Trustworthy Stigmergic Service Composition and Adaptation in Decentralized Environments

Ahmed Mostafa, Minjie Zhang, Senior Member, IEEE, and Quan Bai

1 INTRODUCTION

The emerging area of Service-Oriented Computing (SOC) has drawn a great deal of attention in recent years, prompting researchers to devise a variety of technologies to facilitate loosely coupled and flexible SOC architectures. Web services are autonomous application modules that can be described, published and located in SOC environments. To build a complex application, one may have to link a set of service providers across their heterogeneous environments [29]. Service composition fulfills users' requirements by combining services in different repositories. The users’ requirements can be functional or non-functional. The functional requirements are in relation to the overall outcome concerning the functionalities of the business process, while the non-functional requirements pertain to the quality of the composition as a whole, e.g., availability, reliability, cost and response time. As the number of web services, providing similar functionalities, grows large, the problem of selecting web services with the best guaranteed Quality of Service (QoS) becomes critical.

Web service adaptation is responsible for adjusting web services to changing user requirements, QoS degradations and other unpredictable changes in run-time environments. Any successful service composi-

- A. Mostafa and M. Zhang are with the Department of Computer Science and Software Engineering, University of Wollongong, Wollongong 2500, Australia. E-mail: aase995, minjie@uowmail.edu.au
- Q. Bai is with Auckland University of Technology, Auckland 1142, New Zealand. E-mail: quan.bai@aaut.ac.nz
Such pheromone can encode and describe application-specific information to be used to achieve specific tasks. From a general perspective, stigmergic interactions have two major advantages: (1) stigmergic interactions are mediated by pheromone and are completely decoupled, which makes it suitable for open and dynamic environments and (2) stigmergic interactions naturally support application-specific context awareness, in that pheromone provides agents with an application-specific representation of their operational environment.

A major challenge in the SOC domain is that $QoS$ alone cannot represent an accurate measure for Web service reliability. Some Web service providers may lie and exaggerate the $QoS$ values of their published services. This malicious action cannot be discovered by the mere values of $QoS$. Therefore, employing trust measures becomes indispensable to any successful Web service application [24]. Moreover, the need to calculate the level of trust in composite services, which have complex invocation structures is another rising challenge.

To address these challenges, an approach is proposed in this paper. The novelties of this approach are manifold:

- First, a stigmergic-based modeling approach is proposed to model Web service compositions and adaptations. The proposed approach adopts a decentralized architecture, and allows Web services to compose and adapt in open decentralized environments.
- Second, to consolidate service composition and adaptation, trust concepts are adopted in the proposed approach as a measure to filter functionally equivalent Web services and to select the most trustworthy concrete services, resulting in a robust composite service.
- Third, to overcome the complexity of the scale-free environments, hybridization with local search operators is employed to enable service adaptation in large-scale service environments.
- Fourth, to address continually changing environments, diversity schemes are incorporated to promote continuous adaptation features within the proposed approach.

The rest of this paper is organized as follows. A motivating scenario is introduced in Section 2. Stigmergic modeling for Web service composition and adaptation is introduced in Section 3. An algorithm for trustworthy stigmergic service composition and adaptation along with a self-coordination mechanism are proposed in Section 4. Section 5 presents an algorithm for trustworthy stigmergic service adaptation with the application of local search operators and diversity schemes. In Section 6, experimental results are presented for evaluating the proposed algorithms. Section 7 gives a brief review of related work and discussion. Finally, the paper is concluded in Section 8.

2 Motivating Scenario

This section presents an application scenario from the big data analysis field to motivate the challenge of service composition and adaptation in decentralized environments. Let us consider Company A that has terabytes of transactional sales records and needs to analyse these records to evaluate its customers’ loyalty. It becomes difficult for Company A to process this large amount of data using in-house processing tools. Therefore, the cloud computing platform would represent a cost-effective option to outsource the data analysis job. Company A then decides to submit the transactional data to the analytics cloud service, e.g., Service S. Service S, in turn, generates a composite data analysis job on behalf of Company A. This data analysis job, in particular, will have the following tasks/abstract services, i.e., Software as a Service (SaaS), which are, the data cleaning service (Task 1), the data transformation service (Task 2), the pattern analysis service (Task 3), the pattern evaluation and representation service (Task 4). Besides these software services, the data analysis job also needs CPU, network and storage resources from Infrastructure as a Service (IaaS) providers, as shown in Fig. 1.

![Motivating Scenario](image)

Fig. 1. Motivating Scenario

Service S does not know where these component services are located and who are providing them. Then, Service S has to search the cloud service network to find a set of services to satisfy Company A requirements. In addition, since there are thousands of services on the cloud that can provide each task individually, Service S has to select the highest quality services according to some criterion. Moreover, Service S has to adapt the running workflow to cope with dynamic scenarios where cloud services can join or leave anytime.

3 Stigmergic Modeling for Service Composition and Adaptation

In this section, an approach (model) called ‘stigmergic service composition and adaptation’ is proposed to address the problem of Web service composition and adaptation in decentralized environments. The proposed approach is inspired by the concept of ‘stigmergy’ [22], which is a pheromone-based mechanism for coordinating and controlling swarming objects.
Examples from natural systems show that stigmergic systems can generate robust, complex, intelligent behaviors at the system level even when the individual agents only possess very limited or even no intelligence. In these systems, intelligence resides not in a single agent (as in centralized control), but in the whole group of agents distributed in the environment [9].

In our approach, each Web service is modeled as a service agent, and a Web service system can then be considered as a multi-agent system with a number of interactive service agents. These agents achieve collaboration and self-organization via exchanging pheromone and performing several pheromone operations. A service is composed if a group of service agents can form an organization in order to collaborate.

Definition 1: Service Agent. Service agent $sa$ is defined as a tuple $sa = \langle id, F \rangle$, where id is the identifier of the service agent, $F$ is the pheromone store for facilitating composition and decentralized self-coordination (as detailed in Definition 2).

Definition 2: Pheromone Store. Pheromone store $F$ is a physical place to store a set of pheromone flavors, i.e., $F = \{f_1, f_2, ..., f_n\}$, where $f_j$ is the $j$th pheromone flavor and each pheromone flavor holds a scalar value, which represents the trail of a certain service agent.

