GPLAG: Detection of Software Plagiarism by PDG Analysis

...

G Neha (IMT2014018) & Meghana Kotagiri (IMT2014034)

Contents:

Primary Objective:

To study how to detect core-part plagiarism both accurately and efficiently.

Problem Statement

Suppose the original program P and the plagiarism suspect P' are represented by PDG: G and G' respectively. Then the problem of plagiarism detection boils down to two sub-problems:

- Given $g \in G$ and $g' \in G'$, how can we decide whether g' is a plagiarized PDG of g?
- How to efficiently locate real-plagiarized PDG pairs?

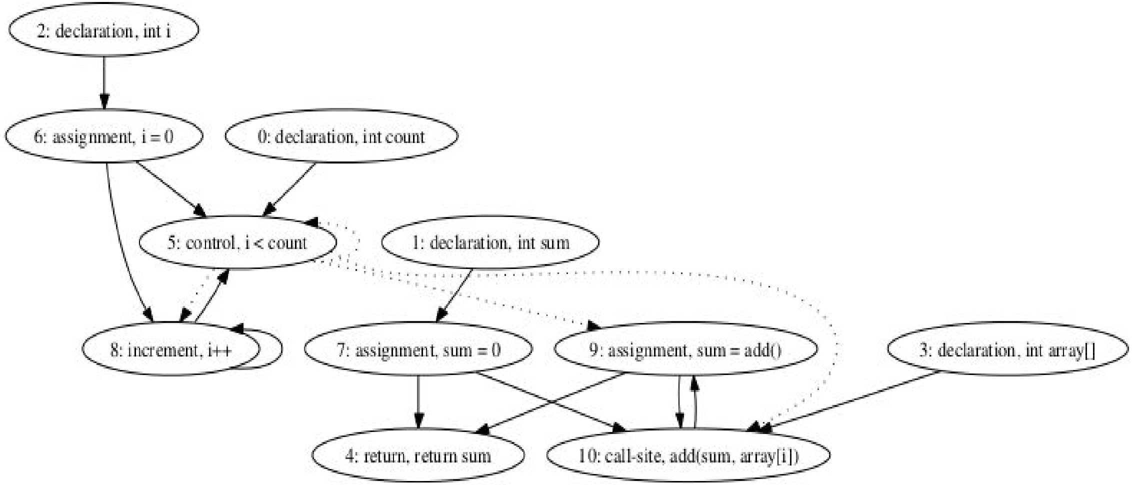
Key Definitions:

1. **Program Dependence Graph:**

    The program dependence graph G for a procedure P is a 4-tuple element

    $G = (V, E, \mu, \delta)$, where

    - V is the set of program vertices in P
    - $E \subseteq V \times V$ is the set of dependency edges, and $|G| = |V|$
    - $\mu : V \rightarrow S$ is a function assigning types to program vertices,
    - $\delta : E \rightarrow T$ is a function assigning dependency types, either data or control, to edges.



(a) Program Dependence Graph of the Procedure `sum`

```
int sum(int array[], int count)
{
  int i, sum;
  sum = 0;
  for(i = 0; i < count; i++){
    sum = add(sum, array[i]);
  }
  return sum;
}

int add(int a, int b)
{
  return a + b;
}
```

(b) Summation over an Array

Key Definitions:

2. **Graph Isomorphism:**

A bijective function $f : V \rightarrow V$ is a graph isomorphism from a graph $G = (V, E, \mu, \delta)$ to a graph $G' = (V', E', \mu', \delta')$ if

- $\mu(v) = \mu'(f(v))$,
- $\forall e = (v_1, v_2) \in E, \; \exists\, e' = (f(v_1), f(v_2)) \in E'$ such that $\delta(e) = \delta(e')$,
- $\forall e' = (v_1', v_2') \in E', \; \exists\, e = (f^{-1}(v_1'), f^{-1}(v_2')) \in E$ such that $\delta(e') = \delta(e)$

Key Definitions:

3. Subgraph Isomorphism:

An injective function $f : V \rightarrow V$ is a subgraph isomorphism from G” to G if there exists a subgraph G’ $\subset$ G such that f is a graph isomorphism from G” to G’.

