Autostabilizing Minimal Clique Decomposition with Byzantine Faults tolerance

Johanne Cohen, Laurence Pilard, Jonas Sénizergues

Autostabilizing Minimal Clique Decomposition with Byzantine Fault tolerance

Johanne Cohen[1], Laurence Pilard[2], and Jonas Sénizergues[3]

[1]LISN, Université Paris-Saclay & CNRS
[2]Li-PARaD, Université Versailles Saint-Quentin-en-Yvelines
[3]LaBRI, Université de Bordeaux

January 7, 2025

Abstract

In this paper, we tackle the problem of finding a minimal clique decomposition of the underlying graph of a distributed system, in the presence of transient and byzantines faults. We propose a distributed algorithm that performs such a decomposition of the underlying graph -minus byzantine nodes and nodes that are "too close" to them-. It does so in $O(\Delta n)$ rounds with high probability.

1 Introduction

The (Vertex) Clique cover problem is a well-known NP-complete problem among Karp's 21 in its minimization version [6]. It has, among other things, been used to approximate Vertex Cover [2].

Definition 1 (Clique cover)**.** *Consider a graph $G = (V, E)$, a* clique cover *$\mathcal{X} = \mathcal{X}_1, \mathcal{X}_2, \dots \mathcal{X}_\ell$ of G is such that:*
- ***clique:*** *$\mathcal{X}_i$ is a clique of G,*
- ***cover:*** *$\bigcup_{i \in [\![1,\ell]\!]} \mathcal{X}_i = V$.*

In centralized computing, it is equivalent to the problem of Clique Decomposition where you expect the cover to be also a partition of V. Any clique decomposition is a clique cover, and to find a clique decomposition it is enough to remove duplicate nodes from one clique until there is no such duplicate left. However, it is not the case in distributed computing, as

choosing which duplicate to remove needs coordination. In this chapter, we will study the Minimal Clique Decomposition problem, specified as follows.

Definition 2 ($\mathcal{MCD}$ Specification)**.** *Consider a graph $G = (V, E)$, a* minimal clique decomposition *$\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \dots \mathcal{X}_\ell\}$ of G is such that:*

- ***clique:*** *$\mathcal{X}_i$ is a clique of G,*
- ***partition:*** *$\mathcal{X}$ is a partition of V,*
- ***minimal:*** *For any i, j distinct in $[\![1, \ell]\!]$, $\mathcal{X}_i \cup \mathcal{X}_j$ is not a clique of G.*

It is equivalent to the Coloring problem as it is enough to replace the edge set by its complement to go from one problem to the other. Or so is the case when you consider centralized computing. In distributed systems where you search for the solution of a graph problem on the underlying graph of the said system, the connectivity directly affects the computations. Here, we deal with the problem of finding a minimal clique decomposition of the graph of communications under the state model, with the possible presence of Byzantine nodes.

2 State of the art

Searching for cliques already has a long history in distributed systems: building upon the work in parallel computing [1], Jennings and al. [5] give a distributed algorithm to find all maximal cliques in a graph, in the context of a message-passing model with message of size $O(\log n)$. It has been followed by many other works improving the performance of such a search (see [8, 9] for example).

Closer to Clique Decomposition, Ishii and Kakugawa [4] give in a self-stabilizing algorithm that operates in the state model under the unfair centralized daemon to compute multiple cliques of "maximal size" for each node under some constraints. In this work, there are agreement constraints between nodes, as if a node computes a given clique, every member of that clique must compute the same clique (among other cliques they may have computed), it is thus closer to the problem of finding a clique decomposition.

The first attempt to our knowledge to tackle the Minimal Clique Decomposition problem in a distributed self-stabilizing setting has been made by Delbot and al. [3]. Their algorithm operates under a fair distributed daemon in $O(n)$ rounds provided a spanning tree has been computed beforehand.

In this chapter, we follow these footsteps by proposing the first algorithm that tackles the Minimal Clique Decomposition problem while handling Byzantine faults.

3 Description of the algorithm

Algorithm 1 Minimal Clique Decomposition Algorithm ($\mathcal{MCD}$)

Variables:
$\Omega_v \subseteq V$: the set of nodes supposed to be the clique v belongs to

$\mathcal{N}_v \subseteq V$: the (closed) neighborhood made apparent for the neighbors to read

$\beta_v \in N(v) \cup \{\perp\}$: the current merge target for the clique leader v

Funtions:

$$min(A) \equiv \begin{cases} \text{the smallest value of a set A} & \text{if } A \neq \emptyset \\ \perp & \text{otherwise} \end{cases}$$

$leader(v) \equiv min(\Omega_v)$
$merge\_candidate(v) \equiv$
$$\left\{ u \in N(v) | merge\_ready(u) \wedge \Omega_u \cap \Omega_v = \emptyset \wedge \left(\Omega_v \cup \Omega_u \subseteq \bigcap_{x \in \Omega_v \cup \Omega_u} \mathcal{N}_x \right) \right\}$$

$choose(A)$ is an element of the non-empty set A taken uniformly at random.

$\Omega^0(v) \equiv \{v\}$
$\Omega^{k+1}(v) \equiv \bigcup_{x \in \Omega^k(v)} \Omega_x$
$\Omega^*(v) \equiv \bigcup_{i \geq 0} \Omega^i(v)$: the Ω-closure of node v

Predicates:

$$merge\_ready(v) \equiv (leader(v) = v) \wedge Stab(v) \wedge coherent\_clique(v)$$

$$Stab(v) \equiv \forall x \in \Omega_v, \Omega_x = \Omega_v$$

$$coherent\_local(v) \equiv \mathcal{N}_v = N(v) \cup \{v\} \wedge \{v\} \subset \Omega_v \subset \mathcal{N}_v \wedge \beta_v \in \{\perp\} \cup \mathcal{N}_v \setminus \Omega_v$$

$coherent\_clique(v) \equiv$
$$\Omega^*(v) \subseteq \mathcal{N}_v \quad \wedge |\{x \in \Omega^*(v) | \beta_x \neq \perp\}| \leq 1$$
$$\wedge \forall x \in \Omega^*(v), \begin{cases} \Omega^*(v) \subseteq \mathcal{N}_x \\ \{x\} \subseteq \Omega_x \subseteq \Omega_{leader(x)} = \Omega^*(x) \\ \forall y \in \Omega_x, \Omega_y \subseteq \Omega_x \vee \Omega_y = \Omega^*(x) \\ \beta_x \neq \perp \Rightarrow (Stab(x) \wedge leader(x) = x \wedge \beta_x \in \mathcal{N}_x \setminus \Omega_x) \\ \min \Omega^*(v) \in \left\{ leader(leader(x)), \beta_{leader(x)} \right\} \end{cases}$$

$$well\_defined(v) \equiv coherent\_local(v) \wedge coherent\_clique(v)$$

Abandonment ▷ priority 5
if $(\beta_v = u \neq \bot) \wedge (\beta_u \notin \{\bot, v\} \vee u \notin merge\_candidate(v))$ **then**
$\beta_v = \bot$
end if

Mariage ▷ priority 4
if $leader(v) = v \wedge Stab(v) \wedge (\beta_v = \bot) \wedge (\exists u \in merge\_candidate(v), \beta_u = v)$ **then**
$\beta_v := choose(\{u \in merge\_candidate(v) | \beta_u = v\})$
end if

Seduction ▷ priority 4
if $leader(v) = v \wedge Stab(v) \wedge (\beta_v = \bot) \wedge (\forall u \in merge\_candidate(v), \beta_u \neq v) \wedge (\exists u \in merge\_candidate(v), \beta_u = \bot)$ **then**
$\beta_v := choose(\{u \in merge\_candidate(v) | \beta_u = \bot\})$
end if

Merge lead ▷ priority 4
if $leader(v) = v \wedge Stab(v) \wedge \beta_v = u \neq \bot \wedge u \in merge\_candidate(v) \wedge \beta_u = v \wedge v < u$ **then**
$\Omega_v := \Omega_v \cup \Omega_u$
$\beta_v := \bot$
end if

Merge follow ▷ priority 4
if $leader(v) = v \wedge Stab(v) \wedge \beta_v = u \neq \bot \wedge leader(u) = u \wedge v \in \Omega_u \wedge \Omega^*(u) \subseteq \mathcal{N}_v \wedge coherent\_clique(u)$ **then**
$\Omega_v := \Omega_u$
$\beta_v := \bot$
end if

Update ▷ priority 2
if $leader(v) = u \wedge \Omega_v \subsetneq \Omega_u$ **then**
$\Omega_v := \Omega_u$
end if

Reset ▷ priority 1
if $\neg well\_defined(v)$ **then**
$\Omega_v := \{v\}$
$\mathcal{N}_v := N(v) \cup \{v\}$
$\beta_v := \bot$
end if

3.1 Local variables

Each node v has three variables $\mathcal{N}_v$, Ω_v and β_v:

- $\mathcal{N}_v$ represents its neighborhood,
- Ω_v corresponding to the clique it belongs to,
- β_v represents its current target for merging cliques if there is one. Else it has value $\perp$.

3.2 About the Ω-closure, Ω^*

The Ω-closure of a node v, $\Omega^*(v)$, is the clique that v will have in its variable Ω_v when the potential merging process currently going is finished. Note that a locally coherent node v (such that $coherent\_local(v)$ is true) can always see -and thus read variables of- every node of its Ω-closure, as if it was not the case $\Omega^*(v)$ would not be a clique containing v. From the local coherence $\mathcal{N}_v = N(v) \cup \{v\}$ the node v can read the variables of the nodes of Ω_v. Then v can check whether the nodes of $\Omega^2(v)$ (which we can compute since v can read variables of Ω_v) have their Ω-values included in $\mathcal{N}_v$. Then v can do the same for $\Omega^3(v)$ and so on. Until $\Omega^*(v) \subseteq \mathcal{N}_v$ is proven false or v stops seeing new nodes, in which case we do have that $\Omega^*_v \subseteq \mathcal{N}_v$.

3.3 How to merge two cliques

Each clique has a distinguished node called the leader corresponding to the one with the smallest identifier. Only the clique leader can decide with which clique it will merge. The leader v of a clique starts looking for a merge target as soon as all nodes in its clique have the same view: the predicate $Stab(v)$ is satisfied ($\forall x \in \Omega_v, \Omega_x = \Omega_v$). When it is the case, v seeks a suitable clique leader u to merge clique with. In order for v to merge with u, the local variables of all nodes in Ω_u must have the following properties:

- All nodes of this clique have the same view of this clique: the predicate $Stab(u)$ is satisfied (*i.e.* $\forall x \in \Omega_u, \Omega_x = \Omega_u$).
- All nodes of this clique and the clique of v form a clique in the graph: $\Omega_v \cup \Omega_u \subseteq \bigcap_{x \in \Omega_v \cup \Omega_u} \mathcal{N}_x$.
- The clique is not merging with another clique : $\beta_u = \perp$.

If all these properties are satisfied, v can propose clique merging. To do so, it changes the value of β_v to u, the leader of one of the cliques suitable for merging with, chosen at random. Then, u answers or does not respond positively to this proposal. Of course, it checks that the first two conditions above are verified too. Note that it is basically applying locally the algorithm proposed by Kunne and al. [7] on the well-formed nodes, hence the borrowed names for the rules dedicated to this: **Seduction**, **Marriage** and **Abandonment**. Once the two leaders agree to merge ($\beta_v = u$ and $\beta_u = v$) the merging process begins. The one among u and v with the smallest identifier changes its variable Ω value to the union of the two cliques, and sets its variable β to $\perp$ by executing rule **Merge lead**. Then the other one updates its two local variables by executing rule **Merge follow**. All the other nodes refresh their variables by executing rule **Update** to complete the merging process.