The proposed approach works as follows. Each Web service can be a requester service agent, a provider service agent, or both. Requester service agent $sa$ requests the composition of an abstract workflow $W$ such that $W = \{s_1, s_2, ..., s_n\}$, where $s_i$ represents a concrete service or a resource that needs to be composed for satisfying a single unit of functionality or a subtask, $s_1$ is the first service component of the workflow and $s_n$ is the final service component of the workflow, respectively. To this end, the requester service agent sends a set of walker agents to traverse a network of directly linked service agents looking for matching services and resources to construct the required workflow $W$.

Definition 3: Walker Agent. Walker agent $wa$ is a clone agent instantiated by the service agent for the purpose of traversing a network of service agents. The number of walker agents is task dependant.

To this end, each walker agent begins from start service $s_1$ following a path till reaching end service $s_n$. Once constructing the required workflow $W$, the walker agent deposits/withdraws a certain amount (i.e., $Q_{f_j}$) of pheromone flavor $f_j$ into/from the digital pheromone stores of all provider service agents, which took part in this composition round. The decision of depositing or withdrawing depends on the quality of the service provided. Based upon the strength of the trail of this pheromone flavor $f_j$, in future rounds, similar walker agents will be guided towards the highest quality workflow quickly without the need to traverse the entire network again.

As the walker agents communicate through the environment via the pheromone trail, these agents need not to establish a direct communication channel, and so keep the communication overhead to the minimum. On the other hand, setting the number of walker agents will affect the convergence speed. Theoretically, the bigger the number of the walker agents, the faster those walker agents can converge to a good solution. Technically, too many walker agents may add extra computational cost. The solution to this dilemma is fairly task dependent. In the proposed model, the number of walker agents is set equal to the number of abstract services included in the composition task. Another possible solution to this dilemma would be using reinforcement learning techniques to learn the adequate number of walker agents per task [18]; this point is set to future research.

4 TRUSTWORTHY STIGMERGIC SERVICE COMPOSITION

Based on the aforementioned model, in the following subsections, we propose a composition algorithm and a self-coordination mechanism to facilitate stigmergic service composition in decentralized environments. Besides, the most common symbols used throughout the rest of the proposed approach have been summarized into Table 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$sa$</td>
<td>Service agent</td>
</tr>
<tr>
<td>$F$</td>
<td>Pheromone store</td>
</tr>
<tr>
<td>$wa$</td>
<td>Walker agent</td>
</tr>
<tr>
<td>$W$</td>
<td>Composition task</td>
</tr>
<tr>
<td>$T(sa_j)$</td>
<td>Concrete service trust</td>
</tr>
<tr>
<td>$TD(x,y)$</td>
<td>Data-dependency trust between two services $x$ and $y$</td>
</tr>
<tr>
<td>$Q_{f_j}$</td>
<td>A quantity of a certain Pheromone flavor $f_j$</td>
</tr>
<tr>
<td>$E_{f_j}$</td>
<td>The evaporation factor of pheromone flavor $f_j$ over time</td>
</tr>
<tr>
<td>$Q_{f_j}$</td>
<td>The propagation of pheromone flavor $f_j$ from service agent into another</td>
</tr>
<tr>
<td>$M$</td>
<td>Memory/Population list to store the iteration best walker agent</td>
</tr>
<tr>
<td>$LS$</td>
<td>Local Search operators</td>
</tr>
<tr>
<td>$BI$</td>
<td>Blind Inversion operator</td>
</tr>
<tr>
<td>$GI$</td>
<td>Guided Inversion operator</td>
</tr>
<tr>
<td>$AIO$</td>
<td>Adaptive Inver-Over mechanism combining both BI and GI</td>
</tr>
</tbody>
</table>

4.1 Service Composition

The proposed stigmergic service composition algorithm, i.e., Algorithm 1, works in two steps, the edge selection and the pheromone update, which are described below.
Algorithm 1 Stigmergic Service Composition

1: Set parameters $\alpha$, $\beta$ and $\mu$
2: Set pheromone trails $GT_{init}$
3: while the termination condition isn’t satisfied do
4:     for all $t = 1 : \mu$ do
5:         repeat
6:             Start from requester service agent $sa_i$
7:             Select service agent $sa_j$ to visit next (not visited before) according to an update rule
8:         until the walker agent has composed a workflow
9:     end for
10: end while
11: Evaporate pheromone
12: Update pheromone s.t. walker agent $wa_t$
13: end for
14: end for

4.1.1 Edge Selection

This step works as follows. For each provider service agent, the walker agent tries to extend its current state seeking a more complete intermediate solution via selecting from a group of provider service agents (Lines 3-9 in Algorithm 1). The walker agent selects amongst provider service agents based upon two decision policies, i.e., visibility and trail. Visibility represents the desirability of the extension based upon \textit{a priori} heuristic information. Trail represents the performance of this selection, i.e., how good this selection was in the past. So, a trail is deemed \textit{a posteriori} judgment made by other walker agents who made this selection before.

The agent moves to the next state with probability

$$P_{i,j} = \frac{T(sa_{i,j})^\alpha GT(sa_{i,j})^\beta}{\sum T(sa_{i,j})^\alpha GT(sa_{i,j})^\beta},$$

where $T(sa_{i,j})$ is the visibility of the transition from one service agent to another service agent, determined by the walker agent, $\alpha \geq 0$ is the parameter to control the influence of $T(sa_{i,j})$, $GT(sa_{i,j})$ is the amount of pheromone deposited for transition from one state to another state and $\beta \geq 0$ is the parameter to control the influence of $GT(sa_{i,j})$.

Traditional service selection approaches used to employ QoS as a measurement of visibility for functionally equivalent Web services [26], [28]. An inherent drawback of these approaches is that fraudulent service providers may use attractive advertisements to deceive service consumers, who become victims. Hence, the trust evaluation of services is highly desirable and critical in SOC environments.

Definition 4: Service Trust. Service trust is the probability by which one service expects another service to perform a specific action.

Although trust based service selection has been proposed in the literature [16], a new idea is presented in this paper for selecting services based on the balance between the trust ranks of the concrete services and the trust ranks of the whole workflow. Towards this end, the trust ranks of the concrete services are calculated using the model proposed by [13]. Then, the data dependency patterns and the conditional probability are used to calculate the trust ranks of the whole workflow. As the concrete web services could be part of different workflows simultaneously, this trust balance will differ in its turn.

