# A Comparative Analysis of Trust-based Service Composition Algorithms in Service-Oriented Ad Hoc Networks

Yating Wang, Ing-Ray Chen
Virginia Tech
Department of Computer Science
{yatingw, irchen}@vt.edu

Jin-Hee Cho
U.S. Army Research Laboratory
Computational and Information Sciences Directorate
jinhee.cho@us.army.mil

Jeffrey J.P. Tsai
Asia University
Department of Bioinformatics and Biomedical Engineering
jjptsai@gmail.com

## ABSTRACT
In this paper we analyze performance characteristics of trust-based service composition and binding algorithms for composing composite services in service-oriented ad hoc networks. We first show that trust is an effective mechanism to cope with malicious nodes acting for their own gain and colluding with each other to increase their chances of being selected for service execution. Then we perform a comparative performance analysis of single-trust-based, multi-trust-based, and context-aware trust-based service composition and binding algorithms for composing composite services in service-oriented ad hoc networks. Using an encounter-based service composition application with multi-objective optimization goals as a running example, we show that context-aware trust-based service composition and binding outperforms single-trust-based and multi-trust-based counterparts for achieving multi-objective optimization and user satisfaction. Physical interpretations of the results are provided.

## Keywords
Service-oriented ad hoc networks; service-oriented computing; service composition; trust management.

## 1. INTRODUCTION
Service oriented architecture (SOA) based composite services have been widely employed in Internet-based web service systems. With the proliferation of smart devices carried by human operators, SOA-based composite services now are extended to service-oriented ad hoc networks (SOANETs). A SOANET is populated with service providers (SPs) and service requesters (SRs) all of which are mobile. An example SOANET is a vehicular ad hoc network or a mobile ad hoc network populated with smart objects. Unlike a traditional web service system in which nodes are connected to the Internet, nodes in SOANETs are mobile and an SR will need to request services from available SPs with which it encounters and interacts dynamically. With a service request in hand, an SR has to first formulate a service composition plan based on the available SPs, and then determine the best node-to-service binding for executing the composite service.

An effective way to deal with malicious behavior is to use trust for decision making [1], [2], [5]. The use of trust for service composition and binding is still in its infancy. Wahab et al. [9] provided a comprehensive survey on service composition and binding and discussed the associated challenges in web services. Very recently, Wang et al. [10] considered integrity as an additional trust metric to cope with malicious attacks in SOANET environments. That is, "integrity trust" is used as confidence to assess the validity of "competence trust" in that competence trust ultimately ensures service success. Wang et al. [11] associated context with service quality such that a SP's service behaviors (affecting quality of information, service delay, and service cost) are inherently associated with rapid context environment changes in SOANETs.

Relative to existing works cited above, our paper aims to provide a guidance of how and what trust protocol should be used to maximize service composition and binding application performance under a hostile environment. This paper has the following unique contributions:

1. We demonstrate that trust can effectively prevent malicious nodes from disrupting composite services in SOANETs. We conduct a detailed performance analysis and demonstrate that trust-based service composition and binding outperforms the non-trust-based, blacklist-based counterpart, and can effectively filter out malicious nodes, which ultimately leads to high user satisfaction.

2. We are the first to conduct a comparative performance analysis of non-trust vs. single-trust (BRS [5]) vs. multi-trust (TRM, SRM [10]) vs. context-aware trust (CATrust [11]) protocols for peer-to-peer trust evaluation in SOANETs. Our results demonstrate that context-aware trust-based service composition and binding (CATrust) can significantly outperform single-trust-based (BRS) and multi-trust-based (TRM/SRM) counterparts by leveraging the association between context and service quality when composing composite services in SOANETs.

## 2. SYSTEM MODEL
We consider three service quality criteria: QoI (quality of information), QoS (quality of service with service delay as the main attribute), and cost. We denote them by *Q*, *D*, and *C* which may be measured after service invocations are performed. While *D* and *C* are easily measureable physical quantities, *Q* is specific to the application domain. For example, in environment monitoring service, *Q* is measured by the extent to which the output contributes to the ground truth data. The objective is to maximize *Q* while minimizing *D* and *C* with multi-objective optimization (MOO) goals.

