechanism [23] to accomplish the last task and write the last result to the text file named “cfile.txt”. Meanwhile, we change the calculation result, provided by the customer, to the text file named “pfile.txt”. Comparing the differences between the two files using “beyond compare”, a tool for comparing files, results show there are no differences. Our algorithm gains the expected result but spends a little time.

Table 3. Toposort Order of Customer’s Dataset

Level	Formulas
1	ST5_A,ST5_C,ST5_D,ST5_B,ST3_BF,ST3_AD,ST3_AE,ST3_AF,ST3_AG,ST3_AJ,ST3_AO,ST3_AP,ST3_AR,ST3_AS,ST3_CL,ST3_DC,ST3_DP,ST3_DQ,ST3_ED
2	ST5_F,ST5_O,ST5_AA,ST3_AH,ST3_AQ,ST3_EC,ST5_I,ST5_H,ST5_J,ST5_K,ST5_M,ST5_G,ST5_AB,ST3_AI,ST3_AV,ST5_Z
3	ST5_N,ST5_L,ST5_R,ST5_S,ST5_T,ST5_U,ST5_V,ST5_W,ST5_X,ST5_Y,ST3_AU
4	ST5_P,ST5_Q,ST5_AE,ST5_AF,ST5_AG,ST5_AI,ST5_AK,ST5_AM,ST3_BE
5	ST5_E,ST5_AL,ST5_AJ,ST5_AN,ST5_AO,ST5_AP,ST5_AS,ST5_BB
6	ST5_AC,ST5_AD,ST5_AH,ST5_AQ,ST5_AR,ST5_AT,ST5_AU,ST5_AV,ST5_AY,ST5_BC,ST5_BD,ST3_AC
7	ST5_AW,ST5_AX,ST5_AZ,ST5_BA,ST5_BE,ST5_BF,ST3_AK,ST3_AL,ST3_AT,ST3_CR,ST5_BJ,ST5_BG,ST5_BH,ST5_BK,ST5_BI,ST3_AM
8	ST5_BM,ST5_BL,ST5_BN,ST3_AB,ST3_CE
9	ST5_BQ,ST5_BO,ST5_BP,ST5_BR,ST3_AN
10	ST5_BS,ST5_BT,ST5_BV,ST5_BW,ST3_DR,ST5_BU,ST5_BX,ST3_DX,ST3_AX,ST3_BG,ST3_BH,ST3_DW,ST3_AW,ST3_BI,ST3_DB,ST3_DI,ST3_BJ,ST3_DD,ST3_DS,ST3_AZ,ST3_BC,ST3_BK,ST3_BL,ST3_BP,ST3_BQ,ST3_BR,ST3_BA,ST3_BM,ST3_BN,ST3_BO,ST3_BS,ST3_BX,ST3_CG,ST3_CH,ST3_CI,ST3_DY,ST3_AY,ST3_BT,ST3_CB,ST3_BU,ST3_BV,ST3_BW,ST3_CP,ST3_BB,ST3_BY,ST3_BZ,ST3_CA,ST3_CD,ST3_CJ,ST3_CK,ST3_CC,ST3_CM,ST3_CN,ST3_CO,ST3_CF,ST3_CQ,ST3_DH,ST3_DJ,ST3_CS,ST3_CT,ST3_CU,ST3_CW,ST3_CX,ST3_CY,ST3_DK,ST3_CV,ST3_CZ,ST3_DL,ST3_DM,ST3_DO,ST3_DA,ST3_DE,ST3_DF,ST3_DN,ST3_DT,ST3_DZ,ST3_BD,ST3_DG,ST3_DV,ST3_EB,ST3_DU,ST3_EA