In the edge selection step, the concrete trust of partner services is interpreted as the visibility parameter and used as the criterion to assess individual services for edge selection.

Definition 5: Concrete Service Trust. Concrete service trust $T$ represents the trust value of a single Web service (service agent) and is used as a visibility parameter to guide the edge selection.

The concrete service trust $T$ can be calculated according to the following equation [13],

$$T(sa_i) = \frac{\sum_{c=1}^{l} eval^{sa_i} \cdot f(d) \cdot c_r(c)}{\sum_{c=1}^{l} c_r(c)},$$

where $L$ is the set of service consumers that have invoked $sa_i$, $eval^{sa_i} \cdot f(d)$ is the personal evaluation value given by the consumer $c$, $c_r(c)$ is the creditability of $c$ and $f(d)$ is a function that makes the evaluation fades with time.

4.1.2 Pheromone Update

In this step, when all walker agents have completed their solutions, the pheromone trails are updated (Lines 10-13 in Algorithm 1) by:

$$GT(sa_{i,j}) = (1-E_{f_j})GT(sa_{i,j}) + \sum \Delta GT(sa_{i,j})^{n},$$

where $GT(sa_{i,j})$ is the amount of pheromone deposited for transition from one service agent to another, $E_{f_j}$ is the evaporation factor such that $1 > E_{f_j} > 0$ and $\Delta GT(sa_{i,j})^{n} = GT(W)^n$ is the amount of pheromone deposited by the $n^{th}$ walker agent.

The trail parameter $GT(W)$ is interpreted as the group trust of the whole workflow constructed by this walker agent and employed to guide future composition rounds.

Definition 6: Group Trust. Group trust $GT$ represents the trust value for the whole invocation structure, i.e., the workflow in this case, and is used as a pheromone trail value instead of a QoS value.

Group trust $GT$ is derived based upon the data dependency pattern and conditional probability as follows:

$$GT(W) = \frac{\sum_{i=1}^{M} T(sa_i) + \sum_{j=1}^{N} TD_j}{M + N},$$

where $T(S_i)$ is the trust value of a single service on this workflow, $M$ is the number of partner services in this workflow, $N$ is the occurrence number of data-dependencies between partner services, and $TD$ is the
value of data-dependency trust between two services $x$ and $y$. This value can be calculated by the following equation:

$$TD(x:y) = \prod_{i=1}^{M} sa_i,$$

where $sa_i$ is a service on this workflow from $x$ to $y$ such that $sa_1 = x$ and $sa_M = y$.

### 4.2 Self-Coordination

In order to facilitate decentralized composition, a network of service agents self-organizes by exchanging trails of digital pheromone. During the course of self-organization, this digital pheromone is subject to three primary operations. These operations represent the underpinnings for the self-organization mechanism. In the following subsections, pheromone operations are defined first. Then, coordination and self-organization mechanisms are demonstrated later.

#### 4.2.1 Pheromone Operations

In this approach, we define three primary pheromone operations for coordinating service agents, and enabling those service agents to achieve self-organizations.

**Pheromone Aggregation:** A pheromone aggregation is the operation/process wherein digital pheromone quantity $Q_f$ can be deposited or withdrawn from any service agent $sa_i$. Deposits of a certain flavor $f_j$ are added to the current amount of that flavor of pheromone $s(Q_f)$ located at service agent $sa_i$ at time $t$. By its very nature, the process of pheromone aggregation is related to the time when this process had taken place.

**Pheromone Evaporation:** A pheromone evaporation is the operation wherein a pheromone flavor $f_j$ evaporates over time $t$. This operation will generate an evaporation factor $E_f (0 < E_f \leq 1)$ to weaken obsolete information. $E_f$ can be obtained via Equation 6:

$$E_f = e^{-\frac{\Delta t(i)}{\lambda}},$$

where $\Delta t(i)$ is the time difference between the current time and when $f_j$ has been left, and $\lambda$ is the parameter to control the evaporation speed. The authors have chosen to calculate the evaporation factor using an exponential equation rather than others, e.g., a linear equation, to allow the evaporation speed to be faster. The reason behind this is that we want the evaporation speed to be faster at the beginning, but does not equal to 0 when time passes on. Slow evaporating speeds will allow significant amounts of pheromone to accumulate. In addition, fast evaporating speeds will enable the proposed model to adapt quickly to highly dynamic environments.

**Pheromone Propagation:** A pheromone propagation $G_f$ is the operation wherein pheromone flavor $f_j$ is propagated from service agent $sa_i$ to service agent $sa_k$ based upon neighborhood relation $N$, where $sa_k \in N(sa_i)$. The act of propagation causes pheromone gradients $g(Q_f, sa_i, t)$ to be formed.

#### 4.2.2 Exchange Strategies

Using the three operations introduced in the above subsection, a stigmergic self-coordination algorithm (i.e., Algorithm 2) is designed to expedite agents to learn the best composition path, and adapt to the changing environment.

**Algorithm 2 Stigmergic Self-Coordination**

1: for each $p_i \in P$ do
2: \hspace{1em} while $s(Q_f, sa_i, t) \geq thr$ do
3: \hspace{2em} for $t = 1: n$ do
4: \hspace{3em} $g(Q_f, sa_i, t) = \sum_{sfa_k \in N(sa_i)} \frac{G_f}{N(sfa_k)}(s(Q_f, sfa_k, t - 1) + d(Q_f, sfa_k, t))$
5: \hspace{3em} if $g(Q_f, sa_i, t) < \alpha$ then
6: \hspace{4em} $g(Q_f, sa_i, t) = g(Q_f, sa_i, t)$;
7: \hspace{4em} else
8: \hspace{5em} $g(Q_f, sa_i, t) = \alpha$;
9: \hspace{3em} end if
10: $s_{in} = g(Q_f, sa_i, t)$;
11: $s_{out} = (G_f) * (s(Q_f, sa_i, t - 1) + d(Q_f, sa_i, t))$
12: \hspace{1em} while $s_{out} < s_{in}$ do
13: \hspace{2em} $s(Q_f, sa_i, t) = E_f * [(1 - G_f) * (s(Q_f, sa_i, t - 1) + d(Q_f, sa_i, t)) + g(Q_f, sa_i, t)]$
14: \hspace{2em} end while
15: \hspace{1em} end while
16: \hspace{1em} end while
17: \hspace{1em} end for
where \( E_j \) is the evaporation factor of pheromone flavor \( f_j \) (refer to Equation 6), \( (1 - G_j) \) calculates the remaining amount after propagation to neighboring service agents, \( s(Q_{f_j}, sa_i, t - 1) \) represents the amount of pheromone flavor \( f_j \) from the previous cycle, \( d(Q_{f_j}, sa_i, t) \) represents the total deposits made since the last update cycle (including pump auto-deposits) and \( g(Q_{f_j}, sa_i, t) \) represents the total pheromone flavor \( f_j \) propagated in from all the neighbors of \( sa_i \). Every service agent \( sa_i \) applies this equation to every pheromone flavor \( f_j \) once during every update cycle. The propagation result received from the neighboring service agents is calculated by Equation 8 (Line 13 in Algorithm 2):