3.4 How to handle errors

Now we describe how the algorithm handles errors in the local variables and avoids creating new ones when cliques merge. This part of the explanation of the algorithm is the most technical.

Each time a node v is activated, it checks whether it detects any inconsistencies in its local variables (for example the local variable $\mathcal{N}_v$ must correspond to its closed neighborhood.) or those of its current clique nodes (for example the clique must be included in the displayed neighborhood $\mathcal{N}$ of every node of the clique). If it detects any inconsistency that way, it executes the rule **Reset**.

The predicate *coherent_local* allows a node v to detect that its local variables are well initialized : $\mathcal{N}_v = N(v) \cup \{v\}$ and $\{v\} \subseteq \Omega_v \subseteq \mathcal{N}_v$. Since variable β_v designates the clique's leader with whom the clique Ω_v must merge, β_v must not be in Ω_v. Note that *coherent_local*(v) can only be computed by v itself since it needs to know $N(v)$.

The predicate *coherent_clique* allows a node to check that the state of the (future) clique of a node v is coherent (assuming local coherence for all the nodes involved). Note that a locally coherent node u may only evaluate *coherent_clique*(v) when $\Omega^*(v) \subseteq \mathcal{N}_u$. As we have said above, it is always the case when $u = v$. It will otherwise only be evaluated when u considers v as a potential target to merge their cliques. In such a case, we will have checked that $\Omega^*(v) \subseteq \mathcal{N}_u$ beforehand, and there will be no problem. We structure the explanation of *coherent_clique*(v) by following the structure of the predicate for better intelligibility:

- $\Omega^*(v) \subseteq \mathcal{N}_v$: As $\Omega^*(v)$ is supposed to be either the future clique of v (or current if there is no merging in progress), this condition is necessary for $\Omega^*(v)$ to be a clique containing v.

- $|\{x \in \Omega^*(v) | \beta_x \neq \bot\}| \leq 1$: If multiple β-values were non-$\bot$ in $\Omega^*(v)$, it would mean that multiple merging processes are taking place at the same time. As the algorithm waits for a clique to have finished merging before making it merge again, we do not want that.

- Then v checks all nodes x of its Ω-closure for potential inconsistencies:

  - $\forall x \in \Omega^*(v), \Omega^*(v) \subseteq \mathcal{N}_x$: $\Omega^*(v)$ is supposed to be a clique, thus every node of $\Omega^*(v)$ must have every other node of $\Omega^*(v)$ in its neighborhood.
  - $\forall x \in \Omega^*(v), \{x\} \subseteq \Omega_x \subseteq \Omega_{leader(x)} = \Omega^*(x)$: As leaders are those that move first when merging, we have $\Omega_x \subseteq \Omega_{leader(x)}$. $\{x\} \subseteq \Omega_x$ is just a part of the local coherence of x that happens to be checkable by neighbors. As $x \in \Omega^*(v)$, we have $\Omega^*(x) \subseteq \Omega^*(v)$, and thus if a node can see every node of $\Omega^*(v)$, it can see every node of $\Omega^*(x)$. Observe that $\Omega^*(x)$ may be different from $\Omega^*(v)$ in the case of an x that is part of a clique whose leader has yet to execute **Merge follow**. Finally, there are two cases for the leader of x:

    * It has already the final value as Ω-value (it has already executed **Merge lead** or **Merge follow**) and in this case $\Omega^*(x) = \Omega^*(v)$.
    * It is waiting to execute **Merge follow** and thus still has its old clique value.

    In both cases $\Omega_{leader(x)} = \Omega^*(x)$.
  - $\forall x \in \Omega^*(v), \forall y \in \Omega_x, \Omega_y \subseteq \Omega_x \vee \Omega_y = \Omega^*(x)$: When a clique merging begins, every node in one of the old cliques have only two possible values: the old, and the new one. The old being included in the new one. Note that either $\Omega^*(x) = \Omega^*(v)$, or we are in a case where the leader of x is waiting to execute **Merge follow**.
  - $\forall x \in \Omega^*(v), \beta_x \neq \bot \Rightarrow (Stab(x) \wedge leader(x) = x \wedge \beta_x \in \mathcal{N}_x \setminus \Omega_x)$: Only a leader may have a non-$\bot$ β-value, and it should be in a valid shape to have such a value. It means that x should be its own leader, having every node in its clique having the same Ω-value, and having a merge target β_x that is not already in its

clique. $\beta_x \neq \perp$ can either happen for $x = \min \Omega^*(v)$ if the clique of v is ready to merge, or to another node that would be the leader of the clique $\min \Omega^*(v)$ is merging with (which has yet to execute **Merge follow**).

- $\forall x \in \Omega^*(v), \min \Omega^*(v) \in \{leader(leader(x)), \beta_{leader(x)}\}$: $\min \Omega^*(v)$ will be the leader of the future clique of v, $\Omega^*(v)$, when every node in it has updated its Ω-value. The case $\min \Omega^*(v) = leader(leader(x))$ corresponds to two types of situations. The first one is when either x has been part of the clique that started the merging process with **Merge lead** (it was already true before the merging in this case). The second is when x has been part of the the other clique involved in the merging and the leader of that second clique has already performed **Merge follow**. The case $\min \Omega^*(v) = \beta_{leader(x)}$ corresponds to situations where $leader(x)$ is the leader of that other clique, but has yet to perform **Merge follow**. If $\min \Omega^*(v)$ is equal to neither of those two options, it means that the clique x belongs to is not really aware that it should be merging. It may happen as a result of errors in the starting configuration.

The predicate *coherent_clique* will be used to check for local coherence in the predicate $well\_defined$, and also to avoid merging with nodes that can be detected as not well defined (because *coherent_clique* is false on them).

4 Convergence

To prove the convergence of our algorithm, we present our reasoning in five steps.

First, in Subsection 4.1, we observe that after at most 1 round the variable $\mathcal{N}$ of non-Byzantine nodes contains the closed neighborhood of the node (Lemma 4.1), and it cannot change afterward (Lemma 4.2). As such we can, without loss of generality, only consider $\mathcal{N}$-stabilized configurations in the remaining of the proof.

In Subsection 4.2, we discuss the properties of well-definedness (corresponding to the $well\_defined$ predicate). We use this to define V_1'' (Definition 4) a superset of V_1 (recall that V_1 is the set of the nodes that have no Byzantine neighbors) on which well-definedness is guaranteed (Lemma 4.4). We prove the convergence of our algorithm on V_1'', as it would not be possible on V_1. on which we can hope to define a convergence property for our algorithm.

In Subsection 4.3, we focus on the merging of two cliques. We prove that when a merging between two nodes of V_1'' has been started, they do not interact with nodes outside their cliques while the process is not finished (Lemma 4.8), and it ends after a few rounds (Lemmas 4.10 and 4.12). It allows us to focus on the events that lead nodes to begin such a merging.

In Subsection 4.4, we focus on the progression of the algorithm. To do so we define a notion corresponding to the "current" state of the clique decomposition (Definition 5). Using this, we then prove a succession of lemmas that draw a pattern by which the decomposition progresses probabilistically (summarized in Figure 1).

Finally, in Subsection 4.5 using a concentration inequality, we deduce that our algorithm converges, and ends within $O(\Delta n)$ rounds with high probability (Theorem 4.23).

4.1 Neighborhood stabilization

Definition 3. *We say that a configuration is $\mathcal{N}$-stabilized when every non-Byzantine node has its $\mathcal{N}$-value equal to its actual closed neighborhood (i.e $\forall x \in V_0, \mathcal{N}_x = N(x) \cup \{x\}$).*

It is a condition needed for the rules to behave as intended, and as such we need to know when we can ensure that the condition is met.

Lemma 4.1. *Let γ be a configuration. The configuration γ' reached after one round from γ is $\mathcal{N}$-stabilized.*

Proof. **Reset** is the highest priority rule, and it is enabled on any node that does not have its closed neighborhood as $\mathcal{N}$-value. After it has been executed on a non-Byzantine node, this node will have its closed neighborhood as $\mathcal{N}$-value, and it is the only rule that may change a $\mathcal{N}$-value.

Then, after at most one round, every node that didn't have its closed neighborhood as $\mathcal{N}$-value has then been activated and performed **Reset**. As no node having its closed neighborhood as $\mathcal{N}$-values may change that property, it follows that γ' is $\mathcal{N}$-stabilized. □

Then, it is easy to see that a $\mathcal{N}$-stabilized configuration will stay $\mathcal{N}$-stabilized across a transition.

Lemma 4.2. *Let γ be a $\mathcal{N}$-stabilized configuration, and $\gamma \to \gamma'$ be a transition. Then γ' is $\mathcal{N}$-stabilized.*

Proof. No rule may change the $\mathcal{N}$-value of a node that has its $\mathcal{N}$-value match its actual closed neighborhood. □

From any configuration, after one round a $\mathcal{N}$-stabilized configuration is reached, and after that, in the execution, every transition will be $\mathcal{N}$-stabilized. Using this fact, in most of the lemmas, we will suppose that we start directly in a $\mathcal{N}$-stabilized configuration without loss of generality.

4.2 Well-definedness

The $well\_defined$ predicate expresses a bunch of "good" properties that we would like to be true, in the sense that it would be always true if we started from a clean starting configuration without Byzantine nodes and where every node u would be such that $\Omega_u = \{u\}$, $\mathcal{N}_u = N(u) \cup \{u\}$ and $\beta_u = \bot$. As it is not the case, we cannot hope for it to be true everywhere and every time. But we can try to understand when it's the case.

First, we note that the $well\_defined$ property is inherited by nodes that are in the Ω-closure of a $well\_defined$ node.

Lemma 4.3. *Let γ be a $\mathcal{N}$-stabilized configuration and u, v two nodes such that $u \in \Omega^*(v)$. In γ, if v is well-defined, u is well-defined too.*

Proof. Let u and v be two nodes such that $u \in \Omega^*(v)$ and suppose that v is well-defined. Since v is supposed well-defined, we have either $\Omega^*(u) = \Omega^*(v)$ or $\beta_{leader(u)} = leader(v)$. Since γ is supposed $\mathcal{N}$-stabilized we have $\mathcal{N}_u = N(u) \cup \{u\}$, and $coherent\_clique(v)$ implies that $\{u\} \subseteq \mathcal{N}_u \subseteq \Omega^*(u)$ and $\Omega^*(v) \subseteq \mathcal{N}_u$. Thus $\{u\} \subseteq \Omega_u \subseteq \mathcal{N}_u$. It also implies that $\beta_u \in \mathcal{N}_u \setminus \Omega_u$. Thus, $coherent\_local(u)$ is true.

Let's then prove $coherent\_clique(u)$:

- Suppose $\Omega^*(u) = \Omega^*(v)$. As $coherent\_clique(v)$ is true, it only remains to prove $\Omega^*(u) \subseteq \mathcal{N}_u$, $\Omega_{leader(u)} = \Omega^*(u)$ and $|\{x \in \Omega^*(u) | \beta_x \neq \perp\}| \leq 1$. Using the hypothesis $\Omega^*(u) = \Omega^*(v)$ and the fact that $u \in \Omega^*(v)$, they are direct consequences of $coherent\_clique(v)$.

