A Formalized Polynomial-Time Reduction From Clique to Vertex Cover

Violeta Kastreva

August 22, 2024

Abstract

This report presents a formalization of the well-known polynomial-time reduction from the Clique problem to the Vertex Cover problem. The reduction is implemented and verified in the Coq proof assistant. We provide a formal proof of correctness, ensuring that the reduction preserves the problem's complexity.

1 Introduction

In computational complexity theory, reductions between NP-complete problems are fundamental for understanding the relationships among various problems. A classic example is the reduction from the Clique problem to the Vertex Cover problem. This report provides a formalization of this reduction, including a proof of correctness, using the Coq proof assistant.

2 Preliminaries

2.1 Graphs

We define an undirected graph $G = (V, E)$, where V is a finite set of vertices, and E is a set of edges, represented as unordered pairs of distinct vertices from V.

2.2 The Clique Problem

Definition 1 (Clique)**.** *Given a graph $G = (V, E)$ and an integer k, the Clique problem asks whether there exists a subset $V' \subseteq V$ such that $|V'| = k$ and every pair of vertices in V' is connected by an edge in G.*

Definition 2 (Vertex Cover)**.** *Given a graph $G = (V, E)$ and an integer k, the Vertex Cover problem asks whether there exists a subset $S \subseteq V$ such that $|S| = k$ and every edge in E has at least one endpoint in S.*

3 Reduction from Clique to Vertex Cover

In this section, we describe the reduction from the Clique problem to the Vertex Cover problem. The key insight is that a k-clique in a graph G corresponds to a vertex cover of size $|V| - k$ in the complement of G.

3.1 Complement Graph

Definition 3 (Complement Graph)**.** *Given a graph $G = (V, E)$, the complement graph $\overline{G} = (V, \overline{E})$ is defined such that two distinct vertices $u, v \in V$ are adjacent in $\overline{G}$ if and only if they are not adjacent in G.*

The complement graph allows us to transform cliques in the original graph into independent sets, which directly correspond to vertex covers in the complement.

3.2 Reduction Construction

The reduction from the Clique problem to the Vertex Cover problem proceeds as follows:

1. **Input:** A graph $G = (V, E)$ and an integer k.
2. **Construct the complement graph:** Compute $\overline{G} = (V, \overline{E})$.
3. **Compute the target vertex cover size:** Set $k' = |V| - k$.
4. **Output:** The pair $(\overline{G}, k')$ forms the instance of the Vertex Cover problem.

Proposition 1 (Correctness of the Reduction)**.** *There exists a k-clique in the graph G if and only if there exists a vertex cover of size $|V| - k$ in the complement graph $\overline{G}$.*

Proof. Assume $V' \subseteq V$ is a k-clique in G. Then, every pair of vertices in V' is connected by an edge in G. Therefore, in $\overline{G}$, no two vertices in V' are connected by an edge, meaning V' forms an independent set in $\overline{G}$. The complement of this set $V \setminus V'$ is a vertex cover in $\overline{G}$ of size $|V| - k$.

Conversely, if $S \subseteq V$ is a vertex cover of size $|V| - k$ in $\overline{G}$, then the set $V \setminus S$ is an independent set of size k in $\overline{G}$, which implies that $V \setminus S$ is a k-clique in G. □

4 Formalization in Coq

To ensure the correctness of the reduction, we formalize the reduction and its proof in the Coq proof assistant. Coq allows us to define the relevant graph structures, implement the reduction, and verify the correctness of the reduction step by step.

4.1 Graph Representation in Coq

We represent graphs using the ‘UGraph‘ structure, where the vertex set is a finite type and the edge set is a decidable symmetric relation.

