# Online Risk Assessment of Cloud Service Providers

Jose Herrera[b], Chris Peabody[a], David Cabral[b]

[a]*Missouri University of Science and Technology*
[b]*University of California, Merced*

**Abstract**

Applications have their own security risks before they are deployed in a cloud environment. To be able to obtain all the benefits that come along with utilizing the cloud environment, a client should know the different variety of security risks that will arise from deploying their application on the cloud. Security risks increase for applications deployed on the cloud because the client has no input in how the cloud platform plans to mitigate and prevent attacks. Additionally, cloud providers having different security plans in place for their own services. We propose a risk assessment framework that is dependent on four attributes, coming in the form of qualitative and quantitative data like cost, quality of service and experience, and security of a cloud service provider(CSP). These attributes are then represented as a graphical structure and modeled using Bayes' theorem to compute a net trust score for a cloud service provider. This will help a client to compare between different CSPs when migrating their applications to their cloud platform. Our proposed framework has been implemented as a web application where users can input certain preferences and conduct calculations on five CSPs.

*Keywords:* Cloud Computing, Risk Assessment, Trust, Quality of Service, Quality of Experience

## 1. Introduction

Cloud computing has grown to be a widely used method to utilize huge computer resources and not have to pay huge upfront costs for those resources. This benefits many start up companies who don't have the capital to purchase huge amounts of hardware and pay to maintain it. The main benefit of the pay-as-you-go pricing model is that customers can pay for the resources they actually consume and utilize. The scalability of these resources allows companies to never lose money when the demand for their application is low or high, the cloud platform allows them to obtain the correct amount of resources they need, as in Central Processing Units(CPUs), Virtual Machines(VMs) and Random Access Memory(RAM), for any given moment. The company or customer will need not to worry about the under- or over- utilization of their resources.

As the cloud's popularity continues to grow, different cloud computing companies have risen, causing an increase in the diversification of a CSP's implementation, payment method and security measures of their cloud platform. With the rise of diversity in cloud providers, the ability for a customer to find a cloud provider that would best suit them for their needs becomes increasingly difficult. Deciding among CSPs becomes more difficult because of the differences that would be offered among them. One cloud provider may offer a better security plan for an application while another provider offers a better user experience and security in terms of data protection. This leads to a customer then having to make decisions and decide what they are willing to give up, like a better user experience, so they could get more of something else, like better application security. These decisions are tough for a customer to make and thus puts them in a position where they cannot pick a CSP based on what they want to accomplish. Therefore, it would be beneficial, for the customer, to be able to compare different CSPs and obtain a meaningful difference between them.

We call this meaningful difference a trust score. Trust is defined as a value that takes into consideration the cost-effectiveness, security, and quality of service(QoS) and user experience(QoE) to enable the user to have a trust score customizable to them. Since we account for measurable attributes, like latency and run time, and subjective attributes, such as reviews; a major challenge for us was converting qualitative data into quantifiable values and then accurately combining subjective with objective data. The conversion and combination of these data types will naturally cause data to be lost in the process of achieving a trust score.

We propose a method to compare CSPs by utilizing a directed acyclic graph that represent relations between common cloud computing factors and having the value of the nodes propagate, using Bayes' Theorem, to produce a trust score based on the customer's needs and preferences for each CSPs being evaluated. The use of Bayes' Theorem allows us to still obtain a meaningful and reliable trust score even with the loss of data, as adding any type of data to this theorem will only make it stronger. We had to give each of the

graph nodes a prior probability value representing the chances of that factor occurring independently. The nodes prior probability value would depend upon their input. Input would vary from a collection of real textual reviews, a simulated value within a given range and user input. Attributes that we put into the graph were based on what we saw as important in calculating a CSP's trust score. For example, we believe performance, availability and throughput are vital attributes to the quality of service a customer should expect out of a given CSP. Likewise, star ratings and reviews given by other individuals greatly impact the quality of experience a customer will receive from a particular CSP.
We incorporated Bayesian Network properties and used the product rule for the backward and forward propagation for the nodes' value in the graph. Like many others in the past, we discovered that using the product rule for probabilities caused us to acquire trust scores close to zero, which did not allow the user to understand the difference when comparing cloud providers. To better scale our final trust score in a more user friendly range, we utilized algorithm 2 in [1], that applied weights based on the individual paths of the graph. This algorithm helped us give the user a trust score in a higher range, where now the user can see a difference when comparing cloud providers. This algorithm is intended more for social network graphs, were the dynamic nature of such graphs causes the path lengths to change constantly, but we realized this algorithm can still have implications in our static graph. Since our graph is static and always propagates level-by-level, we will always know the length of individual paths and the number of nodes in that path, thus giving us a static weight and making it irrelevant for our graph. Seeing this, we only take the multiplication of each child, with its parent, and averaging these values for a final posterior probability for the parent. Bayes theorem was modelled in order to represent probability of a factor given some states of their children. Once all nodes have a prior probability, the values get propagated from the children all the way to the trust node to receive a final trust score for the CSP based upon the input that was received and translated into the graph. In producing the final trust score, we would have the Bayes' propagation combine the subjective and objective data. To be able to see the trust score and take in user input, we have implemented a web application where users can select from five CSPs that we have attained our input ranges and reviews inputs. The paper is organized as follows. Section 2 is related works. Section 3 includes thorough definitions of selected nodes from our graph. Section 4 explains how the values of the nodes propagate up

the graph, the process of converting subjective data into quantifiable values and how we implement a user's security preference. Section 5 goes in depth on the model's representation and toolkits we used to develop our web application. Section 6 shows a test run of our model with prior probabilities for the leaf nodes and the trust score outputted. In section 7, we explain the challenges we experienced while conducting this research and also other ideas and approaches we could have taken for this research project. Section 8 then explains what future work needs to be done and section 9 finally concludes the paper.