We first scale our service quality metrics, *Q*, *D* and *C*, to the range [0, 1] so that the higher the value, the better the quality, as follows:

$$\bar{Q} = \frac{Q - Q_{min}}{Q_{max} - Q_{min}}; \quad \bar{D} = \frac{D_{max} - D}{D_{max} - D_{min}}; \; \bar{C} = \frac{C_{max} - C}{C_{max} - C_{min}}. \tag{1}$$

Here $Q_{max}$ and $Q_{min}$, $D_{max}$ and $D_{min}$, and $C_{max}$ and $C_{min}$ are the maximum and minimum possible values of *Q*, *D*, and *C*, respectively. They are known a priori. With this normalization we transform MOO into multi-objective maximization, i.e., from maximizing *Q* and minimizing *D* and *C*, into maximizing $\bar{Q}$, $\bar{D}$ and $\bar{C}$. From a pragmatic perspective, scaling facilitates a fair quantitative comparison of different service quality criteria, as each service quality criterion is in the range of [0, 1] with a higher value representing a higher service quality.

# 3. PROBLEM DEFINITION AND METRICS

## 3.1 Problem Definition

We use *weighted sum* [8] allowing a user to express its preferences regarding service quality criteria. Let $\bar{Q}_m$, $\bar{D}_m$ and $\bar{C}_m$ be the scaled *Q*, *D*, and *C* scores for service request *m* (denoted by $O_m$ for short) obtainable after service binding. Let $\omega_{Q,m}$, $\omega_{D,m}$ and $\omega_{C,m}$ be the weights associated with $\bar{Q}_m$, $\bar{D}_m$ and $\bar{C}_m$ issued by the user, with $\omega_{Q,m} + \omega_{D,m} + \omega_{C,m} = 1$. With this simple additive weighting technique, we formulate our MOO problem at the *service-request level* as:

$$\text{Maximize } \text{MOO}_\text{m} = \omega_{Q,m}\bar{Q}_m + \omega_{D,m}\bar{D}_m + \omega_{C,m}\bar{C}_m \quad (2)$$

As there may be multiple SRs issuing service requests and performing service composition and binding concurrently, we formulate our MOO problem at the *system level* as:

$$\text{Maximize } \text{MOO} = \sum_{\text{m}\in\mathcal{T}} (\omega_{Q,m}\bar{Q}_m + \omega_{D,m}\bar{D}_m + \omega_{C,m}\bar{C}_m) \quad (3)$$

where $\mathcal{T}$ is the set of concurrent service requests issued by multiple SRs who are competing for the use of SPs available to them.

## 3.2 Performance Metrics

While the MOO value defined in Equation 3 above can be used to measure MOO performance, *user satisfaction* ultimately determines if a service request is a success or a failure. The *user satisfaction* level of the SR toward SPs selected for executing $O_m$, denoted as $US_m$, can be measured by the ratio of the actual service quality received to the best service quality available among all SPs. We allow a user to specify a minimum *user satisfaction threshold*, denoted as $UST_m$, which specifies the minimum service quality the user can accept. This is to be compared against $US_m$ to decide if the service experience of the user toward SPs selected is positive or negative. If the service experience is negative, culprit SPs are identified and penalized with reputation loss. Conversely, if the service experience is positive, all constituent SPs are rewarded with reputation gain. For notational convenience, let $\overline{SQ}_m^R = \omega_{Q,m}\bar{Q}_m^R + \omega_{D,m}\bar{D}_m^R + \omega_{C,m}\bar{C}_m^R$ denoting the actual service quality received after service binding and execution of $O_m$, $\overline{SQ}_m^{max} = \omega_{Q,m}\bar{Q}_m^{max} + \omega_{D,m}\bar{D}_m^{max} + \omega_{C,m}\bar{C}_m^{max}$ denoting the best service quality that can ever be achieved, and $\overline{SQ}_m^{min} = \omega_{Q,m}\bar{Q}_m^{min} + \omega_{D,m}\bar{D}_m^{min} + \omega_{C,m}\bar{C}_m^{min}$ denoting the minimum service quality that must be obtained in order to satisfy the service request level constraint. Then, with score scaling, $US_m$ can be computed as:

$$US_m = \begin{cases} \dfrac{\overline{SQ}_m^R - \overline{SQ}_m^{min}}{\overline{SQ}_m^{max} - \overline{SQ}_m^{min}} & if\ \overline{SQ}_m^R \geq \overline{SQ}_m^{min} \\ 0 & otherwise \end{cases} \quad (4)$$

