Constructing a good C-path

Daniil Demidov, Maximilian Wittmer

07.11.2024

Abstract

We define the concept of a special subset of vertices of an undirected graph called a *clique* and set the rules for when one can"jump" from one clique to another. Such a pair is called a *move*. An (ordered) sequence of cliques, such that every pair of adjacent cliques form a move, is called a *good path* if it satisfies specific properties that we define later. The goal is to establish when a good path is exists and, if so, how to construct it.

1. Introduction

There will be a detailed motivation as I finish digging through the required chapters of WPiG

Since we work with graphs, we need to outline a convention we will use throughout this document.

Convention 1.1: Let $\Gamma = (V, E)$ be an undirected graph with V being the (finite) set of vertices, and $E \subset \mathcal{P}(V)$ being the set of edges with $\forall e \in E, e = \{x, y\}$ for some $x, y \in V$.

Definition 1.2: *(Link)* For some $v \in V$, we call $lk_\Gamma(v) := \{v\} \cup \{u \in V \mid \{v, u\} \in E\}$ the **link** of v.

1.1. Joins

We need to define a specific type of graph that will be studied later.

Definition 1.1.1: *(Join Decomposition)* We call a tuple (L, O) with $L, O \subseteq V$ a **join decomposition**, if the following conditions are satisfied:
(i) $V = L \sqcup O$
(ii) $\forall l \in L, \forall o \in O : \{l, o\} \in E$.

Definition 1.1.2: *(Join Graph)* If, for a graph Γ, there exists a join decomposition, then we call Γ a **join**.

1.2. Cliques

Now, we define the already mentioned concept of special subsets of V.

Definition 1.2.1: *(Clique)* A subset $\alpha \subset V$ is called a **clique** if $\forall x, y \in \alpha, \{x, y\} \in E$.

Notation 1.2.2: *(Set of All Cliques)* For a graph Γ, we denote the set of all cliques as $\mathcal{C}(\Gamma) := \{\alpha \in \mathcal{P}(V) \mid \forall x, y \in \alpha, \{x, y\} \in E\}$.

Notation 1.2.3: *(Set of All Moves)*

$$\mathcal{M}(\Gamma) := \left\{ (\alpha, \beta) \in \mathcal{C}(\Gamma)^2 \mid (\beta \setminus \alpha) \cap \left(\bigcap_{v \in \alpha} lk_\Gamma(v) \right) = \emptyset \right\}. \tag{1}$$

If $m \in \mathcal{M}(\Gamma)$, we call m a **move**.

Remark 1.2.4: $\forall \alpha \in \mathcal{C}(\Gamma), (\alpha, \alpha) \in \mathcal{M}(\Gamma)$.

Definition 1.2.5: *(Maximal Clique)* A clique $\alpha \in \mathcal{C}(\Gamma)$ is called **maximal**, if $\forall \beta \in \mathcal{C}(\Gamma)$ holds

$$(\alpha, \beta) \in \mathcal{M}(\Gamma) \Longrightarrow |\beta| \leq |\alpha|.$$

1.3. Paths

Definition 1.3.1: *(C-path)* A sequence of cliques $(\omega_n)_{n \in \mathbb{N}}$ is called a **c-path** (**c**liques **path**), if $(\omega_{n+1}, \omega_n) \in \mathcal{M}(\Gamma)$ holds for all $n \in \mathbb{N}$.

At first, it might seem counterintuitive to define a c-path in a reverse order. But it is motivated by the goal of finding an algorithm for constructing such a path, which will be done top-to-bottom.

Remark 1.3.2: The definition of a c-path requires it to be an *infinite* sequence of cliques which would not translate to any real world application. So, at first glance, it seems impossible to have a finite path of cliques expressed as a c-path. But this problem can easily be solved by setting $\omega_n := \alpha$ for all $n \in \mathbb{N}, n \geq m$, whereby $m \in \mathbb{N}$ is the length of the path and α is the starting clique. Indeed, it satisfies the definition of a c-path because of Remark 1.2.4.

Definition 1.3.3: *(A good c-path)* A c-path ω is called **good** if
(i) ω_1 is maximal
(ii) $\exists M \in \mathbb{N} : |\omega_M| = 1$.