- Suppose now that $\Omega^*(u) \neq \Omega^*(v)$. It implies $\Omega^*(u) \subsetneq \Omega^*(v)$. Then, using the well-definedness of v we get:

  - $\Omega^*(u) \subsetneq \Omega^*(v) \subseteq \mathcal{N}_u$;
  - Since $\beta(u) \neq \perp$, from the fact that $|\{x \in \Omega^*(v) | \beta_x \neq \perp\}| \leq 1$, u is the only node in $\Omega^*(v)$ to have a non-$\perp$ value of β. Thus, as $\Omega^*(u) \subsetneq \Omega^*(v)$, we get $|\{x \in \Omega^*(u) | \beta_x \neq \perp\}| = 1 \leq 1$;
  - The first four properties stated in the quantified part does not depend on v, thus they are still true for every node of $\Omega^*(u) \subsetneq \Omega^*(v)$. The fifth one comes from the fact that as $\beta(u) \neq \perp$, we have $Stab(u)$ and $leader(u) = u$, thus $leader(leader(.))$ has value $u = min\Omega^*(u)$ for every node of $\Omega^*(u) = \Omega_u$.

  Thus, $coherent\_clique(u)$ is true.

Thus, in every case, u is well-defined. □

To contain the influence of Byzantine nodes, we must identify on which space we want our algorithm to converge. V_0 is out of the picture as a node neighbor to two Byzantine nodes could be fooled by those pretending to be neighbors of each other. The next set to naturally consider is V_1, but as nodes of V_1 may legitimately have some nodes of V_0 in their Ω-value it is not possible to take exactly V_1. We have to widen a bit V_1 in order to include

those legitimately included V_0 nodes. As we just proved that $well\_defined(v)$ implies that all the nodes of $\Omega^*(v)$ are well-defined as well, we might want to consider $V_1' = \Omega^*(V_1)$.

As it is easier for us to handle well-defined nodes, we define the following:

Definition 4. $V_1'' = \Omega^*(\{u \in V_1 | well\_defined(u)\})$.

We will later on prove that after some time it must contain V_1. Do note that contrary to V_1, V_1'' depends on the variables' values of a configuration. We write $V_1''(\gamma)$ for "V_1'' in configuration γ" when there could be some ambiguity.

As we constructed it with this in mind, let's prove that every node of V_1'' is indeed well-defined.

Lemma 4.4. *Let γ be a $\mathcal{N}$-stabilized configuration. Every node of V_1'' is well-defined.*

Proof. Let u be a node of V_1''. By definition, it must be in $\Omega^*(v)$ for some well-defined $v \in V_1$. Then by Lemma 4.3, as v is well-defined, u must also be well-defined. □

To be a concept of use to express the convergence of an algorithm, we need V_1'' to be non-decreasing.

Lemma 4.5. *Let γ be a $\mathcal{N}$-stabilized configuration, and $\gamma \xrightarrow{t} \gamma'$ a transition. Consider $u \in V$, well-defined in γ, such that $leader(u) = u$ and Ω_u does not contain Byzantine nodes.*

1. *If $\neg Stab(u)$, u is well-defined in γ';*
2. *Else, if u does not perform a move in the transition, u is well-defined in γ';*
3. *Else, if $\beta_u = \perp$, u is well-defined in γ';*
4. *Else, if u performs **Abandonment** in the transition, u is well-defined in γ';*
5. *Else, if Ω_{β_u} does not contain Byzantine nodes, u is well-defined in γ'.*

Proof. Let γ be a $\mathcal{N}$-stabilized configuration, and $\gamma \xrightarrow{t} \gamma'$ a transition. Consider $u \in V$ such that u is well-defined in γ, $leader(u) = u$ and Ω_u does not contain Byzantine nodes. Let us first prove Point 1.

Proof of Point 1. Suppose $\neg Stab(u)$. With the well-definedness of u, it implies that none of **Mariage**, **Seduction**, **Abandonment** or **Merge lead** is enabled on any node of $\Omega^*(u) = \Omega_u$ in γ (recall that no node of Ω_u^γ is Byzantine by hypothesis).

Using one of these rules, the Ω-value of a node may only grow by setting it to the value of its current leader (or to the target of the merge in progress in the case of the only potential node being enabled for the rule **Merge follow**) which is already a subset of $\Omega^*(u)$ in γ. As the value of Ω_u does not change in the transition since u is its own leader, the value of $\Omega^*(u)$ does not change in the transition.

Then, let's prove what is needed for *coherent_clique* to be true in γ' (we cut the proof according to the structure of the predicate to make the reading easier):

- As $\Omega^*(u)$ does not change in the transition $\Omega^*(u) \subseteq \mathcal{N}_u$ is still true in γ'.
- As $|\{x \in \Omega^*(v) | \beta_x \neq \bot\}| \leq 1$ and no rule that may give a non-$\bot$ value of β is activable on any node of $\Omega^*(u)$ in γ, the same can be said in γ'.
- Let $x \in \Omega^*(u)$,
  - Neither $\Omega^*(u)$ nor $\mathcal{N}_x$ may change in the transition, thus as it was true in γ from well-definedness, $\Omega^*(u) \subseteq \mathcal{N}_x$ is still true in γ'.
  - As the Ω-value of a well-defined node may only change by taking the current Ω-value of its leader, $\{x\} \subseteq \Omega_x \subseteq \Omega_{leader(x)} = \Omega^*(x)$ in γ implies the same in γ'.
  - Let y be a node of Ω_x^γ, we have in γ that $\Omega_y \subseteq \Omega_x \vee \Omega_y = \Omega^*(x)$ in γ by well-definedness. If none of x and y are activated in the transition, or if $x = y$, there is nothing to prove, suppose then that they are distinct and at least one of them is activated.
    - Suppose $\Omega_y \subseteq \Omega_x$ in γ. If only x was activated this remains true in γ', and there is nothing to prove. We then suppose y is activated in the transition.
      - If $\Omega^*(x) = \Omega^*(y)$ in γ, y must have performed **Update** in the transition (as $\Omega_y \subsetneq \Omega_x$ implies $\neg Stab(y)$). As well-definedness implies that $\Omega_x = \Omega^*(x)$ and $\Omega_{leader(y)} = \Omega^*(x) = \Omega_x$ in γ, we have $\Omega_y^{\gamma'} = \Omega_x^\gamma \subseteq \Omega_x^{\gamma'}$.
      - Else, we have $\Omega^*(y) \subsetneq \Omega^*(x) \subseteq \Omega^*(v)$. Well-definedness implies $\min(\Omega^*(v)) \in \left\{leader(leader(y)), \beta_{leader(y)}\right\}$ in γ.

The previous inequality implies that $leader(leader(y)) \neq \min(\Omega^*(v))$ in γ, thus we have $\beta_{leader(y)} = \min(\Omega^*(v))$ in γ. Well-definedness implies $Stab(leader(y))$, and thus since we supposed that y is activated in the transition we must have $y = leader(y)$, and y performed **Merge follow** in the transition. We can also deduce that $\Omega^*(x) = \Omega^*(v)$ from the fact that it would otherwise imply that $leader(x) \neq y$ has a non-$\bot$ value of β which would contradict well-definedness. Thus, in γ', $\Omega_y = \Omega_x = \Omega^*(v)$.

  - Else, we have $\Omega_y = \Omega^*(x)$. If $\Omega^*(x) = \Omega^*(v)$, there is nothing to prove as Ω-values may only grow in the transition and $\Omega^*(v)$ does not change. Suppose then that $\Omega^*(x) \neq \Omega^*(v)$, *i.e.* $\Omega^*(x) \subsetneq \Omega^*(v)$. This implies by well-definedness that $\beta_{leader(x)} = \min(\Omega^*(v))$, and $Stab(leader(x))$. Then, only $leader(x)$ is activable (for the **Merge follow** rule) among the nodes of $\Omega_{leader(x)}$. As x and y are in $\Omega_{leader(x)} = \Omega^*(x)$, one of them must be $leader(x)$, and the other one is not activated in the transition. Then, if y performs **Merge follow** in the transition, we have $\Omega_y = \Omega^*(v) = \Omega^*(x)$ in γ'. Else, x performs **Merge follow** in the transition, and $\Omega_y \subseteq \Omega^*(v) = \Omega_x = \Omega^*(x)$ in γ'.

- We have $\beta_x \neq \bot \Rightarrow (Stab(x) \wedge leader(x) = x \wedge \beta_x \in \mathcal{N}_x \setminus \Omega_x)$ in γ. If $\beta_x^{\gamma'} = \bot$ there is nothing to prove. Let's then focus on the other case: $\beta_x \neq \bot$ in γ'. In this case we have $\beta_x \neq \bot$ in γ too as no rule that could give a non-$\bot$ β-value is enabled on $\Omega^*(v)$ in γ. Thus, by well-definedness, $Stab(x)$ and $leader(x) = x$ in γ and only x may be activable among nodes of Ω_x in γ, for the rule **Merge follow**. As $\beta_x \neq \bot$, it executes **Merge follow** in the transition, and no node of Ω_x was activated. Thus $Stab(x)$ and $leader(x) = x$ are still true in γ', and as no variable of x changes value, $\beta_x \in \mathcal{N}_x \setminus \Omega_x$ in γ' too.
- By well-definedness, $\min(\Omega^*(v)) \in \{leader(leader(x)), \beta_{leader(x)}\}$ in γ. As $\Omega^*(v)$ does not change in the transition, so does its minimum $\min(\Omega^*(v))$. As Ω-values may only grow in the transition, if $leader(leader(x)) = \min(\Omega^*(v))$ in γ, the same is also true in γ'. Suppose now that $leader(leader(x)) \neq \min(\Omega^*(v))$ in γ, which means that $\beta_{leader(x)} = \min(\Omega^*(v))$ in γ. Using well-definedness we have $Stab(leader(x))$ and $leader(leader(x)) = leader(x) \neq \min(\Omega^*(v))$. If $leader(x)$ is not activated in the transition there

is nothing to prove. Else, it performs **Merge follow** and we have $\Omega^{\gamma'}_{leader(x)} = \Omega^{\gamma}_{\min(\Omega^*(v))} = \Omega^*(v)$. Thus, in γ', $leader(leader(x)) = \min(\Omega^*(v))$.

Thus, when $\neg Stab(u)$ in γ, u is well-defined in γ', which proves Point 1. □

Now suppose $Stab(u)$ in γ. As u is well-defined and there is no Byzantine node in Ω_u by hypothesis, no rule is enabled on $\Omega_u \setminus \{u\}$.

Proof of Point 2. If u does not execute any rule in the transition, no node of $\Omega_u = \Omega^*(u)$ does, and u is well-defined in γ' which proves Point 2. □

Suppose then that u executes a rule in the transition.

Proof of Point 3. Suppose $\beta^{\gamma}_u = \perp$. As u is well-defined, the only rules it can perform in the transition are **Seduction** and **Mariage**. From the guards of those rules $\beta^{\gamma'}_{leader(u)}$ is a node in $merge\_candidate(leader(u))$ in γ, which ensures that $\beta^{\gamma'}_{leader(u)} \notin \Omega^{\gamma}_{leader(u)} = \Omega^{\gamma'}_{leader(u)}$. The guards of both rules imply that $Stab(u)$ is true in γ. This gives by well-definedness that no node of Ω^{γ}_u had β-value non-$\perp$ in γ, and that in Ω^{γ}_u only u is activable in γ. As no Ω-value in Ω^{γ}_u changes in the transition, we have that u is well-defined in γ'. □

Suppose now that $\beta^{\gamma}_u \neq \perp$, u must have performed either **Abandonment**, **Merge lead**, or **Merge follow**.