## 2. Related Works

In previous work, Sen *et al.* [2] introduced a risk assessment model that aided organizations on better understanding the security of their application on a cloud platform and its cost effectiveness and how much responsibility falls, in certain security aspects, on the organization versus the cloud provider. We build on this previous work by also adding qualitative measures like reviews and ratings. We also further improve the security assessment of each cloud provider by allowing the user to give their own security preference on different security domains; we use the Consensus Assessments Initiative Questionnaire(CAIQ, Version 3.0.1) by the Cloud Security Alliance(CSA) to acquire a CSP's security assessment which helps us diversify a CSP's security measures. Krieg [3] gives a thorough explanation of Bayesian probability and belief networks. Using these ideas, we created a directed graph that uses bayesian propagation to give us a final trust score. Correctly combining qualitative with quantitative data was a vital part to obtaining a valid trust score, Qu *et al.* [4] combine objective and subjective by classifying subjective and objective attributes of the cloud and then use fuzzy logic to turn those attributes into fuzzy numbers. Then, by the use of a fuzzy matrix, a user's importance weight on certain attributes and an application of the Euclidean distance formula; they aggregate all the attributes to determine a final score for a CSP. We convert qualitative parameters into quantifiable numbers by using opinion mining for reviews and a user security preference scale of high, medium, low and no preference. We then normalize these values and use a variation of Baye's theorem to combine subjective and objective data in our directed graph. Baye's theorem allows for the propagation of missing data and thus, is an advantage if data is loss from the conversion of subjective data to quantifiable numbers.

Pei *et al.* [5] proposed a methodology that involved evidential reasoning, subjective data manipulation and taking into account multiple factors affecting trust. They incorporate a mapping function that allows for combination of a user's overall trust from their experience from utilizing the CSP before previously along with other users experience with that CSP to form a trust score. They also make it so that the user can have some input on how much they trust other user's opinions on the CSP, so the trust score would also take that into account. They call the raw score from other users the reputation value and they map it to being a trust value based off of how much a user trusts other user's opinions. To take into account multiple factors of trust, they ask the user what factors of the CSP do they want to evaluate for trust. To determine the final trust score, they first retrieve trust evidence which they identify as direct and indirect. Direct evidence being the experiences the user has had with that CSP in the past and indirect being the experiences of other users on that CSP. Then two types of trust scores will be calculated from the evidence that was collected, perception based trust and reputation based trust. These values then update the trust values for the services being evaluated in the user's own trust history of those CSPs. The final trust value will depend upon the user's trust value and the reputation value, which will be aggregated with weights assigned by the user. Then once, the user is done using the CSP that the system selected, they will be asked to give their own review of the CSP and then this will be added as trust evidence. They will be asked specifically rate how well the CSP did according to the factors that they initially chose. Puthal *et al.* [6] discuss the current issues of cloud computing, specifically security, and explain the open challenges of today's cloud. They thoroughly explain the cloud computing architecture and security issues in all three service models, IaaS, PaaS, and SaaS. Their insight helped us formulate what issues were most apparent to the cloud in our view; which we then kept in mind when choosing our leaf nodes for our graph. Li *et al.* [7] developed a tool called CloudCmp with the goal of comparing CSPs by their common services to help customers get a better generalized understanding of the diverse population of cloud platforms. They look at four common sets of functionality between CSPs: elastic compute cluster, persistent storage, intra-cloud and wide-area networks. They conclude by giving results from their test runs and show the comparison of four cloud providers in the different functionalities. To help users better understand the performance and cost difference between CSPs, we also look at different common performance and cost metrics for all five CSPs we are looking at. On top of giving the final aggregated trust score, we also give the score for each subgraph, including network QoS and Cost, which help the user compare different CSPs by their specific subgraphs. Garg *et al.* [8] proposes a trust model that considers performance variation between cloud services based off the requirements for a virtual appliance. They monitor the performance of different cloud services and their selection algorithm will dynamically choose the cloud service that is most trustworthy with the least cost. Due to the nature of Bayes' theorem, our model will also change the trust scores for each CSP dynamically based on when new information is inputed into the leaf nodes. As their algorithm selects a cloud service based on it being the most dependable with the least cost; we give the user trust scores for five CSPs, each personalized by the user's needs, and give the power of comparing and choosing a CSP to the user.

## 3. Definitions

### *3.1. Directed Graph with Adjacency Matrix modeled as a Bayesian Network*

In order for the trust to resemble a meaningful score, the placement of the nodes are significant for the calculations. You would not want to have causal relations between nodes that do not have any relation with each other. For example, you would not want a node that resembles User QoE to have a relationship with a node that impacts Network Performance. It would not make sense because User QoE does not have a direct impact on Network Performance and the value coming from their propagation would not have any meaning with the causal relationships between nodes in a Bayesian Belief Network[3].

We followed this methodology in the construction of our graph structure, with all of the connections representing a causal relationship in the direction of the arrows that we modeled off of Bayesian Belief Networks [3]. This graph structure is beneficial because it is able to handle cases of uncertainties, and represent a logical flow of which nodes affect each other. We take the concept of OR gates from Sen [9], that talks about risk assessment utilizing attack graphs that apply OR/AND gates to represent different attack progressions that can be done within a sensor network. We do this because there are situations in our analysis where not all of the parent nodes will be present to affect the trust score. For example, in the cost subtree, we realized that some CSPs did not provide all of the types of cloud service models(IaaS, Paas, and SaaS). In order to represent this situation, we placed OR gates for the arrows connecting the Cost subgraph nodes with it's parents to show that in some cases, not all of the Cost nodes will be taken into consideration when propagating. In order for propagation to take place, the nodes would have to be initialized with a value between zero and one and the way that the nodes obtain this value differ for their situations. The different types of inputs for the nodes range from:

- Simulated user input
- Actual user input
- Text
- Yes/No/NA scoring
- Random entry value within a certain range

| **Leaf Nodes** | **Range** | **From** |
|---|---|---|
| Total Runtime | 0-1200 ms | [10] |
| VM type | 1-4 | [11] |
| Network Bandwidth | 1.6-6.4 mbps | |
| Multi-tenancy | 1-10 VM's | [12] |
| Global latency | 15-188 ms | |
| Uplink | 1.87-100.97 mbps | |
| Downlink | 3.4-179.22 mbps | [13] |
| Latency | 26-344 ms | |
| Uptime Percentage | 0-100 | |
| CPU Speed | 1000-20000 MIPS | [14] |
| Overall STAR score | 0-295 questions | [15] |
| Storage Size | 2-2048 GB | |
| Disk I/O | 43636-160000 IOPS | |
| Number of Cores | 1-128 | [16][17][18][19][20] |
| Bandwidth | 100-2048 GB | |
| Transactions Processed | 1-1000000 | |

Table 1: Ranges found for certain leaf nodes

For a large majority of the nodes, values are obtained through randomly selecting a value within a minimum and maximum range. These ranges, shown on **Table 1**, being different for all the nodes who obtain their value in this manner. The reason that we went about doing ranges is because some nodes have values that are not in the range between 0 and 1. For example, a node like Latency can have an input like 27 ms. So finding the range possible, for our 5 CSPs, for each node allowed us to set a range for possible values for that node, which we then normalized using that range with the normalization equation. These ranges would be determined for the individual nodes and they were found through 5 CSP's cost calculators, other research papers and Cloudharmony.com.

