J Comput Virol (2009) 5:335–343
DOI 10.1007/s11416-008-0100-6

Evaluation of malware phylogeny modelling systems using automated variant generation

Matthew Hayes · Andrew Walenstein · Arun Lakhotia

Received: 20 January 2008 / Revised: 1 July 2008 / Accepted: 8 July 2008 / Published online: 29 July 2008
© Springer-Verlag France 2008

Abstract A malware phylogeny model is an estimation of the derivation relationships between a set of malware samples. Systems that construct phylogeny models are expected to be useful for malware analysts. While several such systems have been proposed, little is known about the consistency of their results on different data sets, about their generalizability across different types of malware evolution. This paper explores these issues using two artificial malware history generators: systems that simulate malware evolution according to different evolution models. A quantitative study was conducted using two phylogeny model construction systems and multiple samples of artificial evolution. High variability was found in the quality of their results on different data sets, and the systems were shown to be sensitive to the characteristics of evolution in the data sets. The results call into question the adequacy of evaluations typical in the field, raise pragmatic concerns about tool choice for malware analysts, and underscore the important role that model-based simulation is expected to play in evaluating and selecting suitable malware phylogeny construction systems.

1 Introduction

Of the millions of malicious programs known to anti-virus companies, the clear majority of them are variants of some previously generated program [21]. That is, malware authors modify, reuse, maintain, and tweak. They are also known to share code, use libraries, and employ generators and kits. These are examples of *software evolution*, in which *derivation* relationships are created between malicious programs, i.e., between variants. As a result, it is understood that there exist various families or species of malware with relationships between them. This creates a need to identify, understand, relate, classify, organize, and name the variants, species, or families.

In biology, a "phylogeny" is the set of derivation relationships between a set of species. The actual phylogenetic relationships are rarely, if ever known in biology. Rather, they must be inferred or "reconstructed" [17] through painstaking sleuthing and analysis, often with the help of automated systems that can generate estimated models of the phylogenies. Such a system can be called a "phylogeny model generator", or "PMG" for short. Similarly in malware, the phylogenetic relationships are frequently unknown for malicious programs, and so the phylogenetic models likewise need to be constructed. Tools to do so are expected to help malware analysts. Several malware PMGs have been proposed in the literature to meet this specific goal.

Existing PMGs have been subjected to relatively simple evaluations. The system evaluations we are aware of assess only a single PMG, tend to be informal and non-quantitative in assessment criteria, and operate on a limited or ad hoc collection of evolution histories. Moreover, the question of how to properly evaluate malware PMGs has not yet been addressed in depth. One question of particular importance is whether or not tests on limited sets of malicious samples can reasonably be considered sufficient for evaluation since: (a) phylogeny constructors may produce variable results depending upon the specific test set, and (b) they may be sensitive to the class of malware evolution present in the test set. Important questions are unanswered regarding such evaluations: How important is random sampling? What measures of goodness are suitable? What evaluation approaches are helpful?

M. Hayes is presently at Case Western Reserve University.

M. Hayes (✉) · A. Walenstein · A. Lakhotia
Center for Advanced Computer Studies,
University of Louisiana at Lafayette, Lafayette, LA, USA
e-mail: meh1666@louisiana.edu

This paper explores answers to such questions using a quantitative, model-driven simulation approach to evaluation. Models of distinct malware evolution classes are proposed, and then used to drive an evolution simulation that constructs artificially generated reference corpora. Each corpus consists of a collection of related variants, plus an explicit record of their derivation relationships (i.e., the phylogeny). Two forms of evolution models are employed: a straightforward code-mutation based model that simulates evolution by fine-grained program modification, and a feature-based model that simulates a coarser evolution by addition of new features among family members. These models, while limited, are utilized to begin exploring the questions posed above.

A study was conducted using reference corpora generated by the two evolution simulators. Reference sets were generated, and the outputs of two different PMGs were compared to the reference phylogenies. Graph distance measures were used to quantify the divergence from the reference phylogenies. The results show high variance in the output quality for different samples from the same population; the variance calls into question the sufficiency of evaluating phylogeny model constructors using limited reference corpora. The results of the study also highlight the importance of considering accuracy versus stability or reliability in the constructor. Finally, the study illustrates the important role that the quantitative approach may play in evaluating phylogeny model construction systems.

Problems in evaluating malware phylogenies are reviewed in Sect. 2, the evaluation approach through model-based artificial evolution systems is introduced in Sect. 3, and the study using these is described in Sect. 4. Conclusions and implications are presented in Sect. 5.

2 Problems in evaluating malware phylogeny model constructors

A variety of approaches to constructing malware phylogeny models have been proposed. Table 1 summarizes the known publications using the PMG taxonomy of Karim et al. [12]. The taxonomy distinguishes generators on the basis of three properties: (1) what features of the programs they examine, (2) the class of graphs they generate, and (3) the construction algorithm used to generate the graph.

