March 2006

Simulation-Based Graph Similarity

Oleg Sokolsky
University of Pennsylvania, sokolsky@cis.upenn.edu

Sampath Kannan
University of Pennsylvania, kannan@cis.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Postprint version. Published in *Lecture Notes in Computer Science*, Volume 3920, Tools and Algorithms for the Construction and Analysis of Systems: Proceedings of 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS 2006), pages 426-440.
Publisher URL: http://dx.doi.org/10.1007/11691372_28

Simulation-Based Graph Similarity

Abstract
We present symmetric and asymmetric similarity measures for labeled directed rooted graphs that are inspired by the simulation and bisimulation relations on labeled transition systems. Computation of the similarity measures has close connections to discounted Markov decision processes in the asymmetric case and to perfect-information stochastic games in the symmetric case. For the symmetric case, we also give a polynomial-time algorithm that approximates the similarity to any desired precision.

Comments
Postprint version. Published in *Lecture Notes in Computer Science*, Volume 3920, Tools and Algorithms for the Construction and Analysis of Systems: Proceedings of 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS 2006), pages 426-440.
Publisher URL: http://dx.doi.org/10.1007/11691372_28

Simulation-Based Graph Similarity*

Oleg Sokolsky, Sampath Kannan, and Insup Lee

Department of Computer and Information Science
University of Pennsylvania
{sokolsky,kannan,lee}@cis.upenn.edu

Abstract. We present symmetric and asymmetric similarity measures for labeled directed rooted graphs that are inspired by the simulation and bisimulation relations on labeled transition systems. Computation of the similarity measures has close connections to discounted Markov decision processes in the asymmetric case and to perfect-information stochastic games in the symmetric case. For the symmetric case, we also give a polynomial-time algorithm that approximates the similarity to any desired precision.

1 Introduction

The motivation for this work comes from the need for rapid detection of new computer viruses. The proliferation of virus development kits that can be downloaded from the Internet has dramatically lowered the entry threshold for virus developers [16]. What used to require considerable skill and substantial knowledge can now be accomplished by a relatively inexperienced hacker. As a result, large numbers of new virus programs appear every week. They are different enough from known viruses that conventional signature-based techniques become ineffective. Yet, since these viruses are developed using the same development kits, they share distinctive similarities with known representatives of viruses developed using the same kit. The classification of viruses into families is an attempt to capture such similarity.

The starting point of this work was the question of how the similarity between viruses of the same family can be captured and quantified. Our approach to similarity is *behavioral*, by which we mean that similar virus programs should be able to perform similar actions, arranged in similar ways. In order to make this intuition precise, we define a similarity metric on *control flow graphs* of programs. Control flow graphs, which can be defined either at the object code level or at the level of high-level programming language, are directed graphs, whose nodes and possibly edges are labeled with code fragments. There is also a dedicated initial node. Such labeled rooted graphs are often formalized as *labeled transition systems* (LTS).

One notion commonly used for semantic comparison for LTSs is *simulation*. The simulation relation defined on pairs of LTS nodes captures whether one node

* Research has been supported in part by the ONR MURI N00014-04-1-0735 and ARO DAAD19-01-1-0473.

simulates the other, or not. Intuitively, a node s_1 in an LTS G_1 simulates a node t_1 in another LTS G_2 if any outgoing edge of $t_1 \to t_2$ labeled by, say, a can be matched by an edge $s_1 \to s_2$, also labeled by a, in such a way that s_2 simulates t_2. A symmetric version of the simulation relation is known as *bisimulation*.

For the purpose of comparing control flow graphs of similar programs, we need to generalize the simulation relation into a function that captures *how well* one control flow graph simulates another. The first step in defining such a function is to introduce *local* similarity between graph nodes and edges. Local similarity functions define how well the label of a node matches the label of another node, and how well the label of an edge matches the label of another edge. The definition of local similarity functions depends on the nature of the labels of nodes and edges of the graphs. The labels may be viewed as symbols in a given alphabet, variable-length strings over a fixed alphabet, or have a more complicated structure such as, for example, assembly-language instructions or short fragments of assembly-language code. Different local similarity functions are appropriate in each of these situations. For the purpose of this paper, we assume that local similarity functions are given to us, and abstract away the nature of the graph labels.

In designing the similarity measure, we also need to decide whether we are pursuing an *aggregate* or *extremal* measures. In extremal measures, the computed value is based on the closest (or the most distant) match between components in the graph. Aggregate measures take into consideration matches from all paths. We argue that extremal measures can lead to counterintuitive results, where a single unmatched component yields a very low similarity value for two otherwise quite similar graphs. The advantage of extremal measures, on the other hand, is that they allow us to easily obtain distance metrics that obey the triangle inequality: the distance between two graphs does not exceed the sum of distances between each of the graphs and an arbitrary third graph. In general, it is useful to work with distance metrics. In our case, however, we do not necessarily need a distance metric. Our goal is to raise an alarm when a graph of the new program is similar enough to a known malicious program to warrant a closer inspection. If the program is found to be close to two quite different malicious programs, so much the better.

Guided by this motivation, we present new aggregate similarity measures that generalize simulation and bisimulation. We call these measures *quantitative* simulation and bisimulation. The contribution of this paper is a collection of algorithms for computing quantitative simulation and bisimulation on finite graphs. The algorithms rely on techniques similar to the ones used for the analysis of Markov decision processes and perfect-information stochastic games. Of particular practical importance are polynomial-time approximation algorithms for quantitative simulation and bisimulation.

Related work. In [6], several metrics have been proposed for *quantitative transition systems*, which are labeled with tuples of numbers instead of more conventional symbolic labels. The introduced measures are extremal and have the advantage of relatively low computational complexity. Similar metrics for la-

beled Markov processes have been considered in [8] as means of approximating bisimulation relations. In both cases, quantitative information is present in the transition system and serves as the basis for the metric. In our case, however, transition systems do not contain quantitative information.

