An Agent-based Integrated Self-evolving Service Composition Approach in Networked Environments

Dayong Ye, Qiang He, Yanchun Wang and Yun Yang

***Abstract*—Service composition is an important research problem in service computing systems, which combines simple and individual services into composite services to fulfill users' complex requirements. Service composition usually consists of four stages, i.e., service discovery, candidate selection, service negotiation and task execution. In self-organising systems, there is the fifth stage of service composition: self-evolution. Most of existing works study only some of the five stages. However, these five stages should be systematically studied so as to develop an integrated and efficient service composition approach. Against this background, this paper proposes an agent-based integrated self-evolving service composition approach. This approach systematically takes the five stages of service composition into consideration. It is also decentralised and self-evolvable. Experimental results demonstrate that the proposed approach can achieve almost the same success rates while uses much less communication overhead and time consumption in comparison with three existing representative approaches.**

Keywords - Service Composition, Self-organisation, Multi-agent Systems

I. Introduction

With the development of computing, service-oriented computing becomes a new paradigm for engineering modern complex systems which are composed of various services for users' consumption [1]. Due to the increase of the complexity of users' requirements, individual services sometimes cannot fulfill users' requirements. Service composition becomes a necessary and effective technique to satisfy users' complex requirements. Service composition enables simple and individual services to be dynamically combined into composite services by coordinating service providers and configuring existing services so as to meet users' complex requirements [2]. Currently, service composition has been widely studied in many systems, e.g., cloud systems [3], multi-agent systems [4], mobile ad hoc networks [5] and sensor networks [6]. Specifically, service composition is an aggregation of atomic or composite services which collaborate to implement a set of operations in order to carry out a complex task or a business process [7].

In the last decade, a number of service composition approaches have been proposed. Most of these approaches are centralised [8], [9], while only a few of them are decentralised [10], [11]. In these centralised approaches, there is a central manager or controller who has the full knowledge of the environments and is responsible to organise service composition for consumers in the environments. The centralised approaches are very efficient in small scale environments. However, they incur computation bottleneck and suffer the single point of failure. They are also inapplicable in some environments where there is not a central manager, e.g., sensor networks [6]. The decentralised approaches overcome the computation bottleneck and the single point of failure drawback of the centralised approaches by enabling a service consumer to autonomously organise service composition. However, some decentralised approaches [12], [13] assume that each service consumer has the full knowledge of the environment. This assumption is infeasible in some large-scale environments, e.g., multi-cloud environments [14]. Other decentralised approaches [10] waive this assumption by modeling a networked environment, where each participant has a set of neighbours or acquaintances and each participant has knowledge only about its neighbours or acquaintances. These decentralised approaches, however, overlook the evolution of the operating environments, where participants may change the acquaintance relationship between each other over time and service providers may buy extra services from other participants or sell infrequently used services to other participants. Such evolution exists in a number of real world systems, e.g., multi-agent systems [15], [16], social networks [17] and peer-to-peer systems [18]. Therefore, to adapt to such evolving environments rapidly, it is important to develop a service composition approach which can self-evolve over time. Moreover, service composition consists of five stages: service discovery, candidate selection, service negotiation, task execution and self-evolution, whereas most of the existing approaches study only some of the five stages [19], [5], [20], [21]. To develop an integrated and efficient service composition approach, all the five stages should be systematically studied.

Based on this background, in this paper, an agent-based integrated self-evolving service composition approach is proposed. The proposed approach tackles two research challenges of service composition. First, self-evolution is an important property of self-organising systems [22], [16]. However, self-evolution in service composition has not been addressed properly in existing approaches [19]. Second, as described above, service composition consists of five stages which should be systematically considered to achieve an integrated service composition approach. Existing approaches, however, consider only some of the five stages [5].

In summary, the advantages of the proposed approach are as follows.

1) The proposed approach is decentralised which can avoid the computation bottleneck and the single point of failure and does not need global information.

2) The existing approaches, which consider self-evolution, study service migration only [19], whereas the proposed approach studies both service migration and relation modification. Service migration is for service exchange among participants in order to increase utilisation of services. Relation

This work is supported by an ARC Discovery Project (DP150101775) from Australian Research Council, Australia.