Definition 1.3.4: *(A good vertex)* $v \in V$ is called a **good vertex** if there exists a good c-path ω, such that $\omega_M = \{v\}$ for some $M \in \mathbb{N}$.

1.4. Goal

The goal is to prove the following theorem:

Theorem 1.4.1: A graph Γ is a join $\Longleftrightarrow$ no vertex of Γ is good.

Proof:
- "$\Longrightarrow$" has been shown in our previous paper. Write it down anyway!
- "$\Longleftarrow$" Is equivalent to Lemma 1.4.2.

■

Lemma 1.4.2: A graph Γ is not a join $\Longrightarrow$ there is a good vertex.

The rest of this paper is dedicated to proving Lemma 1.4.2. We will do is constructively by describing an algorithm for finding a c-path to an arbitrary maximal clique and proving that it is good.

2. Algorithms

2.1. Finding a Join Decomposition

The first goal is to determine whether a given graph Γ is a join or not.

Construction 2.1.1: *(Finding a Join Decomposition)* The goal is to construct two sequences $L_n, O_n \in \mathcal{P}(V)$, such that $\forall n \in \mathbb{N}_0$

$$L_n \cap O_n = \emptyset \tag{2}$$

$$L_n \cup O_n = V \tag{3}$$

We start by choosing an arbitrary vertex $v \in V$. We set $L_0 := \{v\}$, $O_0 := V \setminus \{v\}$. Thus, both (2) and (3) are satisfied. For $n \in \mathbb{N}$, we define

$$L_n = L_{n-1} \cup \{o \in O_{n-1} \mid \exists l \in L_{n-1}, \{l, o\} \notin E\} \tag{4}$$

$$O_n = V \setminus L_n. \tag{5}$$

(5) $\Rightarrow$ (2). Inductively, (3) also applies.

However, the algorithm must terminate. To determine when the recursion has to be stopped, we introduce $\Delta L_n := L_{n+1} \setminus L_n$ for all $n \in \mathbb{N}_0$. If $\Delta L_s = \emptyset$ for some $s \in \mathbb{N}_0$, then we have $L_{s+1} = L_s$. That means that *either* $O_s = \emptyset$ *or* $O \neq \emptyset$ (then, (4) would imply that $\forall o \in O_s, \forall l \in L_s : \{l, o\} \in E$). Both cases imply that $\forall n \geq s : \Delta L_n = \emptyset$. That is why it makes sense to stop the recursion as soon as $\Delta L_n = \emptyset$. The recursion depth is represented by $J_\Gamma(v) := \min\{s \in \mathbb{N}_0 \mid \Delta L_n = \emptyset \text{ wenn } L_0 = \{v\}\}$. Proposition 2.1.2 shows that $J_\Gamma(v)$ always exists.

Proposition 2.1.2: $\forall v \in V, \exists s \in \mathbb{N} : s = J_\Gamma(v)$.

Proof: By construction, s exists $\iff \Delta L_n = \emptyset$ for some $n \in \mathbb{N}$. Suppose that, for some starting vertex $v \in \Gamma$, $\Delta L_n \neq \emptyset$ for all $n \in \mathbb{N}$. That would imply $|L_{n+1}| > |L_n|$ for all $n \in \mathbb{N}$. This contradicts $|V| < \infty$ since $\forall n \in \mathbb{N}, L_n \subseteq V$. Thus, the assumption that such vertex $v \in V$ exists, that $J_\Gamma(v) \notin \mathbb{N}$, is false. ■

Now, the goal is to show that (L_s, O_s) is indeed a join decomposition of Γ.

Proposition 2.1.3: Γ is a join $\iff$ the algorithm described in Construction 2.1.1 terminates with $O_s \neq \emptyset$ for all $v \in V$, whereby $s = J_\Gamma(v)$.

Proof: "$\Longleftarrow$": Suppose that $\forall v \in V, O_s \neq \emptyset$, whereby $s = J_\Gamma(v)$. According to Construction 2.1.1, we have $\emptyset \neq L_s \subsetneq V$, such that $L_s \cap O_s = \emptyset$ and $\forall o \in O_s, \forall l \in L_s : \{l, o\} \in E$. By definition, Γ is a join.

