Directed Graphs and Topological Analysi

A.Mahir Khalfallah Mohamed Zain

Dr. Adel Ahmed Hassan

Abstract:

This research aims to study how to preserve the topological ordering of a Directed Acyclic Graph (DAG) in the presence of edge insertion and deletion operations, and to develop mathematical models for understanding the relationships between different elements in directed graphs. The research presents a new algorithm that contributes to the theoretical development of directed graphs and their graphical structures. It also helps improve the understanding of interconnected and complex networks such as communication and data networks.Furthermore, the study provides a connection between topological approaches in analyzing global graphical structures over sets of vertices in directed graphs, combining theoretical foundations with practical applications. The results demonstrate the development of accurate mathematical models for understanding and analyzing global graphical structures in directed graphs, as well as identifying new properties of these structures and their relationships to practical applications in various fields. The study recommends applying this algorithm in the topological analysis of directed graphs.

Keywords:"Directed graphs and topological analysis are two concepts used together to understand complex relationships and systems.

الرسوم البيانية الموجهة والتحليل الطوبولوجي

أ. ماهر خلف الله محمد زين - طالب دكتوراه - جامعة وادي النيل
د. عادل احمد حسن - قسم الرياضيات - جامعة وادي النيل

المستخلص:

يهدف هذا البحث الى دراسة كيفية الحفاظ علي الترتيب الطوبولوجي للرسم البياني الموجه غير الدوري في وجود عمليات ادخال وحذف الحواف وتطوير نماذج رياضية لفهم العلاقات بين العناصر المختلفة في الرسوم البيانية الموجهة ويقدم هذا البحث خوارزمية جديدة تساهم في التطوير النظري للرسوم البيانية الموجهة وهياكلها الرسومية كما تساعد في تحسين فهم الشبكات المترابطة والمعقدة مثل شبكات الأتصالات والبيانات وتقدم دراسة تربط الأساليب الطوبولوجية في تحليل الهياكل الرسومية الكلية علي مجموعات الروؤس في الرسوم البيانية الموجهة بين الأسس النظرية والتطبيقات العملية وتظهر النتائج تطوير نمازج رياضية دقيقة لفهم وتحليل الهياكل الرسومية الكلية في الرسوم البيانية الموجهة وتحديد خصائص جديدة لهذه الهياكل وعلاقتها بالتطبيقات العملية في مختلف المجالات ونوصى بتطبيق هذه الخوارزمية في التحليل الطوبولوجي للرسوم البيانية الموجهة .

الكلمة الافتتاحية: الرسومات البيانية، الموجهة والتحليل الطوبولوجي، مفهومان يستخدمان معاً لفهم العلاقات والأنظمة المعقدة .

Introduction

A topological ordering, ord, of a directed acyclic graph G = (V, E) maps each vertex to a priority value such that, for all edges x→y ∈ E, it is the case that ord(x) < ord(y). There exist it's ll known linear time algorithms for comput ing the topological order of a DAG (see e.g. [4]). Hoit's ver, these solutions are considered offline as they compute the solution from scratch. In this paper it's examine online algorithms, which only perform work neces sary to update the solution after a graph change. It's say that an online algorithm is fully dynamic if it supports both edge insertions and deletions. The main con tributions of this paper are as follows: 1. A new fully dynamic algorithm for maintaining the topological order of a directed acyclic graph. 2. The first experimental study of algorithms for this problem. It's compare against two online algorithms [15, 1] and a simple offline solution. It's show that, compared with [1], our algorithm has marginally inferior time complexity, but its simplicity leads to better overall performance

in practice. This is mainly because our algorithm does not need the Dietz and Sleator or dered list structure [7]. It's also find that, although [15] has the worst theoretical complexity overall, it outperforms the others when the graphs are dense. Organisation: Section 2 covers related work; Section 3 begins with the pre sentation of our new algorithm, folloit's d by a detailed discussion of the tw previous solutions [1, 15]. Section 4 details our experimental work.