\[
g(Q_{f_j}, sa_i, t) = \sum_{sa_k \in N(sa_i)} \frac{G_{f_j}}{N(sa_k)} (s(Q_{f_j}, sa_k, t - 1) + d(Q_{f_j}, sa_k, t)),
\]

(8)

In Equation 8, it can be found that each neighboring service agent \( sa_k \) (\( sa_k \in N(sa_i) \)) propagates a portion of its pheromone to \( sa_i \) in each update cycle \( t \). This portion depends on the parameter \( G_{f_j} \) and the total number of \( sa_k \)'s neighbors \( N(sa_k) \).

Using Equations 7 and 8, we can demonstrate several critical rules including local stability, propagated stability and global stability. These rules are considered in Algorithm 2 design to manage the process of exchanging pheromone through a network of service agents. These rules are used to prevent pheromone depletion, pheromone overloading and to keep the elitist members in the service agent network.

**Rule 1: Local Stability.** A local stability \( S_l \) is the rule in which the strength of the pheromone output, \( sa_i^{out} \), propagated from any set of service agents \( M \subset sa \) to their neighbors \( N \subset sa \) at \( t + 1 \), is strictly less than the strength of the aggregate pheromone input \( sa_i^{in} \) (external plus propagated) to those service agents at \( t \) (Lines 10-12 in Algorithm 2).

**Rule 2: Propagated Stability.** A propagated stability \( S_p \) is the rule in which there exists a fixed upper limit \( \alpha \) to the aggregate sum of all propagated pheromone inputs at an arbitrary service agent \( sa_i \) if one-update cycle \( t \) and one-service agent \( sa_i \)'s external input are considered (Lines 5-9 in Algorithm 2).

**Rule 3: Global Stability.** A global stability \( S_g \) is the rule in which the pheromone strength \( s(Q_{f_j}, sa_i, t) \) at any service agent is bounded. If the pheromone strength \( s(Q_{f_j}, sa_i, t) \) drops below a predefined threshold \( thr \), it automatically disappears from the pheromone store, i.e., \( s(Q_{f_j}, sa_i, t) \geq thr \) (Line 2 in Algorithm 2).

### 5 Trustworthy Stigmergic Service Adaptation

By always choosing the trail with the strongest pheromone flavor, the stigmergic service composition algorithm can naturally select the best services in decentralized environments. However in open dynamic environments, two additional challenges arise: (1) Web services can join and leave the environment at any time, and hence the pheromone trails deposited at iteration \( t \), may not make sense at iteration \( t + 1 \), and (2) the rise and fall of the Web services trust ranks will misguide the pheromone trails.

To tackle these limitations, the basic stigmergic service composition algorithm is amended by attaching a memory (population list), which stores the iteration best walker agent’s solution/workflow. Once an environment change is encountered, these stored solutions are used to restore the pheromone trails. Also, in case of dynamic arrival or departure of services, this memory is repaired heuristically using two strategies: (1) Web services that are no longer part of the stored solutions are immediately omitted and (2) new rising Web services with promising trust values are placed in a way that maximizes service trust.

#### 5.1 Population List

The memory-based trustworthy service adaptation algorithm is represented by Algorithm 3.

```
Algorithm 3 Memory-based Trustworthy Service Adaptation
1: Set parameters \( \alpha, \beta, \mu \) and \( K \)
2: Set pheromone trails \( GT_{init} \)
3: while the termination condition is not satisfied do
4:  for all \( t = 1 : \mu \) do
5:    repeat
6:      \( sa_i \) = start from requestor service agent
7:      \( sa_j \) = move to next service agent (not visited before) according to an updating rule
8:    until the walker agent has composed a workflow
9:  end for
10:  best = find the best walker agent
11:  if \( M \) is full then
12:    remove the first best walker agent
13:  remove pheromone s.t. Eq. 9
14:  end if
15:  insert the new best walker agent into \( M \)
16:  update pheromone s.t. Eq. 3
17: end while
```

Algorithm 3 works as follows. First, memory \( M \) is set by default size \( K \) (Line 1 in Algorithm 3). Second, the proposed memory-based algorithm proceeds through two phases, the edge selection and the pheromone update. The edge selection phase of the memory based algorithm works the same way as defined in Equation 1 (Lines 6-7 in Algorithm 3). The pheromone update phase works as follows. On the first \( K \) iterations, the best walker agent is stored in memory \( M \) and is used...
to update the pheromone trail of the workflow, which the walker agent has composed using Equation 3.

On iteration $K+1$, the new best walker agent enters the memory and updates its pheromone trail positively. Simultaneously, to make free room available for that walker agent, the first walker agent, which has entered the list, is omitted and its pheromone trail is removed (Lines 10-16 in Algorithm 3) according to the following equation:

$$GT(sa_{i,j}) = (1 - E_{f_j})GT(sa_{i,j}) - \sum \Delta GT(sa_{i,j})^n,$$  \hspace{1cm} (9)

where $GT(sa_{i,j})$, $E_{f_j}$ and $\Delta GT(sa_{i,j})^n$ are defined as in Equation 3.