D. Ye is with the State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China and the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia. Q. He, Y. Wang and Y. Yang are with the School of Software and Electrical Engineering, Swinburne University of Technology, Australia. Email: {dye,qhe,yanchunwang,yyang}@swin.edu.au

modification enables participants to remove existing and/or add new acquaintances so as to build neighbourhood with other useful participants. Both of service migration and relation modification are important features of self-organising systems [22], [16].

3) Unlike the existing approaches which consider only some of the five stages of service composition [5], the proposed approach is the first integrated one which combines all the five stages.

4) The proposed approach is developed in a general environment instead of specific systems, e.g., mobile ad hoc networks [20]. This means that the proposed approach could be applied in various systems.

The rest of the paper is organised as follows. Section II presents related studies about service composition. Section III provides a motivating example of our research. Section IV describes the detail of our integrated self-evolving service composition approach. Experiments and the corresponding analysis are given in Section V. Section VI concludes the paper and points out the future research directions.

II. Related Work

Various approaches have been proposed to handle the service composition problem [8], [9]. Most of these approaches are centralised [23], [1], [24], [25], [14].

Ardagna and Pernici [1] formalised the service composition problem as a mixed integer linear programming problem. They used loops peeling in the optimisation and took constraints posed by stateful services into consideration. In their approach, there is a centralised *Concretisator* module which selects the best concrete services to be invoked from a service registry for each task of the composed service according to user's requirements and the optimisation criteria.

Kurdi et al. [14] developed a centralised combinatorial optimisation algorithm for service composition in multi-cloud environments. Their algorithm includes a cloud combiner and a service composer. The cloud combiner is used to select the appropriate cloud combination from the multi-cloud environment. The service composer takes the cloud combination as input and generates a final service composition sequence which meets user's request.

The centralised approaches are very efficient in small scale environments. However, as all the computation and communication are handled by a central manager, the centralised approaches may incur computation and communication bottlenecks and suffer the single point of failure. In order to overcome the drawbacks of centralised approaches, decentralised approaches have also been proposed.

Moustafa and Zhang [12] proposed a reinforcement learning approach for multi-objective service composition and adaptation in dynamic uncertain environments. They used multi-objective Markov decision process to model the service composition problem. Then, two algorithms were devised to handle single policy and multiple policy multi-objective service composition based on user preferences. The solution of each of the two algorithms is a procedure that indicates how an agent selects a service in each state. By using the two algorithms, an agent can find a set of optimal workflows which fulfill the trade-offs among multiple QoS objectives.

Wang et al. [26] proposed a reinforcement learning approach for large scale and adaptive service composition. They modeled the service composition problem as a multi-dimensional transition graph and developed a multi-agent learning algorithm to handle the problem. The solution of their algorithm is a procedure that indicates how a team of agents selects services in each state. Unlike Moustafa and Zhang's single agent learning algorithm, Wang et al.'s algorithm is a multi-agent one which can perform better than a single agent learning algorithm.

Moustafa et al. [11] proposed a stigmergy-based approach to model service interactions and handle service composition. In their approach, interactions among service agents are indirect by leaving and sensing artificial pheromone. Such pheromone encodes specific information which is used to achieve service composition. Also, they developed a trust model for service selection based on the balance between the trust ranks of the concrete services and the trust ranks of the whole workflow.

These decentralised approaches distribute the decision process about service composition to each service consumer. Thus, the computation bottleneck and the single point of failure can be avoided. These decentralised approaches, however, have to use global information for service composition. For example, a service consumer is allowed to select services in the whole environment, which implies that a service consumer knows all the service providers in the environment. The use of global information is infeasible in large environments. In order to avoid using global information, some decentralised approaches were developed in networked environments, where each participant has a set of acquaintances and each participant has knowledge only about these acquaintances.

Sim [3] proposed an agent-based paradigm for cloud resource management which includes cloud search, cloud commerce and cloud service composition. In his approach, each participant has a service capability table which records a list of neighbouring participants and their services. Then, a consumer selects services for composition by consulting its service capability table to determine to which participants it should send its request message. Through this way, the number of messages exchanged among participants in the system can be significantly reduced compared to those approaches which broadcast messages to all the participants in the system.

Gutierrez-Garcia and Sim [10] devised a semi-recursive contract net protocol enhanced with service capability tables. The protocol is based on recursive calls of the contract net protocol [27] which is a distributed problem solving technique used for achieving service agreements among service consumers and service providers. By using the protocol, a service consumer can efficiently find feasible service providers which are capable of carrying out a given task.