Proof of Point 4. Suppose u performed **Abandonment** in the transition. As only u was activable in γ among nodes of Ω^{γ}_u, it's easy to see that u is still well-defined in γ' which proves Point 4. □

Suppose now that u did not perform **Abandonment** in the transition. It means that either **Merge follow** or **Merge lead** has been executed by u.

Proof of Point 5. Let's write $v = \beta^{\gamma}_u$, and suppose Ω_v does not contain Byzantine nodes. Then v is well-defined in γ, as $v \in merge\_candidate(u)$ (from the guard of both possible rules) and γ is $\mathcal{N}$-stabilized (by hypothesis).

- Suppose **Merge follow** was executed by u in the transition. Then we may apply Point 1 of the Lemma to v. Hence v is well-defined in γ'. From the guard of **Merge follow** we have $u \in \Omega^{\gamma}_v$, thus $u \in \Omega^{\gamma'}_v$ as v does not execute **Reset** in the transition. Then, applying Lemma 4.3, we get that u is well-defined.

- Suppose **Merge lead** was executed by u in the transition. We know from the guard of the rule that $v = \beta_u^\gamma$ is in $merge\_candidate(u)$ in γ, which implies $leader(v) = v$, $Stab(v)$, and $coherent\_neighborhood(v)$. Since the configuration is supposed $\mathcal{N}$-stabilized, v is then well-defined in γ. By hypothesis on u that there is no Byzantine node in Ω_v^γ. Since $Stab(v)$ is true, no node of $\Omega_v^\gamma \setminus \{v\}$ is activable in γ. Moreover, the guard of **Merge lead** also implies that $leader(u) < v$ and $\beta_v^\gamma = u$, which implies that v is not activable in γ. Thus u is the only node of $\Omega_{leader(u)}^\gamma \cup \Omega_v^\gamma$ that has been activated in the transition, and no other node of that set had the values of its variables changed. As $\Omega_u^{\gamma'} = \Omega_u^\gamma \cup \Omega_v^\gamma$ from **Merge lead** command, we get $\Omega^*(u)^{\gamma'} = \Omega_u^{\gamma'} = \Omega_u^\gamma \cup \Omega_v^\gamma$. Let's check that $coherent\_clique(u)$ is true in γ' (again we cut the proof according to the structure of the predicate to make the reading easier):

  - As only u was executed in the transition, we have $\Omega^*(u)^{\gamma'} = \Omega_u^{\gamma'} = \Omega_u^\gamma \cup \Omega_v^\gamma$. From the guard of **Merge lead**, $v \in merge\_candidate(u)$ in γ, hence $\Omega^*(u)^{\gamma'} = \Omega_u^\gamma \cup \Omega_v^\gamma \subseteq \mathcal{N}_u$
  - As $\beta_u = v$ and $\beta_v = u$ in γ from the guard of **Merge lead**. By hypothesis u and v are well-defined in γ, thus u and v are the only nodes in $\Omega_u^\gamma \cup \Omega_v^\gamma$ to have non-$\perp$ β-value. Hence we have, in γ', $\{x \in \Omega^*(u) | \beta_x \neq \perp\} = \{v\}$, of size 1.
  - Let $x \in \Omega^*(u)^{\gamma'}$,
    - $\Omega^*(u) \subseteq \mathcal{N}_x$ in γ' is a consequence of $v \in merge\_candidate(u)$ in γ.
    - As $Stab(u)$ and $Stab(v)$ in γ, we know that $leader(x)^\gamma$ is either u or v. If $x = u$ there is nothing to prove. Else x was not activated in the transition, and thus $\{x\} \subseteq \Omega_x \subseteq \Omega_{leader(x)}$ as the Ω-value of its leader may only have grown. Moreover, we still have $\Omega_v^{\gamma'} = \Omega^*(v)^{\gamma'}$ as no node of $\Omega_v^{\gamma'}$ is activated in the transition, and we've already seen that $\Omega_u^{\gamma'} = \Omega^*(u)^{\gamma'}$, thus $\Omega_{leader(x)} = \Omega^*(x)$.
    - Let y be a node of $\Omega_x^{\gamma'}$. If $x \in \Omega_u^\gamma$, either $y = u$, and then $\Omega_y^{\gamma'} = \Omega^*(u)^{\gamma'} = \Omega^*(x)^{\gamma'}$, or $\Omega_y^{\gamma'} = \Omega_u^\gamma \subseteq \Omega_x^{\gamma'}$. Else, $x \in \Omega_v^\gamma$, and $\Omega_x^{\gamma'} = \Omega_y^{\gamma'}$ as neither node was activated in the transition.
    - If $x \neq v$, $\beta_x^{\gamma'} = \perp$ and there is nothing to prove. Else, $x = v$, and as no node of Ω_v^γ is activated in the transition we still have $Stab(v), leader(v) = v$ and $\beta_v^\gamma \in \mathcal{N}_v \setminus \Omega_v$ in γ' .

* If $x \in \Omega_u^\gamma$, we have $leader(x)^{\gamma'} = u$, thus $leader(leader(x)) = leader(u) = u$ in γ'. Else, $x \in \Omega_v^\gamma$, and then $leader(x)^{\gamma'} = v$ and thus $\beta_{leader(x)} = \beta_v = u$ in γ'.

Thus $coherent\_clique(v)$ is true in γ', and hence v is well-defined. Which proves Point 5. □

All five points have been proved, hence the result. □

Lemma 4.6. *Let γ be a $\mathcal{N}$-stabilized configuration, and $\gamma \xrightarrow{t} \gamma'$ a transition. Then $u \in V_1''(\gamma) \Rightarrow u \in V_1''(\gamma')$.*

Proof. As $u \in V_1''(\gamma)$, u is well-defined in γ from Lemma 4.4. By definition of V_1'', there is $x \in V_1$ well-defined in γ such that $u \in \Omega^*(x)$.

Let's write $u' = leader(leader(u))$. As $u' = leader(leader(u)) \in \Omega_{leader(u)}$ and $leader(u) \in \Omega_u$, we have $u' \in \Omega^*(x)$. Thus by definition of V_1'', $u' \in V_1''(\gamma)$ and u' is well-defined in γ from Lemma 4.4. As $\Omega_{leader(u)} = \Omega^*(u)^\gamma$ from well-definedness, we get $u' = leader(leader(u)) = \min \Omega^*(u)^\gamma$, and thus $leader(u') = u'$ in γ.

If $\neg Stab(u')$, u' does not perform a move in the transition, $\beta = \bot$, or u' executes **Abandonment**, we can apply Lemma 4.5 and thus u' is well-defined in γ'.

Suppose now that $Stab(u)$, $\beta_{u'}^\gamma \neq \bot$, and u performs a non-**Abandonment** move in the transition. By well-definedness we know that $\Omega_{u'}^\gamma = \Omega^*(u')^\gamma$

Let's write $v = \beta_{u'}^\gamma$. Given the conditions, the rule executed by u' in the transition is either **Merge lead** or **Merge follow**.

- Suppose it is **Merge follow**. As $u' \in \Omega_v^\gamma$ from the guard of **Merge follow**, $x \in \Omega^*(v)^\gamma$ by definition of Ω^*. Since we have $coherent\_clique(v)$ and $leader(v) = v$ in γ, we get $\Omega_v^\gamma = \Omega^*(v)^\gamma$. Thus, using again the fact that $coherent\_clique(v)$ is true in γ, we get $\Omega_v^\gamma = \Omega^*(v)^\gamma \subseteq \mathcal{N}_x$.

- Suppose now that it is **Merge lead**. The guard of the rule implies $v \in merge\_candidate(u')$, which leads to $\Omega_v \subseteq \mathcal{N}_x$.

In both cases, we have $\Omega_v \subseteq \mathcal{N}_x$. As $x \in V_1$ and the configuration is supposed $\mathcal{N}$-stabilized, it implies $\Omega_v^\gamma \subseteq V_0$ *i.e.* does not contain Byzantine nodes. We can then apply Lemma 4.5 and thus u' is well-defined in γ'.

Well-definedness of u' in γ gives $x \in \Omega^*(u')^\gamma$. As every node of $\Omega^*(x)^\gamma$ is well-defined by Lemma 4.4, no such node may have executed **Reset** in the transition, thus their Ω-value can only grow in the transition. Thus we have $x \in \Omega^*(u')^{\gamma'}$ which implies that x is well-defined in γ' from Lemma 4.3, and $u \in \Omega^*(x)$ which then gives $u \in V_1''(\gamma')$. □

There is still to prove that our V_1'' will contain V_1 at some point in the execution, as we advertised that the algorithm would converge on "at least" V_1.

Lemma 4.7. *Let γ be a $\mathcal{N}$-stabilized configuration. Every node of V_1 is well-defined in the configuration γ' reached after one round from γ.*

Proof. Suppose there are some nodes of V_1 that are not well-defined in γ. **Reset** is activable on them until they become well-defined or execute **Reset**, in which case they are well-defined in the configuration following this execution. By definition of a round one of them must happen before the round ends. Then from Lemma 4.6, those may not stop being well-defined, thus they are well-defined in γ'. □

4.3 Any merging process ends

On $\mathcal{N}$-stabilized configurations, the algorithm behaves on a large time scale as if the only existing nodes were the ones that are currently their own leaders with their neighborhood modified to be the intersection of the neighborhood of every node under their rule. Once such a node has performed a move to merge with another leader (using the **Merge lead** rule), their subjects simply follow. Here, we prove that when a merging has begun, a leader must wait for its followers before doing anything more, and that followers end up synchronized with their leader at some point.

To do this, we first prove that in the clique of such a leader node, the only rules that can be enabled are those that are designed to synchronize a follower to its leader, either **Update** (for those that were already followers before), or **Merge follow** (for the leader that is to become a follower after the merging is completed).

Lemma 4.8. *Let γ be a $\mathcal{N}$-stabilized configuration. If $u \in V_1''$, $leader(u) = u$, and $\neg Stab(u)$, only rules **Update** and **Merge follow** may be enabled on nodes of $\Omega^*(u)$ in γ.*

Proof. From Lemma 4.3 every node of $\Omega^*(u)$ is also well-defined. Consider then $v \in \Omega^*_u$.

- If $\neg Stab(v)$, only **Update** can be enabled on v.
- Else, we have $Stab(v)$. If $leader(v) \neq v$, v is not activable, and there is nothing to prove. Else, from well-definedness of u, we get $\min(\Omega^*(u)) = u \in \{leader(leader(v)), \beta_{leader(v)}\}$. As $leader(leader(v)) = v$ by hypothesis, we must have $\beta_v = u$, and then the only rule that can be enabled on v is **Merge follow**.

□

When the said **Update** or **Merge follow** are executed, we can observe that it leads to synchronization with the leader.