Here the second condition in Equation 4 is for the case in which the received service quality is less than the required minimum service quality. Again we note that because of scaling, the large the $\bar{Q}$, $\bar{D}$ and $\bar{C}$ values, the better the service quality. Also note that $\overline{SQ}_m^R = \omega_{Q,m}\bar{Q}_m^R + \omega_{D,m}\bar{D}_m^R + \omega_{C,m}\bar{C}_m^R$, so maximizing $MOO_m$ in Equation 2 is equivalent to maximizing $US_m$ in Equation 4. Therefore, the MOO problem to solve is in effect a user satisfaction maximization problem.

# 4. TRUST MANAGEMENT PROTOCOLS

In this section, we describe four trust protocols to apply to the service composition and binding application with MOO goals for performance comparison, namely, BRS [5], TRM, SRM [10], and CATrust [11].

## 4.1 Single-Trust Baseline Protocol Design

The baseline single-trust protocol (called BRS [5]) is based on Bayesian inference with the trust value modeled as a random variable in the range of [0, 1] following the Beta ($\alpha$, $\beta$) distribution; the numbers of positive and negative service experiences are modeled as binomial random variables. Here $\alpha/(\alpha+\beta)$ is the estimated mean of "direct" trust evidence of an SP, where $\alpha$ is the number of positive experiences and $\beta$ is the number of negative experiences. Positive evidence is observed when the service is satisfactory. More specifically, when $US_m$ exceeds $UST_m$, a positive service experience is counted and $\alpha$ is incremented by 1 for all SPs in $O_m$. On the other hand, when $US_m$ is less than $UST_m$, a negative service experience is counted against all culprits with low performance (i.e., the actual service quality is lower than the advertised service quality) and $\beta$ is increased by 1 for all identified culprits. SPs with expected performance (i.e., the actual service quality is about the same as the advertised service quality) are identified as benign and will not be penalized. After a service request is completed, the SR can propagate its updated trust of the SPs involved in the service request to other nodes in the system as recommendations.

## 4.2 Multi-Trust Protocol Design

Multi-trust refers to the use of multiple dimensions of trust for more accurately describing multiple and often distinct factors contributing to successful service execution. TRM and SRM [10] are based on two trust properties: competence and integrity. Denote node *i*'s trust toward node *j* in trust property *X* (i.e., C for competence and I for integrity) as $T_{i,j}^X$. BRS is used to assess node $i$'s mean *direct trust* toward node *j* in trust property *X* as $\alpha_{i,j}^X/(\alpha_{i,j}^X+\beta_{i,j}^X)$ where $\alpha_{i,j}^X$ is the number of positive and $\beta_{i,j}^X$ is the number of negative experiences in trust property $X$, which are accumulated upon trust update. To update $(\alpha_{i,j}^C, \beta_{i,j}^C)$ for competence trust, node *i* (acting as the SR) compares $UST_m$ with $US_m$. To update $(\alpha_{i,j}^I, \beta_{i,j}^I)$ for integrity trust, node *i* considers it positive evidence if it sees node *j*'s observed *Q*, *D* and *C* scores are close to node *j*'s advertised scaled *Q*, *D*, and *C* scores. Node *i* (as the SR) assesses node *j*'s compliance degree ($CD_{m,j}$) as:

$$CD_{m,j} = min\left(\frac{\bar{Q}_{m,j}^{observed}}{\bar{Q}_{m,j}^{advertised}}, \frac{\bar{D}_{m,j}^{observed}}{\bar{D}_{m,j}^{advertised}}, \frac{\bar{C}_{m,j}^{observed}}{\bar{C}_{m,j}^{advertised}}\right) \quad (5)$$