Given a weighted directed acyclic graph (a DAG), put the vertices in order such that all its directed edges point from a vertex earlier in the order to a vertex later in the order (or report that doing so is not possible). This ordering is called a topological ordering.

1 Related Work At this point, it is necessary to clarify some notation used throughout the re mainder. Note, in the following it's assume G = (V, E) is a digraph: Definition 1. The path relation, ❀, holds if

∀x, y ∈ V.[x❀y ⇐⇒ x⟶y∈ET], where GT = (V, ET) is the transitive closure of G. If x❀y, it's say that x reaches y and that y is reachable from x. Definition 2. The set of edges involving vertices from a set, S ⊆ V , is E(S) = {x⟶y | x⟶y ∈ E ∧ (x ∈ S ∨ y ∈ S)}. Definition 3. The extended size of a set of vertices, K ⊆ V , is denoted

||K|| = |K| + |E(K)|. This definition originates from [1]. The offline topological sorting problem has been widely studied and optimal algorithms with Θ(||V ||) (i.e. Θ(|V |+|E|)) time are known (see e.g. [4]). Hoit's ver, the problem of maintaining a topological ordering online appears to have received little attention. Indeed, there are only two existing algorithms which, henceforth, it's refer to as AHRSZ [1] and MNR [15]. It's have implemented both and will detail their working in Section 3. For now, it's wish merely to examine their theoretical complexity. It's begin with results previously obtained: – AHRSZ - Achieves O(||δ||log||δ||) time complexity per edge insertion, where δ is the minimal number

of nodes that must be reprioritised [1, 19]. – MNR - Here, an amortised time complexity of O(|V |) over Θ(|E|) insertions has been shown [15]. There is some difficulty in relating these results as they are expressed differently. Hoit's ver, they both suggest that each algorithm has something of a difference betit's en best and worst cases. This, in turn, indicates that a standard worse- case comparison would be of limited value. Determining average-case performance might be better, but is a difficult undertaking. In an effort to find a simple way of comparing online algorithms the notion of bounded complexity analysis has been proposed [20, 1, 3, 19, 18]. Here, cost is measured in terms of a parameter δ, which captures in some way the minimal amount of work needed to update the solution after some incremental change. For example, an algorithm for the

online topological order problem will update ord, after some edge insertion, to produce a valid ordering ord 0 . Here, δ is vieit's d as the smallest set of nodes whose priority must change betit's en ord and ord 0 . Under this system, an algorithm is described as bounded if its worse-case complexity can be expressed purely in terms of δ.

2 Online Topological Order We now examine the three algorithms for the online topological order prob lem: PK, MNR and AHRSZ. The first being our contribution. Before doing this however, we must first present and discuss our complexity parameter δxy. Definition 4. Let G

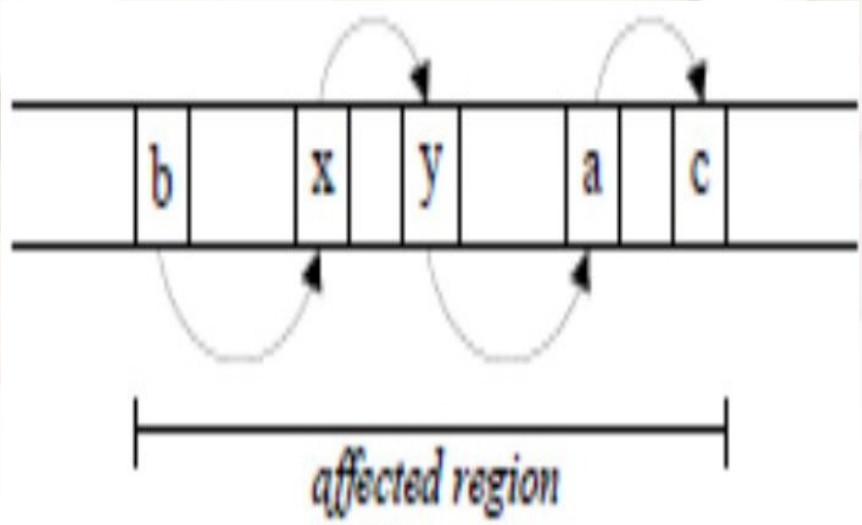
= (V, E) be a directed acyclic graph and ord a valid topo logical order. For an edge insertion x → y, the affected region is denoted ARxy and defined as {k ∈ V | ord(y) ≤ ord(k) ≤ ord(x)}. Definition 5. Let G = (V, E) be a directed acyclic graph and ord a valid topo logical order. For an edge insertion x→y, the complexity parameter δxy is defined as {k ∈ ARxy | y=k ∨ x=k ∨ y❀k ∨ k❀x}. Notice that δxy will be empty when x and y are