The rightmost column of Table 1 indicates the sort of evaluation reported in the relevant publication. In that column, "demonstration" indicates mere demonstration, i.e., that a model can be constructed, but no special consideration is given to ensure the sufficiency of the data set, and no formal comparison to a reference phylogeny is provided. The term "informal evaluation" refers to a demonstration with some ad hoc discussion about the sufficiency of the test data and accuracy of the results. The most carefully constructed evaluation of phylogeny constructors in the field is arguably that of Wehner [22]. Wehner used a convenience sample with no reference phylogeny and informally evaluated the accuracy of the phylogeny models generated; while she quantitatively and formally examined a derived classification heuristic, it evaluates only restricted properties of the trees (i.e., rough grouping). It is listed as a "semi-formal" evaluation because of these properties.

Table 1 makes it clear that no comprehensive assessment is known for any of the systems in the list. None of them employed systematic random sampling or quantitative measures of accuracy with respect to a reference phylogeny. While the bar for evaluation is thus low in relation to that normally desired in science and engineering, it must be acknowledged that the question of how to evaluate such systems has not yet been rigorously addressed. One can expect that progress in the field would be accelerated if more systematic evaluations methods were developed. One important step in doing so is defining suitable reference corpora, i.e., sets of representative samples along with their derivation relationships.

At least two different approaches can be pursued for generating the reference corpora needed for improved evaluation: (1) using actual malware samples collected, or (2) from artificially generated samples. In either case, the phylogeny models created by the PMGs are compared against the reference, i.e., correct data set. In the former approach, sets of (authentic) samples are collected, and their actual relationships are determined through investigation or through knowledge of their construction. In the latter approach, a model of malware evolution is used to drive a simulation which not only generates the data set, it records the actual derivation relationships.

So far in malware phylogeny research, the approach through authentic sample collection is typical, whereas in biology, the simulation based approach is the de facto standard [17]. Many problems are confronted with either approach. Several issues for the hand-crafted reference corpora approach are reviewed below; these will be used to motivate our exploration of the model-based simulation approach. Since phylogeny model evaluation has been studied in biology, points of comparison are offered when relevant.

2.1 Measurement and comparison problems

A key issue in evaluating PMGs is how well their outputs correspond to the actual derivation relationships. In biology, the correspondence has been measured using formal measures of graph differences, or graph *distances*. The so-called "Nodal Distance" [2] is a simple measure for comparing arbitrary graphs by measuring the sum of the differences in path lengths between two graphs. Calculation is straight-

Table 1 Malware phylogeny systems and their evaluations

System	Features	Output type	Generation algorithm	Evaluation
Goldberg et al. [7]	20-grams on bytes	Directed acyclic graph	Variants of minimum phyloDAG	None
Erdélyi and Carrera [5]	Call graph	Binary tree	Graph similarity + unspecified clusterer	Demonstration
Karim et al. [12]	n-perms on operations	Binary tree	Program similarity + UPGMA clusterer	Informal
Ma et al. [15]	Instruction sequences	Binary tree	Exedit distance + agglomerative clustering	Informal
Wehner [22]	Bytes	Binary tree	Normalized compression distance + unspecified clusterer	Semi-formal

forward: the differences in the path lengths between each corresponding pair of nodes in the graphs are summed. The "Robinson–Foulds" distance [19] is a more popular edit-distance approach, but is restricted to trees and its exact solution can be too expensive for some bioinformatics applications, although various approximations have been proposed. Any number of other graph distance or similarity measures might possibly be used.

Whatever graph measure is selected, one inevitable concern is how to interpret the results of the measures. In the ideal case the PMG in question reproduces the phylogeny exactly for any imaginable evolution history. In that case, the measured distances between the outputs and the reference graph should be 0. Since the ideal is unlikely to ever be met, the problem reverts an engineering concern of managing trade-offs. One traditional engineering goal is to ensure that, on average, the difference between the constructed models and the true phylogeny should be as small as manageable. Evaluation of a PMG would examine the mean distances; a comparison between two PMGs could compare the mean distances to see if there is a significant difference. While this approach seems straightforward, it is possible that other engineering goals might also be sought. For example, if two systems produce results with similar mean distances, but one has much higher variance and occasionally generates extremely poor results, then the user may have reasons not to choose the PMG with the better mean score. That is, average distance captures only a portion of the concerns that a user may have. Without relevant data about the performance of existing PMGs to consult, however, it is not possible at this time to know how important the variance issue is.