5.2 Local Search Operators

Using pheromone trails, the proposed stigmergic service composition algorithm is able to traverse the search space looking for high quality solutions/workflows, however, in large-scale service environments it may need a relatively long time to locate good solutions.

Incorporating Local Search (LS) operators can help to accelerate the search process and guide the algorithm to more promising areas in the search space. This usage of local search operators is called memetic algorithms [17].

In this subsection, a memetic trustworthy service adaptation algorithm is proposed to improve the results of the standard stigmergic service composition algorithm. The proposed algorithm works as follows. After each iteration, the best walker agent is selected to be improved by a local search operator. If a better workflow is found, it replaces that walker agent’s workflow in the memory and the pheromone trails are updated.

The inver-over operator is one of the leading local search operators that has proved efficient in a broad domain of problems [17]. Compared to other local search operators [17], the inver-over operator has the following advantages: (1) it only exploits one operator, a combination between crossover and mutation; and (2) every individual only competes with its own offspring. The inver-over operator is based on the inversion operator where two randomly selected points from a solution segment are reversed. It combines two other operators and a probability threshold is set for selecting which one to apply. The first operator is the Blind Inversion (BI), where the second point, which determines the segment to be reversed, is selected randomly from the same solution of the first point. It can be seen as a mutation operator. The second operator is the Guided Inversion (GI), where the second point is determined from another solution randomly picked from the population list. Also, guided inversion can be seen as a crossover operator.

An example is set to demonstrate the operation of the inver-over operator. Suppose that the current solution/workflow, composed by the iteration best walker agent, is $best' = \{2, 3, 9, 4, 1, 5, 8, 7, 6\}$ and the first randomly chosen point/service agent $sa = 3$. If $rand() \leq p$, then the second service agent will also be chosen randomly from the same workflow, i.e., $best'$. Suppose that $sa' = 8$, in the next step, the section between $sa$ and $sa'$ will be inverted and the result workflow is $best = \{2, 3, 8, 5, 1, 4, 9, 7, 6\}$. This step is called the inverse step. On the other hand, if $rand() > p$, another workflow will be chosen randomly from the memory list $M$. Suppose that $best'' = \{1, 6, 4, 3, 5, 7, 9, 2, 8\}$, the second service agent $sa'$ is chosen from this workflow, i.e., the adjacent to the first service agent. In this case, $sa' = 5$. The new section between $sa$ and $sa'$ will be inverted in $best'$, i.e., from 3 to 5. As a result, the new workflow $best\prime = \{2, 3, 5, 1, 4, 9, 8, 6, 7\}$ is produced in which the section from 3 to 5 comes from the second workflow $best''$.

The inver-over operator-based Web service local search algorithm is presented in Algorithm 4.

**Algorithm 4 Inver over operator**

1: Set parameters $best$ and $p$
2: $best' = best$
3: Choose a service $sa$ from workflow $best'$
4: if $rand() \leq p$ then
5: Choose a random service agent $sa'$ from other service agents in $best'$
6: else
7: Select randomly another workflow $best''$ from the population
8: Assign to $sa'$ the next service agent to service agent $sa \in best''$
9: end if
10: Inverse the workflow from service agent $sa$ to service agent $sa'$ in $best'$
11: if $best'$ is better than $best$ then
12: $best = best'$
13: end if

In Algorithm 4, parameter $p$ determines the selection probability of BI and GI (Lines 1-3). If $p = 0$, the inversion becomes GI (Lines 7-9). Conversely, if $p = 1$, the inversion becomes BI (Lines 4-6).

Adaptive mechanisms have been proposed in the literature to address the problem dependency of LS operators and to promote competition and cooperation between different LS operators [27]. In this research, we employ an Adaptive Inver-Over (AIO) mechanism in which the two inver-over operators (BI, GI) compete and cooperate together in order to get the advantages of both of them during different solution evolution periods. Using a similar mechanism as proposed by [27], the probability of the LS operator with the higher achievement will be increased. The proposed memetic trustworthy service adaptation algorithm is shown in Algorithm 5.

Let $p_{gi}$ and $p_{bi}$ represent the probability of applying GI and BI to the workflow selected for local search,
Algorithm 5 Memetic Trustworthy Service Adaptation
1: Set parameters α, β, µ and K
2: Set pheromone trails GTinit
3: set \( p_{bi} = p_{gi} = 0.5 \)
4: while the termination condition isn’t satisfied do
5:     for all \( t = 1 : \mu \) do
6:         repeat
7:             Start from requester service agent \( sa_i \)
8:             Select service agent \( sa_j \) to visit next (not visited before) according to an updating rule
9:         until walker agent has composed a workflow
10:     end for
11: best = find the best walker agent
12: \( \delta_{bi} = \delta_{gi} = 0 \)
13: for \( v = 1 : ls \) do
14:     if \( \text{rand}() \leq p_{bi} \) then
15:         inver-over(best,1) using Algorithm 4
16:     Update \( \delta_{bi} \)
17:     else
18:         inver-over(best,0) using Algorithm 4
19:     Update \( \delta_{gi} \)
20: end if
21: Recalculate \( p_{bi} \) and \( p_{gi} \)
22: end for
23: if \( M \) is full then
24: Remove the first best walker agent from \( M \)
25: Remove pheromone s.t. Eq. 9
26: end if
27: Insert best into \( M \)
28: Update Pheromone s.t. Eq. 3
29: end while

respectively (Lines 11-12 in Algorithm 5). To allow fair competition between the two operators, the initial probabilities are set to be equal, i.e., \( p_{bi} = p_{gi} = 0.5 \), where both \( p_{bi} + p_{gi} = 1 \). As in a dynamic service environment, services can join or leave at any time, and the biased selection towards a certain operator, i.e., by giving this operator more initial weight, would trip the search process into suboptimal regions of the search space. Let \( \delta \) represent the degree of improvement of the selected workflow after a local search step, such that:

\[
\delta = \frac{|GT(\text{best'}) - GT(\text{best})|}{GT(\text{best})},
\]

(10)

where \( GT(\text{best}) \) and \( GT(\text{best'}) \) are the group trust values for the iteration best workflow before and after local search (refer to Equation 4), respectively. After a preset number of local search steps \( ls \), the degree of improvement achieved by each operator is calculated as \( \delta_{bi} \) and \( \delta_{gi} \), respectively. Those values are used to adjust the probabilities of selecting both BI and GI (Lines 14-21 in Algorithm 5) in the next iteration \( (t+1) \) as follows:

\[
p_{bi}(t+1) = p_{bi}(t) + \delta_{bi}(t),
\]