Existing work on graph similarity has been applied mostly in pattern recognition and data mining contexts. In these domains, graphs can have rich structures, but little associated semantics with graphs nodes and edges. In our case, however, similarity is rooted in graph labels and the structure is less important and serves only to capture relationships between labels. The idea that similarity between two graphs can be computed as a fixed point of local similarities propagated through the transition relation has appeared in [2, 10], but was treated in an *ad hoc* fashion.

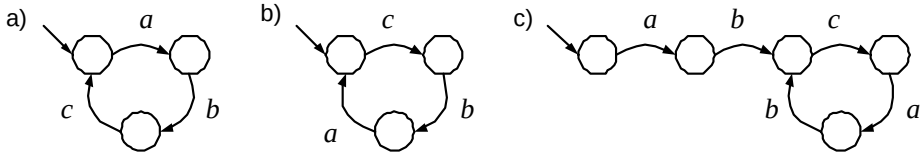

Fig. 1. Different notions of graph similarity

Most other existing graph similarity measures can be grouped into two categories [14]. Cost-based distance measures (also known as *edit distances*) are based on the number of modifications that are needed to transform one graph into the other. On the other hand, feature-based distance measures, for example [13], rely on extracting a set of structural features (such as vertex degrees) from the graphs and comparing vectors of features. For our purposes, both kinds of measures have the disadvantage that they are purely structural, while the measures we are proposing utilize significant amount of semantic information contained in the graph and its labels. To see the difference between our approach and the approach based on edit distances, consider the graphs is Figure 1. Graphs a) and b) have very small edit distance (high similarity): one needs only to swap two of the labels to make the graphs identical. However, if similarity between a and c is small, we would consider these graphs to be quite different as they admit different executions. Conversely, graphs a) and c) are quite different structurally, but they are, in fact, bisimilar and thus should be considered identical for our purposes.

Algorithmic techniques studied in this paper are similar to the approaches used in solving perfect-information quantitative stochastic games [4].

Structure of the paper. The rest of the paper is organized as follows. In Section 2 we briefly consider an extremal simulation-like measure and discuss its disadvantages. In Sections 3 and 4, we define strong and weak versions of an aggregate measure called *quantitative simulation*, discuss its properties and in-

troduce an approach to compute quantitative simulation that is based on linear programming. Section 5 introduces quantitative bisimulation, which is a symmetric version of quantitative simulation. Computing quantitative bisimulation turns out to be naturally related to finding the value of an appropriately defined infinite stochastic game and has a number of interesting connections to the recent work such as [4]. Finally, we offer a polynomial-time procedure to approximate quantitative bisimulation to any desired precision.

2 Extremal Quantitative Simulation

Preliminaries. A *labeled transition system* (LTS) is a labeled directed graph $G = \langle S, \mathcal{L}, T, s_0 \rangle$, where S is a set of states, $\mathcal{L}$ is a set of labels, $T \subseteq S \times \mathcal{L} \times S$ is a transition relation, and $s_0 \in S$ is the initial state. As usual, we denote $(s_1, a, s_2) \in T$ as $s_1 \xrightarrow{a} s_2$. In addition, assume two similarity functions: a *node similarity* function $N : S \times S \to [0, 1]$, and *label similarity* function $L : \mathcal{L} \times \mathcal{L} \to [0, 1]$. We assume that each node and each label is perfectly similar to itself: $N(s, s) = L(l, l) = 1$, and that the similarity functions are symmetric: $N(s_1, s_2) = N(s_2, s_1)$ and $L(a, b) = L(b, a)$.

Definition 1. Extremal quantitative simulation *(mq-simulation, for short) is a function $Q : S \times S \to [0, 1]$, such that for all states s_1, s_2, the following condition holds:*

$$Q(s_1, s_2) = \begin{cases} N(s_1, s_2) & \text{if } \forall a, s_1 \overset{a}{\not\to} \\ N(s_1, s_2) \cdot \prod_{s_1 \xrightarrow{a} s_1'} \max_{s_2 \xrightarrow{b} s_2'} L(a, b) \cdot Q(s_1', s_2') & \text{otherwise} \end{cases} \tag{1}$$

If $Q(s_1, s_2) = n$, we say that s_2 simulates s_1 up to n. If we have two LTSs, G_1 and G_2 with initial states s_0^1 and s_0^2, respectively, we say that G_2 simulates G_1 up to n if $Q(s_0^1, s_0^2) = n$.

As a motivation for this definition, consider the case when the node similarity function does not distinguish nodes, that is, $\forall s_1, s_2 \in S, N(s_1, s_2) = 1$ and the label similarity function is binary, that is, $L(l_1, l_2) = 1$ if $l_1 = l_2$ and 0 otherwise. In this case, we recover the traditional definition for simulation relation $\precsim$ on labeled transition systems: $s_1 \precsim s_2$ *iff* s_2 simulates s_1 up to 1.

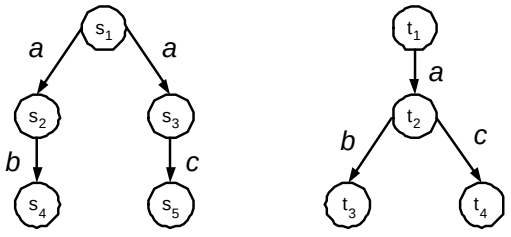

Fig. 2. Example of quantitative similarity

Example. Consider the classical example from the (bi)simulation literature, shown in Figure 2. It is well-known that t_1 simulates s_1, but not vice versa. Let us assume, however, that b and c are not completely dissimilar. Let $L(b, c) = 0.5$. With the definition above, we have that $Q(s_1, t_1) = 1$, which is not surprising because t_1 simulates s_1. In the reverse direction, we have that $Q(t_2, s_2) = Q(t_2, s_3) = 0.5$. This is because s_2 (and, similarly, s_3) matches one transition of t_2 perfectly and the other by 0.5, the product of this yields 0.5. Finally, $Q(t_1, s_1) = 0.5$.