2.2 Difficulty of using authentic data sets

One of the established problems in phylogeny constructor evaluation in biology is the difficulty of constructing the reference corpora that can be used to compare the constructed phylogenies against [18]. The true derivation relationships may not be known and, indeed, the techniques one might use to try to establish such a reference model may involve the very phylogeny reconstruction techniques under evaluation. In order to advance the field past case studies it is desirable that multiple reference corpora be constructed; moreover the mechanics of statistical hypothesis testing make it desirable that the reference models are proved to be selected randomly from a population of family histories with common evolution characteristics. The need for representative samples of reasonable sizes exacerbates the problem of hand-constructing of the reference models.

This problem may be addressed, in part, through aggregation and sharing of effort. It may be feasible to establish standardized, shareable reference data sets, complete with carefully checked derivation information. This approach is similar in spirit to the TREC efforts in the field of text retrieval [3], as well as to benchmarking efforts in software engineering [20]. In this vein, standardized malware data sets could be constructed, much like the WildList effort for anti-virus testing [16]. Unfortunately, the fact that malware is involved may add special challenges to sharing authentic reference corpora: sharing malicious samples is notoriously difficult in practice, and introduces many legal and safety challenges. While shareability of reference models is perhaps not strictly required for the field to advance, if they cannot be shared then key pillars of science and engineering are likely to be affected in practice: independent repeatability and verification of studies and fair comparison between systems. We know of no instance of malware phylogeny modelling system evaluators sharing their data sets to enable direct comparison of systems.

2.3 Variation and idiosyncrasy in malware evolution

In biology it may be reasonable to assume a uniform and stable set of mechanics and characteristics for evolution. The same sorts of transcription errors may be expected to occur, for example, in large numbers of species over long periods of time. Malware evolution may not enjoy stability and universality to the same degrees. For example, certain malware families may evolve in special ways due to the specific tools the malware author employs, the particular ways that the author attacks the defense infrastructure and, in general, the constantly and rapidly changing nature of the malware/anti-malware battle. Further, mutants can be generated automatically through distinct forms of polymorphism and metamorphism [1].

If one can expect that malware evolution be highly variable and idiosyncratic, it creates additional problems for the approach through hand-crafted reference sets. Specifically, it calls into further question the sufficiency of a small or fixed number of reference sets, as they may fail to represent the overall and varied characteristics of malware evolution.

3 The approach through artificial evolution histories

The use of artificial evolution histories can address many of the problems listed in Sect. 2. Consider the efforts of Nakhleh et al. [17] or Rambaut et al. [18], for example. They construct reference models using simulations of genetic evolution. In their approaches, they randomly selected (i.e., created) evolution paths and then simulated mutation events to match those paths.

A similar approach may be taken in creating artificially constructed malicious reference sets. Several benefits may accrue from the use of simulations based on evolution models:

1. Large numbers of reference sets may be feasibly generated. This reduces the threat to external validity posed by using only a small number of hand-constructed reference sets, while enabling the measuring of both mean performance and variance.
2. The characteristics of the evolution histories can be tailored to match the type of evolution history the user is expecting. Thus, unlike biology in which a modeller may seek to find an accurate and general model, malware phylogeny constructor evaluators may use only limited-purpose but relevant models.
3. If the simulator creates benign samples, or uses existing malware samples in benign ways, the threat in evaluation can be controlled, and it may be simpler to share the outputs or the simulator itself.

While these are clear benefits for the artificial history approach, the approach does suffer one important drawback: in order to construct artificial malware evolution histories, suitable models of evolution are needed so that an evolution simulator can be constructed. This simulator would generate the required reference data, namely, a corpus of samples related through derivation. It would also generate the corresponding reference derivation graph. Thus a question is raised as to what models could be used.

One approach to answering this question is to adopt a goal of creating an "ideal" malware evolution model that captures all important characteristics of known evolution, and could thus serve as an effective proxy for reality. While this is a daunting task well beyond the scope of this work, it could perhaps be approached incrementally. However, it is not clear that a comprehensive and authentic model is absolutely required in order to create pragmatically useful evaluations of PMGs.

From a pragmatic point of view, a malware analyst may have only a certain class of malware evolution histories to deal with. In terms of creating a model phylogeny, the analyst's main concern is the selection of a suitable system to use on her particular data. In addition, at the moment there is no reason to believe that a singular PMG can exist that performs optimally on all classes of malware evolution. Said another way, at the moment we can reasonably expect that every existing phylogeny construction system will be associated with some classes of malware evolution for which it performs better than other classes. Moreover, the best tool for the analyst's job may actually be sub-optimal with respect to the full panoply of malware evolution classes. Thus to serve the analyst's practical problem, a comprehensive evolution model is not only not required, it may not be as effective as a restrictive evolution model that matches her specific situation.

