Improving PDG Vector Creation for AnDarwin

Julia Matsieva • ECS 235A • Fall 2012

UC Davis

December 6, 2012

Background - AnDarwin

AnDarwin project identifies plagiarized Android applications by

- constructing a *program dependency graph* for each application
- converting connected components of each PDG into vectors
- using Locality Sensitive Hashing algorithm to identify clusters of similar vectors

Background - AnDarwin

Advantages of this approach

- avoid solving *maximum common subgraph isomorphism* problem on PDG's, which is known to be NP-hard
- avoid pairwise comparisons between all n Android programs in the data set, which would require $O(n^2)$ comparisons

Background - PDG

A *program dependency graph* G is constructed by

- creating a node for each statement s in the program
- for each pair of statements s, t creating edge (s, t) if there is a variable in t whose value depends on statement s

Thus, PDG's are resistant to code reordering, variable renaming and other simple obfuscation techniques.

Background - PDG Vectors

AnDarwin constructs d-dimensional PDG vector v by

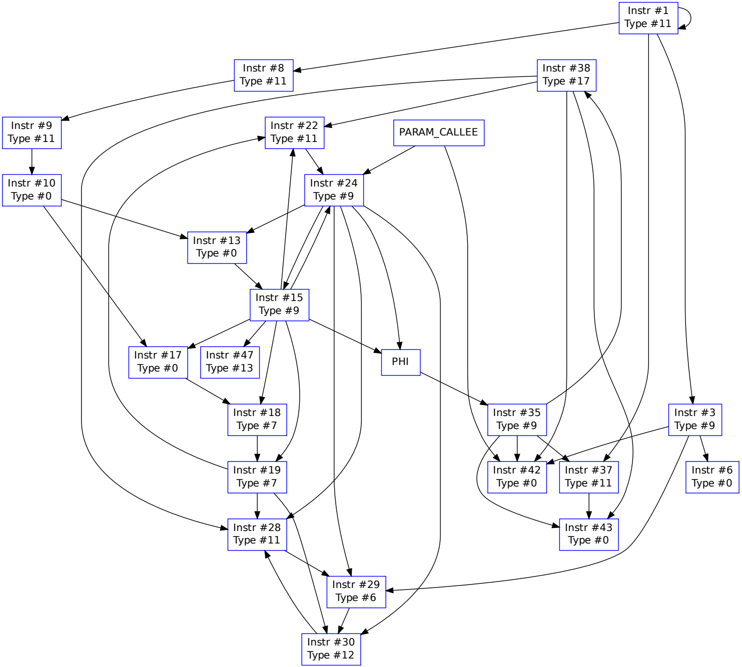
- classifying program statements into d types, i.e., conditionals, binary operations, etc.
- selecting an ordering on the types of statements in the program
- setting the i^{th} component of v to be the number of statements of type i found in the PDG

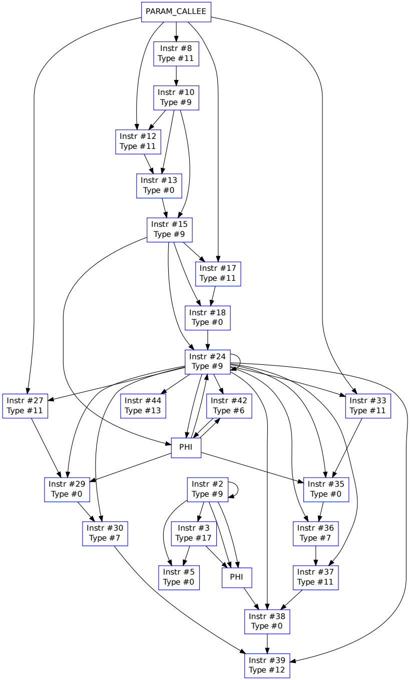
Background - PDG Vectors

AnDarwin constructs d-dimensional PDG vector v by

- classifying program statements into d types, i.e., conditionals, binary operations, etc.
- selecting an ordering on the types of statements in the program
- setting the i^{th} component of v to be the number of statements of type i found in the PDG

Unfortunately PDG vectors only encode node count and do not contain any structural information about the graph