Fixpoint characterization of mq-simulation. Consider the set of tuples $\mathcal{Q} = \{Q_{i,j}\}$ with $i, j \in 1 \ldots |S|$ and every $Q_{i,j} \in [0, 1]$. Equipped with the operations of pointwise minimum and maximum, the set forms a complete lattice.

Consider the function $f : \mathcal{Q} \to \mathcal{Q}$, which is the simultaneous application of Equation (1) for all i, j. The function is monotonic. By the Tarski-Knaster theorem, a fixed point of f exists, and is an mq-simulation.

Discussion. The attractive feature of mq-simulation is that it is an immediate generalization of the classical simulation relation to the domain of reals. However, it does not make a good measure for LTS similarity, because it can produce quite counterintuitive results. Consider the following two examples.

One problem of quantitative simulation is that it assigns too much importance to "eventual dissimilarity." Consider, for example, the LTSs

$$s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \; s_{n-1} \xrightarrow{a_{n-1}} s_n \text{ and } t_1 \xrightarrow{b_1} t_2 \xrightarrow{b_2} \ldots t_{n-1} \xrightarrow{b_{n-1}} t_n. \tag{2}$$

Let $L(a_i, b_i) = 1$, except for the last pair, where $L(a_{n-1}, b_{n-1}) = 0$, and $N(s_i, t_i) = 1$ for all i, j. Intuitively, one would expect to see these LTSs rather similar, because the dissimilarity is far down the road. However, $Q(s_1, t_1) = 0$.

The second problem is that mq-simulation is too strict in combining alternatives. Consider two star-shaped LTSs constructed as follows: $s_0 \xrightarrow{a_i} s_i$ and $t_0 \xrightarrow{b_i} t_i$. Again, let all nodes be similar and let labels be pairwise similar $L(a_i, b_i) = 1$ for all i except $i = n$. In all other cases, that is, $i \neq j$ or $i = j = n$, $L(a_i, b_j) = 0$. Again, our intuition suggests that these LTSs should be considered similar since all but one branches are matched perfectly. Still, $Q(s_1, t_1) = 0$.

Analysis of these two problems suggest that perhaps a more useful definition would be a function, additive both in terms of different branches leaving the same state and in terms of number of states along a path. To address the first problem, this function should give more weight to close states and progressively less weight to more distant states. It is important, however, to ensure that the similarity measure remains bounded over long paths and for states with large branching factors. The next section will be devoted to defining such a function.

3 Weighted Quantitative Simulation

3.1 Similarity on Paths

We begin by comparing states in paths of an LTS, and then generalize the obtained notion of similarity to LTSs in general.

In order to overcome the problems with the mq-simulation, we introduce a parameter p that describes the relative importance of local similarity vs. "step similarity." Consider two edges in an LTS: $s_0 \xrightarrow{a} s_1$ and $t_0 \xrightarrow{b} t_1$. Similarity between s_0 and t_0 based on these two edges can be defined to be $(1-p) \cdot N(s_0, t_0) + p \cdot L(a,b) \cdot N(s_1, t_1)$. Consider now two paths of the same length, $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \dots s_n$ and $t_0 \xrightarrow{b_1} t_1 \xrightarrow{b_2} t_2 \xrightarrow{b_3} \dots t_n$. We can extend the similarity between s_0 and t_0 to recursively consider paths instead of single edges as follows: $Q(s_0, t_0) = (1-p) \cdot N(s_0, t_0) + p \cdot L(a_1, b_1) \cdot Q(s_1, t_1)$. Unfolding the recursive expression, we obtain that, for the paths of length n, the similarity between the paths is

$$Q(s_0, t_0) = \sum_{i=0}^{n} (1-p) \cdot p^i \cdot N(s_i, t_i) \cdot \prod_{k=1}^{i} L(a_k, b_k), \tag{3}$$

with the convention that the product over an empty set is 1.

The value of the similarity between two paths is bounded from above by the similarity of two identical traces. Since all node and edge similarity functions yield 1 in this case, it is clear that the similarity between two identical infinite paths is 1.

Consider again the LTSs of (2). The effect of dissimilarity after n steps is now negligible for large n: $Q(s_1, t_1) = 1 - 2^{-n}$ for $p = 1/2$.

3.2 Graph Similarity

Definition 2. *For a parameter $0 < p < 1$, p-*weighted quantitative simulation *(q-simulation) is a function $Q : S \times S \to [0,1]$, such that for all states s_1, s_2, the following condition holds:*

$$Q_p(s_1, s_2) = \begin{cases} N(s_1, s_2) & \text{if } \forall a, s_1 \overset{a}{\not\to} \\ (1-p) \cdot N(s_1, s_2) + \frac{p}{n} \cdot \sum_{s_1 \xrightarrow{a} s_1'} \max_{s_2 \xrightarrow{b} s_2'} L(a,b) \cdot Q_p(s_1', s_2') & \text{otherwise,} \end{cases} \tag{4}$$

where n is the number of transitions leaving s_1.

Consider the same example of Figure 2. Let $p = 1/2$. Here we again have $Q_{\frac{1}{2}}(s_1, t_1) = 1$. Indeed, $Q_{\frac{1}{2}}(s_2, t_2) = (1-p) + p \cdot Q_{\frac{1}{2}}(s_4, t_3) = 1$ (as well as $Q_{\frac{1}{2}}(s_3, t_2)$), and $Q_{\frac{1}{2}}(s_1, t_1) = (1-p) + p \cdot \frac{1}{2} \cdot (Q_{\frac{1}{2}}(s_2, t_2) + Q_{\frac{1}{2}}(s_3, t_2)) = 1$. In the reverse direction, $Q_{\frac{1}{2}}(t_2, s_2) = (1-p) + p \cdot \frac{1}{2} \cdot (1 + \frac{1}{2}) = \frac{7}{8}$. Finally, $Q_{\frac{1}{2}}(t_1, s_1) = (1-p) + p \cdot Q_{\frac{1}{2}}(t_2, s_2) = \frac{15}{16}$. The number is much higher than for the mq-simulation, because here considerable weight is given to node similarity,

which in this case does not make any distinction between nodes and only drives the similarity up.