Here $\bar{Q}_{m,j}^{advertised}$, $\bar{D}_{m,j}^{advertised}$ and $\bar{C}_{m,j}^{advertised}$ are node *j*'s advertised "scaled" *Q*, *D*, and *C* scores, while $\bar{Q}_{m,j}^{observed}$, $\bar{D}_{m,j}^{observed}$ and $\bar{C}_{m,j}^{observed}$ are node *j*'s "scaled" *Q*, *D* and *C* scores actually observed by node $i$. Each user defines its minimum compliance degree threshold for service request $m$, denoted by $CDT_m$. If $CD_{m,j} \geq CDT_m$, then it is counted as a

positive experience for node *j* for integrity and $\alpha_{i,j}^{I}$ is incremented by 1; otherwise, it is counted as a negative experience for node *j* and $\beta_{i,j}^{I}$ is incremented by 1. In the *threshold-based relationship model* (TRM), if integrity trust falls below a threshold, $T_{i,j}^{I,THRES}$, competence trust drops to zero. In the *scaling relationship model* (SRM), competence trust scales up (to 1 maximum) or down (to 0 minimum), depending on whether integrity trust is higher or lower than the threshold. More specifically, TRM computes the overall trust as:

$$T_{i,j} = T_{i,j}^{C} \ if \quad T_{i,j}^{I} \geq T_{i,j}^{I,THRES};\ 0 \ otherwise \tag{6}$$

where $T_{i,j}^{I,THRES}$ is the minimum integrity trust threshold.

SRM computes the overall trust as:

$$T_{i,j} = min\left(1, T_{i,j}^{C} \times \frac{T_{i,j}^{I}}{T_{i,j}^{I,THRES}}\right) \tag{7}$$

## 4.3 Context-Aware Trust Protocol Design

The basic idea of CATrust [11] is for SR *i* to predict whether SP *j* will perform satisfactorily or not for a requested abstract service in a particular context environment, given a history of evidence. SR *i*'s observation $s_{ij}^{t}$ at time $t$ of the service quality received from SP *j* is either "satisfactory" or "unsatisfactory." If the service quality is satisfactory, then the service assessment $s_{ij}^{t}$=1 and SP *j* is considered *trustworthy* in this context environment; otherwise, $s_{ij}^{t}$ =0 and SP *j* is considered *untrustworthy*. For the service composition and binding application in hand, if $US_m \geq UST_m$and $CD_{m,j} \geq CDT_m$, then $s_{ij}^{t}$=1; otherwise, $s_{ij}^{t}$=0. Let the operational and environmental conditions at time $t$ be characterized by a set of distinct context variables$\underline{x}^{t} = [x_0^t, \dots, x_M^t]$. Then, SP *j*'s service trust is the probability that SP *j* is capable of providing satisfactory service given context variables $\underline{x}^{t}$, i.e., $T_j^t \triangleq \Pr(s_j^t = 1)$. Let $\underline{\tilde{S}}_{ij} = \left[\underline{\tilde{S}}_{ij}^{t_0}, \dots, \underline{\tilde{S}}_{ij}^{t_n}\right]$, $i \neq j$, denote the cumulative evidence gathered by SR *i* over $[t_0, \dots, t_n]$, including self-observations and recommendations. Also let $\underline{X} = \left[\underline{x}^{t_0}, \dots, \underline{x}^{t_n}\right]$ denote the corresponding context matrix. CATrust [11] learns the service behavior pattern of SP *j* based on $\underline{\tilde{S}}_{ij}$ and $\underline{X}$, and predicts the probability that SP *j* is trustworthy at time $t_{n+1}$, given $\underline{x}^{t_{n+1}}$ as input. Suppose that node *j* follows service behavior pattern $\underline{\beta}_j$. Then, CATrust estimates $T_{ij}^{t_{n+1}} = \Pr\{\hat{s}_{ij}^{t_{n+1}} = 1 | \underline{x}^{t_{n+1}}, \underline{\hat{\beta}}_{ij}\}$, where $\underline{\hat{\beta}}_{ij}$ is node *i*'s estimate of $\underline{\beta}_j$. Essentially, $T_{ij}^{t_{n+1}}$ obtained above is the service trust of SP *j* at time $t_{n+1}$ from SR *i*'s perspective.

To apply CATrust to solving the service composition and binding application with MOO, SR *i* would apply $T_j^t = \Pr\left\{s_j^t = 1 \middle| \underline{x}^t, \underline{\beta}_j\right\}$ to predict SP *j*'s trust. Since service quality is defined by *Q*, *D*, and *C*, local traffic (which influences *Q* and *D*) and payoff (which influences *C*) are chosen as the context variables. The payoff to SP *j* is determined by SR *i* itself so it is easily measurable by SR *i*. The local traffic can be estimated by SR *i* based on the location information which determines collision probability or the packet retransmission probability after transmitting a sequence of packets for initiating a service request.