Lemma 4.9. *Let γ be a $\mathcal{N}$-stabilized configuration. Consider $u \in V_1''$ such that leader$(u) = u$ and $\neg Stab(u)$ in γ, and a transition $\gamma \xrightarrow{t} \gamma'$. We have:*

- $\Omega^*(u)^\gamma = \Omega^*(u)^{\gamma'}$;
- *If $v \in \Omega^*(u)^\gamma$ appears in a move of t, $\beta_v^{\gamma'} = \perp$ and $\Omega_v^{\gamma'} = \Omega^*(u)^\gamma$.*

Proof. Let's prove separately the two points of the lemma:

- Lemma 4.8 only allows **Update** and **Merge follow** to be enabled on nodes of $\Omega^*(u)^\gamma$ in γ. By well-definedness $\Omega^*(u)^\gamma = \Omega_u^\gamma$, and $\beta_u^\gamma = \perp$ thus no rule is enabled on u. Then **Update** and **Merge follow** may only change the Ω-value of nodes in Ω_u^γ to values that are included in Ω_u^γ in the transition. Thus $\Omega^*(u)^{\gamma'} = \Omega_u^\gamma = \Omega^*(u)^\gamma$.
- If $v \in \Omega^*(u)^\gamma$ appears in a move t, it means that it executed either **Update** and **Merge follow** in the transition. In both cases, the well-definedness of u implies that the new Ω-value of v must be $\Omega^*(u)^\gamma$.

□

To bound the time it takes for every node to synchronize with the new leader, we begin by removing **Merge follow** from the equation, which takes at most one round.

Lemma 4.10. *Let γ be a $\mathcal{N}$-stabilized configuration. Consider $u \in V_1''$ such that leader$(u) = u$, and $\neg Stab(u)$ in γ. Then after at most one round a configuration γ' is reached such that $\Omega^*(u)^{\gamma'} = \Omega^*(u)^\gamma$ and $\forall x \in \Omega^*(u)^{\gamma'}, \beta_x^{\gamma'} = \perp$.*

Proof. Using well-definedness, we know that at most one node of $\Omega^*(u)$ with β-value non-$\perp$ in γ. Remember that from Lemma 4.6 u will be well-defined in every future configuration.

If there is no such node, then γ already verifies the condition, and there is nothing to prove.

Suppose then such a node v exists, well-definedness gives that $\beta_v^\gamma = u$ and guarantees that **Merge follow** is enabled on it until this β-value changes. Thus, after at most one round, **Merge follow** is executed by this node. Consider the first transition when it happens $\gamma'' \to \gamma'$.

By hypothesis v does not execute **Merge follow** in any transition in $\gamma \rightarrow^* \gamma''$. Thus $\beta_v = u$ in every configuration between γ and γ''.

By immediate induction using our Lemma 4.9 and the fact that $\beta_v = u$, every configuration in $\gamma \rightarrow^* \gamma''$ is such that $leader(u) = u$ and $\neg Stab(u)$. We then have $\Omega^*(u)^{\gamma''} = \Omega^*(u)^{\gamma}$, and v is the only node of $\Omega^*(\gamma'')$ with β-value non-$\perp$.

Then using Lemma 4.9 we get in γ' that $\Omega^*(u)^{\gamma'} = \Omega^*(u)^{\gamma''}$ and since v is activated in the transition we have $\beta_v^{\gamma'} = \perp$. As the other nodes of $\Omega^*(u)^{\gamma'}$ had already β-value $\perp$ and no rule may have changed that in the transition by Lemma 4.8, we have $\forall x \in \Omega^*(u)^{\gamma'}, \beta_x^{\gamma'} = \perp$. □

Lemma 4.11. *Let γ be a $\mathcal{N}$-stabilized configuration. Consider $u \in V_1''$ such that $leader(u) = u$ and $\neg Stab(u)$ and $\forall x \in \Omega^*(u), \beta_x = \perp$ in γ.*

*Then $\forall x \in \Omega^*(u)$ either $\Omega_x = \Omega^*(u)$ or **Update** is enabled on x in γ.*

Proof. From Lemma 4.8 only **Update** and **Merge follow**. As the guard of **Merge follow** requires a non-$\perp$ β-value it cannot be enabled.

Consider $v \in \Omega^*(u)^{\gamma}$. The well-definedness of u gives $\min(\Omega^*(u)) \in \{leader(leader(v)), \beta_{leader(v)}\}$. As $u = \min(\Omega^*(v))$ and with the constraints on β-values, we get that $leader(leader(v)) = u$, thus $\Omega^*(v) = \Omega^*(u)$. Well-definedness gives also $\Omega_{leader(v)} = \Omega^*(u)$.

Then, either $\Omega_v = \Omega_{leader(v)} = \Omega^*(u)$ (and **Update** is not enabled on v), or $\Omega_v \neq \Omega_{leader(v)}$ and **Update** is enabled on v. □

Then, we prove that if only **Update** is enabled on the followers of a leader, after at most one round, a configuration where every follower is synchronized with the leader is reached.

Lemma 4.12. *Let γ be a $\mathcal{N}$-stabilized configuration. Consider $u \in V_1''$ such that $leader(u) = u$, $\neg Stab(u)$, and $\forall x \in \Omega^*(u)^{\gamma}, \beta_x^{\gamma} = \perp$. Then after at most one round a configuration γ' is reached such that $\Omega^*(u)^{\gamma'} = \Omega^*(u)^{\gamma}$, $Stab(u)$ is true and $\beta_u = \perp$ in γ'.*

Proof. Consider such a configuration γ and such a node u. Let v be a node such that $\Omega_v \neq \Omega^*(u)$.

Let's prove by induction that in every configuration until v executes **Update**, $\Omega^*(u)$ does not change, and $\neg Stab(u)$ is true. It is enough for that purpose to see that $\Omega_v \neq \Omega^*(u)$ prevents $Stab(u)$ to be true, thus the induction works using Lemma 4.9.

Consider then $S = \{x \in \Omega^*(u)^{\gamma} | \Omega_v^{\gamma} \neq \Omega^*(u)^{\gamma}\}$. As every node in S will be activable until it has executed **Update**, after at most one round every

one of them will have done this. Consider then the configuration just after the last of them is activated for the first time since γ, γ'.

By the above argument, every configuration in $\gamma \to^* \gamma'$ before γ' is such that $\neg Stab(u)$ and $\Omega^*(u)$ is the same as in γ, and then using Lemma 4.9 for the last transition we have $\Omega^*(u)^{\gamma'} = \Omega^*(u)^{\gamma}$. Then as Ω-value may only grow on well-defined nodes, in γ', we must have $Stab(u)$. Morever as in every configuration in $\gamma \to^* \gamma'$ before γ' we have $\neg Stab(u)$, no rule may have been executed to change a β-value in $\Omega^*(u)$, thus $\beta_u^{\gamma'} = \bot$. □

As a summary, it takes at most 2 rounds for every follower to synchronize with the leader.

Lemma 4.13. *Let γ be a $\mathcal{N}$-stabilized configuration. Consider $u \in V_1''$ such that $\neg Stab(u)$ in γ. Then after at most two rounds a configuration γ' is reached such that $\Omega^*(u)^{\gamma'} = \Omega^*(u)^{\gamma}$ and $Stab(u)$ in γ'.*

Proof. Using Lemma 4.10, from γ, after at most 1 round, a configuration γ'' is reached and is such that $\Omega^*(u)^{\gamma''} = \Omega^*(u)^{\gamma}$ and $\forall x \in \Omega^*(u), \beta_x^{\gamma''} = \bot$.

Using Lemma 4.6 we know that u is still well-defined in γ'', which gives us that $\Omega_u^{\gamma''} = \Omega^*(u)^{\gamma''} = \Omega^*(u)^{\gamma} = \Omega_u^{\gamma}$. Thus $leader(u)^{\gamma''} = leader(u)^{\gamma} = u$.

If $Stab(u)^{\gamma''}$, we can take $\gamma' = \gamma''$, and there is nothing left to prove. Suppose now that $\neg Stab(u)^{\gamma''}$. Then, using Lemma 4.12, from γ'', after at most 1 round, a configuration γ' is reached and is such that $\Omega^*(u)^{\gamma'} = \Omega^*(u)^{\gamma''}$ and $Stab(u)$ is true in γ'. Which concludes the proof, as $\Omega^*(u)^{\gamma''} = \Omega^*(u)^{\gamma}$. □

4.4 Merging happens and makes the solution progress

Now that we know that once a merging has begun it ends in a small number of rounds, we want to be sure that some merging happens.

Definition 5. *Consider a configuration γ, and $K = \{\Omega_x | x \in V_1''(\gamma)\})$ the set of all Ω-values in $V_1''(\gamma)$.*

We define: $\mathcal{C}(\gamma) = \{\omega \in K | \omega$ is maximal for inclusion in $K\}$

To begin with, we prove that it is indeed a clique decomposition.

Lemma 4.14. *Let γ be a $\mathcal{N}$-stabilized configuration, then $\mathcal{C}(\gamma)$ is a clique decomposition of V_1''.*

Proof. Being a clique decomposition of V_1'' is to be a set of cliques, to cover the entire set, and to have pairwise disjoint members:

- Since every node of V_1'' is well-defined by Lemma 4.4, and since γ is $\mathcal{N}$-stabilized, every Ω-value of nodes in V_1'' is a clique of G.
- From well-definedness we have that every node is contained in its Ω-value thus $\mathcal{C}(\gamma)$ is a cover of V_1''.
- Suppose by contradiction that c and c' distinct elements of $\mathcal{C}(\gamma)$ are such that $c \cap c' \neq \emptyset$. Consider then $u \in c \cap c'$.

  Consider v and v' in V_1'' such that $\Omega_v = c$ and $\Omega_{v'} = c'$ which exist by definition of $\mathcal{C}(\gamma)$. They are distinct since $\Omega_v = c \neq c' = \Omega_{v'}$. If $v = u$ then by well-definedness we have $c \subseteq c'$, which is a contradiction to c being a member of $\mathcal{C}(\gamma)$. Symmetrically the same can be said if $v' = u$. Suppose then that neither v nor v' is equal to u. Well-definedness of v and v' implies that they are both in $\{leader(leader(u)), \beta_u\}$. If $\beta_u = \perp$, this is impossible as $\min(c)$ and $\min(c')$ are distinct. Else $\beta_u \neq \perp$, and well-definedness of u implies that $leader(leader(u)) = u$, which is impossible as u, v and v' are supposed distinct. Thus, by contradiction, we have that $c \cap c' = \emptyset$.

Thus $\mathcal{C}(\gamma)$ is a clique decomposition of V_1''. □

The next lemma is about some well-formed property of the cliques of $\mathcal{C}(\gamma)$: every such clique must have a leader, and every node in a clique must have an Ω-value included in the clique. It is what we expect from the way the algorithm forms new cliques by merging.

Lemma 4.15. *Let γ be a $\mathcal{N}$-stabilized configuration, and consider $c \in \mathcal{C}(\gamma)$. We have $\forall x \in c, \Omega_x \subseteq c$. Moreover, $\exists! u \in V_1'', leader(u) = u \wedge \Omega_u = c$.*

Proof. Consider $x \in c$. Consider also $v \in V_1''$ such that $\Omega_v = c$, which exists by definition of $\mathcal{C}(\gamma)$. By well-definedness of v we have that $\Omega_x \subseteq \Omega_v = c$.

Consider then $u = \min(c)$. We have by definition of *leader* that $leader(v) = u$, and then by well-definedness $c \subseteq \Omega_u$. As $u \in c$, we also have $\Omega_u \subseteq c$, thus $\Omega_u = c$.