The reason why we have separated user input as simulated and some of the data actually coming from user input is because for the situations that we utilized simulated user input are cases where the user would've need to utilize certain tools for measurements. We considered this too much of a hassle for the user to obtain, so as instead of asking the user to utilize certain tools to obtain values necessary to input to certain nodes we simulated them instead. The input that would be coming from the user, whether it be simulated or actual user input, would would always be converted to a value between zero and one. For simulated user input, the nodes would take in a random number within a set range and determine the converted value by normalizing with the range for that specific node. For actual user input, choices of measurement, such as high, medium or low, would be given to the user and we would set specific values in the zero to one range to represent the choices. The value being put into the actual user input nodes would depend on the set value of the user choices.

Two nodes, User Feedback and 3rd party reviews, take in initially plain text that represents reviews on the CSP being evaluated. To obtain numerical values from text reviews, we utilized an opinion mining tool from nltk [21], a Python package, that would give three separate scores for a text review in the form of a tuple: [positive, negative, neutral]; the sum of the score would be one. The positive and negative scores are then aggregated for the total number of reviews and then combined in such a manner that they form a value between zero and one. This process is explained in more detail in the methodology section. The nodes receiving text aren't the only nodes who receive a tuple scoring. All of the control group nodes also receive a tuple of three values for each entry, but instead, this tuple is in the form of: [Yes, No, N/A] and it represents percentages of questions answered yes, no and NA by a single CSP. These values are normalized similarly to the values produced from opinion mining the review texts and are described in more detail in the methodology section.

In the following subsections, certain nodes are described in more depth as to clear up possible confusions of their meaning and to lead to better understanding of their parents and their purpose in our final trust score.

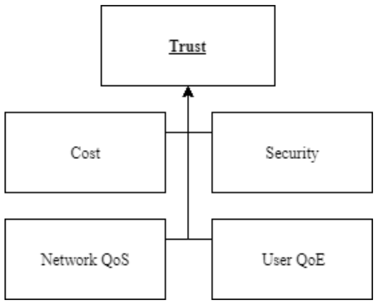


Figure 1: Root

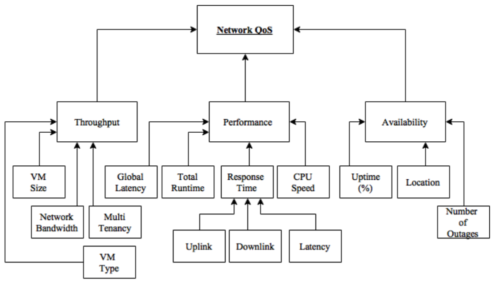


Figure 2: Network QoS

## *3.2. Network Quality of Service (QoS)*

Network QoS subgraph **Figure 2**, whose probabilities represents the likelihood that the services being provided through a CSP's network will be satisfactory. Throughput, Performance and Availability are nodes chosen as children of Network QoS because they are seen as major factors of Network QoS. They affect the network speed, how much data is outputted, and whether the network would be able to provide data that you request in a timely manner. A large part of the node ranges in this subtree were obtained through finding reliable ranges for our 5 CSPs from other research papers **table 1**.

### *3.3. Cost*

Cost subgraph **Figure 3**, whose value represents the possibility of saving or losing money based on resources being utilized and pricing of services. We use the OR gate to signify the possibility that a CSP might not have certain cloud service models or may have a different payment method.

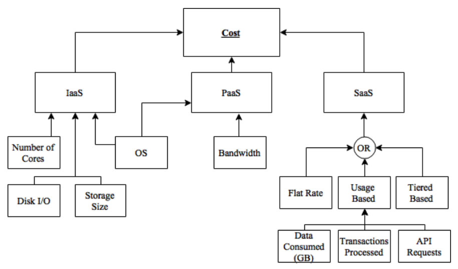


Figure 3: Cost

### *3.4. Cloud Service Models (IaaS, PaaS, SaaS)*

These nodes, in **Figure 3** take in a value within a range of minimum and maximum cost ranges for utilizing those services with certain inputs. Inputs being varied among the different service models. These minimum and maximum cost values were estimated from utilizing the various cost calculators from the 5 CSPs that we analyzed. The ranges were calculated by taking into account the factors that effected those services in the cost calculator or indirectly could've effected the cost.

### *3.5. Bandwidth*

The Bandwidth node in **Figure 3** represents the amount of outgoing traffic per month that the user produces from utilizing the service. The range in **Table 1** for Bandwidth was estimated as a good range of the amount of traffic that could be produced with the help from the Google cloud platform cost calculator[20]. This node is different from the bandwidth node in **Figure 2** because it represents outgoing traffic per month while the one in **Figure 2** represents a rate to present maximum amount of data to be processed during a given time.

### *3.6. Transactions Processed*

Transactions processed, in **Figure 3**, is a form of the pay-as-you-go model where a CSP will charge a customer based on the number of transactions they incur in a given time period. To turn this input into a value between zero and one, we first find the minimum and maximum possible values for the amount of transactions processed in a month. Then we divide the simulated value by the max value of transactions that we set per month. With the result being input into the Transactions Processed node that represents the prior probability of the node being a large factor of resource usage in the cloud environment. This is important for our cost subgraph because it helps to analyze how much the client will be utilizing CSP resources helping update the usage based value.

### *3.7. User Quality of Experience (QoE)*

The User QoE subgraph **Figure 4**, represents the likelihood on how good of an experience the user should expect based upon a CSP's reputation from previous customers along with the user's own subjective view of the CSP in the form of a review and star rating.