already prioritised correctly (i.e. when ord(x) < ord(y)). We say that invalidating edge insertions are those which cause |δxy| > 0. To understand how δxy compares with δ and the idea of minimal work, we must consider the minimal cover (from [1]): Definition 6. For a directed acyclic graph G

= (V, E) and an invalidated topo logical order ord, the set K of vertices is a cover if ∀x, y ∈ V.[x❀y ∧ ord(y)< ord(x) ⇒ x ∈ K∨y ∈ K]. This states that, for any connected x and y which are incorrectly prioritised, a cover K must include x or y or both. We say that K is minimal, written Kmin, if it is not larger than any valid cover. Furthermore, we now show that Kmin ⊆ δxy: Lemma 1. Let G = (V, E) be a directed acyclic graph and ord a valid topological order. For an edge insertion x→y, it holds that Kmin ⊆ δxy. Proof. Suppose this were not the case. Then a node a ∈ Kmin, where a ∈/ δxy must be possible. By Definition 6, a is incorrectly prioritised with respect to some node b. Let us assume (for now) that b❀a and, hence, ord(a)

2.1 The PK Algorithm We now present our algorithm for maintaining the topological order of a graph online. As we will see in the coming Sections, it is similar in design to MNR, but achieves a much tighter complexity bound on execution time. For a DAG G, the algorithm implements the topological ordering, ord, using an array of

size |V |, called the node-to-index map or n2i for short. This maps each vertex to a unique integer in {1 . . . |V |} such that, for any edge x→y in G, n2i[x] < n2i[y]. Thus, when an invalidating edge insertion x →y is made, the algorithm must update n2i to preserve the topological order property. The key insight is that we can do this by simply reorganising nodes in δxy. That is, in the new ordering, n2i 0 , nodes in δxy are repositioned to ensure a valid ordering, using only positions previously held by members of δxy. All other nodes remain unaffected. Consider the following caused by invalidating edge x→y:



Here, nodes are laid out in topological order (i.e. increasing in n2i value from left to right) with members of δxy shown. As n2i is a total and contiguous ordering, the gaps must contain nodes, omitted to simplify the discussion. The affected region contains all nodes (including those not shown) between y and x. Now, let us partition the nodes of δxy into two sets: Definition 7. Assume G = (V, E) is a DAG and ord a valid topological order. Let x → y be an invalidating edge insertion, which does not introduce a cycle. The sets RF and RB are defined as RF = {z ∈ ARxy | z = y ∨ y ❀z} and RB = {z ∈ ARxy | z

=x ∨ z❀x}. Note, there can be no edge from a member of RF to any in RB, otherwise x→ y would a introduce a cycle. Thus, for the above graph, we have RF = {y, a, c} and RB = {b, x}. Now, we can obtain a correct ordering by repositioning nodes to ensure all of RB are left of RF , giving:

In doing this, the original order of nodes in RF must be preserved and likewise for RB. The reason is that a subtle invariant is being maintained:

$$\forall x \in RF .\ n2i[x] \leq n2i\ 0\ [x] \ \wedge \forall y \in RB.\ n2i\ 0\ [y] \leq n2i[y]$$

This states that members of RF cannot be given lower priorities than t already have, whilst those in RB cannot get higher ones. This is because, for any procedure

```
procedure add_edge(x, y)
  lb = n2i[y]; ub = n2i[x]; if lb < ub then dfs-f(y); dfs-b(x); reassign();

procedure dfs-f(n)
  mark n as visited; R_F ∪= {n};
  forall n→w ∈ E do
    if n2i[w] = ub then abort; //cycle
    if w not visited ∧ n2i[w] < ub then dfs-f(w);

procedure reassign()
  sort(R_B); sort(R_F); L = ∅;
  for i = 0 to |R_B|−1 do w = R_B[i]; R_B[i] = n2i[w]; unmark w; push(w, L);
  for i = 0 to |R_F|−1 do w = R_F[i]; R_F[i] = n2i[w]; unmark w; push(w, L);
  merge(R_B, R_F, R);
  for i = 0 to |L|−1 do n2i[L[i]] = R[i];
```

add edge(x, y) lb = n2i[y]; ub = n2i[x]; if lb < ub then dfs-f(y); dfs- b(x); reassign();

procedure dfs-f(n)

mark n as visited; R_F ∪= {n}; forall n→w ∈ E do

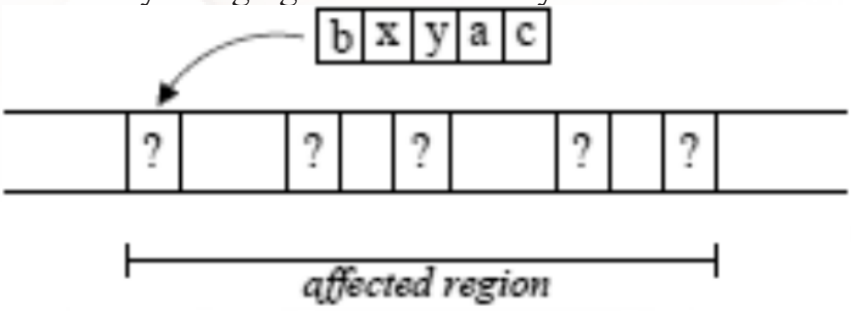
if n2i[w] = ub then abort; //cycle if w not visited ∧ n2i[w]< n2i[y].

"merge" combines two arrays into one whilst maintaining sortedness. "dfs-b" is similar to "dfs-f" except it traverses in the reverse direction, loads into RB and compares against lb. Note, L is a temporary. node

in RF , we have identified all in the affected region which must be higher (i.e. right) than it. However, we have not determined all those which must come lower and, hence, cannot safely move them in this direction. A similar argument holds for RB. Thus, we begin to see how the algorithm works: it first identifies RB and RF . Then, it pools the indices occupied by their nodes and, starting with the lowest, allocates increasing indices first to members of RB and then RF . So, in the above example, the algorithm proceeds by allocating b the lowest available index, like so: affected region b x y a c ? ? ? ? ? after this, it will allocate x the next lowest index,

then y and so on. The algo rithm is presented in

Figure 1 and the following summarises the two stages: Discovery: The set δxy is identified using a forward depth-first search from y and a backward depth-first search from x. Nodes outside the affected region are not explored. Those visited by the forward and backward search are placed into RF and RB respectively. The total time required for this stage is Θ(||δxy||). Reassignment: The two sets are now sorted separately into increasing topolog ical order (i.e. according to n2i), which we assume takes Θ(|δxy|log |δxy|) time. We then load RB into array L followed by RF . In addition, the pool of avail able indices, R, is constructed by merging indices used by elements of RB an



Topologies on the Edges:

One of the most important structures in discrete mathematics is graph theory. It is a prominent mathematical tool in many subjects . Their importance can be attributed to two reasons. First, graphs are mathematically elegant from theoretical viewpoint. Even though

graphs are simple relational combinations, they can represent topological spaces, combinatorial objects and many other mathematical combinations. Many concepts will be very useful from practical perspective when they abstractly represented by graphs and this represents the second reason for importance of graphs Topology is an interesting and important field of

mathematics.