We now show two important properties of q-simulation. Proofs of these properties use the functional $\mathcal{T}(Q)(s_1, s_2)$ derived from (4). That is, let $\mathcal{Q}$ be the set of functions $Q : S \times S \to [0, 1]$. $\mathcal{T} : \mathcal{Q} \to \mathcal{Q}$ is defined as

$$\mathcal{T}(Q(s_1, s_2)) = \begin{cases} N(s_1, s_2) & \text{if } \forall a, s_1 \overset{a}{\not\to} \\ (1-p) \cdot N(s_1, s_2) + \frac{p}{n} \cdot \sum_{s_1 \overset{a}{\to} s_1'} \max_{s_2 \overset{b}{\to} s_2'} L(a,b) \cdot Q(s_1', s_2') & \text{otherwise,} \end{cases}$$

The following theorem shows that q-simulation is well-defined:

Theorem 1. *Equation (4) has a unique solution.*

Proof (Sketch): We reduce the problem of computing q-simulation to the problem of optimal control in discrete event systems, studied in [1]. Given an LTS G, we construct the dynamical system

$$x_{i+1} = f(x_i, c_i, w_i),$$

where the next state x_{i+1} depends on the current state x_i, the current control input c_i, and random disturbance w_i. The state space is given by pairs of graph nodes, $x \in S \times S$. Control inputs $c_i \in C(x)$ are state-dependent. Given the current state $x_i = (s_1, s_2)$, $C(x) = \{s_2 \overset{b}{\to} s_2'\}$. That is, control inputs represent the choice of transition in the simulating state. Disturbance $w_i \in W(x) = \{s_1 \overset{a}{\to} s_1'\}$, on the other hand, represents the choice of transition in the simulated state. Fixing a control strategy, that is, the sequence of control inputs, yields a random trajectory $x_0, x_1, \ldots$. The value of a given trajectory is given according to (3). The optimal control problem for the given dynamical system is to determine the maximum expected value over the set of possible control strategies. For uniformly distributed disturbances, independently chosen in each step, this maximum expected value coincides with 4. The proof then closely follows [1], p. 182ff, which shows that the control problem – albeit for a different but also monotonic value function – has a unique solution. We show that the functional $\mathcal{T}(Q)$ is a *contraction mapping* over the space of functions $Q : S \times S \to [0, 1]$, measured by the distance function $d(Q, Q') = max_{x \in S \times S} |Q(x) - Q'(x)|$. Since $\mathcal{T}$ is a contraction mapping, it has a unique fixed point by the Banach's theorem, and the fixed point coincides with Q_p. □

A more direct proof in the style of Theorem 3 is also possible; however, the proof we chose here points to a clear connection with an established approach. The next theorem shows that q-simulation generalizes the simulation relation:

Theorem 2. *Let $N(s_1, s_2) = 1$ for all s_1, s_2, and $L(a, b) = 1$ iff $a = b$, and 0 otherwise. Then $s_1 \precsim s_2$* iff *$Q_p(s_1, s_2) = 1$ for any $0 < p < 1$.*

Proof: ($\Rightarrow$) We proceed by contradiction. Assume that $s_1 \precsim s_2$ but $Q_p(s_1, s_2) \neq 1$. Consider the function $Q^+ : S \times S \to [0, 1]$, such that $Q^+(s_1, s_2) = 1$ if $s_1 \precsim s_2$ and $Q^+(s_1, s_2) = Q_p(s_1, s_2)$ otherwise. Clearly, $Q_p < Q^+$ in the lattice

of functions $S \times S \to [0,1]$. Consider the functional $\mathcal{T}(Q)$ defined above. It is clear that when $s_1 \precsim s_2$, $\mathcal{T}(Q^+)(s_1,s_2) = Q^+(s_1,s_2)$, because the maximum value over the transitions of s_2 in Equation (4) will always be 1. It is also easy to see that, for any s_1 and s_2, $\mathcal{T}(Q^+)(s_1,s_2) \geq Q^+(s_1,s_2)$. By repeating this argument, we see that $\mathcal{T}^n(Q^+)(s_1,s_2) \geq Q^+(s_1,s_2)$. Since T has a unique fixed point by Theorem 1, the sequence $\mathcal{T}^n$ converges to the fixed point of $\mathcal{T}$, which is Q_p. Thus we have $Q_p < Q^+ \leq Q_p$, which is a contradiction. ($\Leftarrow$) Consider relation $R \subseteq S \times S$ such that $(s_1,s_2) \in R \Leftrightarrow Q_p(s_1,s_2) = 1$. We show that R is a simulation relation. The case when s_1 does not have outgoing transitions is obvious. Otherwise, for each transition $s_1 \xrightarrow{a} s_1'$, $\max_{s_2 \xrightarrow{b} s_2'} L(a,b) \cdot Q_p(s_1',s_2') = 1$, which is possible only if $a = b$ and $Q_p(s_1',s_2') = 1$, that is, $(s_1',s_2') \in R$. □

3.3 Computing Weighted Quantitative Simulation

Given an LTS G and functions N and L, we transform (4) into an instance of the linear programming problem in the following way. For every two states s_i, s_j, we introduce a variable $Q_{i,j}$. For every edge $e = s_i \xrightarrow{a} s_k$ and state s_j, we introduce a variable $X_{e,j}$. The objective function minimizes the sum of all variables:

$$min \sum_{i,j \in S} Q_{i,j} + \sum_{e \in T, s_j \in S} X_{e,j}.$$

We represent the relationship between variables using the following constraints: $0 \leq Q_{i,j} \leq 1$, $0 \leq X_{e,j} \leq 1$ for all i, j, and e. For m such that $s_j \xrightarrow{b} s_m$, $X_{e,j} \geq L(a,b) \cdot Q_{k,m}$. Finally, for all i, j,

$$Q_{i,j} = (1-p) \cdot N(s_i,s_j) + \sum_{e \in s_i \xrightarrow{a} s_k} \frac{p}{n} X_{e,j}$$

By Theorem 1, this linear programming problem has a unique solution, so that $Q_{i,j} = Q_p(s_i,s_j)$.