# 5. TRUST-BASED SERVICE COMPOSITION ALGORITHMS

In this section, we describe four trust-based algorithms and a non-trust-based algorithm for solving our service composition and binding MOO problem as follows:

- ***Non-trust-based***: While there is no trust in place, each SR keeps a blacklist of SPs with which it has negative interaction experience, i.e.,$CD_{m,j} < CDT_m$. When selecting the best node-to-service assignment, it only considers SPs that are not blacklisted.
- ***Trust-based*** (BRS, TRM, SRM, or CATrust): each SR selects the best node-to-service assignment that maximizes $MOO_m$ in Equation 2 with $\bar{Q}_{m,j}$, $\bar{D}_{m,j}$ and $\bar{C}_{m,j}$ (of node *j* at the bottom level of the SCS) multiplying by $T_{SR,j}$ which is the SR's overall trust toward node *j* obtained from running a trust protocol (BRS, TRM, SRM, or CATrust) as discussed in Section 5. The basic idea of trust-based service composition and binding is that an SP's advertised *Q*, *D* and *C* scores are discounted by the SR's trust towards the SP. When the trust estimate is accurate, it can effectively defend against malicious nodes performing attacks discussed in Section 2.

To circumvent high runtime complexity which renders it infeasible for runtime operations, a heuristic-based solution with linear runtime complexity of $O(|\mathcal{N}|)$ is used for solution efficiency. For all algorithms, an SR simply ranks all eligible SPs for executing an abstract service by $\omega_{Q,m}\bar{Q}_{m,j}T_{SR,j} + \omega_{D,m}\bar{D}_{m,j}T_{SR,j} + \omega_{C,m}\bar{C}_{m,j}T_{SR,j}$ and selects the highest ranked SP as the winner for executing that particular abstract service. It examines all SPs which responded to its query in a single round and performs ranking and service binding for all abstract services needed in the service request. Then, the SR notifies SPs that are selected without coordination with other concurrent SRs. In the *non-trust-based* algorithm, an SP receiving multiple offers randomly selects one SR among all to serve. In the trust-based algorithm, an SP resolves the tie-breaker by selecting the SR for which it has the highest trust to ensure the highest success probability as it will increase its chance of gaining good reputation. The other SRs not selected will be informed of the decision by the SP and will then select other SPs that are still available to provide the particular abstract service. The time to complete the node-to-service selection thus is linear to the number of eligible SPs multiplied by the number of concurrent service requests because each SR will only examine and rank the advertised service quality scores by all eligible SPs once to select a subset of SPs that maximizes its own ranking.

TABLE 1: INPUT PARAMETERS VALUES/RANGES

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $\lvert\mathcal{T}\rvert$ | 80 | $\lvert\mathcal{N}\rvert$ | 60 |
| $\lvert\mathcal{S}\rvert$ | 9 | $p_e$ | 5% |
| $n_{rec}$ | 3 | $\lvert\mathcal{S}_m\rvert$ | 4 |
| $P_{bad}$ | 10-50% | $P_{risk}$ | 0-100% |
| $Q_{bad}$ | [1-3] | $Q_{good}$ | [3-5] |
| $D_{bad}$ | [3-5] | $D_{good}$ | [1-4] |
| $C_{bad}$ | [2-5] | $C_{good}$ | [1-2] |
| $(Q_k^{min}, D_k^{min}, C_k^{min})$ | (1,5,5) | $(Q_m^{min}, D_m^{min}, C_m^{min})$ | (4,20,20) |
| $T_{i,j}^{I,THRES}$ | 0.5 | $CDT_m$ | 0.9 |
| $\omega_{Q,m}$: $\omega_{D,m}$: $\omega_{C,m}$ | 1/3:1/3:1/3 | $UST_m$ | 50-100% |
| ($\alpha$, $\beta$) for BMA | (1, 10) | ($\alpha$, $\beta$) for BSA | (10, 1) |