(11)

\[
p_{gi}(t+1) = p_{gi}(t) + \delta_{gi}(t),
\]

(12)

\[
p_{bi}(t+1) = \frac{p_{bi}(t+1)}{p_{bi}(t+1) + p_{gi}(t+1)},
\]

(13)

\[
p_{gi}(t+1) = 1 - p_{bi}(t+1),
\]

(14)

5.3 Diversity Schemes

Due to the strong exploitation that LS operators provide, the memetic trustworthy service adaptation algorithm faces the risk of keeping identical solutions in the population list and prevents the walker agents from exploring more areas in the service environment. Diversity schemes have been found useful when applied to dynamic environments due to their ability to keep a specified level of diversity within the population list [17]. To this end, a diversity scheme-based algorithm is introduced in this subsection to enhance the memetic trustworthy service adaptation algorithm and it works as follows.

Algorithm 6 Diversity Scheme
1: Set parameter \( K \)
2: \( sim_t = 0 \)
3: for \( i = 1 : K - 1 \) do
4:     for \( j = 1 : K \) do
5:         \( sim(W_i, W_j) = (1\text{-common service agents}/ \text{number of service agents}) \)
6:         \( sim_t = sim_t + sim(W_i, W_j) \)
7:     end for
8: end for
9: \( div_t = (1/(k \ast (k - 1))) \ast sim_t \)
10: if \( div_t = 0 \) then
11:     Randomly generate \( migrant \)
12:     Replace a present workflow in \( M \) with \( migrant \)
13: end if

On every iteration \( t \), the diversity of the population-list (Lines 3-9 in Algorithm 6) is calculated as follows:

\[
div_t = \frac{1}{K(K - 1)} \sum_{i=1}^{K} \sum_{j \neq i}^{K} \text{sim}(W_i, W_j),
\]

(15)

where \( K \) is the size of the population list and \( \text{sim}(W_i, W_j) \) is the similarity between workflow \( W_i \) and \( W_j \), which is calculated as follows:

\[
\text{sim}(W_i, W_j) = 1 - \frac{\text{common service agents}}{\text{number of service agents}}.
\]

(16)

When a predefined diversity threshold is reached, random immigrants are activated (Lines 10-12 in Algorithm 6) to replace one walker agent in the population list with a random walker agent, in order to better adapt to the next iteration.

For clarity, Table 2 summarizes the set of algorithms presented throughout the proposed approach.

6 Experimentation and Results

A set of experiments have been conducted to explore the efficiency of the proposed approach wherein a set of benchmark measurements are used to assess the robustness of the proposed approach across several parameters.

1939-1374 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
6.1 Experiment Setup

Five experiments have been conducted to assess different facets of the proposed trustworthy stigmergic service composition and adaptation approach. The first experiment examines the effectiveness of the proposed approach in composing Web services in a decentralized environment. The second experiment inspects the performance and the efficiency of the proposed approach as related to the scale of the environment. The purpose of the third experiment is to examine the performance of the proposed approach using three different local search operators to optimize the pheromone update after every iteration. The fourth experiment studies the effect of the diversity provided by the random immigrants scheme on the performance of the proposed approach. Finally, the fifth experiment studies how well the proposed approach adapts to random changes in dynamic environments.

The proposed approach runs in successive iterations until reaching a convergence point. The proposed approach converges to an optimal solution once it receives the same approximate value of trust $T$ for a number of successive iterations. Those trust $T$ values are compared iteration by iteration and the difference is projected against a threshold. To reflect tight convergence settings, in our experiments, this threshold value is set to 0.0001, and the number of successive iterations is set to 100.

Since there is no any sizeable web service test case that is available in the public domain and can be used for experimentation purpose, we focus on evaluating the proposed algorithms using a synthetic data set. This data set is designed to mimic the characteristics of real-world web service data sets and is generated using the same approach as in [26]. We use a degree-based Internet topology generator, Inet 3.0 [23] to generate a power-law random graph with 6000 nodes to represent a service network. We assume an equal-degree random graph topology. The numbers of service classes in the process are supposed to range from 100 to 1000. For simulation, the initial trust rating for each node, i.e., web service, is randomly generated with a uniform distribution between [0, 1]. All the experiments have been conducted on a 3.33 GHz Intel core 2 Duo PC with 3 GB of RAM.

6.2 Experiment 1: Decentralization

To explore the effectiveness of the proposed stigmergic service composition and adaptation approach in decentralized environments, a composition request is submitted to the proposed approach and the $QoS$ values of the composed workflow is contrasted against their counterparts obtained by a centralized service composition approach [21]. The proposed approach assumes no prior knowledge of the environment including available service agents and their $QoS$ values, while other centralized approaches assume the existence of a central service registry, in which all the functional and $QoS$ qualities of Web services are recorded.

The centralized approach is used to benchmark the proposed approach. The centralized approach represents the optimal case with full knowledge of the available services and a deterministic composition plan. The purpose of this experiment is to show that the proposed approach can reach near optimal solutions although working under uncertainty and dynamicity that are deeply rooted in decentralized environments.

Table 3 shows the progression of the $QoS$ values of the composed workflow using the proposed approach against those values obtained by centrally composing the same workflow using centralized approaches, i.e., Greedy Approach [21]. Those values are recorded on a 100-iteration basis.