4 Weak Weighted Quantitative Simulation

A deficiency of q-simulation as defined by Equation 4 is that it requires the graphs to unfold synchronously - that is, every step of one graph has to be matched by a similar step of another graph. Consider two paths $s_1 \xrightarrow{a} s_2 \xrightarrow{b} s_3 \xrightarrow{c} \ldots$ and $t_1 \xrightarrow{a} t_2 \xrightarrow{a'} t_3 \xrightarrow{b} t_4 \xrightarrow{c} \ldots$. The paths are identical, except for the insertion of an a'-step in the second path. If $L(a',b) = 0$, the similarity of the two paths will be very low, despite the fact that almost every step in the paths can be matched to a step in the other path in the same order. Intuitively, the role of the inserted step should be heavily discounted.

We can compare graph similarity to the well-known notion of string similarity, known as the *string alignment problem*, widely used in biological sequence analysis. We consider a particular dynamic-programming formulation of this problem

that serves as the basis for the Needleman-Wunsch alignment algorithm [12]. We assume a similarity score between elements of the alphabets of the two strings. Consider strings $s_1 = as'_1$ and $s_2 = bs'_2$. The optimal alignment score of s_1 and s_2, denoted $F(s_1, s_2)$, is computed as the maximum of $F(s'_1, s'_2) + s(a, b)$, $F(as'_1, s'_2) - d$, and $F(s'_1, bs'_2) - d$. Here, $s(a, b)$ is the similarity score of a and b, and d is the *gap penalty*.

We want to introduce a similar notion – that "skipping a step" is permissible but carries a penalty – into the q-simulation framework and define *weak q-simulation*. We remind the reader, that the classical definition of the weak simulation relation, a transition of the simulated state labeled with the action a can be matched by a finite sequence of transitions from the simulating state, exactly one of which is labeled with a and the rest are labeled with a special internal action τ[1]. The intuition for our definition comes from the fact that, in the binary world, classical weak (bi)simulation is strong (bi)simulation applied to the τ-*closed* transition system, in which every such sequence of transitions is represented by a single transition.

It is tempting to use this intuition directly. Consider a special "skip" action $\epsilon \notin \mathcal{L}$. Suppose we construct the ϵ-closure of G. Since ϵ is a new action, the closure amounts to adding a self-loop transition labeled ϵ to every state in G. The edge similarity function $L(a, \epsilon) = L(\epsilon, a)$ serves as the label-sensitive "gap penalty". Treating ϵ differently than any other action, we require that $L(\epsilon, \epsilon) = 0$. This precludes the pathological case when both states stutter and yet similarity increases. We can then use the same equation (4) to define weak q-simulation using the extended transition relation instead of T.

Although such simple solution gives a very intuitive definition, it is easy to see that this is not the definition we want. We lose a desirable property that q-simulation reduces to classical simulation in the binary case. Indeed, according to the definition above, a state will not be weakly q-similar to itself! To see this, consider a deadlocked state. Its weak q-similarity to itself would be $1 - p$ instead of 1. The source of this problem is that equation (4) is including the stuttering step in computing the average of matches. To fix this problem, we use the following definition.

Definition 3. *A (weighted) weak quantitative simulation is a function* $Q^W : S \times S \to [0, 1]$*, such that for all* s_1, s_2*, the following condition holds:*

$$Q^W(s_1, s_2) = \begin{cases} N(s_1, s_2) & \text{if } \forall a, s_1 \overset{a}{\not\to} \\ (1 - p) \cdot N(s_1, s_2) + \max(W_1, W_2) & \text{otherwise,} \end{cases} \tag{5}$$

where

$$W_1 = \max_{s_2 \overset{b}{\to} s'_2} L(b, \epsilon) \cdot Q^W(s_1, s'_2)$$

$$W_2 = \tfrac{p}{n} \cdot \sum_{s_1 \overset{a}{\to} s'_1} \max(\max_{s_2 \overset{b}{\to} s'_2} (L(a, b) \cdot Q^W(s'_1, s'_2)), L(a, \epsilon) \cdot Q^W(s'_1, s_2)),$$

[1] Note that "skipping a step" is more similar to stuttering than to executing an internal step. Therefore, our weak q-simulation is closer in spirit to stuttering (bi)simulation [3, 11] than to classical weak (bi)simulation.

and n is the number of transitions leaving s_1.

Weak q-simulation can be computed using a slightly modified linear programming problem from the previous section.

5 Quantitative Bisimulation

A natural extension of the q-simulation idea is to define a symmetric similarity function. That is, for all states s_1 and s_2, $B(s_1, s_2) = B(s_2, s_1)$. It is natural to think of this function as quantitative bisimulation (q-bisimulation). We construct such a function by taking the minimum of the asymmetric one-step similarities between two states.

Definition 4. *Given the graph $G = \langle S, \mathcal{L}, T\rangle$, the (p-weighted) quantitative bisimulation is the function $B_p : S \times S \to [0, 1]$, defined as*

$$B_p(s_1, s_2) = \min(B^l_p(s_1, s_2), B^r_p(s_1, s_2)),$$

where B^l_p, B^r_p are left and right similarities, respectively, are defined as

$$B^l_p(s_1, s_2) = \begin{cases} N(s_1, s_2) & \text{if } \forall a, s_1 \overset{a}{\not\to} \\ (1-p)\cdot N(s_1, s_2) + W(s_1, s_2) & \text{otherwise} \end{cases}$$

$$B^r_p(s_1, s_2) = \begin{cases} N(s_2, s_1) & \text{if } \forall b, s_2 \overset{b}{\not\to} \\ (1-p)\cdot N(s_2, s_1) + W(s_2, s_1) & \text{otherwise,} \end{cases}$$

where $W(s,t) = \frac{p}{n} \cdot \sum_{s \overset{a}{\to} s'} \max_{t \overset{b}{\to} t'} L(a,b) \cdot B_p(s', t')$

Similarly to q-simulation, q-bisimulation can be considered in the strong as well as the weak form by constructing a symmetric version of Definition 3.