"$\Longrightarrow$": Let Γ be a join graph. There could be different join decompositions (L, O) of this graph. As a reminder, a join decomposition of a graph is a tuple $(L, O) \in \mathcal{P}(V)^2$, such that

$$L \cap O = \emptyset \ , \ \ L \cup O = V \text{ and } \forall o \in O, \forall l \in L : \{l, o\} \in E. \tag{6}$$

Now, let us choose an arbitrary vertex $v \in V$. Then, there exists a join decomposition $(L, O) \in \mathcal{P}(V)^2$ such that $v \in L$. Join decomposition is not unambiguous, so we choose (wlog) such a decomposition that $|L|$ is minimal across all possible decomposition. Now

we let the algorithm Construction 2.1.1 run with $L_0 = \{v\}$. By induction we show that $O \subseteq O_n$ and $L_n \subseteq L$ for all $n \in \mathbb{N}_0$.

Clearly, we have $O \subseteq O_0$, as well as $L_0 \subseteq L$. Now we assume that $O \subseteq O_n$ and $L_n \subseteq L$ for some $n \in \mathbb{N}$. By construction, $\forall o \in O_{n+1}, \forall l \in L_n : \{l, o\} \in E$, therefore the assumption $L_n \subseteq L$ implies $O \subseteq O_{n+1}$. By construction, $L_{n+1} \cap O_{n+1} = \emptyset$. By definition, $L \cap O = \emptyset$. Thus, $\forall l \in L_{n+1} : l \notin O_{n+1} \Longrightarrow l \notin O \Longleftrightarrow l \in L$, which means $L_{n+1} \subseteq L$.

Obviously, $\forall n < s = J_\Gamma(v), \Delta L_{n+1} \neq \emptyset \Longrightarrow |L_n| < |L_{n+1}| \leq |L|$ as well as $|O| \geq |O_n| > |O_{n+1}|$. Proposition 2.1.2 shows, that the algorithm halts after $s = J_\Gamma(v))$ steps. Hence, $|L_s| \leq |L| < |V| \Longrightarrow L_s \neq V \Longrightarrow O_s \neq \emptyset$. That means that $\forall l \in L_s, \forall o \in O_s, \{l, o\} \in E \Longrightarrow L_s = L$ sowie $O_s = O$. ■

2.2. Adjacency Table

Now, we introduce one last construction that will be used to prove Lemma 1.4.2.

Construction 2.2.1: *(Adjacency Table)* Let $T_\Gamma(v)$ be a tuple of $(s+1)$ subsets of V, whereby $s = J_\Gamma(v)$ and

$$(T_\Gamma(v))_i \coloneqq \Delta L_{i-1} = L_i \setminus L_{i-1} \quad \text{for} \quad i \in \{1, ..., s\} \quad \text{and} \quad (T_\Gamma(v))_0 \coloneqq \{v\} \tag{7}$$

.

We call $T_\Gamma(v)$ the **adjacency table** of v (over Γ).

Example 2.2.2: Let $\Gamma = (\{1, 2, ..., 10\}, E)$. Then, the adjacency table might look something like

$(T_\Gamma(1))_0$	$(T_\Gamma(1))_1$	$(T_\Gamma(1))_2$	$(T_\Gamma(1))_3$	$(T_\Gamma(1))_4$
1	3 7	4 2 8	6 9	5 10

Convention 2.2.3: From now on, let $\Gamma = (V, E)$ be a non-join graph, q an arbitrary, but fixed vertex of Γ (as in Construction 2.1.1). Let $s = J_\Gamma(q) \in \mathbb{N}$ be the number of steps which the algorithm (Construction 2.1.1) terminates after with $L_s = V, O_s = \emptyset$. Let $T \coloneqq T_\Gamma(q)$ be the adjacency table of this vertex.

Let $I \coloneqq \{1, ..., s\}$ and $I_0 = I \cup \{0\}$ be the index sets for the adjacency table T.

Now, let's make the following observation:

Lemma 2.2.4: $\forall i, j \in I_0 : i \neq j \Rightarrow T_i \cap T_j = \emptyset$.