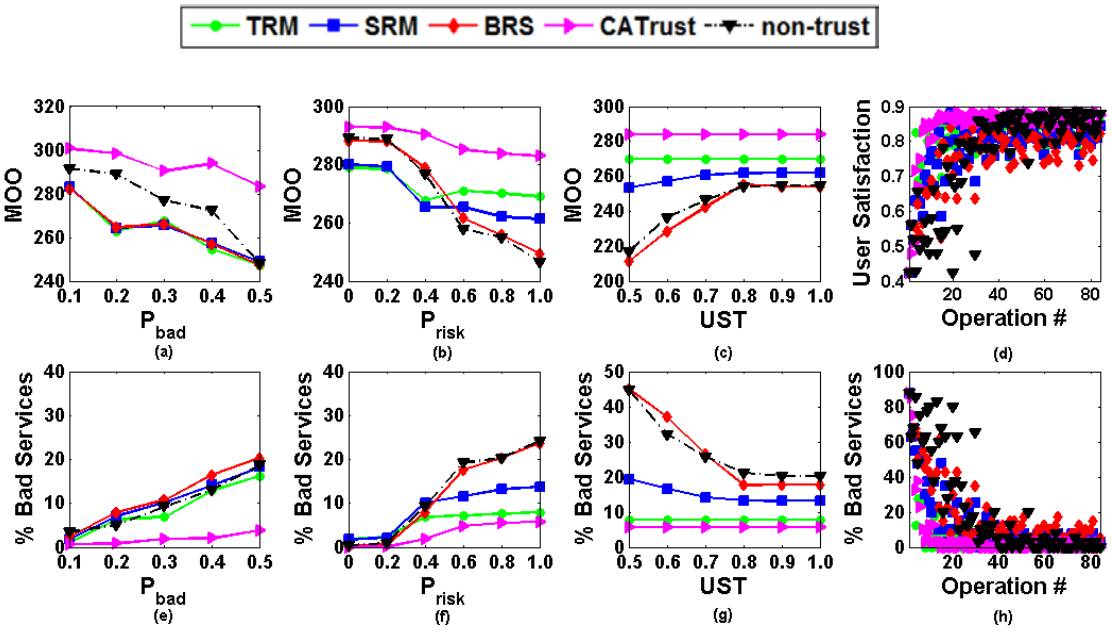


Figure 1: Performance Comparison of Trust-based vs. Non-trust-based Service Composition Algorithms.

# 6. COMPARATIVE ANALYSIS

## 7.1 Experiment Setup

Table 1 lists input parameters and their default values for performance analysis as in [10] for fair comparison. Below we explain each parameter in the table. We consider $|\mathcal{N}|$=60 SPs in the system. There are $|\mathcal{S}|$=9 abstract services, $S_0$ to $S_8$, provided by these SPs. For simplicity, each SP is assumed to be specialized to one abstract service $S_k$ which is randomly assigned initially. All nodes follow the SWIM mobility model [6] with wireless transmission error probability $p_e = 5\%$. We consider a scenario with 80 service requests ($|\mathcal{T}|$=80) divided into 50 chunks, e.g., {1, 2}, {3}, {4, 5}, {6, 7, 8}, {9}, {10}, {11, 12}, {13}, {14, 15}, etc. where a chunk is defined as a set of concurrent service requests overlapping in execution time. For simplicity, each service request has only one service composition specification consisting of $|\mathcal{S}_m| = 4$ abstract services connected in a series structure. The abstract services are randomly selected from $S_0$ to $S_8$ so that the demand to each node is roughly equal in these different sized problems. In case an SR cannot find enough SPs to satisfy the requirement, the service request fails and $US_m = 0$.