These decentralised approaches were designed based only on local information, so they can be used in large scale environments. However, these approaches do not consider the multiple stages of service composition, including the evolution of environments, where participants may change the acquaintance relationship between each other over time and participants may buy extra services from others or sell infrequently used services to others. Some other decentralised approaches take multiple stages of service composition into consideration but they still have some limitations or differences compared to our approach.

Prinz et al. [19] studied dynamic service composition in P2P systems by introducing logical peer groups. They then proposed an approach which enables runtime composite service reconfiguration including service exchange among peers.

The formation of a logical service group is carried out by an initial coordinator which collects information about all peers required sub-service. During execution, new peers may arrive and peers' execution properties may vary. Then, sub-services may be exchanged among peers to guarantee the proper execution of the system. Their approach, however, is based on global information, while our approach is based only on local information. Moreover, self-evolution in their approach considers service exchange only, while self-evolution in our approach considers both service exchange and relation modification.

Cardellini et al. [21] proposed a decentralised service composition approach by using a gossip protocol to support information dissemination and decision making. In their approach, each peer has three components: network latency estimator, workflow manager and gossip manager. Network latency estimator estimates the network delay between pairs of peers in order to meet QoS requirements. Workflow manager is responsible for initialising the composition of a new workflow or for repairing an existing workflow. Gossip manager is responsible for decentralised information dissemination and decision making so as to achieve dynamic service composition and adaptation of workflows. Their approach is fully decentralised which is based only on local information. However, their study focuses on service discovery and selection while overlooks other stages. In comparison, our approach considers all the five stages.

Groba and Clarke [20] presented a service composition protocol which opportunistically allocates and invokes service providers. They modeled a service composite as a directed acyclic graph and a service provision as a time-varying graph. Then, service composition is to find a mapping between the directed acyclic graph and the sequence of time-varying graph such that the edges between two services are enabled for the time they interact. Their protocol is independent of network topology and can reduce failure probability. Their study is different from ours in the following aspects. First, their protocol is developed in dynamic ad hoc environments, while our approach is designed in a general environment, which means that our approach could be applied in various environments. Second, for service composition, their protocol mainly focuses on three stages: service discovery, service selection and service execution (known as service allocation and invocation in the paper)[1], while our approach studies all the five stages. Third, in their protocol, the service composition problem is modeled as a graph mapping problem, while in our approach, the service composition problem is modeled as a multi-agent resource allocation problem. Since the models and environments are different, the solutions are entirely different.

Chen and Clarke [5] proposed a decentralised service composition model, where a system dynamically adapts its business process by composing its fragments on-demand to meet the constraints of service providers and consumers. Their model matches L ($L \leq 1$) service queries to M ($M \leq 1$) services by adapting abstract workflows and their concrete implementation details. Their work is carried out in ad hoc environments and focuses on two stages of service composition: service discovery and service selection, while our work is conducted in general environments and focuses on all the five stages of service composition.

Compared with existing approaches, our approach is decentralised, uses only local information and takes multiple stages, including self-evolution, of service composition into consideration.

III. A Motivating Example

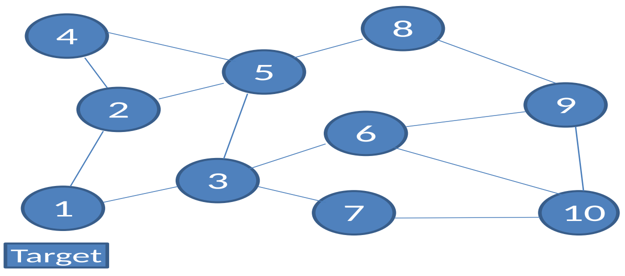


Fig. 1. A motivating example

This section provides an application example from target tracking in wireless sensor networks (WSNs). It should be noted that the target tracking in WSNs is an application example only. The proposed service composition approach will be developed in a general environment rather than any specific networks or systems.