Proof: Without loss of generality, we assume $i > j$. By construction, we have then $T_j \subseteq L_{i-1}$ and $T_i = \Delta L_{i-1} = \{o \in O_{i-1} \mid \exists l \in L_{i-1} : \{l, o\} \notin E\} \subseteq O_{i-1}$. Because of $L_{i-1} \cap O_{i-1} = \emptyset$ we obtain $T_i \cap T_j = \emptyset$. ■

Lemma 2.2.5: For each adjacency table, the following holds:

$$\forall i \in I : \forall w \in T_i \ \exists v \in T_{i-1} : \{v, w\} \notin E \tag{8}$$

and

$$\forall i, j \in I_0 : |i - j| > 1 \Longrightarrow \forall v \in T_i, w \in T_j : \{v, w\} \in E. \tag{9}$$

Proof: (8) follows directly from Construction 2.1.1 and Construction 2.2.1. ■

Proof: (9): We assume that $\exists i, j \in I_0$ with $|i - j| > 1$, such that $\exists v \in T_i, w \in T_j : \{v, w\} \notin E$. Without loss of generality, let $i > j$. Lemma 2.2.4 implies $v \neq w$

and $T_i \cap T_j = \emptyset$ as well as $T_j \cap T_{j+1} = \emptyset$. $w \in T_j = \Delta L_{j-1} \Rightarrow$ by Construction 2.1.1, $w \in L_j$. We know that $T_{j+1} = \Delta L_j = \{o \in O_j \mid \exists l \in L_j : \{l, o\} \notin E\}$. $v \in T_i$ and $i > j$ imply $\forall k \leq j < i, v \notin T_k$, which means that $v \in O_j \Longrightarrow v \in \Delta L_j = T_{j+1} \Longrightarrow T_i = T_{j+1} \Longrightarrow j + 1 = i$. That contradicts $|i - j| > 1$, thus, the assumption that such v and w exist is false. That completes the proof of (9). ■

3. Construction of a Good C-path

Now our goal is to take advantage of the adjacency table defined above and its properties. For readability, we call the entries of the adjacency table T_i *cells*. The direction "left to right" corresponds to the direction "0 to s".

Notation 3.1: *(Selection)* We set $\Omega_n^i = \omega_n \cap T_i$ and call Ω_n^i the **selection** from the cell T_i in step n.

3.1. Zebras are good!

Definition 3.1.1: *(Zebra)* We call $\alpha \in \mathcal{C}(\Gamma)$ a **zebra**, if the following conditions are met:
(i) $\forall k \in I, k \leq \frac{s}{2} : \alpha \cap T_{2k+1} = \emptyset$
(ii) $\exists m \in I : \forall k \in I, k \leq m : |\alpha \cap T_{2k}| = 1$ and $\forall k \in I, k > m : |\alpha \cap T_{2k}| = 0$.
We call m the **zebra index** of α.

Remark 3.1.2: Lemma 2.2.5 applies that zebras exist. Should I explain this in more detail?

Example 3.1.3: $\alpha = \{q, d, g\}$ is a *zebra*.

T_0	T_1	T_2	T_3	T_4
q	a b	c **d** e	e f	**g** h

Definition 3.1.4: A zebra with zebra index $m = \left\lceil \frac{s+1}{2} \right\rceil$ is called a **maximal zebra** of q. α from Example 3.1.3 is the maximal zebra of q.

Remark 3.1.5: Each vertex has at least one maximal zebra because it is constructed using a deterministic algorithm described in Construction 2.1.1.

Definition 3.1.6: *(Reducibility)* A clique $\alpha \in \mathcal{C}(\Gamma)$ is called **reducible**, if a c-path $(\omega_n)_n$ exists, such that $\exists M, N \in \mathbb{N}, M > N$, with $\omega_N = \alpha$ and $|\omega_M| < |\omega_N|$.

Proposition 3.1.7: If $\alpha \in \mathcal{C}(\Gamma)$ is a zebra, then α is reducible in 2 steps.