5.1 Computing Quantitative Bisimulation

We can reduce the problem of computing strong quantitative bisimulation to the problem of computing the value of a *stochastic game with extended payoffs and Büchi winning condition*. Such games are extensions of simple stochastic games [9, 5] to include value derived from infinite runs.

A stochastic game with payoffs is a graph $\mathcal{B} = \langle V, E, N, L\rangle$, where V is a set of vertices partitioned into three subsets V_{min}, V_{max}, and V_{avg}, $E \subseteq V \times V$ is the transition relation, $N : V \to [0, 1]$ is the node payoff function, and $L : E \to [0, 1]$ is the edge payoff function. We use $v_1 \overset{l}{\to} v_2$ to denote $L(v_1 \to v_2) = l$.

Given a graph $G = \langle S, \mathcal{L}, T\rangle$, we construct $\mathcal{B}(G)$ as follows. We introduce two parameters, $\lambda, \delta \in (0, 1)$, to define edge payoffs and game values to match the discounting structure of q-bisimulation. We set $\lambda = \sqrt[3]{p}$ and $\delta = 1 - p$. For each pair of states $s_i, s_j \in S$, we introduce vertices $v^b_{ij}, v^t_{ij} \in V_{min}$, $v^s_{ij} \in V_{avg}$, and, for each edge $e : s_i \to s_k$, $v^m_{ej} \in V_{max}$. The edges in $\mathcal{B}$ are introduced according to the following rules:

– $v^b_{ij} \xrightarrow{1} v^s_{ij}$, $v^b_{ij} \xrightarrow{1} v^s_{ji}$;
– $v^t_{ij} \xrightarrow{\lambda^2} v^t_{ij}$;
– for each edge $e : s_i \xrightarrow{a} s_k$, $v^s_{ij} \xrightarrow{1} v^m_{ej}$;
– for each s_n such that $s_j \xrightarrow{b} s_n$, $v^m_{ej} \xrightarrow{L(a,b)} v^b_{kn}$.
– if s_i does not have outgoing transitions, then $v^s_{ij} \xrightarrow{\lambda} v^t_{ij}$ for every s_j.
– if s_j does not have outgoing transitions, then $v^m_{ej} \xrightarrow{0} v^t_{ij}$ for every $e : s_i \xrightarrow{a} s_k$.

For every $s_i, s_j \in S$ and $e \in T$, $N(v^b_{ij}) = N(v^t_{ij}) = N(s_i, s_j)$, $N(v^s_{ij}) = N(v^m_{ej}) = 0$.

The game has two players, one of which selects the transitions at the *max* vertices, while the other selects the transitions at the *min* vertices. The choice of a transition by a player is given by a *strategy* of the player. We consider only *pure memoryless strategies*, in which the choice depends only on the current vertex and not on the history of the game. Such a strategy for the *max* player is represented by a function $\sigma : V_{max} \to V$ (respectively, $\pi : V_{min} \to V$ for the *min* player). The choices at a vertex $v \in V_{avg}$ are made randomly according to a uniform distribution over the successors of v.

Given strategies σ, π and a starting vertex v, a play $w_v^{\sigma,\pi}$ is an infinite random path through the game graph, in which steps from the *min* and *max* vertices comply with the strategies.

The Büchi winning condition for this game is defined as follows. A play is a winning play for player *min* if it contains an infinite number of the vertices from V_{min}. It is easy to see from the construction of the game graph that every play is a winning play for player *min*.

The *discounted payoff* of a play $w_{v_0} = v_0 \xrightarrow{l_1} v_1 \xrightarrow{l_2} \dots$ for a discount factor λ and a scaling factor δ is defined as

$$Q(w_v) = \delta \cdot \sum_{i=0}^{\infty} \lambda^i \cdot N(v_i) \cdot \prod_{k=1}^{i} l_k. \tag{6}$$

The value of the game for an initial state v and given strategies σ, π is given as the expected payoff of $w_v^{\sigma,\pi}$. Strategies σ_o, π_o are called optimal for v if $w_v^{\sigma_o,\pi_o} = \min_\pi \max_\sigma w_v^{\sigma,\pi}$ The optimal value of the game $\mathcal{B}(G)$ for a node v, denoted $B(v)$, is the value of the play $w_v^{\sigma_o,\pi_o}$ yielded by the optimal strategies.

Considering the structure of the game graph, we conclude that relations between optimal values of the nodes are as shown in Figure 3. Putting these equations together, we can see that the value of a node $v^b_{i,j}$ is the q-bisimulation $B_p(s_i, s_j)$.