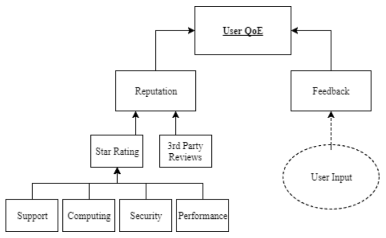


Figure 4: User QoE

### *3.8. Cloud Security*

The Cloud Security subgraph **Figure 5**, represents how well a CSP does in terms of security based on the user's security preference and the Cloud Security Alliance(CSA) Consensus Assessment Initiative Questionnaire(CAIQ) and Cloud Controls Matrix(CCM) [15]. We use another OR gate here because if the user gives no preference, then we only look at the Overall STAR score. If a preference is given by the user, then we take the Control Group Score as the prior probability for our propagation and ignore Overall STAR score. Control group bias is a node obtaining the value of relevancy of each control group to a security domain like network or storage; the relevancies coming from Cloud Controls Matrix [15].

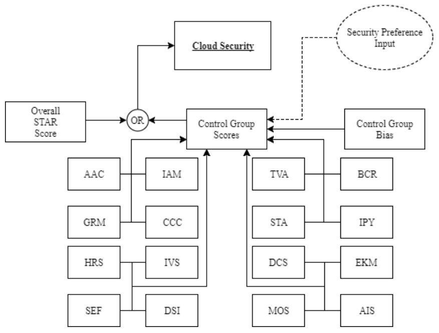


Figure 5: Security

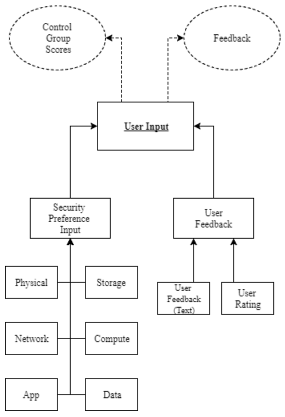


Figure 6: User Input

*3.9. Overall STAR Score*

The Overall Security Trust and Assurance Registry(STAR) Score represents a score for security based upon answers to security questions that could be answered either with a yes, no or N/A forming a tuple for every control group. The sixteen individual tuples are then aggregated together without taking into account any preference from the user and then turned into a normalized value, the conversion is explained later in methodology.

*3.10. Control Group Score*

Control Group Score has the the same goal as the Overall STAR Score but this node's value also depends upon a user security preference on certain domains, and the importance of each control group in the chosen domains. Control groups are categories in the CAIQ, from Cloud Security Alliance [15], that represent different aspects of security being evaluated in the CSP's cloud platform. For example, some of the categories are Data-center Security(DCS), Encryption and Key Management(EKM), and Threat and Vulnerability Management(TVM). These categories of questions will check to see if the CSP's policies for those categories are in-line with standard security polices. Such as, for DCS, they will check to see how secure the data-center locations are. They ask questions in the DCS control group in regards of the people who have access to physical servers, and how well their data-centers are protected by natural disasters.

## 4. Methodology

x = a value between range [min,max]:

$$Normalization = \frac{(x - min)}{(max - min)} \tag{1}$$

Given user and measured inputs, we try to calculate a final trust score for a certain cloud service provider. Using the product and chain rule (2) to propagate up the subgraphs, **figures: 2, 3, 4, 5**, and finally to the root of our entire graph **figure 1**, we are left with a final trust score dependent on the cost, security, quality of service and experience for a cloud provider. At the end, the user will be left with 5 final scores, each from top cloud providers, to which they can then compare. All leaf node values need to be in a range between 0 and 1 for our conversion of prior to posterior probabilities to work. We have certain input that does not come in the form of a number, for example reviews that are in text, and also values that take into account a certain level of preference given by the user, like User Security Preference where the user decides what domains it prefers. We needed to make sure we could turn review text to a quantifiable value and normalize it so it is between 0 and 1. Same goes for any other node that had a preference associated with it or was in a specific range, for example, runtime for all 5 cloud providers was found to be between 0 and 1200 ms, so we randomize a value between 0 and 1200 and then normalize it using it's known range and (1). This same method is applied to all other leaf nodes that have integer values coming in. We find the range possible for all 5 cloud providers for a leaf node and then randomize a value between that range and then use (1) to get a value between 0 and 1 and put that value into the leaf node.

### *4.1. Propagation of Value of Nodes*

We first estimate and store the unconditional or prior probabilities of every node in the graph. Every leaf node's unconditional probability is obtained from user input or CloudHarmony.com. Every Parent node is either a belief node, meaning the user enters their belief on what this unconditional probability should be based on their subjective view, or a measurable node, where the user has the ability to obtain this unconditional probability for their own application using benchmark testing. **Table 2** shows the classification of all non-leaf nodes.

| **Belief** | **Measuring** |
|---|---|
| Cloud Security | Availability |
| Cost | Cost Subgraph |
| Feedback | Performance |
| Network QoS | Response Time |
| Reputation | Throughput |
| User Feedback | |
| User QoE | |

Table 2: Non-leaf nodes Classifications

After these probabilities are set, we use the chain and product rule to propagate the values up the graph from the leaf nodes to the root. We first calculate the conditional probability for the parent nodes of all the leaf nodes using the product rule. $c_i \epsilon P_a$, $P_a$ denotes the parent node connected to its children, $c_i$. The posterior probability, $P_r(Pa_j|c_n)$, of the parent of the leaf nodes is then calculated from set S of its children, S = $(c_1,...,c_n)$, with $P_p(x)$ denoting prior probability.

$$P_r(Pa_j|c_n) = P_p(Pa_j) * [\prod_{i=1}^{n} P_p(c_i)] \qquad (2)$$

After calculating all the posterior probabilities for the parents of the leaf nodes, The parents of the leaf nodes become the children of its parent node and equation (2) is reused to calculate the posterior probability of its parent node, the leaf node's ancestors. We continue to do this, turning everything into a posterior probability until we reach the root node. A challenge that we came upon, and many others have in the past, was that using this form of bayes probability and multiplying the probabilities until we reached the root was resulting in trust scores being extremely small values close to zero. To mitigate this issue, we used algorithm 2 in [1]. The way this algorithm is intended to work is by looking at each individual path from the root(destination) to the leaf node of that path(source), and applying a certain weight, depending on the path length and number of nodes,to the multiplication of all the probabilities held in that path. For example, a path with one node in between source and destination would have a path length of 2 and 3 nodes, from this, the weight will then be 1. Each path then becomes an effective path, after being multiplied by its weight. After all paths are turned into