Proof: We will construct a c-path $(\omega_n)_n$ with $\omega_1 = \alpha$ and $\omega_3 = \alpha \setminus T_{2m}$, whereby m is the zebra index of α. Thus, we only have to define ω_2. Further, let Ω_n^i denote the *selection* (Notation 3.1) with respect to ω. By Definition 3.1.1, we have $\forall k \leq m : |\Omega_n^{2k}| = 1$, and for all other $i \notin \{2k \mid k \in \mathbb{N} \text{ and } 0 \leq k \leq m\}$ $|\Omega_n^i| = 0$, namely, every index $0 \leq k \leq m$ corresponds to exactly one unique v_k with $\Omega_n^{2k} = \{v_k\}$. By Lemma 2.2.5, $\forall k \in \{1, ..., m\}, \exists u_k \in T_{2k-1} : \{v_k, u_k\} \notin E$. For each v_k we pick such u_k and define $\omega_2 := \{u_k \mid 0 < k \leq m\}$. But we do not know yet if $\{u_2, q\} \notin E$, because then and only then

will (ω_1, ω_2) be a move. It turns out to always be the case, since $T_0 = \{q\}$ by construction and all $x \in T_1$ (in particular, u_1) satisfy $\{q, x\} \notin E$. By construction, $\omega_1 \cap \omega_2 = \emptyset$. That implies

$$\omega_1 \cap \left(\bigcap_{v \in \omega_2} lk_\Gamma(v) \right) = \emptyset \Longrightarrow (\omega_2, \omega_1) \in \mathcal{M}(\Gamma). \tag{10}$$

ω_3 is defined similarly, namely, for every $u_k \in T_{2k-1} (0 < k \leq m)$, we pick a vertex $w_k \in T_{2k-2}$ such that $\{w_k, u_k\} \notin E$. In particular, $w_1 \in \alpha \cap T_0 = \{q\}$, which ultimately means $w_1 = q$, and $u_1 \in T_1$ implies $\{w_1, u_1\} \notin E$. Therefore, every vertex $x \in \omega_3$ corresponds to exactly one other vertex $y \in \omega_2$, such that $\{x, y\} \notin E$. Thus, (ω_3, ω_2) is also a move. ■

Corollary 3.1.8: Let $\alpha \in \mathcal{C}(\Gamma)$ be a zebra. Then, there exists a path $(\omega_n)_n$ with $\omega_1 = \alpha, \omega_M = \alpha \cap T_0 = \{q\}$, whereby $M = 2(|\alpha| - 1)$. In particular, ω is a *good c-path*.

Proof: If $|\omega_1| = 1$, then there is nothing to show. Otherwise let $\omega_1 = \alpha$. Then define ω_2 and ω_3 as in Proposition 3.1.7. This implies $|\omega_3| = |\omega_1| - 1$. Thus, ω_3 is another zebra, therefore Proposition 3.1.7 is applicable to ω_3 if $|\omega_3| > 1$. After repeating this $|\alpha| - 1$ times, whereby size of the current clique is reduced by 1 every other step, you get $|\omega_{2(|\alpha|-1)}| = 1$. By Definition 1.3.3, ω is good. ■

Example 3.1.9: Going back to the example table that we used earlier to illustrate zebras, let us demonstrate how such a c-path would look like. Suppose we have a c-path ω with $\omega_z = \{q, d, g\}$ being a zebra for some $z \in \mathbb{N}$.

n	T_0	T_1	T_2	T_3	T_4
z	**q**	a b	c **d** e	e f	**g** h
z + 1	q	**a** b	c d e	e **f**	g h
z + 2	**q**	a b	**c** d e	e f	g h
z + 3	q	a **b**	c d e	e f	g h
z + 4	**q**	a b	c d e	e f	g h

Note that even at the step $z + 3$ we already have found a good vertex. But our goal is to show that *every* vertex is good. For that purpose, we chose one arbitrary vertex q as the starting vertex for our algorithm and show that we can construct a good c-path from it to a maximal clique.

3.2. Putting the Pieces Together

Now, we have an algorithm that solves the problem if we start with a zebra. But not every maximal clique is a zebra. Example? On the other hand, every zebra is a subset of some maximal clique. Thus, in order to show that our arbitrarily chosen vertex q leads to *a* maximal clique (which is our goal), we need to be able to construct a c-path from a maximal clique to q. The easiest way is to choose some maximal clique that has one of the maximal zebras of q as a subset, then construct a c-path from that maximal clique to zebra, after which we complete this c-path using Corollary 3.1.8.

The idea is that we identify cells that do not meet the definition of a zebra (Definition 3.1.1), calling them *problems*, and provide a systematic approach for resolving them. By resolving, I mean constructing a c-path such that the amount of these

problems or the "size" of one of them gets smaller after each step. In order to be able to work out the solution, we need to define what we mean by the *size* of a problem.