Another approach to the evolution modelling challenge, therefore, is to aspire not to create an ideal evolution model, but to produce a toolkit of restricted but useful artificial evolution systems such that each captures essential characteristics of some class of malware evolution. The restricted models will have utility in the case that they are relevant to some non-empty set of analyst situations. Because analyst situations differ, a beneficial quality of the resulting simulator is that it can be in some way parametrized or specialized to customize the artificial evolution to match the analyst's situation. Note that a new matching problem is created: the analyst must select the evolution model that matches her problem best. One possible way of easing the matching problem is to construct models with clearly recognizable characteristics—that is, they generate evolution histories that are in some sense prototypical for a class of evolution types. If a given phylogeny construction system performs well on one of these, the potential user may be able to choose the system for which the prototype seems to match known characteristics best.

The preceding analysis produces a number of research questions that might be explored empirically, including:

1. How variable are the outputs of malware phylogeny constructors? If they vary greatly, it may severely limit the value of small numbers of hand-crafted reference sets.
2. How sensitive are the outputs to different classes of malware evolution? If the types of changes have significant

effects, it may suggest that specialized models be pursued instead of waiting for a comprehensive, idealized model of malware evolution to be developed.

Some evolution simulators are required to explore these questions. The simulations need not be "ideal" models of malware evolution in order to yield interesting answers. We propose here two models that are intended to capture some important but distinct characteristics of malware evolution. Each of these evolution models are inspired by knowledge about software evolution in general, and malware evolution in particular. Neither are intended to be comprehensive models of all different types of malware evolution.

3.1 Non-uniform, mutation-based evolution model

One of the ways of generating simulated biological evolution is to develop a model of the mechanics of genetic change [18]; transcription errors, for example, are one of the ways that mutations are known to occur. A similar approach in malicious software is to start with an authentic sample of malware and then perform a sequence of code-mutation operations on it, recording the derivation. Variations of this approach have been described for the purpose of testing malware detectors [4,6]. One advantage is that a potentially large selection of initial seed programs can be selected as authentic starting points for the artificial evolution history.

When considering a mutation-based model, perhaps the important questions to ask—from an evolution history point of view—are: which mutations does one perform, and what characteristics should the resulting graph of derived samples have as a whole? Consider a probabilistic generator type of simulator that randomly selects from a fixed set of mutation mechanisms. These mechanisms might include, for example, semantics-preserving transformations, and random add/delete/change operations. Control of the evolution class would then amount to selecting the set of mutation mechanisms, and assigning their associated probabilities for being employed. However, it may not be obvious how to use such a system to tailor such systems to match the evolution characteristics desired. For example, it has been pointed out that ordinary software evolution is non-uniform in the sense that changes between versions are frequently discontinuous and characterized by periods of small, localized change interspersed with periods of rapid or more global change [8,23]. A similar concern exists in biology, in which simulations are careful to follow known properties of evolution [9]. If some malware evolves along similar principles, then a mutation-based simulator may fail to capture important characteristics of the evolution class if it generates artificial evolution histories in which the change rates are relatively constant, even if the underlying mutations are randomized because of the probabilistic generation process.

To address this issue we propose a mutation model that is simple and abstract, and yet can generate artificial evolution sets that alternate large and small changes in ways that are consistent with a mixture of probabilistic modification. The model assumes a single mutation type: replacement of either a "small" or a "large" amount of code with new, mutated pieces of code. Any number of different mutation mechanisms (add/delete, permute, etc.) might be used for the mutations. The model assumes small changes between generations happen at a particular ratio to the number of large changes, i.e., a "Small-to-Large" ratio. It also assumes that the small changes are all smaller than a given threshold "Small Threshold", and the large changes all larger. Although the resulting changes sizes will have a bimodal distribution instead of a power function distribution observed by Gorshenev et al. [8], the changes will exhibit the critical property of non-uniformity.

3.2 Feature accretion model

One property of software evolution is commonly discerned: new features creep into code as it is incrementally modified. In malware, this is known to occur as a malicious code base matures and the developers add new exploit or payload capabilities [11]. An evolution simulator for this type of evolution would need to be able to add realistic new code; perhaps in the ideal case, it would automatically create the features, exploits, and payloads that a real malicious program writer would create. One would, of course, expect it to be extremely difficult to create such an automated evolution system (else malware writers might already be using such systems). However, it is possible to simulate some facets of this type of evolution history using an existing mature code base as a starting point.