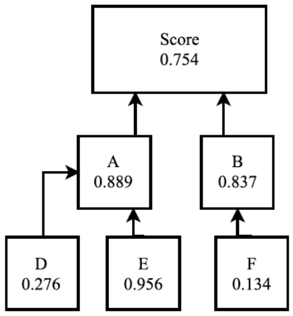


Figure 7: Example of how chain and product rule propagate

effective paths, the algorithm then takes the average of the effective paths and that becomes the value you receive at the root of the graph.

$$weight = (1 - \frac{Pathlength - 2}{Nodes - 1}) \tag{3}$$

The creators of this algorithm intended to use this for a social networking graph where dynamic changes are inevitable and they needed to take into account that their graph could easily become smaller or bigger. This algorithm puts more weight on smaller paths and smaller weights on longer paths. In our case, we know the path lengths and number of nodes due to the nature of our graph being static at all times. Looking at **Figure 7**, level by level means we first calculate the posterior probability for A, B and C from their respective child nodes and then using the calculated posterior probabilities to calculate the posterior probability for the Score or root node. We are using the product rule level by level and thus will always have a path length of 1 with 2 nodes from parent to each of its respective children's paths. Since we will always have the same path length and number of nodes, our weight will stay constant and will not factor in as intended to by the algorithm. We took the concept of algorithm 2 without the weight factor, so we multiplied each node value in the path from child to its parent by its prior probabilities and averaged all paths that go to the same parent.

If we have a graph with its unconditional probabilities as in **Figure 7**, we can calculate the posterior probability of the score with the product rule and using the path length algorithm [1]:

Path Length = 1 Nodes = 2

$E_t(s,d)$ = Effective Path from source (s) to destination (d)

$$\text{Score} = P_r(Score|A,B) = \frac{E_t(A,Score)+E_t(B,Score)}{2}$$

$$E_t(A,Score) = P_p(Score) * P_r(A|D,E)$$

$$E_t(B,Score) = P_p(Score) * P_r(B|F)$$

$$P_r(A|D,E) = \frac{E_t(D,A)+E_t(E,A)}{2}$$

$$P_r(B|F) = \frac{E_t(F,B)}{1}$$

$$E_t(F,B) = P_p(B) * P_p(F)$$
$$E_t(D,A) = P_p(A) * P_p(D)$$
$$E_t(E,A) = P_p(A) * P_p(E)$$

For **figure 7**

1) Get effective paths for A and B from their respective children's paths

$$E_t(F,B) = 0.837 * 0.134 = 0.112$$
$$E_t(D,A) = 0.889 * 0.276 = 0.245$$
$$E_t(E,A) = 0.889 * 0.956 = 0.850$$

2) Obtain Posterior probabilities for A and B taking average of effective paths

$$P_r(A|D,E) = \frac{0.245+0.850}{2} = 0.548$$

$$P_r(B|F) = \frac{0.112}{1} = 0.112$$

3) Calculate effective paths from A and B to Score

$$E_t(A,Score) = 0.754 * 0.548 = 0.413$$
$$E_t(B,Score) = 0.754 * 0.112 = 0.084$$

4) Trust score can now be measured

$$P_r(Score) = \frac{0.413+0.084}{2} = 0.249$$

**Score = 0.249**

## *4.2. Control Group Score*

The Cloud Security Alliance provides us with a questionnaire that allows cloud providers to self-assess themselves on different areas of their cloud service based on security. The Consensus Assessments Initiative Questionnaire is divided into 16 control groups that offer different areas of a cloud service. The cloud provider can fill in this questionnaire by either answering yes, no or not available(n/a) for every question. The questionnaire asks a wide range of questions with the goal of obtaining a better understanding of a cloud provider's security measures. We use this aggregation of all the yes's, no's and n/a's to obtain an Overall STAR score. We also would like to take this one step further and add a user preference on six different domains. With the help of the Cloud Controls Matrix, which specifies how many questions from each control group is related to a specific domain, we can then create a Control Group Score based on the aggregation of yes's, no's, and n/a's and the security preference given by the user. Now the user can specify if they see the network as more important to their security needs than another domain like storage. From this, we get a Control Group Score between 0 and 1 that is more relevant to the user.

$$ControlGroupScore(\alpha, \beta, \delta) \tag{4}$$

$\alpha_i$ = Control Groups(i = [1,16])
$\beta$ = Cloud Controls Matrix(CCM) Bias
$\delta$ = User Security Preference
$T_{\alpha_i}$ = Total number of questions for control group($\alpha_i$)

Input: Control Groups individual scores in tuples of (Yes, No, N/A)

$$\sum_{n=1}^{T_{\alpha_i}} (Yes_n, No_n, N/A_n) \tag{5}$$

1) Get average for each control group:

$$\left(\frac{Yes_n}{T_{\alpha_i}}, \frac{No_n}{T_{\alpha_i}}, \frac{N/A_n}{T_{\alpha_i}}\right)$$

2) We will only consider the Yes's and No's since a CSP answering N/A should not go against or for them, we then find the difference between the Yes's and No's:

$$\alpha_{i_{diff}} = (\frac{Yes_n}{T_{\alpha_i}} - \frac{No_n}{T_{\alpha_i}})$$

3) Giving us a value for $\alpha_{i_{diff}}$ between [-1,1], we then normalize this value using (1) for the final value for $\alpha_i$:

$$\alpha_i = \frac{(\frac{Yes_n}{T_{\alpha_i}} - \frac{No_n}{T_{\alpha_i}}) - min}{max - min}$$

4) Steps 1-3 are applied for the Overall STAR score which is the aggregation of the total number of (Yes, No, N/A) for all control groups. To incorporate a user preference, we need to do steps 1-3 individually for every control group and then we will be ready to include a user security preference for all domains.**Table 3** shows how user can answer for each domain and their respective weights:

| **User Preference** | **Weight** |
|---|---|
| High | 3 |
| Medium | 2 |
| Low | 1 |
| No Preference | 0 |