Unicity comes from that the leader of such a node must have $\min(c)$ as a leader. □

Definition 6 (Representative)**.** *Since we know the existence and unicity in every clique of $\mathcal{C}(\gamma)$ of a node that is its own leader and has the clique as Ω-value, we call $min(c)$ the representative of the clique $c \in \mathcal{C}(\gamma)$ in configuration γ.*

To progress toward our goal, we need to prove that $\mathcal{C}$ makes progress in some sense. To this aim, we prove a heredity property as well as a non-regression property across transitions for $\mathcal{C}$.

Lemma 4.16. *Let γ be a $\mathcal{N}$-stabilized configuration and $\gamma \to \gamma'$ a transition. We have:*

- *(Non-regression) $\forall c \in \mathcal{C}(\gamma)$, $\exists c' \in \mathcal{C}(\gamma')$, $c \subseteq c'$.*
- *(Heredity) $\forall c' \in \mathcal{C}(\gamma')$, $\exists c \in \mathcal{C}(\gamma)$, $c \subseteq c'$.*

Proof. Let's first prove the first point of the lemma. Consider $c \in \mathcal{C}(\gamma)$, and consider u the representative of c in γ (see Lemma 4.15). Consider also $c' \in \mathcal{C}(\gamma')$ such that $u \in c'$ (which exists by Lemma 4.14), with v the representative of c' in γ' (see Lemma 4.15).

As $u \in \Omega_v^{\gamma'}$, by well-definedness $\Omega_u^{\gamma'} \subseteq \Omega_v^{\gamma'}$. But since $u \in V_1''(\gamma)$, we also have $\Omega_u^{\gamma} \subseteq \Omega_u^{\gamma'}$ (only **Reset** may make the Ω-value shrink). Thus $\Omega_u^{\gamma} \subseteq \Omega_v^{\gamma'}$ *i.e.* $c \subseteq c'$. Hence the first point of the Lemma.

Let's then prove the second point of the lemma. Consider $c' \in \mathcal{C}(\gamma')$ with v the representative of c in γ (see Lemma 4.15). If $c' \in \mathcal{C}(\gamma)$, there is nothing to prove, suppose then w.l.o.g. that it's not the case.

As the Ω-value of v must have changed in the transition, it executed either **Merge lead**, **Merge follow**, or **Update**.

- In fact, **Update** is not a valid option as it is not enabled on u in γ. It would otherwise imply the existence of $u \in V_1''(\gamma)$ such that $\Omega_u = c'$. Then there would be $c \in \mathcal{C}(\gamma)$ such that $\Omega_u \subseteq c$ by definition of $\mathcal{C}$. Then with the first point of the Lemma plus the fact that $\mathcal{C}(\gamma')$ is a clique decomposition by Lemma 4.14 we get $c' \subseteq c \subseteq c'$ *i.e.* $c = c'$ which is false by hypothesis.
- If v performed **Merge lead** in the transition, let's write $u = \beta_v^{\gamma}$. We have also from the guard of **Merge lead** that $\beta_u^{\gamma} = v$, $\Omega_u^{\gamma} \cap \Omega_v^{\gamma} = \emptyset$, $Stab(v)^{\gamma}$, and u is not enabled in γ. By the definition of $V_1''(\gamma')$ and the fact that $Stab(v)$ is true, $\Omega_v^{\gamma'}$ must contain a node of V_1. Then, as $\Omega_v^{\gamma'} = \Omega_u^{\gamma} \cup \Omega_v^{\gamma}$, either Ω_u^{γ} or Ω_v^{γ}.
  - Suppose Ω_u^{γ} contains a node of V_1, we have $u \in V_1''$. Suppose by contradiction that $\Omega_u^{\gamma} \notin \mathcal{C}(\gamma)$. There must exist $u' \in V_1''(\gamma)$ such that $\Omega_u^{\gamma} \subseteq \Omega_{u'}^{\gamma}$. $u' \notin \Omega^*(v)^{\gamma}$ from the fact that $\Omega_u^{\gamma} \cap \Omega_v^{\gamma} = \emptyset$. But then $\beta_u = v$ is a contradiction to the well-definedness of u'. Thus, by contradiction we have $\Omega_u^{\gamma} \in \mathcal{C}(\gamma)$, we then write $c = \Omega_u^{\gamma}$. After the transition we have $\Omega_v^{\gamma'} = \Omega_u^{\gamma} \cup \Omega_v^{\gamma}$, thus $c \subseteq c'$.

– Suppose now Ω_v^γ contains a node of V_1. By the same reasoning as the previous case, we have that $\Omega_v^\gamma \in \mathcal{C}(\gamma)$, and then by taking $c = \Omega_v^\gamma$ we have $c \subseteq c'$.

- Else, v must have performed **Merge follow** in the transition, and let's write $u = \beta_v^\gamma$. The guard of the rule also gives us $Stab(v)^\gamma$, $\Omega_u^\gamma = c'$. Observe that as we supposed that $c' \notin \mathcal{C}(\gamma)$ this implies that $u \notin V_1''$, and that $v \in V_1''(\gamma)$. By the same reasoning than for the **Merge lead** case we get $\Omega_v^\gamma \in \mathcal{C}(\gamma)$, and then by taking $c = \Omega_v^\gamma$ we have $c \subseteq c'$.

□

Remark 1. *Lemma 4.16 implies that in such a transition* $|\mathcal{C}(\gamma')| \leq |\mathcal{C}(\gamma)|$.

When $\mathcal{C}$ changes, it makes progress. But we have to ensure that it does change sometimes. The next four lemmas are a toolbox that will be used to prove that.

Before proving that $\mathcal{C}$ makes progress, we prove a lemma about what happens when a node of $V_0 \setminus V_1''$ is merged with one from V_1''.

Lemma 4.17. *Let γ be $\mathcal{N}$-stabilized configuration and v a node of $V_1''(\gamma)$. If v executes rule **Merge lead** or **Merge follow** in the transition $\gamma \to \gamma'$, then $\beta_v^\gamma \in V_1''(\gamma')$.*

Proof. Suppose v executes **Merge lead** in the transition $\gamma \to \gamma'$, it means that **Merge lead** was enabled on v in γ. Then, as $\beta_v \in merge\_candidate(v)$ in γ, we have $leader(\beta_v) = \beta_v$ in γ, and thus $\beta_v \in \Omega_{\beta_v}$ in γ. Thus, we have $\beta_v^\gamma \in \Omega_v^{\gamma'} = \Omega_v^\gamma \cup \Omega_{\beta_v}^\gamma$.

Suppose now that v executes **Merge follow** in the transition $\gamma \to \gamma'$, it means that **Merge follow** was enabled on v in γ. Then, as $coherent\_clique(\beta_v)$ in γ, we have $\beta_v \in \Omega_{\beta_v}$ in γ. Thus, we have $\beta_v^\gamma \in \Omega_v^{\gamma'} = \Omega_{\beta_v}^\gamma$.

In both cases, $\beta_v^\gamma \in \Omega_v^{\gamma'}$. But as $u \in V_1''(\gamma)$, from Lemma 4.6 we have $u \in V_1''(\gamma')$. Thus $\beta_v^\gamma \in V_1''(\gamma')$. □

Starting in a $\mathcal{N}$-stabilized configuration, either we directly get what we want (Case 4 of Lemma 4.18), or we reach one of two types of configurations (Cases 1,2 and 3 of Lemma 4.18) that will be dealt with in other lemmas.

Lemma 4.18. *Let γ be a $\mathcal{N}$-stabilized configuration. Suppose $\mathcal{C}(\gamma)$ is not a minimal clique decomposition of $V_1''(\gamma)$.*

Then, after at most two rounds, a configuration γ' is reached where one of those is true:

1. ***Seduction*** *or* ***Mariage*** *is enabled on at least one node of* $V_1''(\gamma')$.
2. $\exists u \in V_1'', \exists v \in V_0$ *such that* $\beta_u = v$ *and* ***Mariage*** *is enabled on* v.
3. $\exists u \in V_1'', \exists v \in V_0$ *such that* ***Merge Lead*** *is enabled on* u *or* v.
4. $|\mathcal{C}(\gamma')| < |\mathcal{C}(\gamma)|$ *or* $V_1''(\gamma) \subsetneq V_1''(\gamma')$

Proof. Suppose there exists $c, c' \in \mathcal{C}(\gamma)$ such that $c \cup c'$ is a clique (they exist by hypothesis, as $\mathcal{C}(\gamma)$ would be a minimal clique decomposition of $V_1''(\gamma)$ otherwise).

Consider then u (resp. u') the representative of c (resp. c') in γ. If **Seduction** or **Mariage** is enabled on u there is nothing to prove. The same can be said if $\beta_u \neq \perp$ and **Merge Lead** is enabled on either u or β_u. The same can be said for u'.

Let's then suppose it's not the case in γ. We are in either of those cases :

- $Stab(u)$, $\beta_u^\gamma \neq \perp$, and **Merge follow** is enabled on u. We have $\beta_u^\gamma \notin V_1''$ as if it was not the case, u would not be a representative. Then the guard of **Merge follow** guarantees that $Stab(\beta_u)$ in γ and that this will not change until **Merge follow** is executed by u. Thus after at most one round, u executes **Merge follow**, and in the resulting configuration β_u^γ is in V_1'' from Lemma 4.17 and we have $V_1''(\gamma) \subsetneq V_1''(\gamma')$ (Case 4 of the lemma).
- $Stab(u)$, $\beta_u^\gamma \neq \perp$, and **Merge follow** is not enabled on u. By hypothesis, **Merge lead** is not enabled on u either.
  - If **Abandonment** is not enabled on u we have $\beta_v \in \{\perp, u\}$ and $v \in merge\_candidate(u)$. If $\beta_v = \perp$, **Mariage** is then enabled on v (Case 2 of the lemma), else $\beta_v = u$ and **Merge lead** is enabled on u or v (Case 3 of the lemma)
  - If **Abandonment** is enabled on u, it remains so until either it executes **Abandonment**, or **Abandonment** is not activable anymore, and one of those must happen in at least one round. In the first case in the configuration just after the transition where it executes **Abandonment** we have $\beta_u = \perp$, and $Stab(u)$. In the second case, as in the previous point, we are in either Case 2 or 3 of the lemma.
- $\neg Stab(u)$ in γ, in which case by Lemma 4.13 a configuration where $Stab(u)$ is true and $\beta_u = \perp$ is reached after at most two rounds.

Thus, after at most two rounds, either a configuration that satisfies one of the conditions of the lemma has been reached, or a configuration where $Stab(u)$ and $\beta_u = \perp$ has been reached. Note that when this is the case, the only rules that may be enabled on u are **Seduction** or **Mariage**, so we may assume w.l.o.g. that in the configuration when exactly two rounds have passed $Stab(u)$ and $\beta_u = \perp$ (otherwise we reached a configuration corresponding to Case 1 of the lemma before that).

By symmetry, the exact same argument applies to u'.

Then, after two rounds, we are in a configuration where $Stab(u)$, $\beta_u = \perp$, $Stab(u')$ and $\beta_{u'} = \perp$. Then, as $c \cup c'$ is supposed to be a clique, **Seduction** is enabled on u and u', which concludes the proof. □

Then, starting in a configuration corresponding to Case 1 of Lemma 4.18, we reach with probability at least $\frac{1}{\Delta}$ a configuration having the same properties as the one of Case 3 of Lemma 4.18, *i.e.* a configuration where a clique merging is about to begin.