The idea we propose is to dissect a mature program into sets of independent features and then generate artificial evolution histories that consist entirely of subsets of the original program, with each distinct subset defined by a different set of features. More formally, assume a program M can be decomposed into a set $F = \{f_1, f_2, \ldots, f_k\}$ of features, for some k. The power set $P(F)$ of all feature sets of F is a lattice of size 2^k. Assume that each feature f_i describes one potential behaviour of M, so that the behaviour of a program with a subset of F is defined by the union of the features. Then define a derivation path $D = (d_1, d_2, \ldots, d_l)$ through the lattice starting at point d_1 such that each $d_i + 1 = d_i \cup n$, where n is non-empty and the intersection of n and d_i is empty. That is each evolution step adds one or more new features; it is a model of feature accretion. Then we can define a (rooted) evolution history as a collection of derivation paths starting at a common point and overlapping only at that point. An example of such a derivation tree is shown in Fig. 1.

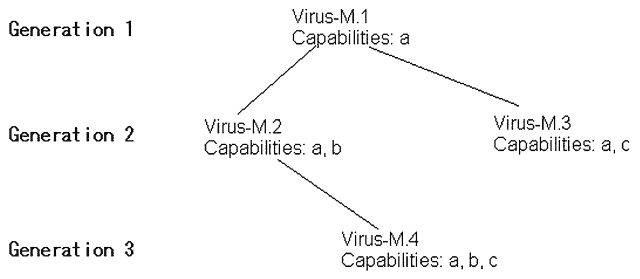


Fig. 1 Example artificial evolution through feature addition

Using this definition it is possible to define a process to randomly select derivation trees when given a set of features of a seed program. If the seed program is the result of a long process of evolution, and this process of evolution worked to gradually add new features, then this random derivation tree selection process serves to select alternative histories by choosing different orderings and paths. The intent is to use the existing features to suggest plausible but artificial alternative derivation histories.

It may be difficult to define an entirely automated process for dissecting the programs and then re-combining the features. We expect the problem to be much harder to solve without the source code for a mature sample. However, in some cases a semi-automated approach may be simple to implement. One possibility is to use a program slicing-based program decomposition scheme to automatically construct executable program subsets [14]. When a source base is available, however, it may be feasible to select groups of program elements (lines, objects, etc.) that form a feature, and then set up a simple infrastructure for compiling only program subsets. We use this approach in the study reported below.

4 Study of phylogeny model constructor behaviour

We performed a two-part study to explore some of the questions raised in the previous sections regarding evaluation of malware phylogeny model constructors. In particular, we wished to provide data that can yield new insight into: the importance of using multiple reference sets, the variability of PMGs, and the degree of generality that can be expected of various PMGs, i.e., their sensitivity to different classes of evolution.

To examine the question of how sensitive malware phylogeny constructors are to evolution class, the output model quality was compared when sampling from different classes of evolution. To examine the question of how important multiple reference sets are, and what measurement issues may arise in analysis, we sought to collect information about the standard deviation in the results of the phylogeny malware constructors for a given treatment.

4.1 Design

Evolution simulators are employed to generate samples from different classes of evolution histories. The experiment followed a factorial design, where the factors were the evolution characteristics of the simulated evolution histories, which were set by selecting a particular evolution simulator and setting its parameters. That is, we ran different evolution simulators with a variety of parameters, generating collections of artificial evolution histories. Treatments consisted of applications of a malware phylogeny model constructor to these collections, producing estimated models (trees), and the dependent variable was the nodal distance between the estimated model and the (known) reference phylogeny. That is, we ran different phylogeny model constructors on the simulated evolution histories and measured how different their outputs were from the reference tree. We used a convenience selection of phylogeny model constructors: Vilo [12], and our own implementation of Wehner's NCD [22]. If these detectors were sensitive to the evolution type, we would expect the dependent measure (distance mean) to vary according to the simulator used and its parameters.

4.2 Apparatus

Two different malware evolution simulators were constructed. The first simulator followed the mutation model of Sect. 3.1. It was constructed as a Perl script that read Windows portable executable (PE) files and wrote them with modified code segments. The simulator takes a PE file to mutate, and two parameters: a ratio of small to large changes, and the threshold value of what is considered a small change. The simulator then constructs an artificial evolution history consisting of a balanced binary tree of depth four (15 samples) by mutating the PE file to create children, with the size of the mutations randomly selected from either a large change population or small change population with the population selected as if by a weighted coin flip with the provided small/large change ratio as the weighting. Mutations are made by replacing code blocks with randomly generated code. Each mutation is randomly split into one to seven different mutations, simulating modifications in multiple places between generations.