It is well known that there exist optimal pure memoryless strategies for both players for similar payoffs. By using techniques similar to [4], we can show that there are optimal pure memoryless strategies in our case as well. Thus the value of the game can be computed, since there are finitely many pure memoryless strategies. Several approaches for computing game values exist [7, 4], with complexity no more than exponential in the size of the game graph. Since, in our

$$B(v^t_{ij}) = \delta \cdot N(v^t_{ij}) \cdot \sum_{i=0}^{\infty} \lambda^i \cdot (\lambda^2)^i = \delta \cdot N(v^t_{ij}) \cdot \frac{1}{1-\lambda^3} = N(v^t_{ij})$$

$$B(v^m_{ej}) = 0 \qquad \text{if } v^m_{ej} \rightarrow v^t_{ij}$$

$$B(v^m_{ej}) = \lambda \cdot \max_{v^m_{ijk} \xrightarrow{l_n} v^b_{nk}} (l_k \cdot B(v^b_k)) \qquad \text{otherwise}$$

$$B(v^s_{ij}) = \lambda^2 \cdot B(v^t_{ij}) \qquad \text{if } v^s_{ij} \rightarrow v^t_{ij}$$

$$B(v^s_{ij}) = \lambda \cdot \text{avg}_{v^s_{ij} \rightarrow v^m_{ej}} Q(v^m_{ej}) \qquad \text{otherwise}$$

$$B(v^b_{ij}) = \delta \cdot N(v^s_{ij}) + \lambda \cdot \min_{v^b_{ij} \rightarrow v^s_k} B(v^s_k)$$

Fig. 3. Relationships between optimal values in graph nodes

case, the game graph $\mathcal{B}(G)$ is polynomial in the size of G ($O(|G|^3)$, to be precise), the complexity of computing q-bisimulation is also no worse than exponential in the size of G.

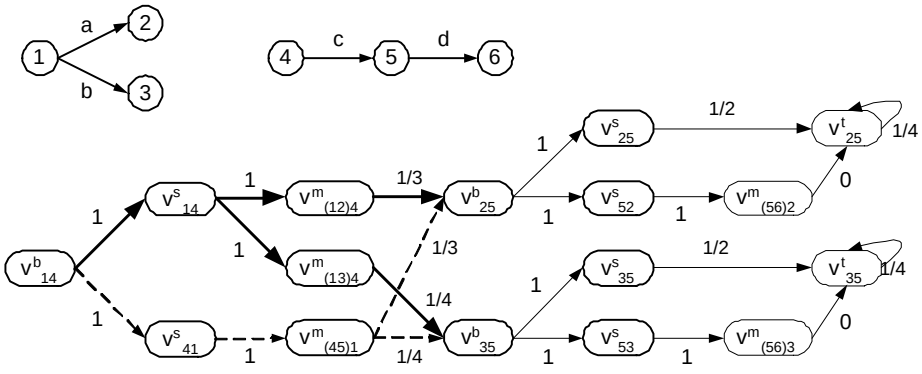

Fig. 4. Game graph construction

To illustrate how the game graph construction works, consider the example in Figure 4. Let $L(a,c) = 1/3$, $L(b,c) = 1/4$, and $p = 1/8$. Let $N(s,t) = 1$ for all nodes s, t. Consider, for example, nodes 2 and 5. $B^l(2,5) = 1$ since node 2 is deadlocked, while $B^r(2,5) = 7/8$ since node 2 cannot match the transition of node 5, and $B(2,5) = 7/8$. Accordingly, $B(v^b_{25}) = \delta + \lambda \cdot \min(\lambda^2, 0) = 7/8$. Similarly, $B^l(1,4) = (1-p) + p \cdot 1/2 \cdot (L(a,c) \cdot B(2,5) + L(b,c) \cdot B(3,5)$ and $B^r(1,4) = (1-p) + p \cdot \max(L(c,a) \cdot B(2,5), L(c,b) \cdot B(3,5)$. Game steps that correspond to the computation of $B^l(1,4)$ and $B^r(1,4)$ shown as block and dashed arrows, respectively. Note that a path from v^b_{ij} to $v^b_{i'j'}$ is always three steps long. Thus, discounting during a game is applied three times, where it is applied once in the definition of q-bisimulation. This observation explains the relationship between p and λ.

5.2 Approximating quantitative bisimulation

In many cases, it is not necessary to compute the precise value of q-bisimulation, especially considering that the node and edge similarity functions are likely to be heuristic estimates. It may be sufficient to know whether it exceeds a certain threshold value. It is therefore useful to have a polynomial algorithm to compute an approximation of q-bisimulation up to a required degree of accuracy.

Given the game $\mathcal{B}(G)$, we compute an approximation of q-bisimulation for all states of G as follows. For a chosen n, we make n copies of $\mathcal{B}(G)$, $\mathcal{B}^{(0)}$,...,$\mathcal{B}^{(n-1)}$. The copy of a node v in $\mathcal{B}^{(i)}$ is denoted $v^{(i)}$. We connect these copies into a single graph by replacing every edge $v^{m(i)} \xrightarrow{l} v^{b(i)}$ $(i > 0)$ with $v^{m(i)} \xrightarrow{l} v^{b(i-1)}$. All nodes $v^{t(i)}$ and all incident edges are removed. In $\mathcal{B}^{(0)}$, we keep only the nodes $v_{ij}^{b(0)}$ and remove all other nodes and edges. As a result, we have an acyclic graph $\mathcal{B}^{+}$. We assign initial values to terminal nodes of $\mathcal{B}^{+}$ as follows: (1) every node $v^{(0)}$ is assigned $\delta \cdot N(v)$; (2) every node $v^{m(i)}$ is assigned 0; (3) every node $v^{s(i)}$ is assigned $N(v)$. With this initialization, we can compute the values for all nodes in a bottom-up fashion according to the formulas in Figure 3, in time linear in the size of the resulting graph. Let the value of a node $v^{(i)}$ computed in this fashion be denoted $B^{+}(v^{(i)})$. It is easy to see that $B^{+}(v^{(i)}) \leq B(v)$. Indeed, $q \cdot N(v) \leq B(v)$ by definition, thus $B^{+}(v^{(0)}) \leq B(v)$. For all other i, the result follows by the monotonicity of the functional defining B.