Table 3: Possible values for Domains Preferenced

5)We will now take preference from CCM for every Control Group using domains preferred by user and use the Weighted Mean Formula (7) to account for the weight of both CCM and user:

$$weight(w) = \sum_{i=1}^{i=16} [\sum_{j=0}^{j=6} (DP_{j,\alpha_i} * \beta_{DP_{j,\alpha_i}})] \tag{6}$$

$$\sum_{i=1}^{i=16}(\frac{\alpha_i * w_i}{w_i}) \tag{7}$$

DP = Domains Preferenced:(App, Compute, Data, Physical, Network, Storage)
N = [0,6], the number of domains preferenced by user
$User_{DP}$ = Weight of all domains preferenced, either in high, medium or low
$\beta_{DP}$ = weight specified by CCM for each domain preferenced for control group $\alpha_i$

| User Preference | Storage: High, Data: Low |
|---|---|
| AIS | 0.6 |
| BCR | 0.8 |
| CCM bias for AIS | Storage: 0.25 Data: 0.50 |
| CCM bias for BCR | Storage: 1.0 Data: 0.75 |

Table 4: Example using 2 control groups and their bias

**Table 4** shows two control groups (AIS and BCR) and their respective scores so we can demonstrate how (7) works:

$Weight_{AIS}$= $(Storage_{User} * Storage_{CCM}) + (Data_{User} * Data_{CCM})$

= (3*0.25)+(1*0.50) = 1.25
$Weight_{BCR} = (Storage_{User} * Storage_{CCM}) + (Data_{User} * Data_{CCM})$
= (3*1.0)+(1*0.75) = 3.75
Now that we have the weights, we use (7):

**Control Group Score** $= \frac{(AIS*weight_{AIS})+(BCR*weight_{BCR})}{weight_{AIS}+weight_{BCR}}$

$= \frac{(1.25*0.6)+(3.75*0.8)}{(1.25+3.75)} = \frac{3.75}{5} = \mathbf{0.75}$

*4.3. Reviews*

Input: Review text for each CSP turned into a positive, negative and neutral rating summing up to 1 using opinion mining:
(Pos:$\alpha_1$, Neg:$\beta_1$, Neu:$\delta_1$)...(Pos:$\alpha_n$, Neg:$\beta_n$, Neu:$\delta_n$)

1) Sum up all positive, negative and neutral values for each review per CSP to get total(T):

$$\sum_{i=1}^{n}(Pos : \alpha_i, Neg : \beta_i, Neu : \delta_i) = (Pos : \alpha_T, Neg : \beta_T, Neu : \delta_T) \tag{8}$$

2) get average of positive, negative and neutral of total reviews by dividing by total number of reviews($Reviews_T$):

$$(\frac{\alpha_T}{Reviews_T}, \frac{\beta_T}{Reviews_T}, \frac{\delta_T}{Reviews_T}) \tag{9}$$

3)Find difference between average of positive and negative reviews($\frac{\alpha_T}{Reviews_T} - \frac{\beta_T}{Reviews_T}$) to get a range between [-1,1] and then use (1):

**3rd Party Reviews** $= \frac{(\frac{\alpha_T}{Reviews_T} - \frac{\beta_T}{Reviews_T}) - min}{(max - min)}$

## 5. Implementation and Simulation of Trust Computations

In order to implement the web application, we use the python coding language along with the *Django* web-development framework [22] This package makes available the bulk of what we need to maintain the application and it's database. In order to represent the graphs, we use directed, weighted adjacency matrices. We use the *Numpy* python package[23] to facilitate the creation and operations of the matrices. We store the values which propagated throughout the graph as the weights on each edge of propagation. First the edges which extended from the leaf nodes of the graph are initialized based on the CSP information in our database. Additionally, the prior belief values of each node on the graph is stored in their respective edges of propagation. Next, our propagation formula is systematically run on each set of children nodes to calculate the value of their parents. This is done until the final trust value is propagated to the root node of the entire graph. In order to obtain the information about the cloud service providers, we use a web crawler to pull information from review sites. We use the *Scrapy* [24] python package to facilitate this. On any textual review data we obtain, we use the *NLTK*(Natural Language Tool Kit) python package[21] to conduct opinion mining on the data. We use this information to determine each user's opinion of the cloud service provider they reviewed.

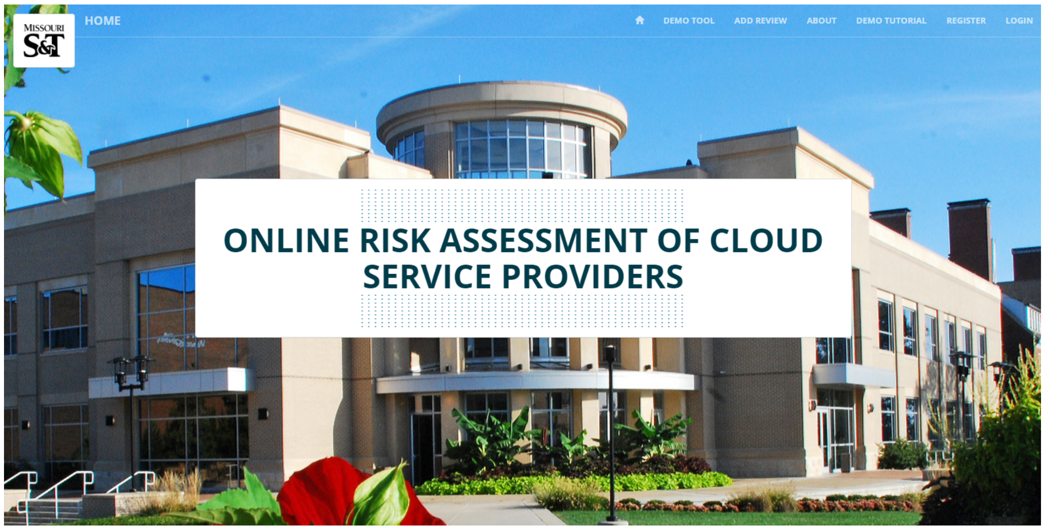


Figure 8: Home page of the Web application.

For the application visuals, we use the HTML and CSS formatting languages, employing a Bootstrap template (Insert here where the template was from) for the overarching style and structure. For displaying graphs, we use the Google Chart API[25] side by side with javascript code.