Convention 3.2.1: Let ω_1 be some maximal clique, such that there exists a subset $\zeta \subseteq \omega_1$, such that ζ is a maximal zebra of q. By Corollary 3.1.8, there exists a c-path from ζ to q.

Definition 3.2.2: *(Problem Size)* For every clique ω_n, we associate each cell of the adjacency table with a number called **cell problem size** $\mathbb{P}_n^i$ defined the following way:

$$\mathbb{P}_n^i := \begin{cases} |\Omega_n^i| - 1 \text{ if } i \text{ is even} \\ |\Omega_n^i| \quad \text{if } i \text{ is odd} \end{cases} \tag{11}$$

The number $\mathbb{P}_n := \sum_{i=0}^{s} \mathbb{P}_n^i$ is called the (global) **problem size** at step n.

Remark 3.2.3: $\mathbb{P}_n^i \geq 0$ since we chose ω_1 to contain a maximal zebra of q.

Remark 3.2.4: $\mathbb{P}_n = 0 \iff \omega_n$ is a zebra. So, in some sense, $\mathbb{P}_n$ represents how "far" ω_n is from being a zebra.

Notation 3.2.5: For $i \in I \setminus \{s\}$ we write $\mathcal{E}_n^i := \Omega_n^{i-1} \cup \Omega_n^i \cup \Omega_n^{i+1}$.

Construction 3.2.6: We start with ω_1 as in Convention 3.2.1. Our goal is to construct a valid c-path, such that ω_z is a zebra for some $z \in \mathbb{N}$. The algorithm goes as follows:

(i) If $\mathbb{P}_n = 0$, ω_n is a zebra. Jump to step (vi). If not, there exists $p := \min\{i \in \{1, ..., s\} \mid \mathbb{P}_n^i > 0\}$. Note that, by definition, there are no problems to the left of the cell T_p.
(ii) $\mathbb{P}_n^p > 0$ means that there is too much selected vertices in the cell p. Thus, we pick some $r \in \Omega_n^p$ to be removed from ω_{n+1}.
(iii) Chose some $x \in T_{p-1} : \{x, r\} \notin E$. By construction, there exists at least one such $x \in T_{p-1}$.
(iv) Set $\omega_{n+1} := \{x\} \cup \omega_n \setminus \mathcal{E}_n^{p-1} \cup (lk_\Gamma(x) \cap \mathcal{E}_n^{p-1})$. Lemma 3.2.8 shows that, indeed, $(\omega_{n+1}, \omega_n) \in \mathcal{M}(\Gamma)$.
(v) Jump to step (i).
(vi) Set $z := n$ and RETURN.

Remark 3.2.7: Note that if p is even, then $\mathcal{E}_n^{p-1} = \Omega_n^{p-2} \cup \Omega_n^p$ since there is (by the choice of p) no problem to the left of the cell p, which means that all the odd cells (including $p-1$) are not subsets of ω_n.

The same argument applies to p is odd $\implies \mathcal{E}_n^{p-1} = \Omega_n^{p-1} \cup \Omega_n^p$.

Lemma 3.2.8: $\forall n \in \mathbb{N} : \omega_{n+1}$ is a clique and $(\omega_{n+1}, \omega_n) \in \mathcal{M}(\Gamma)$ for ω_n, ω_{n+1} from the step (iv) of the algorithm described in Construction 3.2.6.

Proof: Firstly, we need to show that $\omega_{n+1} \in \mathcal{C}(\Gamma)$. Let $v, u \in \omega_{n+1}$ be an arbitrary pair of vertices.

1. $v = x \wedge u \in lk_\Gamma(x) \cap \mathcal{E}_n^{p-1} \implies \{v, u\} \in E$ by Definition 1.2
2. $v = x \wedge u \in \omega_n \setminus \mathcal{E}_n^{p-1} \implies \exists j \in I : u \in T_j \wedge |(p-1) - j| > 1 \implies \{v, u\} \in E$ by Lemma 2.2.5
3. $v \in lk_\Gamma(x) \cap \mathcal{E}_n^{p-1} \wedge \omega_n \setminus \mathcal{E}_n^{p-1} \implies v, u \in \omega_n \implies \{v, u\} \in E$ because $\omega_n \in \mathcal{C}(\Gamma)$.