4. γ-Isomorphic:

A graph G is γ-isomorphic to G’ if there exists a subgraph $S \subseteq G$ such that S is subgraph isomorphic to G’ , and $|S| \geq \gamma|G'|$, $\gamma \in (0, 1]$

Plagiarism Disguises:

1. **Format Alteration:** Inserting and removing blank statements/ comments.
2. **Identifier Renaming:** Identifier names are changed without violating program correctness.
3. **Statement Reordering**: Program statements are reordered without causing errors and affecting sequential dependencies.
4. **Control Replacement:** Replacing while with for, changing if conditions to their negations.
5. **Code Insertion:** Immaterial code insertion which doesn't affect original program logic.

```c
static void
make_blank (struct line *blank, int count)
{
  int i;
  unsigned char *buffer;
  struct field *fields;

  blank->nfields = count;
  blank->buf.size = blank->buf.length = count + 1;
  blank->buf.buffer = (char*) xmalloc (blank->buf.size);
  buffer = (unsigned char *) blank->buf.buffer;
  blank->fields = fields =
   (struct field *) xmalloc (sizeof (struct field) * count);

  for (i = 0; i < count; i++){
      ...
  }
}
```

Original Code

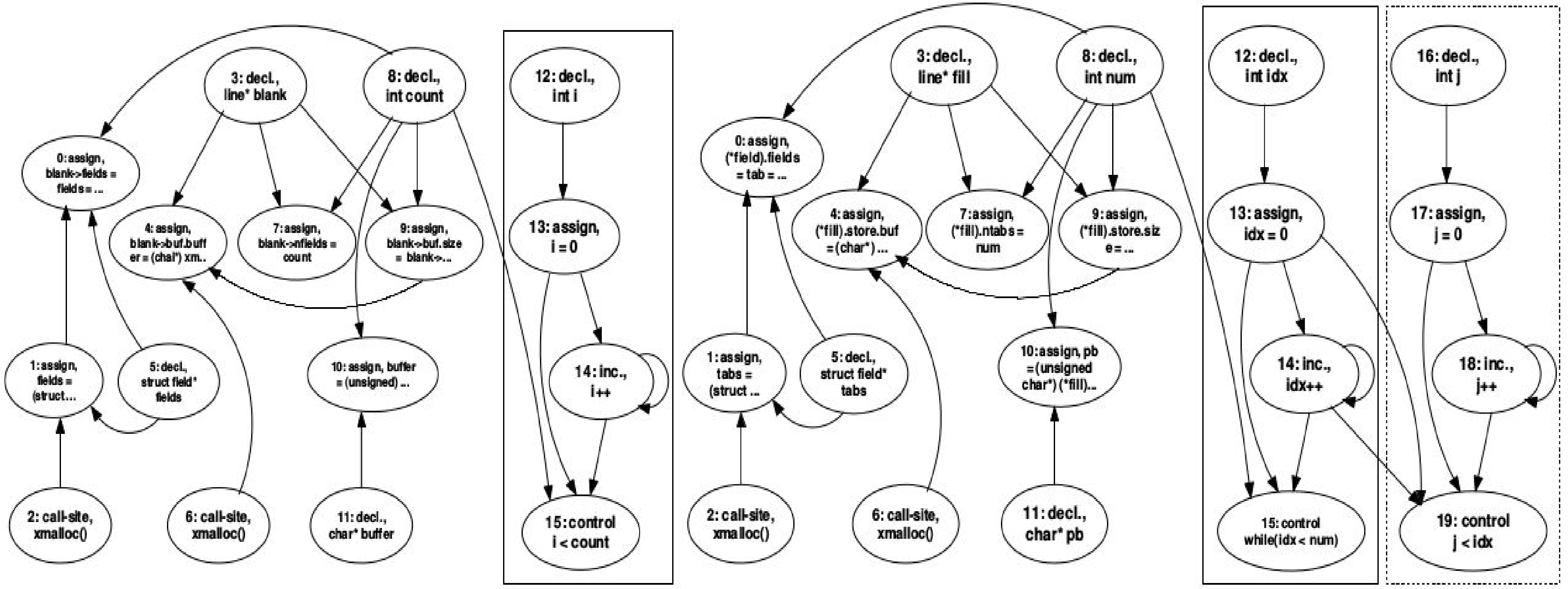
```c
static void
fill_content(int num, struct line* fill)
{
  (*fill).store.size = fill->store.length = num + 1;
  struct field *tabs;
  (*fill).fields = tabs = (struct field *)
                 xmalloc (sizeof (struct field) * num);
  (*fill).store.buffer = (char*) xmalloc (fill->store.size)
  (*fill).ntabs = num;
  unsigned char *pb;
  pb = (unsigned char *) (*fill).store.buffer;

  int idx = 0;
  while(idx < num){ // fill in the storage
    ...
    for(int j = 0; j < idx; j++)
      ...
    idx++;
  }
}
```

