Phylogenetic Analysis of Software Using Cache Miss Statistics

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Sebastiano Verde, Simone Milani, Giancarlo Calvagno - University of Padova
{sebastiano.verde, simone.milani, calvagno}@dei.unipd.it

Abstract

While the phylogenetic analysis of multimedia documents keeps being investigated, some recent studies have shown the possibility of re-using the same strategies to analyze the evolution of computer programs (Software Phylogeny), considering its several applications spanning from copyright enforcement to malware detection.
This work presents a solution for reconstructing the phylogenetic dependencies among different releases of a given program. The proposed method collects cache miss statistics during the program execution, builds a dissimilarity matrix from the results, and then estimates the corresponding Software Phylogenetic Tree (SPT) using a refined minimum spanning tree algorithm.

Cache profiling

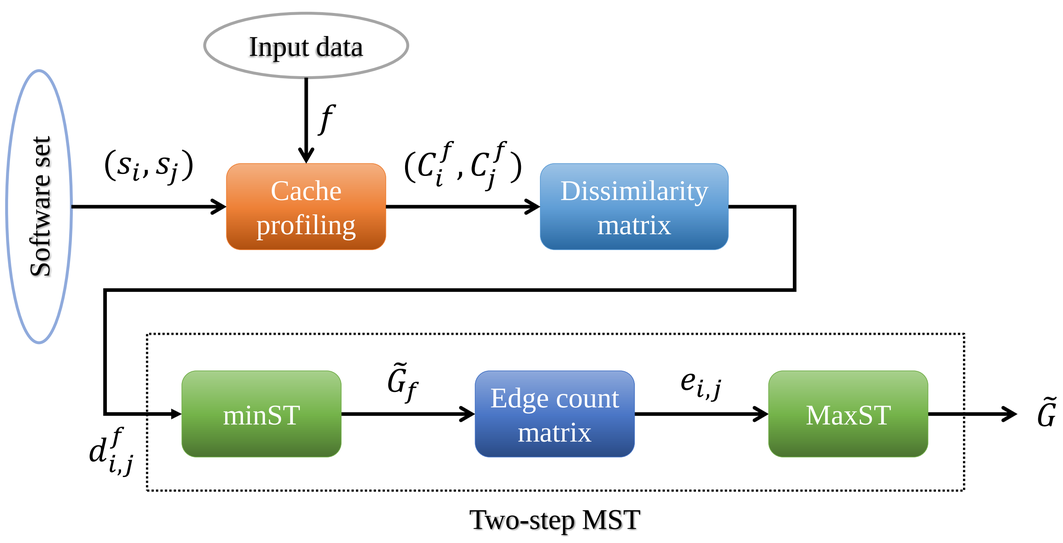
```
--------------------------------------------------------------------------------
Ir        I1mr ILmr Dr        D1mr  DLmr  Dw        D1mw   DLmw    file:function
--------------------------------------------------------------------------------
8,821,482    5    5 2,242,702 1,621    73 1,794,230      0      0  getc.c:_IO_getc
5,222,023    4    4 2,276,334    16    12   875,959      1      1  concord.c:get_word
2,649,248    2    2 1,344,810 7,326 1,385         .      .      .  vg_main.c:strcmp
2,521,927    2    2   591,215     0     0   179,398      0      0  concord.c:hash
2,242,740    2    2 1,046,612   568    22   448,548      0      0  ctype.c:tolower
1,496,937    4    4   630,874 9,000 1,400   279,388      0      0  concord.c:insert
  897,991   51   51   897,831    95    30        62      1      1  ???:???
  598,068    1    1   299,034     0     0   149,517      0      0  ../sysdeps/generic/lockfile.c:__flockfile
  598,068    0    0   299,034     0     0   149,517      0      0  ../sysdeps/generic/lockfile.c:__funlockfile
  598,024    4    4   213,580    35    16   149,506      0      0  vg_clientmalloc.c:malloc
  446,587    1    1   215,973 2,167   430   129,948 14,057 13,957  concord.c:add_existing
  341,760    2    2   128,160     0     0   128,160      0      0  vg_clientmalloc.c:vg_trap_here_WRAPPER
  320,782    4    4   150,711   276     0    56,027     53     53  concord.c:init_hash_table
  298,998    1    1   106,785     0     0    64,071      1      1  concord.c:create
  149,518    0    0   149,516     0     0         1      0      0  ???:tolower@@GLIBC_2.0
  149,518    0    0   149,516     0     0         1      0      0  ???:fgetc@@GLIBC_2.0
   95,983    4    4    38,031     0     0    34,409  3,152  3,150  concord.c:new_word_node
   85,440    0    0    42,720     0     0    21,360      0      0  vg_clientmalloc.c:vg_bogus_epilogue
```

Using Cachegrind tool. Cache statistics are represented with 9 features per function.

Cache type	**Instructions**	**Data (reads)**	**Data (writes)**
Total cache accesses	I_r	D_r	D_w
1st-level cache misses	$I1_{mr}$	$D1_{mr}$	$D1_{mw}$
Last-level cache misses	IL_{mr}	DL_{mr}	DL_{mw}

M functions called by the program $\longrightarrow$ $C \triangleq$ cache data matrix ($M \times 9$)