The second evolution simulator followed the feature-accretion evolution model of Sect. 3.2. It was specially constructed by modifying a version of the "Agobot" construction kit [11]. The Agobot kit was a suitable selection because its source was available to us, it is mature and has a rich feature set that could be selected from, and the features are, by design, implemented in a highly modular manner so that they can be independently selected. Moreover, though the kit we acquired is considered in-the-zoo, many in-the-wild malware belonging to Agobot or Gaobot family are believed to have

Table 2 Features of Agobot selected for building the lattice of possible variants

1	Use computer name as nickname	8	Enable stealth
2	Login using channel messages	9	Auto start enabled
3	Generate random nickname	10	Start as service
4	Melt original server file	11	Enable Identd
5	Execute topic commands	12	Steal Windows product keys
6	Do speedtest on startup	13	Spam AOL
7	Kill AV processes	14	Sniffer enabled
15	Polymorph on install		

been created through variants of this kit [11]. A subset of 15 features were selected for constructing variations; these are listed in Table 2. The code was segmented by (manually) wrapping the features in `#ifdef`/`#endif` delimiters. Arbitrary combinations could be selected by use of a script that invoked Make and the Microsoft Visual C++ 6.0 compiler. Balanced binary trees of depth four were sampled by starting at the minimum point in the lattice (no features on) and then randomly walking up the lattice.

Adequate care was taken that the samples generated could not accidentally be executed and the samples were destroyed immediately after analysis. Further details about the feature-accretion simulator, including the algorithms used for tree sampling and construction, are provided in Hayes [10].

4.3 Subjects and preparation

A malicious sample from a wild collection was used as the seed to the mutation engine. It was identified by four different anti-virus scanners as belonging to the Agobot family. The two parameters (two factors) to the simulator were varied to create 18 different classes of simulated evolution histories, as follows: Small-To-Large Ratio took on values from {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, and Small Threshold (measured in bytes) from {400, 2400}. A 19th sample of size 20 was constructed using the feature-accretion model simulator.

4.4 Protocol

The simulators were run to create the 19 different samples of size 20 with 15 programs in each instance. Dendrograms were constructed for each simulated evolution from the balanced binary trees, using the relative changes between parent and child to determine how to generate pairs in the dendrogram. Each sample was fed to Vilo and NCD, which generated similarity matrices. The similarity matrices were fed through CLUTO [13] such that CLUTO's UPGMA clusterer constructed dendrograms. The nodal distance between these dendrograms and the reference dendrograms were then measured and recorded. Their means and standard deviation values for each parameter setting were then collected.

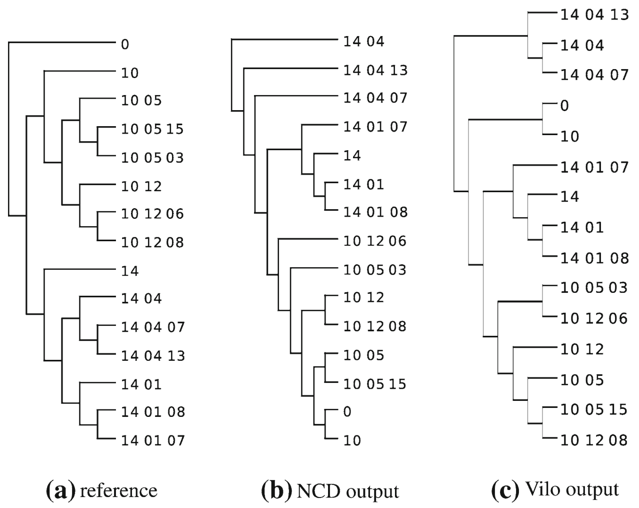


Fig. 2 Example reference and output trees

Table 3 Measures for feature-addition sample

	Mean nodal distance	Standard deviation
NCD	219.7	39.44
Vilo	208.3	35.48

4.5 Results

An example of the reference and constructed trees is shown in Fig. 2. The example is one of the randomly constructed evolution histories using the feature-accretion model. The labels in the leaves indicate the feature numbers included in the program; the numbers correspond to the feature numbers from Table 2. The measures for the feature accretion model are in Table 3. The mean and standard deviation for the mutation simulation are shown in Tables 4 and 5, and the means graphed in Fig. 3.

4.6 Discussion

The data from the sensitivity study, presented in the tables and chart, indicate that the mean distances are affected by the model type and, to a lesser extent, the parameter settings in the models. While this study is limited by the types of evolution models employed, the results appear to signal a need for caution when building or selecting evolution models for evaluation. In particular, if the model does not match the characteristics of the target evolution history, then the evaluation using the simulations may indicate that a suboptimal choice be used.