In Fig. 1, each circle represents a sensor node. Suppose that sensor 1 has detected a target and needs to identify and track the target. To identify and track a target, sensor 1 needs two services: identification service and tracking service. Sensor 1, however, does not have such two services and needs to ask other sensors in the network for help. This is a wireless sensor network, so sensor 1 has knowledge only about its neighbouring sensors, i.e., sensor 2 and sensor 3. Sensor 1 will inquire sensor 2 and sensor 3 that if they can provide any of the services. In the case that sensor 2 and sensor 3 cannot provide the required services, they will inquire their neighbours. This process will continue until the pre-set TTL (Time To Live) value for the discovery of each service is reached (stage 1). After the search process, several service providers may be found for each service request. Then, sensor 1 will select providers for the service requests based on some criteria (stages 2 and 3). After execution of these services (stage 4), sensor 1 will evolve (stage 5). Sensor 1 may remove less efficient neighbours or may add efficient sensors as new neighbours. In this example, the removed or added neighbours are logic neighbours rather than physical neighbours, because physical neighbours are formed based on sensors sensing ranges and thus cannot be changed arbitrarily. In WSNs, sensors can change physical neighbours by adjusting their sensing ranges. The detailed discussion of discovery and change of physical neighbours in WSNs is out of the scope of this paper. Interested readers can refer to [28].

Specifically, the service workflow of the example tracking network is given in Fig. 2. As stated in [29], [30], there are roughly seven types of services used in a tracking network. S_1 is the target discovery service which is used for discovering that a target appears in the network. S_2 is the target detection service which is used to locate the position of the target. S_3, S_4 and S_5 constitute the target identification service, where S_3 is the target feature extraction service, S_4 is the target feature

[1]Groba and Clarke [20]'s protocol also considers other two stages: synchronisation and communication. The two stages have to be considered in ad hoc environments but are not necessary in general environments. If our approach is applied to ad hoc or other communication environments, these two stages will be taken into account.

classification service, and S_5 is the target feature recognition service. S_6 is the target tracking service which is used to chase the target. S_6 may be provided by multiple service providers, i.e., sensors, as the target may travel in the network and a sensor is not powerful enough to cover the area that the target travels through. S_7 is the processing and reporting service which is used to process the collected information and report the outcome to the user.

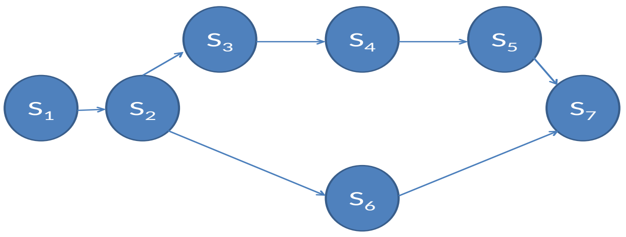


Fig. 2. The service workflow

In this application example, the WSN is a decentralised network. Thus, those centralised service composition approaches are most likely inadequate here. Also, in WSNs, the communication capability of sensors is very limited [31], [32]. Thus, those decentralised service composition approaches, which have to use global information, are also most likely inadequate, as the acquisition of global information involves large overhead of communication. Finally, as WSNs are self-organising systems [33], those decentralised service composition approaches, which do not consider the evolution of environments, are again most likely inadequate here either. Therefore, a decentralised service composition approach, which considers multiple stages of service composition and uses little communication overhead, is necessary and the aim of this paper is to develop such an approach.

IV. The Integrated Self-evolving Approach Design

In this section, we first formulate the problem. Then, an overview of the proposed approach is given. Finally, the details of the proposed approach are presented.

A. Problem Formulation

The proposed approach is developed in a distributed networked environment which is defined as an acquaintance network[2] in **Definition 1**.

Definition 1: (Acquaintance Network). The acquaintance network is defined as an undirected graph $\mathcal{G} = \langle V, E\rangle$. $V = \{v_1, ..., v_n\}$ is the set of participants in the network. $E \subseteq V \times V$ is the set of edges which link participants. Two participants, v_i and v_j, are acquaintances of each other if $(v_i, v_j) \in E$. Each participant, v_i, has an acquaintance set $N_i = \{v_j | \langle v_i, v_j\rangle \in E\}$. In addition, $N_i^+ = N_i \cup \{v_i\}$ is the acquaintance set which includes v_i itself. Each participant can directly interact only with those participants in its acquaintance set.

Each participant, v_i, in the acquaintance network has a set of services recorded as S_i. Each participant has three possible roles: consumer, broker and provider which are defined in **Definition 2**.

Definition 2: (Consumer, Broker and Provider). When a participant initialises a service request, this participant is called a service consumer. When a participant provides a service or services to a consumer, this participant is called a service provider. When a participant receives a commitment from a consumer to find proper providers, this participant is called a broker.