![Fig. 2. Decentralized Environment](image)

Fig. 2. depicts the contrast between the $QoS$ values obtained from the proposed approach against those values obtained from the centralized approach ($y$ axis) and the corresponding number of iterations ($x$ axis). As shown in Fig. 2., the centralized service composition approach shows a steady line with a $QoS$ value of 177, regardless of the number of iterations. This
TABLE 3
Decentralization vs. Centralization

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Stigmergic Adaptation</th>
<th>Centralized Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>115</td>
<td>177</td>
</tr>
<tr>
<td>200</td>
<td>123</td>
<td>177</td>
</tr>
<tr>
<td>300</td>
<td>137</td>
<td>177</td>
</tr>
<tr>
<td>400</td>
<td>145</td>
<td>177</td>
</tr>
<tr>
<td>500</td>
<td>145</td>
<td>177</td>
</tr>
<tr>
<td>600</td>
<td>145</td>
<td>177</td>
</tr>
</tbody>
</table>

is attributed to the assumption of prior knowledge of the available Web services and their qualities, which is not the case in the real world. On the other hand, the proposed stigmergic service composition and adaptation approach shows a progressive line representing an upward trend in the QoS values obtained as the number of iterations increases. Due to its decentralized nature, the proposed approach needs to run for a number of iterations before showing stability in results, which happens in our experiment when the number of iterations goes beyond 300. The best stable QoS obtained is 145, which shows an 82% rate of effectiveness compared to the optimal value of 177 in a fully centralized approach. This result demonstrates a good performance considering a decentralized open environment with no prior knowledge of available Web services and their QoS values.

6.3 Experiment 2: Environment Scales
To examine the scalability of the proposed approach in handling large-scale service environments, a composition request is submitted to the proposed approach with a workflow consisting of 20 abstract services/tasks. The proposed approach in turn sends 20 walker agents to traverse the service agent network to compose the requested workflow. The proposed approach works in iterations till reaching a convergence point. A single iteration is completed when all the walker agents return with their initial solutions. The proposed approach converges to an optimal solution when all the walker agents return with the same solution for a number of successive iterations. To empirically test the scalability of the proposed approach, we run the proposed approach multiple times and change the service network scale every time. The service network scale is represented by the number of member service agents in this network, and has a range from 10000 to 250000. The number of iterations till convergence is accordingly recorded as in Table 4.

<table>
<thead>
<tr>
<th>Environment Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Service Agents</td>
</tr>
<tr>
<td>10000</td>
</tr>
<tr>
<td>40000</td>
</tr>
<tr>
<td>90000</td>
</tr>
<tr>
<td>160000</td>
</tr>
<tr>
<td>250000</td>
</tr>
</tbody>
</table>

6.4 Experiment 3: Local Search Operators
To examine the effect of the proposed memetic trustworthy service adaptation algorithm, a composition request has been submitted to the proposed algorithm and the aggregate service trust value is measured. The purpose of this experiment is to study the effect of BI, GI and AIO, when applied to the list of iteration best workflows.

Table 5 shows the progression of the aggregate trust values of the composed workflows using the memory-based algorithm (without local search) against those values obtained by composing the workflows using the memetic trustworthy service adaptation algorithm with two different local search operators, i.e., BI and GI, and the adaptive search mechanism, i.e., AIO.

Fig. 4. depicts the relationship between the number of workflows evaluated (x axis) and the corresponding best performance (y axis). The best performance is defined as the aggregate trust that the algorithm has achieved after evaluating one workflow, (refer to Equation 1). In Fig. 4., we use blind, guided, and adaptive to denote BI, GI and AIO, respectively. Also, we use standard to denote the memory-based algorithm. From the experimental results shown in Fig. 4., several conclusions can be drawn. First, it is obvious that all memetic algorithms outperform the standard memory-based algorithm since they converge to a
much better solution. Different memetic algorithms, however, work well on different instances. Second, the adaptive search mechanism, AIO, always performs the best since it combines the merits of both BI and GI, i.e., the convergence speed and solution quality. Therefore, AIO is considered a good choice and will be used for the rest of our experiments.

### 6.5 Experiment 4: Using Diversity Schemes

In the fourth experiment, we study the effect provided by the diversity scheme on the performance of the memory-based algorithm. To this end, we compare the memory-based algorithm without the diversity scheme and the memory-based algorithm with the diversity scheme.

Table 6 shows the progression of the aggregate trust values of the composed workflows using the memory-based algorithm without the diversity scheme against those values obtained by incorporating the diversity scheme.

Fig. 5 plots the relationship between the number of workflows evaluated (x axis) and the corresponding aggregate trust values (y axis).

From Fig. 5, the following results can be observed. The proposed diversity scheme efficiently improves the performance of the memory-based algorithm, since the diversity scheme-enhanced version gives significantly better aggregate trust values in almost all the cases, with a varying number of workflows. This is due to the fact that once the iteration best workflow converges to an optimum, random immigrant workflows are generated to replace existing workflows in memory M. This helps to promote diversity and to avoid a possible local optimum.

Due to the strong exploitation that the local search operators provide, diversity schemes are very important to steer the search process away from suboptimal regions in the solution space. Also, the diversity schemes help to fix the premature convergence of memetic algorithms.

The diversity of the memory-based stigmergic service composition and adaptation algorithm, in general, can not reach a very high level. This is due to the guidance that walker agents gain from the visibility information, i.e., concrete service trust, while they construct their solutions. Therefore, even a small increase in the diversity of the population can have a significant impact on the performance of the algorithm.

### 6.6 Experiment 5: Dynamic Adaptation

In this experiment, we study how well the proposed algorithms adapt to changes in dynamic environ-
ments. The dynamic change in service-oriented environments is expressed by the change in the trust ratings of the concrete Web services which may go up or drop down for various reasons, and in turn, affects the quality of the workflow.

![Dynamic Adaptation](image)

### 7 RELATED WORK AND DISCUSSIONS

Various approaches have been proposed to handle the problem of dynamic composition of Web services [5], [26], [30]; however, most of these approaches are based on centralized mechanisms. These centralized mechanisms become impractical in scenarios with dynamic arrivals and departures of service providers, which require frequent updating of the central entities, resulting in a huge system overhead.

Some approaches exploiting distributed architectures have been proposed in [2], [4]. For example, Basu et al. [2] proposed an approach, in which a composite service is represented as a task-graph and sub-trees of the graph are computed in a distributed manner. Their approach, however, assumes that a service requestor is one of the services, and the approach relies on this node to coordinate service composition. A similar approach was proposed by Chakraborty et al. [4], whose work is different from Basu et al. [2] in terms of the way the coordinator is elected. For each composite request, a coordinator is selected from a set of nodes. The service requestor delegates the responsibility of composition to the elected coordinator. Both approaches [2] and [4] have limitations. Although service discovery is performed in a distributed manner, service composition still relies on a coordinator assigned to perform the task of combining and invoking services. Also, both of the approaches assume the direct interaction between nodes responsible for service discovery and service composition. In contrast, our approach overcomes these limitations because: (1) in our approach, there is no special entity to manage the service composition process, (2) service providers communicate only with their local neighbors where no service provider knows the full global information; and (3) our approach adopts an indirect interaction scheme where the set of agents responsible for service composition and adaptation can interact with each other through the environment by exchanging pheromone trails.