Table 4 Mean nodal differences across factors

Small-to-Large	NCD,400	Vilo,400	NCD,2400	Vilo,2400
0.9	946.1	1,004.8	975.2	980.8
0.8	975.9	1,019.3	920.4	925.1
0.7	988.1	1,000.7	1,003.7	1,017.9
0.6	1,014.6	1,055.7	1,016.6	1,017.1
0.5	1,054.0	1,094.9	980.4	992.0
0.4	1,091.4	1,082.1	992.8	994.6
0.3	997.8	996.5	929.7	926.7
0.2	969.2	992.9	905.8	917.8
0.1	959.9	934.2	949.6	930.8

Table 5 Standard deviation across factors

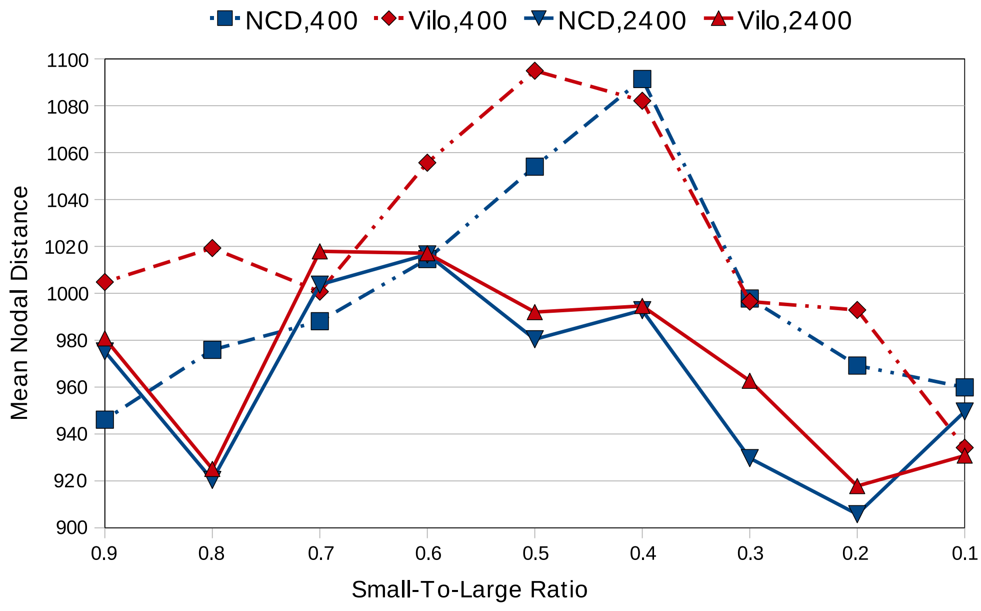
Small-to-large	NCD,400	Vilo,400	NCD,2400	Vilo,2400
0.9	184.56	143.23	132.90	111.01
0.8	205.94	118.72	133.11	158.44
0.7	248.00	103.24	154.89	107.58
0.6	281.69	149.23	118.18	122.98
0.5	217.28	153.73	91.47	110.65
0.4	305.36	131.80	114.42	123.89
0.3	275.23	162.92	100.59	120.09
0.2	232.96	155.43	131.82	103.74
0.1	224.03	132.69	85.21	95.86

Variation is high between individual histories taken from a single population of evolution histories. This fact is captured in Table 2 by the relatively large values of the standard deviation for the case of the accretion model data, or about 18% of the mean. The difference in means is stark when comparing the results across different evolution models. While some variation appears between the mutation models (Fig. 3), the difference between the mutation and feature accretion model is large: from 200 to 1,000.

The study is limited in that only a single measure (nodal distance) is used, and it may be a factor in the variance shown. Nonetheless, similar results were achieved using the Robinson–Foulds measures also. Furthermore, the variance exhibited in the data set appears to present important challenges to the evaluation of phylogeny model construction systems. There are several points that can be considered depending upon the purpose and context of evaluation:

1. The variation calls into question the sufficiency of a small number of tests data sets for evaluation of malware phylogeny model construction systems. It suggests that there may be a need, as in biology, to lean on simulation-based evaluations similar in spirit to the ones in this paper.
2. An anti-malware analyst may value consistency of results in addition to mean performance. For example, if she is constructing a phylogeny model from a specific data set of incoming malware, she may be worried that the result may happen to be egregiously bad, and thus allow a risky piece of software to be misclassified. This possibility suggests that publication of performance results should include indications of consistency in addition to straightforward accuracy.
3. The question of selecting quantitative measures is likely to be critical, especially for the anti-malware analyst. Nodal distance measures the average path distance deviations, but in some circumstances the analyst may be specifically interested in other key measures, such as the number of especially poor classifications. While other

Fig. 3 Mutation chart



measures from biology might be useful, there may be measures of particular interest specifically for malware analysts, such as ones similar to those studied by Wehner [22].