We model the hostility of the environment by the percentage of malicious nodes ($P_{bad}$) in the range of [0-50%] with the default value set at 30%. In a SOANET, malicious nodes do not intend to break the system functionality via traditional packet dropping attacks or impairment attacks [3], [7], [8] but aim to increase their chance of being selected for personal gain. When a bad node is chosen as a recommender, it can perform a bad-mouthing attack (BMA) on a good trustee node to ruin the reputation of that good node by providing bad recommendations so as to decrease the chance of this good node being selected to provide services. This is done by providing a very low α and a very high β, e.g., (α, β) = (1, 10), with α representing the number of positive and β the number of negative experiences, to ruin the reputation of that good trustee node. A bad node serving as a recommender can also perform a ballot-stuffing attack (BSA) to boost the reputation of another bad trustee node by providing good recommendations for that bad node, so as to increase the chance of that bad node being selected to provide services. This is done by providing a very high α, and a very low β, e.g., (α, β) = (10, 1), to boost the reputation of that bad trustee node. A malicious node can also perform a self-promotion attack. This self-promotion behavior is modeled by a risk parameter $P_{risk}$ in the range of [0-100%] with the default set at 50%. A malicious node performs a self-promotion attack by boosting its advertised *Q*, *D* and *C* scores obtained by the product of its true *Q*, *D* and *C* scores and $(1 + P_{risk})$, $(1 - P_{risk})$, and $(1 - P_{risk})$, respectively. $Q_{bad}$, $D_{bad}$ and $C_{bad}$ give the *true Q*, *D* and *C* scores for bad nodes, which can be boosted during service advertisement. $Q_{good}$, $D_{good}$ and $C_{good}$ give the *true Q*, *D* and *C* scores for good nodes, which will be reported by good nodes faithfully during service advertisement. The default values are set for the case in which the service quality of bad nodes is inferior to that of good nodes to reveal interesting design tradeoffs. These *Q*, *D*, and *C* values are context dependent (location and payoff), generated at the beginning, and are not changed during system operation. We set $(Q_k^{min}, D_k^{min}, C_k^{min})$ = (1, 5, 5) at the abstract service level and $(Q_m^{min}, D_m^{min}, C_m^{min})$= (4, 20, 20) at the service request level for the minimum service quality requirement.

The initial trust values (for integrity and competence in the case of multi-trust) are set to 0.5 for all nodes meaning ignorance (no knowledge). The integrity trust threshold $T_{i,j}^{I,THRES}$ for TRM and SRM is set to 0.5 to punish a node with integrity trust less than ignorance trust as time progresses. The protocol compliance degree threshold $CDT_m$ is set to 0.9 to accommodate a 10% maximum detection error for assessing protocol compliance behavior. For simplicity we set $\omega_{Q,m} = \omega_{D,m} = \omega_{C,m} = 1/3$ in Equation 2. The number of recommenders $n_{rec}$ is set to 3 so node *i* will only allow 3 nodes with the highest $T_{i,j}$ values as the recommenders during trust update.

## 7.2 Comparative Performance Analysis

In this section, we perform a comparative performance analysis of CATrust vs. single-trust and multi-trust protocols as the

underlying trust protocol for the trust-based algorithm for solving the service composition and binding MOO problem via MATLAB simulation. Specifically, we simulate non-trust-based and trust-based algorithms under the same environment setting defined in Table 1.

Figure 1 shows the results. Figures 1(a)-(d) compare the MOO and user satisfaction performance. Figures 1(e)-(h) compare the percentage of bad service selected for service execution in this service composition application with MOO requirements.

Figures 1(a)-(b) examine the negative impact of increasing $P_{bad}$ and $P_{risk}$ on MOO performance in Equation 3. Compared with the non-trust-based algorithm, trust-based algorithms show high resilience against increased attack intensity with more malicious entities ($P_{bad}$)or higher self-promotion attack behavior ($P_{risk}$). Figure 1(c) examines the impact of $UST_m$ on MOO performance. Note that $UST_m$ is the service quality threshold compared with $US_m$ to determine if a service experience is positive or negative. We observe that as $UST_m$ increases, MOO performance increases (and levels off). The reason is that a high $UST_m$ has the effect of penalizing bad nodes with low trust and can effectively remove bad nodes from participating in future service requests.

Figure 1(d) compares $US_m$ calculated from Equation 4 as more service requests (labeled as "Operation #" on the x coordinate) are executed over time for the three trust-based algorithms against the non-trust-based algorithm. We consider a combination of $P_{risk} = 70\%$ and $P_{bad} = 30\%$ to reveal interesting trends. Because of high $P_{risk}$, even trust-based algorithms are fooled into selecting bad nodes in the first few service requests. So the first few service requests do not pass $UST_m$. As a result, bad nodes selected to provide services in the first few service requests are penalized with trust decrease and filtered out from later service requests. This is evidenced by the fact that the later service requests have high $US_m$values. In particular, for CATrust (pink dots), $US_m$ is near 90% after 15 service requests. We observe that $US_m$ under CATrust is consistently higher than other trust-based counterparts. On the contrary, the non-trust-based algorithm (black dots) achieves a high $US_m$ only after 50 service requests are processed because it has no effective way of filtering out bad services until in the last few operations at which point it has blacklisted sufficient nodes with bad services. This trend supports our claim that trust-based algorithms can effectively achieve high user satisfaction despite the presence of bad nodes performing self-promotion attacks, especially after trust convergence occurs.