Lemma 4.19. *Let γ be a $\mathcal{N}$-stabilized configuration and suppose **Seduction** or **Mariage** is enabled on at least one node of V_1''. Then, there is a probability at least $\frac{1}{\Delta}$ that after at most two rounds a configuration γ' is reached where: $\exists u \in V_1'', \exists v \in V_0$ such that **Merge Lead** is enabled on u or v.*

Proof. Consider a node $u \in V_1''(\gamma)$ such that **Mariage** is enabled on it. Any node $w \in V_0$ such that $\beta_w^\gamma = u$ and $w \in merge\_candidate(u)$ cannot execute any rule while **Mariage** is enabled on u. There is at least one such node since **Mariage** is enabled on u. Thus after at most one round, u executes **Mariage**, and in the resulting configuration we have $\beta_u = v \in V_0$, $\beta_u = v$, and $\beta_v = u$, with $v \in merge\_candidate(u)$. Thus, in this configuration, **Merge Lead** is enabled on $\min(u, v)$.

Suppose now that **Mariage** is not enabled on any node of V_1''. Consider $u \in V_1''\gamma$ such that **Seduction** is enabled on it. **Seduction** will remain enabled on u until either it is executed, or one member of $merge\_candidate(u)$ executes **Seduction**.

- Suppose that in the first transition where one of those events happens u executes **Seduction** (after at most one round), and γ' the resulting configuration. If $v = \beta_u^{\gamma'}$ did not execute any rule in the transition, then we have $v \in merge\_candidate(u)^{\gamma'}$, $\beta_v^{\gamma'} = \perp$, and thus **Mariage** is enabled on it. It will remain enabled until v executes **Mariage** (after at most one round), and in the resulting configuration with probability at least $\frac{1}{\Delta}$ in the resulting configuration γ'' we have $\beta_v^{\gamma''} = u$, and **Merge lead** is enabled on $\min(u, v)$ in γ''.

- Suppose now that in the first transition where one of those events happens (after at most one round) u does not execute **Seduction** and at least one node $w \in merge\_candidate(u)$ executes **Seduction**. In the resulting configuration γ' we have then $Stab(u)$, $\beta_u^{\gamma'} = \perp$. Moreover, with probability at least $\frac{1}{\Delta}$, $\beta_w = u$. Thus in γ' **Mariage** is now enabled on u until it is executed on u as no rule will be enabled on v until then. When this happens (after at most one round), in the resulting configuration γ'', $v = \beta_u^{\gamma}$ is such that $\beta_u = v$, $\beta_v = u$, $v \in merge\_candidate(u)$ and $u \in merge\_candidate(v)$. Thus, **Merge lead** is enabled on $\min(u, v)$ in γ''.

□

From a configuration of the type of Case 2 of Lemma 4.18 we will reach a configuration of the type of Case 4 of Lemma 4.18 with probability at least $\frac{1}{\Delta}$.

Lemma 4.20. *Let γ be a $\mathcal{N}$-stabilized configuration such that $\exists u \in V_1'', \exists v \in V_0$ such that $\beta_u = v$ and **Mariage** is enabled on v.*

*Then, there is a probability at least $\frac{1}{\Delta}$ that after at most one round a configuration γ' is reached where: $\exists u \in V_1'', \exists v \in V_0$ such that **Merge Lead** is enabled on u or v.*

Proof. As **Mariage** is enabled on v in γ, we have that v is well-defined and $v \in merge\_candidate(u)$, thus Ω_v does not have Byzantine nodes. **Mariage** will then remain enabled on v until it is executed as no node of Ω_u and Ω_v except is activable while **Mariage** is enabled on $\beta_u = v$. When the first time v executes **Mariage** starting in γ (which happens after at most one round), in the resulting configuration γ', there is a probability at least $\frac{1}{\Delta}$ (Δ being the maximum size of $merge\_candidate(v)$) that $\beta_v = u$. If this is the case, as no other node than v in Ω_u and Ω_v

Observe that, as no other node than v in Ω_u and Ω_v was activable before v executed **Mariage**, we still have $v \in merge\_candidate(u)$ and $u \in merge\_candidate(v)$ in γ'. Then, if $\beta_v^{\gamma'} = u$, **Merge lead** is enabled on either u or v in γ'.

Thus, with probability at least $\frac{1}{\Delta}$, **Merge lead** is enabled on either u or v in γ'. □

And then from a configuration of the type of Case 3 of Lemma 4.18 we will reach a configuration of the type of Case 4 of Lemma 4.18.

Lemma 4.21. *Let γ be a $\mathcal{N}$-stabilized configuration such that $\exists u \in V_1'', \exists v \in \{x \in V_0 | well\_defined(x)\}$ such that **Merge Lead** is enabled on u or v. Then after 2 rounds a configuration γ' is reached where $\beta_u = \beta_v = \perp$, and either is true:*

- $|\mathcal{C}(\gamma')| < |\mathcal{C}(\gamma)|$,
- $|\mathcal{C}(\gamma')| \leq |\mathcal{C}(\gamma)|$ *and* $V_1''(\gamma) \subsetneq V_1''(\gamma')$.

Proof. As $u \in V_1''(\gamma)$, by definition of V_1'' there is $w \in V_1$ well-defined in γ such that $u \in \Omega^*(w)^\gamma$. As w is well-defined in γ, we have $\min \Omega^*(w)^\gamma \in \{leader(leader(u)), \beta_u\}$ *i.e.* $\min \Omega^*(w)^\gamma \in \{u, v\}$. If $\min \Omega^*(w)^\gamma$ was v, it would contradict w well-definedness since we would have $\beta_v^\gamma \in \Omega^*(w)^\gamma$, thus by contradiction $\min \Omega^*(w)^\gamma = u$.

Moreover well-definedness of w also implies that $w \in \min \Omega^*(w)^\gamma$, thus $w \in \Omega_u$. From the guard of **Merge lead**, either $u \in merge\_candidate(v)$ or $v \in merge\_candidate(u)$. In both cases it implies that every node of Ω_v is neighbor of $w \in V_1$, thus $\Omega_v \subseteq V_0$.

Let us write $x = \min(u, v)$ and $y = \max(u, v)$

Given the constraints on the configuration, we know that:

- **Merge lead** (and only this rule) is enabled on x in γ,
- No node in $\Omega_x^\gamma \setminus \{x\}$ or $\Omega_y^\gamma \setminus \{y\}$ is activable as we supposed $Stab(x)$, $Stab(y)$, and that x and y are well-defined,
- No rule is activable on y, as $\beta_x = y$.

Those facts will remain true after any transition where x is not activated (as it means that no node in Ω_x^γ or Ω_y^γ made a move, and thus the hypothesis on x and y stay true in the resulting configuration).

Thus, in the first transition where a node of $\Omega_x^\gamma \cup \Omega_y^\gamma$ is activated, **Merge lead** is executed by x and no other node of $\Omega_x^\gamma \cup \Omega_y^\gamma$ is activated. It happens after at most 1 round since **Merge lead** is continuously enabled on x until so. Consider the configuration γ' just after the transition where it happens.

Lemma 4.5 allow us to say that both x and y are still well-defined in γ', as neither Ω_x^γ nor Ω_y^γ contain Byzantine nodes.

We still have $Stab(y)$ and $leader(y) = y$ in γ' since no node of Ω_y was activated. Moreover we also still have $leader(x) = x$ in γ' as $x < y = min\Omega_y^\gamma$, and we have $\Omega_y^{\gamma'} = \Omega_y^\gamma \subseteq \Omega_x^\gamma \cup \Omega_y^\gamma = \Omega_x^{\gamma'}$. Moreover, as x is well-defined in γ', $coherent\_clique(x)$ is true in γ'.

Thus **Merge follow** is enabled on y. Observe that no other node of Ω_y is activable, and that it won't change until y executes a rule. Node x cannot

make any move until $Stab(x)$ becomes true again, thus not before **Merge follow** is executed by y. It happens after at most 1 round since **Merge follow** is continuously enabled on y until so. Consider the configuration γ'' just after the transition where it happens.

Again using Lemma 4.5, x and y are still well-defined in γ''. From the command of **Merge follow** we have $\beta_y^{\gamma''} = \perp$. Moreover, from the guard of **Merge lead**, $\beta_y^{\gamma'} = \perp$, and since x did not get activated afterward $\beta_x^{\gamma''} = \perp$.

Then, as $v \in \Omega_u^{\gamma''}$ and $u \in V_1''(\gamma'')$, we have $v \in V_1''(\gamma'')$.

- If u and v where both in $V_1''(\gamma)$, we have $\Omega_u^\gamma \in \mathcal{C}(\gamma)$ and $\Omega_v^\gamma \in \mathcal{C}(\gamma)$ with $\Omega_u^\gamma \neq \Omega_v^\gamma$. Moreover, we have $\Omega_u^\gamma \cup \Omega_v^\gamma = \Omega_u^{\gamma''} \in \mathcal{C}(\gamma'')$. Using Lemma 4.16 we can then say $|\mathcal{C}(\gamma'')| < |\mathcal{C}(\gamma)|$.
- Else, v was not in $V_1''(\gamma)$. Thus, as we have $v \in V_1''(\gamma'')$, $V_1''(\gamma) \subsetneq V_1''(\gamma'')$. Lemma 4.16 also implies $|\mathcal{C}(\gamma'')| \leq |\mathcal{C}(\gamma)|$.

□

4.5 Convergence and time complexity

Then, as we visually represent in Figure 1, we have a probabilistic pattern that makes cliques grow that will repeat as long as $\mathcal{C}$ is not a minimal clique decomposition of V_1''. Trivially, it implies that the algorithm ends with probability 1, but we will try to be more precise than that.

To do this, we use a concentration inequality (Azuma's inequality) to give a probabilistic bound on the number of rounds it takes to reach a configuration where $\mathcal{C}$ is a minimal clique decomposition of V_1''.

Lemma 4.22. *Let γ be a $\mathcal{N}$-stabilized configuration. With probability at least p, after $4\max(-\Delta^2 \ln p, \frac{\sqrt{2}}{\sqrt{2}-1}\Delta n) + 6n$ rounds a configuration γ' such that $\mathcal{C}(\gamma')$ is a minimal clique decomposition of V_1'' is reached.*

Proof. The size of $\mathcal{C}$ may only decrease from Lemma 4.16, and V_1'' may only increase from Lemma 4.6.

Suppose that $\mathcal{C}(\gamma)$ is not a minimal clique decomposition of V_1''.

Starting in configuration γ, using Lemmas 4.18, 4.19 and 4.20 there is a probability at least $\frac{1}{\Delta}$ to reach a configuration corresponding to the preconditions of Lemma 4.21 in at most 4 rounds.

If it is successful, from Lemma 4.21, starting in the resulting configuration γ', after at most two rounds, a configuration γ'' is reached such that $|\mathcal{C}(\gamma'')| < |\mathcal{C}(\gamma)|$, or $|\mathcal{C}(\gamma'')| \leq |\mathcal{C}(\gamma)|$ and $V_1''(\gamma) \subsetneq V_1''(\gamma'')$.