Now we show that

$$\omega_n \setminus \omega_{n+1} \cap \left(\bigcap_{v \in \omega_{n+1}} lk_\Gamma(v) \right) = \emptyset. \tag{12}$$

By construction, we have $\omega_n \setminus \omega_{n+1} = \mathcal{E}_n^{p-1} \setminus lk_\Gamma(x) = \mathcal{E}_n^{p-1} \cap (lk_\Gamma(x))^C$. That implies that (12) is equivalent to

$$\mathcal{E}_n^{p-1} \cap (lk_\Gamma(x))^C \cap \left(\bigcap_{v \in \omega_{n+1}} lk_\Gamma(v) \right) = \emptyset. \tag{13}$$

Indeed, $x \in \omega_{n+1} \Longrightarrow \left(\bigcap_{v \in \omega_{n+1}} lk_\Gamma(v) \right) \subset lk_\Gamma(x)$, thus $(lk_\Gamma(x))^C \cap lk_\Gamma(x) = \emptyset$ implies (12). ■

Theorem 3.2.9: The algorithm from Construction 3.2.6 always returns.

Proof: The only selections Ω_n^i that are changed in each iteration n of the algorithm are $\Omega_n^{p-2}, \Omega_n^{p-1}$ and Ω_n^p, whereby $p = \min\{i \in I \mid \mathbb{P}_n^i > 0\}$. So, we can see how all the possible changes of their cardinalities (which we denote by $\Delta|\Omega_n^i| := |\Omega_{n+1}^i| - |\Omega_n^i|$) change, which affects $\mathbb{P}_n$:

p is odd:

$\Delta\lvert\Omega_n^{p-2}\rvert$	$\Delta\lvert\Omega_n^{p-1}\rvert$	$\Delta\lvert\Omega_n^{p}\rvert$	$\min\{\text{i in I} \mid \mathbb{P}_{n+1}^i > 0\}$	$\Delta\mathbb{P}_n$
0	0	≤ -1	$\geq p$	≤ -1
0	1	≤ -1	$p-1$	≤ 0

p is even:

$\Delta\lvert\Omega_n^{p-2}\rvert$	$\Delta\lvert\Omega_n^{p-1}\rvert$	$\Delta\lvert\Omega_n^{p}\rvert$	$\min\{\text{i in I} \mid \mathbb{P}_{n+1}^i > 0\}$	$\Delta\mathbb{P}_n$
0	1	≤ -1	$p-1$	≤ 0
-1	1	≤ -1	$p-1$	≤ -1

That shows that the problem does not increase, but because there is a finite amount of cells, it is impossible to only move to the left. That means that the case described in the first row of the *odd* table will always be reached decreasing $\mathbb{P}_n$. This means that there exists a $z \in \mathbb{N} : \mathbb{P}_z = 0$. ■

Remark 3.2.10: There is one case (the last row of the *even* table) when p is even and, for $\{l\} = \Omega_n^{p-2}$, $\{x, l\} \in E$ applies, which would lead to $\Omega_{n+1}^{p-2} = \emptyset$ violating the assumed zebra-ness of everything to the left of $\min\{i \in I \mid \mathbb{P}_{n+1}^i > 0\}$. But this is automatically resolved by the fact that, in the $(n+1)^{\text{th}}$ iteration, such a $u \in \Omega_{n+1}^{p-2}$ will be found, that $\{u, x\} \notin E$, thus, solving the issue, because ω_{n+2} would be equal to $\{u\} \cup \omega_{n+1} \setminus \{x\}$ and making $\min\{i \in I \mid \mathbb{P}_{n+2}^i > 0\} \geq \min\{i \in I \mid \mathbb{P}_{n+1}^i > 0\}$ (say, shifting the problem to the right) without increasing the problem size and repairing the zebra-ness of everything to the left of $\min\{i \in I \mid \mathbb{P}_{n+2}^i > 0\}$ in step $n+2$.

4. What's next?

The next goal is to use this concrete definition of the algorithm to try to optimize it in terms of the length of the c-path to understand the asymptotic behavior of the upper bound with respect to $|V|$ and $|E|$.