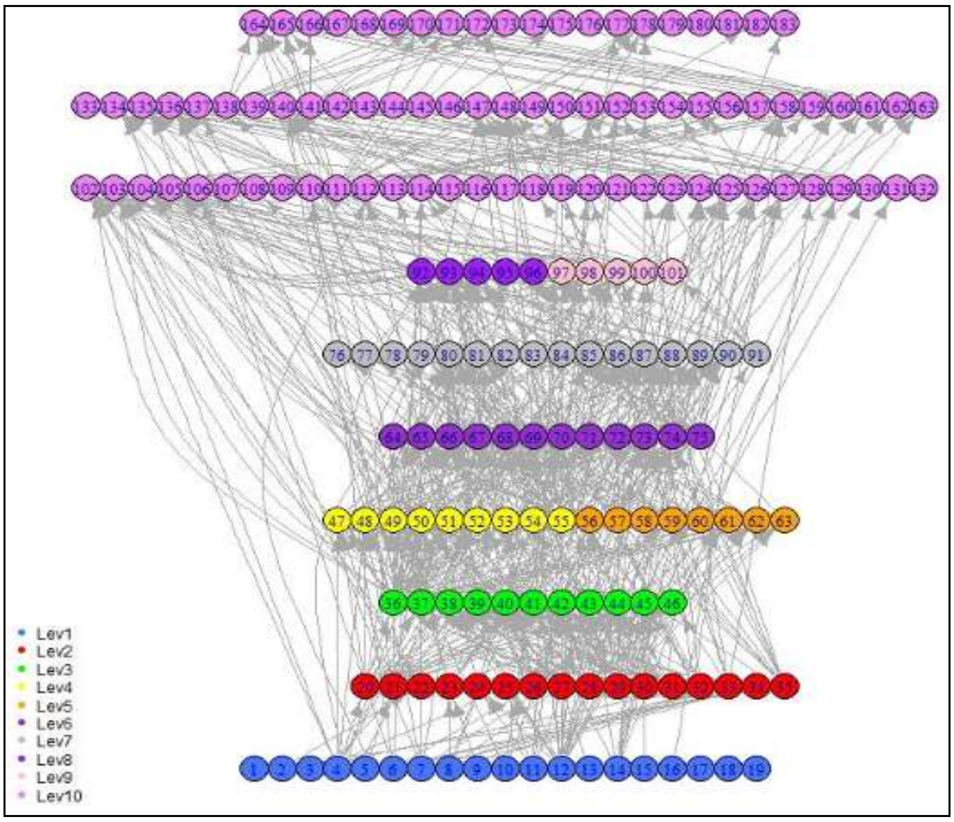


Figure 8. The Results of Data Visualization

6. Conclusion

In this study, we have proposed a modified topological algorithm to gain the feasible execution sequence of the formula dataset with hundreds of formulas. To gain our the target, a feasible order and level, we add a level list to store the current level topological order and a flag distinguished between creating a new list and using the last created list to improve the execution efficiency based on standard topological algorithm.

Compared to the customer's manual calculation result, our system gains the same result but spend a little time. This shows our algorithm is effective in solving automatic calculation of formula dataset which includes many formulas (more than 100) and is constrained by previous or next line data.

Acknowledgments

The authors would like to thank all the reviewers for their constructive comments. This research was supported by the National Natural Science Foundation of China (U1433124,51475065), the Natural Science Foundation of Liaoning Province (2015020013), Open Fund of Key Laboratory of Guangxi High Schools for Complex System & Computational Intelligence(15CI06Y), Open Project Program of Guangxi Key laboratory of hybrid computation and IC design analysis(HCIC201507,HCIC201402), Open Project Program of the Traction Power State Key Laboratory of Southwest Jiaotong University (TPL1403), Open Project Program of State Key Laboratory of Mechanical Transmissions(Chongqing University)(SKLMT-KFKT-201416, SKLMT-KFKT-201513).