Software dissimilarity



Given a pair of matrices (C_i^f, C_j^f), we define their dissimilarity as

$$d_{i,j}^f = \|C_i^f - C_j^f\|_2, \tag{1}$$

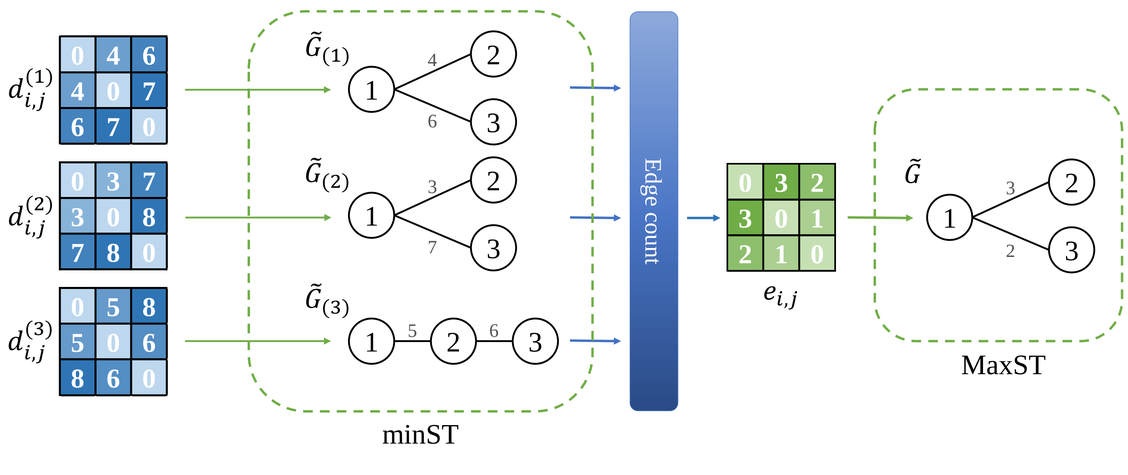
where $\|\cdot\|_2$ denotes the matrix 2-norm.

Different programs may use different functions $\Rightarrow$ incompatible matrix dimensions. Possible workarounds:

1. intersection of the two function sets;
2. union of the two function sets (missing functions represented with zero-vectors).

Two-step minimum spanning tree

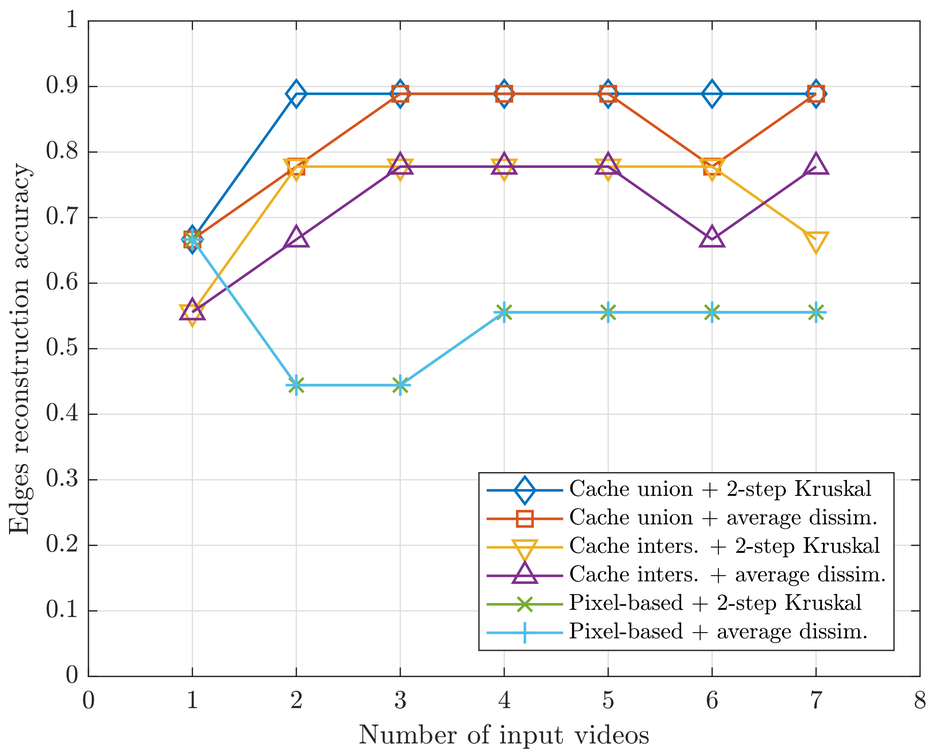
The algorithm allows the estimation of a single minimum spanning tree from multiple realizations of the dissimilarity matrix, through a twofold application of a standard MST algorithm.



1. $\forall f$ run Kruskal's algorithm on $d_{i,j}^f \longrightarrow$ min. spanning tree $\tilde{G}_f$
2. $\forall (i,j)$ count occurrences $e_{i,j}$ of edge (i,j) among all the estimated trees $\longrightarrow$ edge count matrix $E = [e_{i,j}]$
3. Run Kruskal's algorithm on $-E \longrightarrow \tilde{G}$

Method comparison

Percentage of correct edges vs. no. input videos in Thor codec. Comparison between 2-step Kruskal ($2K$) and average dissimilarity ($\bar{D}$) combined with cache intersection, union and pixel-based method [1].



Software set and results

Analyzed software				Percentage of correct edges				
Software name	**Type**	**Input**	**Releases**	$2K_{\text{union}}$	$\bar{D}_{\text{union}}$	$2K_{\text{inters}}$	$\bar{D}_{\text{inters}}$	*Pixel-based* [1]
Thor	Video coding	7 videos	10	0.89	0.89	0.67	0.78	0.56
OpenJPEG	Image coding	30 images	6	0.60	0.60	0.60	0.60	0.20
RNNoise	Audio denoising	40 audios	5	0.50	0.25	0.50	0.25	-
LZ4	Data compression	30 images	7	0.67	0.67	0.67	0.67	-
LIBSVM	Machine learning	5 datasets	7	0.83	0.83	0.50	0.50	-

[1] Sebastiano Verde, Simone Milani, and Giancarlo Calvagno,"Phylogenetic analysis of multimedia codec software," in *Proc. of EUSIPCO 2018*, Sept. 2018.