Early attempts for service composition models in decentralized environments have been proposed lately [8], [20]. Sim [20] proposed a self-organizing multi-agent approach for decentralized service composition in cloud environments. However, the dependency on service capability tables to maintain information about existing cloud services represents a notable drawback. To reflect new changes in a dynamic environment, service capability tables need to be updated periodically, which could limit the adaptability of the approach when deploying in highly dynamic service environments. Fernandez et al. [8] proposed an approach to promote decentralized workflow execution using the chemical reaction paradigm. However, the proposed approach did not provide a clear mechanism for handling open environments where services can join or leave the environment at any time.
Recently, many trust and reputation models have been proposed for Web services selection. Each model uses a different source of trust and proposes a different approach to derive the trust ratings. Nevertheless, no single model can address all the trust challenges [10], [14], [15].

Few trust models consider the direct experience of the service consumer drawn from past interactions with the service provider [10]. In those models, trust is computed as a rating of the performance that the service provider has demonstrated over multiple interactions. This means that the service consumer trusts the service provider because of his good performance in the past. The suitability of those approaches is questionable. In open environments where millions of new services join frequently, those approaches do not allow trust in a new service. Either some experience with that provider is necessary or low trust ratings are given to new service providers.

Reputation-based models are another type of trust models wherein a service consumer trusts a service provider because of the provider’s good reputation [14]. Reputation represents the collective evaluation of a service provider by other members of the community. Hence, a reputation system needs to collect ratings from other members of the community about a particular provider behavior and then computes and publishes the resulting reputation score. In turn, a service requestor uses this score to select the most trustworthy service provider [1]. Reputation-based models, however, also have some limitations. One limitation is the dependency on other members to provide reputation information. This problem arises in the case of new members joining the system with no historical record. Another limitation is that reputation-based models are mostly centralized, and hence make them unsuitable to be adopted in Web service environments, which are decentralized by nature. In addition, reputation-based models count on an adequate number of member services to provide competent ratings. To overcome these limitations, content driven reputation systems [7] have been proposed, which rely on the automatic analysis of the content and the collaboration process, rather than on the explicit user feedbacks, to derive trust ratings. However, the algorithmic nature of content driven systems can play against their success, preventing users from understanding and consequently trusting the reputation values they generate.

Recommendation is the most popular source of trust in current models. An approach is proposed in [15] to target multiple QoS attributes. This approach calculates trust by taking the difference between the consumer’s preferred quality attribute value and the vector including, maximum, minimum, typical and aggregate reputation values obtained from ratings of that quality attribute. The rater credibility is not considered in this approach; however. Another approach is proposed in [28] where collaborative filtering is used to predict QoS values of Web services and to make Web service recommendations, however, this approach lacks personalized models to express different users’ preferences. A key limitation of the recommendation models is the reliance on the existence and awareness of a good community to provide ratings to a centralized recommender system. This feature would not hold in open Web service environments, putting the service consumer in a weak position if the community does not provide a significant rating system.

The centralized nature is a shared drawback of both recommendation-based and reputation-based trust models. Both models rely on a governing entity to grant authentication and calculate trust ratings. To overcome this drawback, referral approaches have been proposed to present a decentralized model based on Web communities [25]. In referral-based models, a service consumer trusts a service provider based on references obtained from other trusted consumers. Unlike recommendation-based models which hide the identity of the source of the recommendations, in referral models, the participants reveal their ratings to whom they trust, therefore, become more honest. Referral approaches, however, are still based on the referral coming from the previous experience of someone else without considering the presence of neighbors that might help.

Trust and reputation management systems for P2P networks and online distributed systems received a lot of attention recently [3], [11], [12]. Eigen Trust [12] is one of the most popular reputation management algorithms for P2P networks. However, the Eigen Trust algorithm is constrained by the fact that it computes the global reputation value by a simple iterative weighted averaging mechanism, which is vulnerable to collaborative attacks from malicious peers. Another algorithm that uses the Bayesian framework has been proposed in [3], in which, each reputation value is computed independent of the other nodes’ reputation values. Therefore, ignoring these dependencies could affect the performance dramatically.

All the aforementioned approaches use the notion of trust to support trustworthy service selection, but, they still have a number of limitations. These include: (1) failing to deal with the dynamic behavior of services, (2) failing to introduce a comprehensive approach that combines direct experience and third-party referral together, (3) failing to take the composite services dependency relationships into account and (4) failing to provide a mechanism for trust dissemination among neighborhoods.

Unlike the above approaches, the approach proposed in this paper addresses those limitations by (1) a hybrid model for trust management through combination between individual experience and group referral, (2) the consideration of dependency rela-
8 CONCLUSION AND FUTURE WORK

This paper presents a trustworthy service composition and adaptation approach using stigmergic interactions to actively adapt to dynamic changes in complex decentralized service environments. The proposed approach is able to adapt actively to trust fluctuations considering the potential emergence and degradation of trust ratings. The experimental results have shown the effectiveness of the proposed approach in highly complex and dynamic service environments. The future work is set to investigate the collaboration among service agent communities and the coding of social preference choices under the proposed model.

REFERENCES


Ahmed Moustafa received the BSc and the MSc degrees from Zagazig University, Egypt in 2003 and 2010, respectively. Now, he is working toward the PhD degree in computer science at the University of Wollongong, Australia. His research interests focus on self-organizing multi-agent systems and their applications.

Minjie Zhang received the BSc and the MSc degrees from Fudan University, P.R. China in 1982 and the PhD degree in computer science from the University of New England, Australia, in 1996. Now, she is an associate professor of computer science at the University of Wollongong, Australia. Her research interests include multiagent systems and agent-based modeling in complex domains. She is a senior member of IEEE.

Quan Bai received the PhD in computer science at the University of Wollongong in 2007. He is presently a senior lecturer at Auckland University of Technology. He specializes in multi-agent coordination, trust-based computing and service composition.