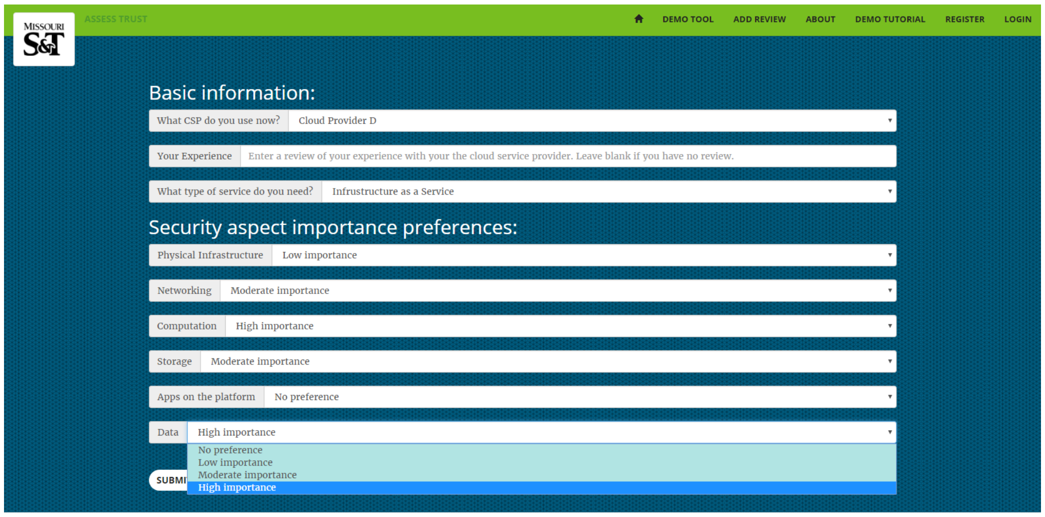
Through these packages and toolkits our web application allows the user to select a service provider, select some security preferences, add textual reviews about the CSP and give a scoring for the CSP. Once all this input is taken in and the user presses the submit button, the application goes through the whole process of propagating through the graph to come up with final trust score. This trust score will be represented at the end along with it's parent nodes values in a graphical view.

## 6. Simulation

### *6.1. Setup*

When first entering the website, you are taken to the home page, but in order for you to run your own test runs on the five CSPs that we have available for simulation, you first have to click on the button called "Demo Button" as the **(Input here reference to home page)**.

Once you clicked the "Demo Button", you should arrive to the page shown as the **Figure 9**. In this page you will provide information that will be



Basic information:

| | |
|---|---|
| What CSP do you use now? | Cloud Provider D |
| Your Experience | Enter a review of your experience with your the cloud service provider. Leave blank if you have no review. |
| What type of service do you need? | Infrustructure as a Service |

Security aspect importance preferences:

| | |
|---|---|
| Physical Infrastructure | Low importance |
| Networking | Moderate importance |
| Computation | High importance |
| Storage | Moderate importance |
| Apps on the platform | No preference |
| Data | High importance |

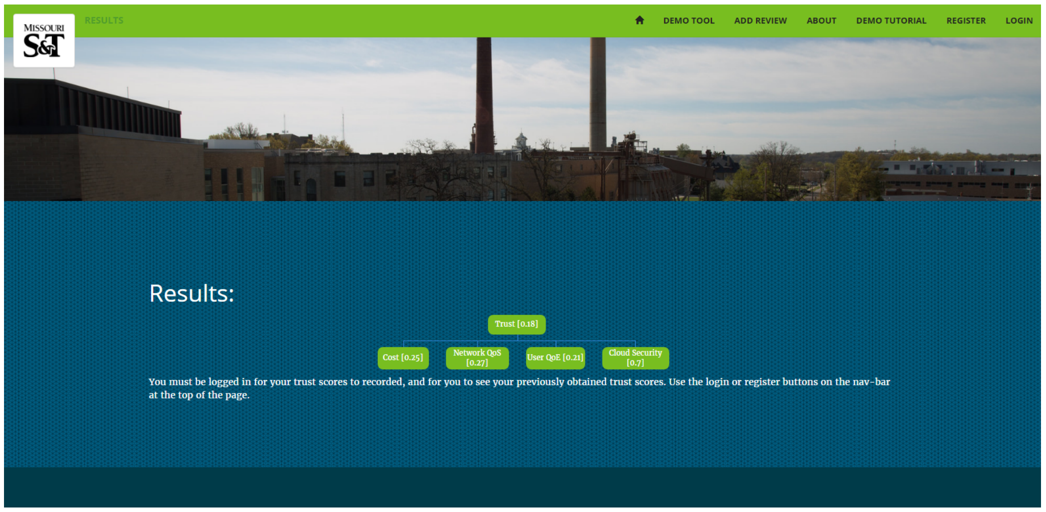
No preference
Low importance
Moderate importance
High importance

Figure 9: Input page for demo tool.

utilized for the calculation of the trust score. The type of information to input ranges from the CSP that you want to analyze, preferences for security domains, textual review of the CSP that you are going to analyze and a scoring that you would like to give the CSP. Once you are done inputting all the information on this page, just press submit and wait for a new page to respond.

## *6.2. Results*

Once the submit button was pressed with all the information obtained from **Figure 9** the web application will take you to **Figure 10**. On this page you will be able to see the final trust score calculated, as well as the values of the category subtrees. To save space, the further divisions of the subtrees remain hidden. For each category, a score between zero and one is given. A value of one is a perfect score, while a value of zero is the worst possible score. You can use these scores to assess which CSPs are better for you in which categories. If you are logged in to the application, your previous score assessments will be listed below. Under the circumstances that the same configurations where given, the highest final trust score will be assigned to the CSP which is considered the best choice to suit your needs in total.



Results:

You must be logged in for your trust scores to recorded, and for you to see your previously obtained trust scores. Use the login or register buttons on the nav-bar at the top of the page.

Figure 10: Output page from utilizing demo tool.

## 7. Discussion

During the progression of this project multiple hurdles came along the way. Some main problems that we came across was that crucial information that would be useful for calculations isn't publicly available. This came along with some problems for crawling certain websites and issues with logic.