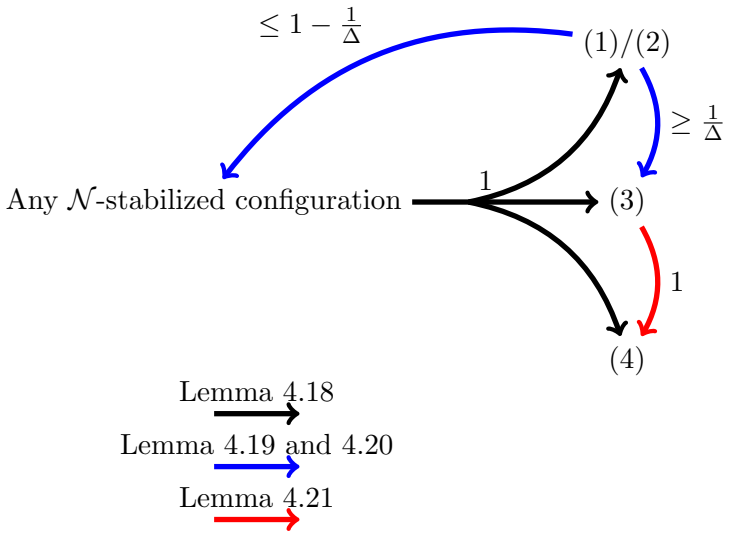

Figure 1: (1), (2), (3), and (4) refer to configurations having properties numbered in Lemma 4.18. Values on arrows are probabilities given by the lemmas.

In both cases (successful or not), either in the resulting configuration $\mathcal{C}$ is a clique decomposition of V_1'' (and there is nothing left to prove), or we may again apply the same set of lemmas.

Then, by Azuma's inequality, with probability at least p, in at most $\max(-\Delta^2 \ln p, \frac{\sqrt{2}}{\sqrt{2}-1}\Delta n)$ iterations, we get a configuration where $\mathcal{C}(\gamma')$ is a minimal clique decomposition of V_1''.

Successful iterations take at most 6 rounds, unsuccessful ones 4. Since there can be at most n successful iterations, we get to that configuration after at most $4\max(-\Delta^2 \ln p, \frac{\sqrt{2}}{\sqrt{2}-1}\Delta n) + 6n$ rounds. □

Follows the theorem as a direct corollary of Lemma 4.22 and 4.1

Theorem 4.23. *From any configuration γ, with probability at least p, after $4\max(-\Delta^2 \ln p, \frac{\sqrt{2}}{\sqrt{2}-1}\Delta n) + 6n + 1$ rounds a configuration γ' such that $\mathcal{C}(\gamma')$ is a minimal clique decomposition of V_1'' (and $V_1 \subseteq V_1''$) is reached .*

5 Specification

As we introduced the notions $\mathcal{C}$ and V_1'' we can now express the specification of our algorithm: the legitimate configurations are configurations γ such that

$V_1 \subseteq V_1''(\gamma)$ and $\mathcal{C}(\gamma)$ is a minimal clique decomposition of $V_1''(\gamma)$.

We cannot guarantee that when such a configuration is reached $\mathcal{C}$ and V_1'' will not change again. It's because nodes that neighbor Byzantine nodes, and were previously entangled in dummy cliques staged by those Byzantine nodes, may at any time execute **Reset** after a move from a Byzantine node. If one of them merges with a clique of V_1'' after that, it makes V_1'' grow, and this can happen arbitrarily far in the execution. But what we can guarantee as a stability property is that when such a configuration is reached the property $\mathcal{C}$ is a minimal clique decomposition of V_1'' will be conserved, even if the values of $\mathcal{C}$ and V_1'' change. And recall that those may only change by "growing" in some sense (see Lemmas 4.16 and 4.6 respectively).

Lemma 5.1. *Let γ be a $\mathcal{N}$-stabilized configuration such that $\mathcal{C}(\gamma)$ is a minimal clique decomposition of $V_1''(\gamma)$. Suppose $\gamma \to \gamma'$.*

Then $\mathcal{C}(\gamma')$ is a minimal clique decomposition of $V_1''(\gamma')$.

Proof. Suppose by contradiction that $\mathcal{C}(\gamma')$ is not a minimal clique decomposition of $V_1''(\gamma')$.

As γ' is $\mathcal{N}$-stabilized by Lemma 4.2, Lemma 4.16 tells us that $\mathcal{C}(\gamma')$ is a clique decomposition of V_1''. Then $\mathcal{C}(\gamma')$ not being a minimal clique decomposition of $V_1''(\gamma')$ means that there exist c and c' in $\mathcal{C}(\gamma')$ such that $c \cup c'$ is a clique of $V_1''(\gamma')$.

But then, using Lemma 4.16, consider k, k' in $\mathcal{C}(\gamma)$ such that $k \subseteq c$ and $k' \subseteq c'$. Since $c \cup c'$ is a clique, it means that $k \cup k' \subseteq c \cup c'$ is also a clique. It is then a clique of $V_1''(\gamma)$, which contradicts the fact that $\mathcal{C}(\gamma)$ is a minimal clique decomposition of $V_1''(\gamma)$. □

6 Correction

Moreover, we want a correction property for our algorithm. A natural thing would be to be in a legitimate configuration when no node is activable in V_1''. However, it's not the case, as a node of V_1'' might not be activable, but waiting for a node in $V_0 \setminus V_1''$ to execute **Merge lead** before being able to execute **Merge follow**. We will then restrict our correction property, expressed in Lemma 6.3 to configurations where β-values of every node in V_1'' is $\perp$. To prove this lemma, we will prove two preliminary lemmas.

First, a lemma that states some properties that are weaker than being a legitimate configuration but which are true whenever no node is activable in V_1''.

Lemma 6.1. *Let γ be a configuration with no node activable in V_1''. For $v \in V_1''$ we have in γ:*

1. $\mathcal{N}_v = N(v)$,
2. $\Omega_v = \Omega_{leader(v)}$,
3. $Stab(v)$,
4. Ω_v *is a clique of* G.

Proof. 1. If we had $\mathcal{N}_v = N(v)$, **Reset** would be activable on v.

2. Suppose $\Omega_v \neq \Omega_{leader(v)}$. Then either v is not well-defined and **Reset** is enabled on v, or **Update** is enabled on v.

3. Suppose $\neg Stab(v)$. There would be $w \in \Omega_v$ $(= \Omega_{leader(v)} \subseteq V_1'')$ such that $\Omega_w \subsetneq \Omega_{leader(v)}$. Then either w is not well-defined, or **Update** is enabled on w, which contradicts the non-activable hypothesis.

4. Suppose that Ω_v is not a clique of G, *i.e.* $\exists s, t \in \Omega_v$ distinct such that s and t are not neighbors. By definition, s and t are in V_1'. From previous points, we have $Stab(v)$, thus $\Omega_v = \Omega_s = \Omega_t$. Then $t \notin N(s)$ implies either that $t \notin \mathcal{N}_s$ or $N(s) \neq \mathcal{N}_s$, and in both cases **Reset** is enabled on s which contradicts the hypothesis that no node is activable in V_1''.

□

Then we prove that in a configuration where no node is activable in V_1'', if two cliques of V_1'' are not already waiting to merge with some node in $V_0 \setminus V_1''$, then their union is not a clique.

Lemma 6.2. *Let γ be a configuration with no activable nodes in V_1'', and u, v two distinct nodes of V_1'' such that $\Omega_u \neq \Omega_v$ and $\beta_u = \beta_v = \perp$ in γ. The set $\Omega_u \cup \Omega_v$ does not form a clique in G.*

Proof. Suppose by contradiction that $\Omega_u \cup \Omega_v$ forms a clique in G with $\Omega_u \neq \Omega_v$.

By hypothesis, we have the properties from Lemma 6.1.

Thus $Stab(u)$ and $Stab(v)$, and we may suppose w.l.o.g. that $leader(u) = u$ and $leader(v) = v$.

The fact that $\Omega_u \cup \Omega_v$ is a clique of G means that $\Omega_u \cup \Omega_v \subseteq \bigcap_{x \in \Omega_u \cup \Omega_v} N(x) \cup \{x\}$, but by well-definedness we have $\forall x \in \Omega_u \cup \Omega_v, \mathcal{N}_x = N(x) \cup \{x\}$.

Moreover, $\Omega_u \cup \Omega_v = \emptyset$, since it would imply that $\Omega_u = \Omega_v$ using $Stab(v)$ and $Stab(u)$.

Then, since $leader(v) = v$ and $Stab(v)$, $v \in merge\_candidate(u)$. Then, since we have $\beta_u = \beta_v = \perp$, either there exists $w \in merge\_candidate(u)$ such that $\beta_w = u$ and **Mariage** is activable on u, or **Seduction** is activable on u. □

Our correction property follows immediately from Lemmas 6.1 and 6.2.

Lemma 6.3. *Let γ be a configuration with no node activable in V_1'' and where every node of γ has β-value $\perp$, $\mathcal{C}(\gamma)$ is a minimal clique decomposition of $V_1''(\gamma)$.*

7 Conclusion

We have proved that Minimal Clique Decomposition can be solved in $O(\Delta n)$ rounds with high probability in the presence of Byzantine faults under the fair daemon.

The same algorithm could be used to do the same in a context without Byzantine nodes but under the adversarial daemon. However, as we used probabilistic rules to prevent the Byzantine nodes to be able to reliably trap us, we could probably remove randomness in this context, by saying $choose(A) = min(A)$ instead of drawing an element uniformly.

References

[1] Elias Dahlhaus and Marek Karpinski. "A fast parallel algorithm for computing all maximal cliques in a graph and the related problems". In: *SWAT 88*. Ed. by Rolf Karlsson and Andrzej Lingas. Springer Berlin Heidelberg, 1988, pp. 139–144.

[2] François Delbot, Christian Laforest, and Raksmey Phan. "New approximation algorithms for the vertex cover problem". In: *International Workshop on Combinatorial Algorithms*. Springer. 2013, pp. 438–442.

[3] François Delbot, Christian Laforest, and Stephane Rovedakis. "Self-stabilizing Algorithms for Connected Vertex Cover and Clique Decomposition Problems". In: *Principles of Distributed Systems*. Ed. by Marcos K. Aguilera, Leonardo Querzoni, and Marc Shapiro. Springer International Publishing, 2014, pp. 307–322.

[4] H. Ishii and H. Kakugawa. “A self-stabilizing algorithm for finding cliques in distributed systems”. In: *21st IEEE Symposium on Reliable Distributed Systems, 2002. Proceedings.* 2002, pp. 390–395.

[5] Esther Jennings and Lenka Motyčková. “A distributed algorithm for finding all maximal cliques in a network graph”. In: *LATIN ’92*. Ed. by Imre Simon. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 281–293.

[6] Richard M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Springer US, 1972, pp. 85–103.

[7] Stephan Kunne, Johanne Cohen, and Laurence Pilard. “Self-stabilization and Byzantine Tolerance for Maximal Matching”. In: *Stabilization, Safety, and Security of Distributed Systems*. Ed. by Taisuke Izumi and Petr Kuznetsov. Cham: Springer International Publishing, 2018, pp. 80–95.

[8] Li Lu, Yunhong Gu, and Robert Grossman. “dMaximalCliques: A Distributed Algorithm for Enumerating All Maximal Cliques and Maximal Clique Distribution”. In: *2010 IEEE International Conference on Data Mining Workshops*. 2010, pp. 1320–1327.

[9] Yanyan Xu et al. “Distributed Maximal Clique Computation”. In: *2014 IEEE International Congress on Big Data*. 2014, pp. 160–167.