Plagiarized Code

Why PDG based plagiarism detection works?

CLAIM 1	Restricted to the five kinds of disguises, if g ($g \in G$) is subgraph isomorphic to g' ($g' \in G'$), the corresponding procedure of g' is regarded plagiarized from that of g.
CLAIM 2	If g ($g \in G$) is γ-isomorphic ($0 < \gamma \leq 1$) to g' ($g' \in G'$), the corresponding procedure of g is regarded plagiarized from that of g', where γ is the mature rate for plagiarism detection.



(a) PDG of the Original Code

(b) PDG of the Plagiarized Code

Proposed Solution:

1. Pruning the plagiarism search space by using filters
2. Applying γ-isomorphism for a PDG pair (g, g'), g ∈ G and g' ∈ G'

Pruning Plagiarism space

1. Lossless filter:
    - PDGs smaller than an interesting size K are excluded from both G and G’
    - Based on the definition of γ-isomorphism, a PDG pair (g, g’), g $\in$ G and g’ $\in$ G’ , can be excluded if $|g'| < \gamma|g|$.
2. Lossy filter:
    - Take vertex histogram as a summarized representation of each PDG
    - PDG g is represented by $h(g) = (n_1 , n_2 , \cdots , n_k)$, where n_i is the frequency of the ith kind of vertices
    - Similarity between g and g’ in terms of their vertex histograms

Proposed Algorithm

Algorithm 1 GPLAG($\mathcal{P}$, $\mathcal{P}'$, K, γ, α)

Input: $\mathcal{P}$: The original program
$\mathcal{P}'$: A plagiarism suspect
K: Minimum size of nontrivial PDGs, default 10
γ: Mature rate in isomorphism testing, default 0.9
α: Significance level in lossy filter, default 0.05
Output: $\mathcal{F}$: PDG pairs regarded to involve plagiarism

$\mathcal{G}$ = The set of PDGs from $\mathcal{P}$
$\mathcal{G}'$ = The set of PDGs from $\mathcal{P}'$
$\mathcal{G}_K = \{g | g \in \mathcal{G}$ and $|g| > K\}$
$\mathcal{G}'_K = \{g' | g' \in \mathcal{G}$ and $|g'| > K\}$
for each $g \in \mathcal{G}_K$
    let $\mathcal{G}'_{K,g} = \{g' | g' \in \mathcal{G}'_K, |g'| \geq \gamma |g|,\ (g, g')$ passes filter$\}$
    for each $g' \in \mathcal{G}'_{K,g}$
        if g is γ-isomorphic to g'
            $\mathcal{F} = \mathcal{F} \cup (g, g')$
return $\mathcal{F}$;

THANK YOU