ISSN: 0974-5823 Vol. 7 No. 4 April, 2022

International Journal of Mechanical Engineering

A STUDY ON NLC2 DECOMPOSITION IN POLYNOMIAL TIME

Vidhya. V [a], Velmurugan. N [b]

[a] M.Phil., Research Scholar, PG and Research Department of Mathematics, Theivanai Ammal College for Women (Autonomous), Villupuram – 605 602, Tamil Nadu, India

[b] Assistant Professor, PG and Research Department of Mathematics, Theivanai Ammal College for Women (Autonomous), Villupuram – 605 602, Tamil Nadu, India

Abstract

In this article, reporter accepted that a group-decomposition can be convert into an identical NLC-decomposition of the same thickness, and an NLC-decomposition of k can be convert into an identical group-decomposition of thickness at greatest in amount 2k. A randomly generated random-data graph on a coordinate of n nodes is very likely to have NLC-thickness and group-thickness algreatest in amount n/2 and n apart, provided that n is sufficiently huge.

Key Words: Graph decomposition, modular decomposition, cographs, NLC-thickness, group-thickness.

1.Introduction

Graphs are middle concepts in computer science. The greatest in amount familiar form of graph consists of a coordinate of entity called vertices (or nodes), and a coordinate of boundary. An boundary join two vertices, showing that they (or entity they correspond to) are interconnected in some way; it could be by a solid connection, a similarity, or a limitation, for example. They are then adjacent, or neighbors. Both algebras we have seen so far are in fact similar to yet another algebra.

This is a system for increase graphs by restore small parts of them, such as individual vertices or boundary, by somewhat enormous graphs relative to to a coordinate of rules. The linguistics type and algebra in are two sides of the same item. The deduction of a graph by exchanges relative to to the linguistics can be report by the sub divide of a tree, from the origin downwards to the leaves. But this tree can be observe also (from the leaves up to the origin) as the illustration of a integratee process, relative to a deduction of the same graph, this time using the operate of the algebra.

This differentiation between integrate and increase can be seen also for NLC- and group-decompositions. However, the linguistics formulation of increase is normally preferably technical. As we shall now see, a preferably simple formulation can be establish when we are association with group-decompositions.

We approach this by observeing first the boundary representation operation. Let D be a group-decomposition (think of it as a expression tree) of an random-data graph G, and let T be any alternate expression of D on the form ηi,j(T'), where, of course, T0 is itself a alternate expression. T defines a (labeled) graph G(T) in which each i-labeled vertex is adjacent to each j-labeled vertex. This is the result of ηi,j, which is the above operator to be register in T, relative to how operate and appearance have been defined. In the improving explanation that we are now going to make, we follow instead the growth of G by following D from the origin and downwards.

All the vertices with label i and j apart in G(T) can then be seen as descendants of two vertices, one with label i, and one with label j, which exist at the point where we reach the ηi,j-operation of T (from above). This operation then introduces an boundary between the two vertices, representing a relationship which will be inherited by their respective descendants. More general approaches to dynamic programming with the help of decompositions can be establish in [6], [7], [14].

2.Definitions and Lemma

Lemma 1: *The total number of vertices in the quotient graphs of the modular decomposition of a graph G is bounded by* $2|V(G)|$.

Proof. Let us view the modular decomposition of G, $DM(G)$, as a tree, T. The leaf nodes of T correspond to the vertices of G, and the nonleaf nodes correspond to the quotient graphs in $DM(G)$. Each nonleaf node has as many children as there are vertices in its quotient graph. Thus, the total number of vertices in the quotient graphs equals the number of nodes in 7, minus one corresponding to the root. Clearly, this is less than twice the number of leaf nodes O.

Definition 2 (Dimension, label-compactness): Let C be an NLC- seeker for a graph G. The dimension of C', dimC, is the number of vertex classes in C, and C is marker-compact with respect to G if *(i) Cls(C) = PG(V(C)), and (ii) the range of labc is* {1,..., *dimC}.*

It's instructional to look closer at what determines the minimal range of a union step. Let $C1$ and $C2$ be disjoint marker-compact NLC- campaigners for a graph G, and let $V *= V(C1)UV(C2)$.We'll see how a marker-compact NLC- seeker for G with vertex set V * can be produced in a union step involving $C1$ and $C2$. Such a step has the form $oR((oR_1, (C1))XS(oR_2, (C2)))$.Let us consider the mappings $R1$ and $R2$. Notice that each vertex class of $C1$ and $C2$ is a subset of some class of $PG(V *)$.Since $C1$ and $C2$ are formerly marker-compact, $R1$ and $R2$ mustn't change these vertex classes — only the associated markers may be changed. As long as this is fulfilled, we can always define the relation S similar that it produces the asked edges between the the two graphs. The point, however, is that a vertex class of op, $(C1)$ and one of $oR(C2)$ can have the same marker, but only if they " belong" to the same class of $PG(V *)$.Hence, we find the minimal range of a union step for $C1$ and $C2$ by casting, for each class V in $PG(V *)$,either the number of $C2$- classes that are subsets of V, or the number of C- classes that are subsets of V, whichever is topmost. Obviously, this range is at least as large as $max\{dimC1, dimC2\}$,, but not larger than $dimC1 + dimC2$