Theorem 3. *For any n and any node v in $\mathcal{B}(G)$, $B(v) - B^{+}(v^{(n)}) \leq \lambda^{n}$.*

Proof: The proof proceeds by induction on the copy number. The base case is immediate, since for any node v, $B(v), B^{+}(v^{(0)}) \in [0,1]$, thus $B(v) - B^{+}(v^{(0)}) \leq \lambda^{0} = 1$. Suppose now that for some i and every node v, $B(v) - B^{+}(v^{(i-1)}) \leq \lambda^{i-1}$. Consider a node $v^{m(i)}$. If it is a terminal node, its initial value is equal to $B(v)$. Otherwise,

$$B(v^{s}) - B^{+}(v^{s(i)}) = \max_{v^{s} \xrightarrow{l_m} v_m^{b}} \lambda \cdot l_m \cdot B(v_m^{b}) - \max_{v^{s} \xrightarrow{l_k} v_k^{b}} \lambda \cdot l_k \cdot B^{+}(v^{b(i-1)}).$$

Here, we have to consider two cases. Either $k = m$, that is, the same strategy is chosen by the original and the approximated computation. In that case, clearly,

$$B(v^{s}) - B^{+}(v^{s(i)}) = \lambda \cdot l_m \cdot (B(v_m^{b}) - B^{+}(v_m^{b(i-1)})) \leq \lambda \cdot (B(v_m^{b}) - B^{+}(v_m^{b(i-1)})) \leq \lambda \cdot \lambda^{(i-1)} = \lambda^{i}.$$

Otherwise,

$$l_m \cdot B(v_m^{b}) - l_k \cdot B^{+}(v_k^{b(i-1)}) \leq l_m \cdot B(v_m^{b}) - l_m \cdot B^{+}(v_m^{b(i-1)}),$$

since $B^{+}(v_k^{b(i-1)}) \geq B^{+}(v_m^{b(i-1)})$, and the result follows by the same argument as above. For a node $v^{s(i)}$ the result is immediate, since the calculation computes the average of the values in $v^{m(i)}$, covered by the case above. Finally, for a node

$v^{b(i)}$, the argument is similar to the first case. The node has two successor nodes, and the computed value is based on the minimum value of the two successors. As above, the interesting case is the one where the exact and the approximate values come from different successors. Let the successors of v^b be v_1^s and v_2^s, considered in the previous case. Without loss of generality, let us assume that $B(v_1^s) \leq B(v_2^s)$ and $B^+(v_1^{s(i)}) \geq B^+(v_2^{s(i)})$. We then have

$$B(v^b) - B^+(v^{b(i)}) = \lambda \cdot (B(v_1^s) - B^+(v_2^{s(i)})) \leq \lambda \cdot (B(v_2^s) - B^+(v_2^{s(i)})) \leq \lambda^i.$$

□

6 Conclusions and future work

We have presented aggregate similarity measures for labeled transition systems. The asymmetric similarity measure, called q-simulation, is well aligned with our original motivation for this work, which involved asymmetric comparisons: a control-flow graph of a new program would compared to well-known representatives of virus families. In general, a symmetric similarity measure may be more useful. Such a symmetric measure, called q-bisimulation, is also more complex to compute. However, for both measures, the more practical approach is to compute an approximate value. Such value, which can be made arbitrarily close to the exact value, can be obtained in time polynomial in the graph size.

A prototype tool for computing q-simulation using the approach of Section 3.3, has been implemented using the `lp_solve` tool as a back-end, and applied to the samples from the Virus Source Code Database [18]. The tool operates on control-flow graphs in the VCG format produced by the GCC compiler. Our initial experiments demonstrated the need for the weak q-simulation, which is currently being implemented. Out current work concentrates on further experimental evaluation. A matter of practical importance is the choice of the parameters of the function: the weight p and, for weak similarity, the gap penalty. We are considering machine learning approaches for determining parameter values.

An interesting direction of future work is to identify areas of very high similarity within graphs, instead of trying to compare the graphs in their entirety. This would allow us to draw the user's attention to programs that have common parts along with substantially different parts. An example of such programs would be viruses that exploit different vulnerabilities but have the same payload. The relationship between these two graph similarity approaches would be reminiscent of the relationship between global [12] and local [15] string alignment in bioinformatics. Another important direction is to modify the definition of q-simulation to obtain a distance metric, which will make it appealing in various applications, particularly, in bioinformatics.

Although the original motivation for this work comes from the goal to identify potential virus programs from their similarity to known virus family representatives, quantitative simulation and bisimulation have many other potential applications. Work on such applications in the domains of bioinformatics and literature citation databases is already under way [17].

Acknowledgments. We are grateful to Lyle Ungar and Ted Sandler for many fruitful discussions on further applications of q-simulation.

References

1. D. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Inc., 1987.
2. V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Review*, 46(4):647–666, 2004.
3. M. C. Browne, E. M. Clarke, and O. Grümberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59(1–2):115–131, 1988.
4. K. Chatterjee, M. Jurdziński, and T. Henzinger. Quantitative stochastic parity games. In *SODA '04: Proceedings of the 15^{th} annual ACM-SIAM Symposium on Discrete Algorithms*, pages 121–130, 2004.
5. A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.
6. L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In *ICALP '04: 31^{st} International Colloquium on Automata, Languages, and Programming*, volume 3142 of *LNCS*, pages 97–109, 2004.
7. L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374–397, 2004.
8. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, 2004.
9. J. A. Filar. Ordered field property for stochastic games when the player who controls transitions changes from state to state. *Journal of Optiraization Theory and Applications*, 34:503–515, 1981.
10. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of 18^{th} ICDE*, 2002.
11. K. Namjoshi. A simple characterization of stuttering bisimulation. In *Proceedings of the 17th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 284–296, 1997.
12. S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.
13. A. Papadopoulos and Y. Manolopoulos. Structure-based similarity search with graph histograms. In *Proceedings International DEXA Workshop on Similarity Search (IWOSS), Florence, Italy*, pages 174–178, 1999.
14. A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3):353–362, 1983.
15. T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
16. P. Szor. *The Art of Computer Virus Research and Defense.* Addison Wesley Professional, 2005.
17. L. Ungar and T. Sandler, Sept. 2005. Personal communication.
18. Virus source code database. `http://www.totallygeek.com/vscdb/`.