Definition(1.2.1) [83] :

Let $D = (V, E, \varphi\square)$ be any directed graph. The compatible edges topology, denote $\mathcal{T}\square\circ\sqcap(D)$, is a topology that associated with the set

E of edges of D and induced by the subbasis SCE whose elements consist of the sets: $B \subseteq E$, $|B| \leq 2$ such that if $e \in B$ and f is adjacent with e in the same direction, then $f \in B$, i.e. $f \in B$ if f is adjacent with e and construct with e a path of length two.

The incompatible edges topology, denote $\mathcal{T}\circ\mathcal{O}\circ\sqcap(D)$, is a topology that associated with the set E of edges of D and induced by the subbasis SIE whose elements consist of the sets: $B \subseteq E$, $|B| \leq 2$ such

that if $e \in B$ and f is adjacent with e in the different direction, then f

$\in B$, i.e. $f \in B$ if f is adjacent with e and not construct a path with e. Example (1.2.2) :

Consider the directed graph in figure 1 such that $V(D) = \{v1,$ $v2, v3,$

$v4\}$, $E(D) = \{e1,e2,e3,e4\}$ and $\varphi D(e1) = (v1, v2)$, $\varphi D(e2) = (v2,$

$v3)$, $\varphi D(e3) = (v3, v4)$, $\varphi D(e4) = (v1, v2)$.

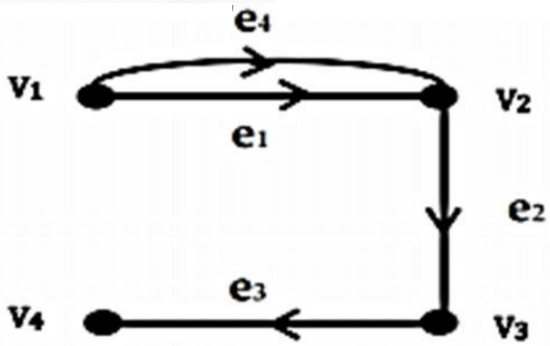


Fig (1.13)

We have, the subbasis for $\mathcal{T}CE(D)$ is $SCE = \{\{e1,e2\},\{e4,e2\},\{e2,e3\}\}$. By taking finitely intersection the basis obtained is $\{e2\}$,

$\{e1,e2\}$, $\{e4,e2\}$, $\{e2,e3\}$, $\emptyset$.

Then by taking all unions the topology $\mathcal{T}$ ○π(D) can be written as:

$\mathcal{T}$ ○π$(D) = \{\emptyset, E,\{e2\},\{e1,e2\},\{e4,e2\},\{e2,e3\},\{e1,e2,e4\},\{e1,e2,e3\},\{e4,e2,e3\}\}$. Now, the subbasis for $\mathcal{T}$○ᘐ○π(D) is

S○ᘐ○π $= \{\{e1,e4\},\{e2\},\{e3\}\}$.

By taking finitely intersection the basis obtained is $\{e1,e4\}$, $\{e2\}$,

$\{e3\}$, $\emptyset$.

Then by taking all unions the topology $\mathcal{T}$○ᘐ○π(D) can be written as:

$\mathcal{T}$○ᘐ○π$(D) = \{\emptyset, E,\{e2\},\{e3\},\{e1,e4\},\{e2,e3\},\{e1,e2,e4\},\{e1,e3,e4\}\}$.