In order to combat the fact that some knowledge just can't be obtained because making that knowledge public would raise the risk of security breaches for the CSP and their clients such as releasing specific places of where security is not as strengthened as other places. Taking this into account we utilized as much of the public knowledge that we could to be able to provide a way to compare providers on limited information. This led us to simulating a large portion of the nodes values, also nodes that would be considered user input.

Some troubles we had revolving around crawling websites was the fact that in certain cases such CloudHarmony and Clutch, there would be moments where the site would take a while to retrieve results. Cases such as when we would've liked to obtain results from CloudHarmony calculations[13] and a series of points from Clutch[26] representing a series of scorings for individual CSPs. The problem that we received from trying to crawl those specific parts of the sites was that it would take some time for the results to pop up on the pages while the crawler that we used would only look for the path at the beginning and return nothing if the path wasn't found. The path that we would state to crawl would only appear after the results were finished displaying. Making the crawler wait for a certain amount of time didn't fix the issue either. In order to obtain these values, we merely grab the values from the site and manually write them down. For the Clutch site, we simply inserted the values obtained from the site and manually put them in a database to be used. For CloudHarmony, we ran multiple tests on the CSPs that we've chosen and found the min and max out of those tests and put them as ranges in the web application.

After further calculations with the process that we set in place. We noticed that within some simple scenarios that the final score would not change much when you would apply the weights or not, when the value from the weights being applied would be normalized.

## 8. Future Work

Gathering input to evaluate CSPs can be difficult to obtain because some information that can be important in evaluations isn't information that a CSP would want to distribute to the public. For example, information such as knowing amount of datacenters/servers at a CSP's disposal, could be harmful to them and their customers if released publicly because an attacker of their systems would have a better idea of the effect of their attacks if successful. In order to get around needing information that would be valuable to have we gathered as much relevant public information in regards to trust score calculations as we could

Something to add to this project in the future could be setting up a way that information that is sensitive towards specific providers, be encrypted throughout the whole process. A way of being able to do calculations on these encrypted values would be beneficial only if more realistic data were to be obtained. Could also go about making more usage of public information.

## 9. Conclusion

In this paper we propose a risk assessment structure, modeled off of Bayesian Belief Networks in order to take advantage of causal relationships between events and the loss of information when combining subjective and objective data. We would initialize all the graph nodes in various ways but always made sure that the value being put in them would be a value between zero and one. Once the node values were initialized to a value within the zero and one range, propagation and combination of those values through the graph would take place by utilizing algorithm 2 [1]. We do this propagation level by level, We first take the prior probabilities of the leaf nodes and their corresponding parent, to create a posterior probability for their that parent. Once we obtain the posterior probabilities from the lowest level(leaf nodes) to the second lowest level(leaf node's parents), we use these posterior probabilities to then propagate up to the next level and so on. Through this process, we would eventually reach the root node, the trust score, and acquire a score for the given CSP based off the cost-effectiveness, quality of user experience and service, and the security of the cloud provider while also taking into account the customer's security needs for their application.

## 10. References

[1] P. S. Chakraborty, S. Karform, Designing trust propagation algorithms based on simple multiplicative strategy for social networks, in: Procedia Technology, volume 6, Elsevier, 2012, pp. 534–539.

[2] S. Madria, A. Sen, Offline risk assessment of cloud service providers, in: IEEE Cloud Computing, volume 2, IEEE, 2015, pp. 50–57.

[3] M. L. Krieg, A tutorial on bayesian belief networks (2001).

[4] L. Qu, Y. Wang, M. A. Orgun, L. Liu, A. Bouguettaya, Context-aware cloud service selection based on comparison and aggregation of user subjective assessment and objective performance assessment, in: 2014 IEEE International Conference on Web Services, IEEE, 2014.

[5] W.-J. Fan, S.-L. Yang, H. Perros, J. Pei, A multi-dimensional trust-aware cloud service selection mechanism based on evidential reasoning approach, Springer, 2015.

[6] D. Puthal, S. Mishra, S. Swain, Cloud computing features, issues and challenges: A big picture, in: 2015 International Conference on Computational Intelligence and Networks, IEEE, 2015.

[7] A. Li, X. Yang, S. Kandula, M. Zhang, Cloudcmp: Comparing public cloud providers, in: 10th ACM SIGCOMM conference on Internet measurement, ACM, 2010.

[8] S. K. Garg, L. Gao, J. Montgomery, Clouds selection for network appliances based on trust credibility, in: 2015 International Telecommunication Networks and Applications Conference, IEEE, 2015.

[9] A. Sen, S. Madria, Risk assessment in a sensor cloudframework using attack graphs (2015).

[10] A. O. Ayodele, J. Rao, T. E. Boult, Performance measurement and interference profiling in multi-tenant clouds, in: 2015 IEEE 8th International Conference on Cloud Computing, IEEE, 2015.

[11] V. Persicoa, P. Marchettaa, A. Bottaa, A. Pescapèa, Measuring network throughput in the cloud: The case of amazon ec2, in: Computer Networks, volume 93, Elsevier, 2015, pp. 408–422.

[12] X. Chen, L. Rupprecht, R. Osman, P. Pietzuch, W. Knottenbelt, F. Franciosi, Cloudscope: Diagnosing and managing performance interference in multi-tenant clouds, in: 2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, IEEE, 2015.

[13] "cloudharmony.com", in: CloudHarmony.

[14] Z. Cai, Q. Li, X. Li, Elasticsim: A toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times, in: Journal of Grid Computing, volume 15, Springer, 2016, pp. 257–272.

[15] "cloudsecurityalliance.org", in: Cloud Security Alliance.

[16] "calculator.s3.amazonaws.com", in: Amazon Web Services Simple Monthly Calculator.

[17] "azure.microsoft.com", in: Microsoft Azure Pricing Calculator.

[18] "console.bluemix.net", in: IBM Bluemix Cost Calculator.

[19] "www.rackspace.com", in: Rackspace Cloud Calculator.

[20] "cloud.google.com", in: Google Cloud Platform Pricing Calculator.

[21] "www.nltk.org", in: Natural Language Toolkit.

[22] "www.djangoproject.com", in: Django 1.11.2.

[23] "www.numpy.org", in: Matrix Arithmetic and Representation for Python.

[24] "scrapy.org", in: Scrapy 1.4.

[25] "developers.google.com", in: Organization Chart.

[26] "clutch.co", in: Clutch Cloud Service Provider Review Site.