According to **Definition 2**, the role of a participant is based on a specific service request. As each participant has a set of services, a participant can be a consumer, a broker and a provider simultaneously. For example, a participant, v_i, needs a service, s_1, so for service s_1, v_i is a consumer. Meanwhile, v_i provides service s_2 to participant v_j, so for service s_2, v_i is a provider. Also, v_i has a commitment from participant v_k for service s_3 to find a proper provider, so for service s_3, v_i is a broker. As these processes can happen concurrently, v_i can be a consumer, a broker and a provider simultaneously.

The service composition problem in this paper is described in **Definition 3**.

Definition 3: (Service Composition). In an acquaintance network, when a consumer has a complex task, T, which needs a set of services to fulfill, $T \leftarrow \{s_1, ..., s_m\}$, then how to find and select proper providers for these services is called service composition in this paper[3].

In **Definition 3**, the number of required services of a complex task depends on the type of the task. For example, in Section III, the target chasing task needs five services: the target discovery service, the target detection service, the target identification service, the target tracking service, and the processing and reporting service. Furthermore, the target identification service can be considered as a sub-task which needs three services: the target feature extraction service, the target feature classification service and the target feature recognition service. As our approach is developed in a general environment, we do not set a specific number of services for a complex task. Instead, we use a variable, m, to represent the number of required services.

B. Overview

To make participants autonomous, each participant is equipped with an intelligent agent which behaves on behalf of the participant. Then, various intelligent agent technologies, e.g., multi-agent learning, multi-agent trust management and multi-agent negotiation, can be used. The use of agent technologies in service composition can overcome the limitations in traditional service composition approaches [11]. Here, an intelligent agent is an entity which can make rational decisions autonomously in a dynamic environment. An intelligent agent blends pro-activeness and re-activeness, shows rational

[2]As this paper aims at developing a service composition approach, the detailed discussion of establishment and maintenance of networks is out of the scope of this paper. Interested readers can refer [34], [35], [36] for the establishment and maintenance of networks.

[3]Accurately, service composition should include the building up process of a composite service. The building up process of a composite service, however, depends on specific applications. As the proposed service composition approach is developed in a general environment, we do not consider the building up process of a composite service in this paper.

commitments to decision making and exhibits flexibility when facing an uncertain and changing environment [37].

When a consumer has a complex task to fulfill, the consumer agent starts the service composition using the proposed approach. The service composition approach consists of the following five stages.

1) Service discovery: For each of the required service, the consumer agent searches the acquaintance network directly or via its acquaintances.

2) Candidate selection: After search, the consumer agent uses a trust model to select the candidate service providers.

3) Service negotiation: After selection, the consumer agent negotiates with these candidate provider agents in order to achieve service agreements for the required service.

4) Task execution: Then, the consumer agent chooses a set of service agreements for the required service. In the set of service agreements, one service is used to carry out the task and other services are used as backup in case that the service in use unexpectedly fails.

5) Network evolution: Finally, the consumer agent evaluates its current status, based on historical data, to make decisions that whether it needs to modify relationships with its acquaintances and whether it needs to buy or sell some services.

The mechanism or algorithm for each stage will be described in detail in the following sub-sections.

C. Stage 1: service discovery

The service discovery process is based only on local information, which is described in **Algorithm 1** as follows.

Algorithm 1: Service discovery process

```
1  for each participant $v_i \in V$ do
2      for each service request, $s_k$, in task $T$ of $v_i$ do
3          $CandList_i(s_k) \leftarrow \emptyset$;
4          if any acquaintance $v_j$, where $v_j \in N_i^+$, can provide service $s_k$ then
5              $CandList_i(s_k) \leftarrow CandList_i(s_k) \cup \{v_j\}$;
6          while $TTL(s_k) > 0$ do
7              select $m$ acquaintances as brokers;
8              if any acquaintance $v_j$ of these brokers can provide service $s_k$ then
9                  $CandList_i(s_k) \leftarrow CandList_i(s_k) \cup \{v_j\}$;
10             $TTL(s_k) \leftarrow TTL(s_k) - 1$;
```