It is easy to see that $\mathcal{T}CE(D)$ and $\mathcal{T}IE(D)$ are non-similar but they produce the same topology on the ground set E of a digraph $D = (V,$

$E,\varphi D)$ when they are both discrete topologies as in the directed cycle graph and the complete symmetric digraph such that $|V| \geq 3$.

References

(1) B. Alpern, R. Hoover, B. K. Rosen, P. F. Sweeney, and F. K. Zadeck. Incremental evaluation of computational circuits. In Proc. 1st Annual ACM-SIAM Symposium on Discrete Algorithms, pages 32– 42, 1990.

(2) S. Baswana, R. Hariharan, and S. Sen. Improved algorithms for maintaining tran sitive closure and all-pairs shortest paths in digraphs under edge deletions. In Proc. ACM Symposium on Theory of Computing, 2002.

(3) A. M. Berman. Lower And Upper Bounds For Incremental Algorithms. PhD thesis, New Brunswick, New Jersey, 1992.

(4) T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algo rithms. MIT Press, 2001.

(5) C. Demetrescu, D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Maintaining shortest paths in digraphs with arbitrary arc weights: An experimental study. In Proc. Workshop on Algorithm Engineering, pages 218–229. LNCS, 2000.

(6) C. Demetrescu and G. F. Italiano. Fully dynamic transitive closure: breaking through the O(n2) barrier. In Proc. IEEE Symposium on Foundations of Computer Science, pages 381–389, 2000.

(7) P. Dietz and D. Sleator. Two algorithms for maintaining order in a list. In Proc. ACM Symposium on Theory of Computing, pages 365– 372, 1987.

(8) H. Djidjev, G. E. Pantziou, and C. D. Zaroliagis. Improved algorithms for dynamic shortest paths. Algorithmica, 28(4):367–389, 2000.

(9) D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic shortest paths and negative cycle detection on digraphs with arbitrary arc weights. In Proc. European Symposium on Algorithms, pages 320–331, 1998.

(10) Y. Ioannidis, R. Ramakrishnan, and L. Winger. Transitive closure algorithms based on graph traversal. ACM Transactions on Database Systems, 18(3):512–576, 1993.

(11) G. F. Italiano, D. Eppstein, and Z. Galil. Dynamic graph algorithms. In Algorithms and Theory of Computation Handbook, CRC Press. 1999.

(12) R. M. Karp. The transitive closure of a random digraph. Random Structures & Algorithms, 1(1):73–94, 1990.
(13) I. Katriel. On algorithms for online topological ordering and sorting. Research Report MPI-I-2004-1-003, Max-Planck-Institut fur ¨ Informatik, 2004.
(14) V. King and G. Sagert. A fully dynamic algorithm for maintaining the transitive closure. In Proc. ACM Symposium on Theory of Computing, pages 492–498, 1999.
(15) A. Marchetti-Spaccamela, U. Nanni, and H. Rohnert. Maintaining a topological order under edge insertions. Information Processing Letters, 59(1):53–58, 1996.
(16) D. J. Pearce. Some directed graph algorithms and their application to pointer analysis (work in progress). PhD thesis, Imperial College, London, 2004.
(17) D. J. Pearce, P. H. J. Kelly, and C. Hankin. Online cycle detection and difference propagation for pointer analysis. In Proc. IEEE workshop on Source Code Analysis and Manipulation, 2003.
(18) G. Ramalingam. Bounded incremental computation, volume 1089 of Lecture Notes in Computer Science. Springer-Verlag, 1996.
(19) G. Ramalingam and T. Reps. On competitive on-line algorithms for the dynamic priority-ordering problem. Information Processing Letters, 51(3):155–161, 1994.
(20) T. Reps. Optimal-time incremental semantic analysis for syntax- directed editors. In Proc. Symp. on Principles of Programming Languages, pages 169–176, 1982.
(21) J. Zhou and M. Muller. ¨ Depth-first discovery algorithm for incremental topological sorting of directed acyclic graphs. Information Processing Letters, 88(4):195–200, 2003