References

[1] H. Fengling, L. Lei, and Z. Xiaozhi, "Solve the problem of formula evaluation seguence by directed graph", Computer Engineering and Application, vol. 39, no. 6, **(2003)**, pp. 87-90.
[2] J. Pichitlamken, S. Kajkamhaeng, and P. Uthayopas, "high performance Spreadsheet Simulation on a desktop Grid", in: Proceeding- Winter Simulation Conf., **(2008)**, pp. 663-670.
[3] W. Q. Ma, J. Z. Li, Z. Y. Jin and K. Ding, "Algorithm for dependent cell re-computation in report system", in: Computer Engineering, vol. 32, no. 13, **(2003)**, pp. 49-51.
[4] C. C. Zhao, L. Y. Zhao, and X. M. Zhang, "Research on enterprise spreadsheet model based-on semantic", in: 6th Int. Conf. Fuzzy Syst. Knowledge Discov. FSKD 2009, vol. 7, **(2009)**, pp. 441-446.
[5] Q. Partitional and C. Algorithm, "Semi-supervised Quadratic Partitional Clustering Algorithm and its Application in Spreadsheet System", Journal of Chines Computer Systems, vol. 32, no. 3, **(2011)**, pp. 409-505.
[6] M. C. Er, "A parallel computation approach to topological sorting", in: Computer Journal, vol. 26, no. 4, **(1983)**, pp. 293-295.
[7] D. J. Pearce and P. H. J. Kelly, "A dynamic topological sort algorithm for directed acyclic graphs", Journal Exp. Algorithmics, vol. 11, no. 1, **(2007)**, p. 1-7.
[8] https://en.wikipedia.org/wiki/Topological_sorting.
[9] D. Ajwani and T. Friedrich, "An O(n 2.75) Algorithm for Incremental", ACM Trans. Algorithms, vol. 4, no. 4, **(2008)**, pp. 1-14.
[10] D. Ajwani, A. Cosgaya-Lozano, and N. Zeh, "A topological sorting algorithm for large graphs", Journal Exp. Algorithmics, vol. 17, no. 1, **(2012)**, pp.1-8.
[11] D. Ajwani, A. Cosgaya-lozano, and N. Zeh, "Engineering a Topological Sorting Algorithm for Massive Graphs", Alenex, vol. 2011, **(2011)**, pp. 139-150.
[12] D. J. Pearce and P. H. J. Kelly, "A batch algorithm for maintaining a topological order", Conf. Res. Pract. Inf. Technol. Ser., vol. 102, no. 1, **(2010)**, pp. 79-87.
[13] I. Katriel and H. L. Bodlaender, "Online topological ordering", ACM Trans. Algorithms, vol. 2, no. 3, **(2006)**, pp. 364-379.
[14] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan, "Incremental Cycle Detection, Topological Ordering, and Strong Component Maintenance", ACM Trans. Algorithms, vol. 8, no. 1, **(2011)**, pp.1-7.
[15] R. Liu, "A Low Complexity Topological Sorting Algorithm for Directed Acyclic Graph", in: Int. J. Mach. Learn. Computer, vol. 4, no. 2, **(2014)**, pp. 194-197.
[16] C. Y. Pang, J. H. Wang, Y. Cheng, H. L. Zhang and T. L. Li, "Topological sorts on DAGs", Information Processing Letter, vol. 114, no. 2, **(2015)**, pp. 298-301.
[17] E. A. A. Tabl, "Topological approaches to generalized definitions of rough multiset approximations", Int. J. Mach. Learn. Cybern., vol. 6, no. 3, **(2015)**, pp. 399-407.
[18] J. Reininghaus, S. Huber, U. Bauer, M. Tu, and R. Kwitt, "A Stable Multi-Scale Kernel for Topological Machine Learning", ARXIV, http://arxiv.org/abs/1412.6821.
[19] J. Barnat, L. Brim, and P. Ročkai, "Parallel partial order reduction with topological sort proviso", in: Proc. - Softw. Eng. Form. Methods, SEFM 2010, no. 10, **(2010)**, pp. 222-231.
[20] G. P. Wang and S. Zhang, "The LAOV network and its topological sorting algorithm", in: Computer Engineering & Science, vol. 34, no. 3, **(2012)**, pp. 170-175.
[21] A. B. Kahn, "Topological sorting of large networks", in: Commun. ACM, vol. 5, no. 11, **(1962)**, pp. 558-562.
[22] http://www.igraph.org
[23] https://msdn.microsoft.com/enus/library/ms173183(VS.80).aspx

Authors

Zhengguang Li, Lecture, received the Master degree in computer science and technology from Dalian Jiaotong University in 2007. His research interests: Artificial intelligence, Algorithm.

Huimin Zhao, Associate Professor, received the Doctor degree in mechanical engineering and automation from Dalian Jiaotong University in 2013. Her research interests: Artificial intelligence, Signal processing, Algorithm.

Wu Deng, Professor, received the Doctor degree in computer science and technology from Dalian Maritime University in 2012. My research interests: Artificial intelligence, Computer application.