For each service request s_k of a participant v_i, v_i creates a candidate list $CandList_i(s_k)$ and initialises the list as an empty set (Lines 2 and 3). First, v_i inquires its acquaintances whether they have the service s_k (Line 4). Those participants which have the service s_k will be added into the candidate list $CandList_i(s_k)$ (Line 5). Then, v_i selects m acquaintances from N_i as brokers and commits the selected acquaintances to find candidates for service s_k (Line 7). This selection is based on the performance of v_i's acquaintances, which is described in **Algorithm 2**. Each broker inquires its acquaintances about service s_k and replies v_i which acquaintances have service s_k (Line 8). These acquaintances, which have service s_k, will be added in the candidate list $CandList_i(s_k)$ (Line 9). Afterwards, each broker continues the search process by selecting m acquaintances as new brokers until $TTL(s_k)$ reaches 0. Here, local information means that each participant knows only its neighbours, i.e., acquaintances. If a participant wants to have more information, it has to inquire its neighbours. In **Algorithm 1**, the loop is used by participants to obtain more information. The loop is controlled by a parameter, TTL, which is set by users, to avoid infinite search in the network. Therefore, how many participants can be searched in a loop entirely depends on the value of TTL. Different services may have different values of TTL. For example, important services can be set a relatively larger TTL compared to less important services so as to have more providers found.

Algorithm 2: Broker selection

```
1  \* Initialisation * \
2  for each acquaintance $v_j \in N_i$ do
3      Let the performance of $v_j$ be $Q(v_j)$;
4      Let the probability of selecting $v_j$ as a broker be $p(v_j)$;
5      The probability distribution over acquaintances is recorded as $P = \langle p(v_1), ..., p(v_{|N_i|}) \rangle$, where $|N_i|$ means the number of acquaintances of $v_i$ and $\sum_{1 \le j \le |N_i|} p(v_j) = 1$ and $0 < p(v_j) < 1$;
6      $v_i$ initialises each $Q(v_j) = 0$ and $p(v_j) = \frac{1}{|N_i|}$;
7  \* Selection and adjustment * \
8  $v_i$ selects $m$ acquaintances as brokers based on probability distribution $P$;
9  Each acquaintance of $v_i$ is evaluated and assigned a reward $r_j$;
10 $\bar{r} \leftarrow \frac{1}{|N_i|} \sum_{v_j \in N_i} r_j$;
11 for each acquaintance $v_j \in N_i$ do
12     $Q(v_j) \leftarrow (1-\alpha)Q(v_j) + \alpha \cdot r_j$;
13     $p(v_j) \leftarrow p(v_j) + \zeta(Q(v_j) - \bar{r})$;
14 $P \leftarrow Normalise(P)$;
```

In **Algorithm 2**, v_i first initialises the performance of each acquaintance and the probability of selecting each acquaintance (Lines 2-6). Because initially, v_i is not aware of its acquaintances' performance, it simply sets the performance of each acquaintance to $Q(v_j) = 0$ and sets the probability of selecting each acquaintance equally to $p(v_j) = \frac{1}{|N_i|}$. Then, v_i selects m acquaintances as brokers based on probability distribution P (Line 8). For example, v_i has three acquaintances and the probability distribution over the three acquaintances is $P = \langle 0.2, 0.3, 0.5 \rangle$. Then, v_i has the probability 0.2 to select the first acquaintance, the probability 0.3 to select the second acquaintance and the probability 0.5 to select the third acquaintance, as a broker.

When these brokers finish their job, v_i evaluates the performance of each acquaintance and assigns each acquaintance a reward r_j (Line 9). For those acquaintances which are not selected as brokers, they are assigned a reward 0. For the brokers, the reward of a broker is based on how many candidates it has found for v_i's task T. The more candidates a broker has found, the more reward the broker is assigned. For example, suppose a threshold is set as 3. Then, if a broker has found 5 candidates, this broker is assigned a reward 2, where $2 = 5 - 3$; whereas if another broker has found only 2 candidates, this broker is assigned a reward -1, where $-1 = 2 - 3$.

After the assignment of reward to each acquaintance, v_i adjusts the performance of each acquaintance and the probability of selecting each acquaintance (Lines 10-13). In Line 12, v_i adjusts the performance of each acquaintance, $Q(v_j)$, based on the acquaintance's current performance and the reward, r_j, assigned during the service discovery for task T, where $0 < \alpha < 1$ is the learning rate which is a constant of deadline, the consumer's reserved price for the service, the probability of reaching an agreement and the competition of the required resource.

• $accept[o]$. When a provider receives a quote o, it can accept the quote, which results in a temporary agreement made with the consumer.

• $counter\_quote[o']$. If a provider is not happy with a quote o, it can send back a counter-quote o' for its service. A counter-quote o' is determined by three aspects which include the provider's reserved price for the service, the probability of reaching an agreement for the serivce and the competition of the service.