Definition 3:*[Label-induced partition, vertex class]* For a set of vertices V and a labeling function Poke defined on V, $Plab(V)$ denotes the partition of V into minimal, slightly labeled sets. A vertex class (or class, simply) of a labeled graph G is a class of Gormandizers, (V (G)). This partition is also denoted by *Cls(G). For* $v \in V(G)$, $Cl(G, v)$ denotes that vertex class of G to which uv belongs, and for any *l*, $Cl(G, l)$ denotes the set of vertices with marker — inG. Therefore, we allow the ultimate set to be empty, although vertex classes are, by description, nonempty.

Definition 4: (Partition induced by graph). For a set of vertices V in a graph G, $PG(V)$denotes that partition of V in which two vertices belong to the same class if and only if (i) they've the same marker in G, and (ii) they aren't distinguished by any vertex in V (G) V.

Lemma6.2.5. Let C be an NIC- seeker for a graph G and suppose that two verter classes in C, let us say with markers 11, *and* $l2$, are subsets of the same class in $PG(V(C))$. Also the graph CJ defined as $oR(C)$, where R maps $l2$ *to* $l1$, but leaves all other markers unchanged, can directly replace C in a derivate ofG.

Proof. Let D be an NLC- corruption of G containing, as a subterm, an NLC- corruption Dc of C, and let D' be what we gain by replacing $DC$with $oR(DC)$ in D. We want to show that the graph G' produced by D' is equal to G. It should be clear also that we only have to check (in G') the marker, and the neighborhood in $V(G)\backslash V(C)$of vertices whose marker is $l2$ in C'. Let $v2$ be such a vertex. We shall compare with a vertex $v1$ whose marker is 11inC. By the description of D', we know the following for G'.

- $v1$and $v2$ have the same marker.
- $v1$and $v2$ have the same neighbors in V (G) V (C).

But these two statements are true for G too, as $v1$ and $v2$ belong to the same class in $PG(V(C))$. Since $v1$ is obviously innocent by the relabeling, we can thus conclude that for v2, the marker, and the neighborhood in $V(G)\backslash V(C)$, are the same in G and G'.

3.Main result

3.1 NLC_2-Decomposition in Polynomial Time:

It has long been an open question whether a potential NLC-decomposition or clique- decomposition with at most k labels can befoundefficiently. For some small values of k, the answer is known to be yes. For instance, unlabeled graphs with NLC-width 1 are known as cographs; these graphs are discussed in [2], for example. Moreover, they are exactly the unlabeled graphs with clique-width 1 and 2. (An argument is provided in [18]). Every cograph has a unique decomposition in the form of a cotree, which can be found in linear time [3]. Such a decomposition can easily be turned into an NLC- decomposition of width 1, as well as a clique-decomposition of width at most 2.

3.2 Algorithm:

The input to this algorithm is a graph G labeled with numbers in $\{1, 2\}$. The output is a relabeling-free $NLC2$-decomposition D of G, if such a decomposition exists.

The algorithm constructs the decomposition in a top-down fashion, by successively partitioning the vertices of G. We know that if D exists, it has the form $(D1)XS(D2)$, where S is one of the 16 subsets of $\{(1, 1), (1, 2), (2, 1), (2, 2)\}$. More interesting, as soon as we find a partition $\{V1, V2\}$ of the vertices of G such that $G = (G1)XS(G2)$ for $G1 = G|V1, G2 = G|V2$, and S among the 16 possibilities above, we know that G has a relabeling-free $NLC2$-decomposition if and only if $G1$ and $G2$ do. Note, for instance, that if D is such a decomposition of G, then the restrictions $D|V1$ and $D|V2$ are themselves relabeling-free $NLC2$-decompositions of $G1$ and $G2$.

We shall now probe how one can find an NLC- corruption of minimal range for a given graphG. Although NLC- corruption has been defined for both labeled and unlabeled input graphs, we will assume that G is labeled, and that the range of $labG$ is $\{1, ...,1\}$ for some integer This will simplify our phrasings, yet of course, it implies no real restriction; an unlabeled graph can be regarded as slightly labeled. In the following, " graph" generally means " labeled graph", but the ultimate expression will occasionally be used explicitly to indicate an intrinsicneed for markers.