Instr #1 Type #11
Instr #8 Type #11
Instr #38 Type #17
Instr #9 Type #11
Instr #22 Type #11
PARAM_CALLEE
Instr #10 Type #0
Instr #24 Type #9
Instr #13 Type #0
Instr #15 Type #9
Instr #17 Type #0
Instr #47 Type #13
PHI
Instr #18 Type #7
Instr #35 Type #9
Instr #3 Type #9
Instr #19 Type #7
Instr #42 Type #0
Instr #37 Type #11
Instr #6 Type #0
Instr #28 Type #11
Instr #43 Type #0
Instr #29 Type #6
Instr #30 Type #12



PARAM_CALLEE
Instr #8
Type #11
Instr #10
Type #9
Instr #12
Type #11
Instr #13
Type #0
Instr #15
Type #9
Instr #17
Type #11
Instr #18
Type #0
Instr #24
Type #9
Instr #27
Type #11
Instr #44
Type #13
Instr #42
Type #6
Instr #33
Type #11
PHI
Instr #29
Type #0
Instr #2
Type #9
Instr #35
Type #0
Instr #30
Type #7
Instr #3
Type #17
Instr #36
Type #7
Instr #5
Type #0
PHI
Instr #37
Type #11
Instr #38
Type #0
Instr #39
Type #12

Proposal

Construct a $2d$-dimensional PDG vector v by

- classifying program statements into d types, i.e., conditionals, binary operations
- selecting an ordering on the types of statements in the program
- setting the i^{th} component of v to be the number of statements of type i found in the PDG

Proposal

Construct a $2d$-dimensional PDG vector v by

- classifying program statements into d types, i.e., conditionals, binary operations
- selecting an ordering on the types of statements in the program
- setting the i^{th} component of v to be the number of statements of type i found in the PDG
- setting the $(d+i)^{th}$ component of v to be the *max out-degree* of the statements of type i

Advantages

Advantages

Recording max out-degree

- capture some measure of the *importance* of the most relevant statement to the rest of the program

Advantages

Recording max out-degree

- capture some measure of the *importance* of the most relevant statement to the rest of the program
- adds distance between some false positives;
  does not create any false negatives

Advantages

Recording max out-degree

- capture some measure of the *importance* of the most relevant statement to the rest of the program
- adds distance between some false positives;
  does not create any false negatives
- harder to tamper with than *in-degree*, since bogus data members can be invented to depend on other statements

Advantages

Recording max out-degree

- capture some measure of the *importance* of the most relevant statement to the rest of the program
- adds distance between some false positives;
  does not create any false negatives
- harder to tamper with than *in-degree*, since bogus data members can be invented to depend on other statements
- not sensitive to small changes in less-important statements

Advantages

Recording max out-degree

- capture some measure of the *importance* of the most relevant statement to the rest of the program
- adds distance between some false positives;
  does not create any false negatives
- harder to tamper with than *in-degree*, since bogus data members can be invented to depend on other statements
- not sensitive to small changes in less-important statements
- decreases distance between vectors created from programs with different node counts but similar structure

Advantages - Performance

The LSH algorithm used by AnDarwin has complexity

$$O(d \sum_{g \in G} |g|^p \log |g|)$$

where d is vector dimension.

Therefore, increasing the vector dimension to $2d$ only increases the runtime by a constant factor.

Disadvantages

- some additional computation time for converting PDG's to vectors
- since we do not have a characterization for the types of graphs induced by the set of Android applications, this method may potentially create many new false positives

Other Ideas/Future work

Other Ideas/Future work

- implementation and testing

Other Ideas/Future work

- implementation and testing
- distorting Euclidean space to account for ease of adding certain types of statements
  (for example, it may be easier to add extra add statements to a program but not extra conditionals)

Other Ideas/Future work

- implementation and testing
- distorting Euclidean space to account for ease of adding certain types of statements
  (for example, it may be easier to add extra add statements to a program but not extra conditionals)
- including *average out-degree* in the vector might also be contain useful structural information about the graph

Other Ideas/Future work

- implementation and testing
- distorting Euclidean space to account for ease of adding certain types of statements
  (for example, it may be easier to add extra add statements to a program but not extra conditionals)
- including *average out-degree* in the vector might also be contain useful structural information about the graph
- implement an automatic method for characterizing false positives

Questions?