Figures 1(e)-(h) compare the percentage of bad service selected in this service composition and binding application with MOO requirements. One can clearly see that CATrust outperforms the single-trust (BRS) and multi-trust (TRM, SRM) counterparts by a wide margin. In particular, the % of bad service selected under CATrust (pink dots) is significantly lower than that under BRS (red dots), TRM (green dots), or SRM (blue dots). We attribute the superiority of CATrust to its ability to associate context to solution quality delivered by an SP. Because a service request issued by an SR is context dependent (location and payoff), CATrust is able to learn an SP's context-dependent service behavior and its delivered service quality, given location and payoff as input. On the contrary, BRS, TRM, and SRM do not take context into consideration and merely predict an SP's "average" delivered service quality across the location and payoff context space.

# 7. CONCLUSION

Trust is known as an effective mechanism in services computing. However how to use it most effectively is less known. In this paper we conducted a comparative performance analysis of single-trust-based (BRS), multi-trust-based (TRM, SRM), and context-aware trust-based (CATrust) algorithms for a service composition and binding SOANET application with the goal of multi-objective optimization to answer this question. Our results demonstrated that context-aware trust-based service composition and binding is the winner for maximizing application performance. We attribute CATrust's superiority to its ability to more accurately learn an SP's context-dependent service behavior and the delivered service quality during service binding, while BRS, TRM, and SRM do not take context into consideration and merely predict an SP's average delivered service quality across the context space. In the future, we plan to generalize the performance comparison analysis to more SOANET applications and validate the comparison results with service quality and mobility traces [4], [12].

# 8. REFERENCES


[1] Chen, I.R., Guo, J., and Bao, F. 2016. Trust Management for SOA-based IoT and Its Application to Service Composition. *IEEE Transactions on Services Computing*. 9, 3 (2016), 482-495.

[2] Chen, I.R., Bao, F., and Guo, J. 2016. Trust-based Service Management for Social Internet of Things Systems. *IEEE Transactions on Dependable and Secure Computing*, 13, 6 (2016), 684-696.

[3] Chen, I. R. and Bastani, F.B. 1991. Effect of Artificial-Intelligence Planning-Procedures on System Reliability. *IEEE Transactions on Reliability*, 40, 3 (1991), 364-369.

[4] Chen, I.R. and Verma, N. 2003. Simulation study of a class of autonomous host-centric mobility prediction algorithms for wireless cellular and ad hoc networks. *36th Annual Symposium on Simulation* (2003), 65-72.

[5] Jøsang, A. and Ismail, R. 2002. The Beta Reputation System. *15th Bled Electronic Commerce Conference* (2002), 1-14.

[6] Kosta, S., Mei, A., and Stefa, J. 2014. Large-Scale Synthetic Social Mobile Networks with SWIM. *IEEE Transactions on Mobile Computing*, 13, 1 (2014), 116-129.

[7] Mitchell, R. and Chen, I.R. 2014. Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles using Behavior Rule Specifications. *IEEE Transactions on Systems, Man and Cybernetics*, 44, 5 (2014), 593-604.

[8] Mitchell, R. and Chen, I.R. 2015. Behavior Rule Specification-based Intrusion Detection for Safety Critical Medical Cyber Physical Systems. *IEEE Transactions on Dependable and Secure Computing*. 12, 1 (2015), 16-30.

[9] Wahab, O.A., Bentahar, J., Otrok, H., and Mourad, A. 2015. A Survey on Trust and Reputation Models for Web Services: Single, Composite, and Communities. *Decision Support Systems*. 74 (2015), 121-134.

[10] Wang, Y. et al. 2017. Trust-based Service Composition and Binding with Multiple Objective Optimization in Service-Oriented Mobile Ad Hoc Networks. *IEEE Transactions on Services Computing.* (2017), in press.

[11] Wang, Y. et al. 2017. CATrust: Context-Aware Trust Management for Service-Oriented Ad Hoc Networks. *IEEE Transactions on Services Computing.* (2017), in press.

[12] Zheng, Z., Zhang, Y., and Lyu, M. R. 2014. Investigating QoS of Real-World Web Services. *IEEE Transactions on Services Computing*, 7, 1 (2014), 32-39.