3.3 Modular Decomposition

Modular decomposition has been defined a number of times for various kinds of structures. It is called substitution decomposition in [23] [24], where an abstract analysis is presented and applied to relations (such as graphs), set systems, and boolean functions. The kind of generalized graphs called 2-structures [11] are also well-suited for modular decomposition, as shown in [11] [12] [13] [22]. The reader is referred to [13], [22], [23], [24] for further references. This section defines modular decomposition for labeled graphs. Its connection with NLC-decomposition is indicated, and some properties are formulated which will be needed later.Finally, it is described how the modular decomposition of a labeled graph can be computed with an existing algorithm for modular decomposition of 2-structures.

3.4 Other Graph Classes

In (27), Wanke also introduced a defined interpretation of the NLC algebra called NLCT. The difference is that in an NLCT union operation $(G)XS(G2)$,either S must be empty, inferring therefore that no edges are drawn between G1 and G2, or differently, one of these graphs must have only one vertex. The reason seems to have been that this defined form is sufficient for expressing a relationship with another, rather different kind of graph corruption called tree- corruption. Wanke showed that graphs with tree-range at most k have NLCT- range at most $2^{k+1}-1$[27].

Because of the below restriction, one can transfigure NLCT- decay of range k into crowd- decay using no further than $k + 1$ markers (18). Compared with the metamorphosis of a general NLC- corruption, union operations are now simpler, but relabeling operations bear further care with only one redundant marker. By combining the last two results, one finds that graphs with tree- range at most k have crowd- range at most 2^{k+1}. Nearly the same bound was attained in (10). Lately still, it has been shown that graphs with tree-range at most k have crowd- range (and therefore NLC- range, but not NLOT- range) at most3.2^{k-1} (5). Apropos, it follows from the results in Chapter 4 that the NLCT- range of a graph G is bounded by logn times the NLC- range. Then n is the number of vertices in G.

3.5 NLC- Campaigners

We first characterize those graphs that can do in derivatives of the input graph, and also take a brief look at the algorithmic idea.

Let u, v, and w be vertices in a graph G. We say that w distinguishes u from v in G if it has an edge in this graph to one of u and v, but not to the other.

Case: *[NLCk-candidate]* An NLC − k- seeker for a graph G is a labeled graph C satisfying the following

• V (C) CV (G).

• The edges of C are those convinced by G.

• For any brace of vertices u and v in C with $labC(u)=labC(v)$,we have

– $labG(u) =labG(v)$.

– No vertex in V (G) V (C) distinguishes u from v inG.

• $C\in NLCk$.

An NLC- seeker (or seeker, simply) is an $NLCk$- seeker for some value ofk.

Property: A labeled graph C is an NLC- seeker for a graph G if and only if C can (and does) do in some derivate ofG.

To NLC- putrefy a graph G, we're going to initialize a seeker set to the single vertices in G, each labeled with 1, and also add constantly to this, NLC- campaigners for G that can be formed (using union and relabeling) from those formerly in the set (that is, from clones of these). More precisely, we will add$NLC1$

For this approach to be successful, we must be suitable to keep down the size of the seeker set. We shall first consider the need for differing campaigners on the same vertex set.

3.6 Relationships and Explanation

Theorem 1. The crowd- range of a graph isn't larger than two times the NLC- range.

Let us now look at how a crowd- corruption D can be converted into an original NLC- corruption D'. Both of these will be regarded as term trees below. This time we will need no further markers than those formerly used inD. We'll not be suitable to transfigure each operation in D collectively, however. Quite naturally, every disjoint union operation in D will correspond to axS union operation in D'. Therefore, D' will get the same overall structure asD. Still, if there should ever be edges between the two graphs that axS-operation bring together, these edges must be added with this operation itself. In D, similar edges may be added by xS- operations anywhere over (that is, on the thrusting path from) the disjoint union operation in question. We must thus transport the effect of xS-operations in D down. It's intriguing to compare then with the growth interpretation of crowd- decay bandied before. In the growth model, the effect of xS- operations is implicitly transported down by an heritage medium. In the present metamorphosis problem, we're also going to follow D from the root and down, but we shall keep a record rather of the accumulated effect H of the xS-operations that we pass. When we reach a disjoint union operation, we will also know how to choose§ in the corresponding xS-operation.