Listing 1: Graph Representation in Coq

```
Record UGraph := {
  V : finType;
  E : V * V -> Prop;
  E_dec : forall v1 v2, {E (v1, v2)} + {~ E (v1, v2)};
  E_symm : forall v1 v2, E (v1, v2) <-> E (v2, v1)
}.
```

4.2 Complement Graph

Definition 4 (Complement Graph)**.** *Given a graph $G = (V, E)$, the complement graph $\overline{G} = (V, \overline{E})$ is defined such that two distinct vertices $u, v \in V$ are adjacent in $\overline{G}$ if and only if they are not adjacent in G.*

The complement graph is implemented in Coq using the following definition:

Listing 2: Complement Graph Construction in Coq

```
Definition V_complement := V.

Definition E_complement (p : V_complement * V_complement) : Prop :=
  let (v1, v2) := p in
  ~ E (v1, v2).

Definition complementGraph :=
  Build_UGraph
    E_complement_dec
    E_complement_symm.
```

4.3 Correctness of the Reduction

The correctness of the reduction is formalized by proving that the reduction preserves the existence of solutions between the two problems. Specifically, we prove that a k-clique in G corresponds to a vertex cover of size $|V| - k$ in the complement graph $\overline{G}$, and vice versa.

Listing 3: Correctness Proof

```
Theorem Clique_reduces_to_KVertexCover :
  forall g k, Clique (g, k) <-> KVertexCover (complementGraph g) (|(V g)| - k).
Proof.
  intros g k.
  split.
```

```
- intros [L Hclique].
  exists (list_diff (elem (V g)) L).
  unfold isKVertexCover. split.
  + (* Step 1: Prove that the size of the vertex cover is |V| - k *)
    rewrite list_diff_length.
    destruct Hclique as [Hlen Hcl].
    assert (Hcount: count_in_list (elem (V g)) L = length L).
    {
      rewrite count_in_list_equals_length_if_clique.
      reflexivity.
    }
    remember (count_in_list (elem (V g)) L) as cnt.
    rewrite Hcount in Heqcnt.
    rewrite Hlen in Heqcnt.
    rewrite Heqcnt.
    reflexivity.
  + (* Step 2: Prove that the resulting set is a valid vertex cover *)
    unfold isVertexCover. split.
    * intros v1 v2 HE_compl.
      unfold E_complement in HE_compl.
      destruct Hcl as [Hedge _].
      destruct (in_dec (fun x y => eqType_dec x y) v1 L) as [Hvin1 | Hnin1];
      destruct (in_dec (fun x y => eqType_dec x y) v2 L) as [Hvin2 | Hnin2].
      -- exfalso. assert (Hneq: v1 <> v2) by auto.
         specialize (Hedge v1 v2 Hvin1 Hvin2 Hneq).
         contradiction HE_compl.
      -- right. apply list_diff_in_iff; auto.
      -- left. apply list_diff_in_iff; auto.
    * apply dupfree_list_diff. apply dupfree_elem.
- intros [S Hcover].
  exists (list_diff (elem (V g)) S).
  unfold isKClique. split.
  + (* Step 1: Prove that the size of the resulting set is exactly k *)
    rewrite list_diff_length.
    assert (Hdiff_size: |V g| - |S| = k) by lia.
    assert (Hcover_modified: isKVertexCover |S| S) by lia.
    assert (Hcount: count_in_list (elem (V g)) S = |S|).
    {
      apply count_in_list_equals_length_if_vertex_cover
      with (g := complementGraph g).
      exact Hcover_modified.
    }
    rewrite Hcount. exact Hdiff_size.
  + (* Step 2: Prove that the resulting set is a valid clique *)
    split.
    * intros v1 v2 H1 H2 Hneq.
```

```
      assert (Hnv1 : ~ v1 el S) by (apply list_diff_in_iff in H1; tauto).
      assert (Hnv2 : ~ v2 el S) by (apply list_diff_in_iff in H2; tauto).
      assert (Hcomp_edge: ~ E_complement (v1, v2)).
      {
        unfold isVertexCover in Hcover. destruct Hcover as [_ Hvc_cover].
        unfold E_complement. simpl. intro HnE.
        apply Hvc_cover in HnE. destruct HnE as [Hin1 | Hin2]; contradiction.
      }
      apply NNPP in Hcomp_edge. exact Hcomp_edge.
    * apply dupfree_list_diff. apply dupfree_elem.
Qed.
```

5 Conclusion

We have presented a formal reduction from the Clique problem to the Vertex Cover problem and proven that this reduction is correct. This formalization demonstrates the applicability of proof assistants like Coq in verifying classical results in computational complexity theory.