5 Conclusions

In biology, phylogeny model construction systems are normally evaluated using simulations that generate large enough samples that are statistically meaningful, quantitative, and objective tests can be performed. This approach is rare in the field of malware phylogeny model generators, but then evaluation in this field is still effectively in its infancy. This paper describes an approach for simulating evolution histories by breaking apart and then recombining existing malware in order to simulate feature evolution. It argues that variance in performance and sensitivity to evolution characteristics may be likely properties of such systems and, if so, then it raises important questions for evaluators. For practitioners in the anti-malware field, the implication is that evaluations of phylogeny construction tools need to be carefully considered if they use only limited sets of data.

The study in this paper, while limited, raises legitimate concerns and provides positive indication that similar sorts of simulation-based evaluations may become important in the field. If so, then important research may lie in characterizing malware evolution and building appropriate models and simulations.

References

1. Beaucamps, P.: Advanced polymorphic techniques. Int. J. Comput. Sci. **2**(3), 194–205 (2007)
2. Bluis, J., Shin, D.: Nodal distance algorithm: calculating a phylogenetic tree comparison metric. In: Proceedings of the Third IEEE Symposium on Bioinformatics and BioEngineering, pp. 87–94 (2003)
3. Buckley, C., Dimmick, D., Soboroff, I., Voorhees, E.: Bias and the limits of pooling for large collections. Inf. Retr. **10**(6), 491–508 (2007)
4. Christodorescu, M., Jha, S.: Testing malware detectors. In Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis, Boston, MA, USA, pp. 34–44 (2004)
5. Erdélyi, G., Carrera, E.: Digital genome mapping: advanced binary malware analysis. In: Martin, H. (ed.) Proceedings of the 15th Virus Bulletin International Conference, Chicago, IL, USA, pp. 187–197. Virus Bulletin Ltd (2004)
6. Filiol, E., Jacob, G., Le Laird, M.: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. J. Comput. Virol. **3**(1), 23–37 (2007)
7. Goldberg, L., Goldberg, P., Phillips, C., Sorkin, G.: Constructing computer virus phylogenies. J. Algorit. **26**(1), 188–208 (1998)
8. Gorshenev, A.A., Pis'mak, Y.M.: Punctuated equilibrium in software evolution. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **70**(6), (2004). Epub 23 December 2004
9. Harding, E.F.: The probabilities of rooted tree shapes generated by random bifurcation. Adv. Appl. Prob. **3**, 44–77 (1971)
10. Hayes, M.: Simulating malware evolution for evaluating program phylogenies. Master's thesis, Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA, USA, 70504 (2008)
11. Infection Vectors. Agobot and the kitchen sink. Retrieved from http://www.infectionvectors.com/vectors/kitchensink.htm, 17 Feb 2008
12. Karim, M.E., Lakhotia, A.W.A., Parida, L.: Malware phylogeny generation using permutations of code. J. Comput. Virol. **1**(1), 13–23 (2005)
13. Karypis, G.: CLUTO—a clustering toolkit. Technical Report TR 02–017, Deptment of Computer Science, University of Minnesota (2003)
14. Lyle, J.R., Gallagher, K.B.: A program decomposition scheme with applications to software modification and testing. In: Proceedings of the 22nd Annual Hawaii International conference on System Sciences, vol. 2, pp. 479–485 (1989)
15. Ma, J., Dunagan, J., Wang, H.J., Savage, S., Voelker, G.M.: Finding diversity in remote code injection exploits. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeiro, Brazil, pp. 53–64 (2006)
16. Marx, A., Dressman, F.: The wildlist is dead: long live the wildlist! In: Martin, H. (ed.) Proceedings of the 18th Virus Bulletin International Conference, Vienna, Austria, pp. 136–147 (2007)
17. Nakhleh, L., Sun, J., Warnow, T., Linder, C., Moret, B., Tholse, A.: Towards the development of computational tools for evaluating phylogenetic network reconstruction. In: Proceedings of the Eighth Pacific Symposium on Biocomputing, pp. 315–326 (2003)
18. Rambaut, A., Grassly, N.: Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Bioinformatics **13**(3), 235–238 (1997)
19. Robinson, D., Foulds, L.: Comparison of phylogenetic trees. Math. Biosci. **53**(1/2), 131–147 (1981)
20. Sim, S.E., Easterbrook, S., Holt, R.C.: Using benchmarking to advance research: a challenge to software engineering. In: Proceedings of the 25th International Conference on Software Engineering (ICSE'03), pp. 74–83 (2003)
21. Symantec. Symantec global internet security threat report volume XIII: trends for July–December 2007, April 2008
22. Wehner, S.: Analyzing worms and network traffic using compression. J. Comput. Secur. **15**, 303–320 (2007)
23. Wu, J., Spitzer, C.W., Hassan, A.E., Holt, R.C.: Evolution spectrographs: Visualizing punctuated change in software evolution. In: Proceedings of the Seventh International Workshop on the Principles of Software Evolution (IWPSE'04), pp. 57–66 (2004)