The details are as follows. With H empty, we start above D, and begin to do down. When we pass an operato$\eta i,j$, we remember this by adding $\{i, j\}$ toH. H willalso be affected by relabelings. The driver $\rho i\rightarrow j$, which changes marker XS to XS in a graph, implies that whatever H says for marker XSincontinently above the driver, should hold also for marker i incontinently below it. Also, the graph defined by the subtree embedded at this driver obviously doesn't contain vertices with marker XS. Hence, if H contains dyads that include i above the $\rho i\rightarrow j$- driver, these are meaningless, and should be removed when we pass $\rho i\rightarrow j$. Rather, for each remaining $\{j,l\}$in H, where l is arbitrary, we add the $\{i,l\}$also.

Conclusion:

To NLC2- decay a graph G that is untagged or labeled with figures in , we have a tendency to 1st cipher the standard corruption of gram D(G), as outlined in Section5.1. With the system delineated in Section5.5, this takes O(n2) time, wherever n = | V (G)|.

We have a tendency to conjointly try and NLC2- decay every quotient graph alphabetic character in DM (G). once alphabetic character is punctually high, we have a tendency to use the algorithmic rule delineated in Section5.6, running in O(n^4 log nQ) time (where nQ = |V (Q)|When alphabetic character is not punctually high, its vertices is combined in any order, and that we will certainly construct a corruption in direct time. By Lemma5.4.4, the overall time for corruption of quotient graphs becomes O(n^4 log n).. The house used noway exceeds O(n^2). , If we have a tendency to currently have AN NLC2- corruption of every quotient graph in DM (G),Still, conjointly we have a tendency to erect along these decay into AN NLC2- corruption of G1 as delineated within the proof of Proposition fifty two.2. Solely direct time is demanded for this last step. In total, we've used O(n^4 log n) time and O(n^2) house.

REFERENCES

1. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algorithms. The MIT Press, 1990. 119Google Scholar
2. D. G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. Discrete Applied Mathematics, 3:163–174, 1981. 110, 111zbMATHCrossRefMathSciNetGoogle Scholar
3. D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. SIAM Journal on Computing, 14:926–934, 1985. 111zbMATHCrossRefMathSciNetGoogle Scholar
4. B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second order logic. To appear in Discrete Applied Mathematics. 111, 111Google Scholar
5. B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. In Graph-Theoretic Concepts in Computer Science, 24th International Workshop, WG'98, volume 1517 of Lecture Notes in Computer Science, pages 1–16. Springer, 1998. 111Google Scholar
6. Bruno Courcelle and Stephan Olariu. Clique-width: A graph complexity measure-preliminary results and open problems. In Proceedings of the Fifth International Workshop on Graph Grammars and Their Application to Computer Science, 1994. 110, 112Google Scholar Ehrenfeucht and G. Rozenberg. Theory of 2-structures, part i: Clans, basic subclasses, and morphisms. Theoretical Computer Science, 70:277–303, 1990. 115zbMATHCrossRefMathSciNetGoogle Scholar
7. Ehrenfeucht and G. Rozenberg. Theory of 2-structures, part ii: Representation through labeled tree families. Theoretical Computer Science, 70:305–342, 1990. 115zbMATHCrossRefMathSciNetGoogle Scholar
8. Andrzej Ehrenfeucht, Harold N. Gabow, Ross M. McConnell, and Stephen J. Sullivan. An O(n 2) divide-and-conquer algorithm for the prime tree decomposition of two-structures and modular decomposition of graphs. Journal of Algorithms, 16:283–294, 1994.zbMATHCrossRefMathSciNetGoogle Scholar
9. Öjvind Johansson. Clique-decomposition, NLC-decomposition, and modular decomposition — relationships and results for random graphs. Congressus Numerantium, 132:39–60, 1998. 111, 114, 114zbMATHMathSciNetGoogle Scholar
10. J. A. Makowsky and U. Rotics. On the clique-width of graphs with few P4's. To appear in IJFCS. 111Google Scholar
11. R. M. McConnell. An O(n 2) incremental algorithm for modular decomposition of graphs and 2-structures. Algorithmica, 14:229–248, 1995. 115zbMATHCrossRefMathSciNetGoogle Scholar
12. R. H. Möhring. Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and boolean functions. Annals of Operations Research, 4:195–225, 1985/6. 114CrossRefMathSciNetGoogle Scholar
13. R. H. Möhring and F. J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. Annals of Discrete Mathematics, 19:257–356, 1984.Google Scholar
14. Egon Wanke. k-NLC graphs and polynomial algorithms. Discrete Applied Mathematics, 54:251–266, 1994. 110, 111, 112, 113, 113zbMATHCrossRefMathSciNetGoogle Scholar