• $cancel[o]$. After a temporary agreement is achieved by a consumer and a provider, any one of them can cancel the temporary agreement without paying penalty. However, cancellation penalty applies on a backup agreement and a final agreement.

The negotiation protocol is described in **Algorithm 4**.

Algorithm 4: Service negotiation

```
1  \* Suppose $v_i$ is a consumer and $v_j$ is a provider * \
2  while $t <$ the deadline of the task do
3      $v_i$ generates a quote $o$ to $v_j$;
4      if $v_j$ accepts $o$ then
5          $\mathcal{A}^{\mathcal{T}}(v_i) \leftarrow \mathcal{A}^{\mathcal{T}}(v_i) \cup \{o\}$;
6          $\mathcal{A}^{\mathcal{T}}(v_j) \leftarrow \mathcal{A}^{\mathcal{T}}(v_j) \cup \{o\}$;
7          return;
8      else
9          $v_j$ generates a counter-quote $o'$ to $v_i$;
10         if $v_i$ accepts $o'$ then
11             $\mathcal{A}^{\mathcal{T}}(v_i) \leftarrow \mathcal{A}^{\mathcal{T}}(v_i) \cup \{o'\}$;
12             $\mathcal{A}^{\mathcal{T}}(v_j) \leftarrow \mathcal{A}^{\mathcal{T}}(v_j) \cup \{o'\}$;
13             return;
14         else
15             continue;
```

In **Algorithm 4**, before the deadline of the task, a consumer and a provider can alternately propose quotes and counter-quotes until an agreement is reached. In Line 3, consumer v_i generates a quote o to provider v_j, where $o = \langle pr, pe\rangle$. Each of the elements in o can be calculated using the following equations.

$$pr = \begin{cases} res\_pr_{v_i} \cdot \frac{t-arrive(s_k)}{start(s_k)-arrive(s_k)} \cdot \frac{\delta(t)}{\Delta(t)}, \text{if } \mathcal{A}^{\mathcal{T}}(v_i) = \emptyset \\ \frac{res\_pr_{v_i}}{|\mathcal{A}^{\mathcal{T}}(v_i)|} \cdot \frac{t-arrive(s_k)}{start(s_k)-arrive(s_k)} \cdot \frac{\delta(t)}{\Delta(t)}, \text{otherwise} \end{cases} \tag{2}$$

$$pe = \frac{pr}{exe_{v_j}} \tag{3}$$

In Equation 2, pr is the price that v_i quotes to v_j. $res\_pr_{v_i}$ is the reserved price of v_i for the requested service. $arrive(s_k)$ is the arrival time of service request s_k, while $start(s_k)$ is the latest start time of s_k. t means current round. $\delta(t)$ is the difference of v_j's quoting prices between the last round, $t-1$, and the current round, t, while $\Delta(t)$ is the difference of v_j's quoting prices between the initial round, 0, and the current round, t. In Equation 2, it can be seen that the quoting price is determined by the v_i's reserved price, $res\_pr_{v_i}$, the left time till the deadline (i.e., the pressure of deadline), $\frac{t-arrive(s_k)}{start(s_k)-arrive(s_k)}$, the number of temporary agreements that v_i has achieved (i.e., the probability of reaching an agreement), $|\mathcal{A}^{\mathcal{T}}(v_i)|$, and the concession that the negotiating partner v_j has made (i.e., the competition of the required resource), $\frac{\delta(t)}{\Delta(t)}$.

In Equation 3, pe is the basic penalty and exe_{v_j} means the total time steps, during which v_i needs to use v_j's service to execute the task. The actual penalty that v_i has to pay v_j depends on how long v_i has used v_j's service. For example, suppose that $pr = 100$ and $exe_{v_j} = 20$, so $pe = \frac{pr}{exe_{v_j}} = 5$. Now, v_i has used v_j's service for 12 time steps and v_i wants to cancel the agreement (i.e., stopping using v_j's service), then v_i has to pay v_j penalty 60, where $60 = pe \times 12$.

After receiving an offer o from v_i, in Line 4, v_j evaluates whether the offer o is acceptable. This evaluation is based on how much revenue v_j could get (Equation 4), where $cost$ is the cost that v_j spends to provide the requested service.

$$rv = pr - cost \tag{4}$$

If the revenue, rv, is larger than a threshold, v_j accepts quote o and a temporary agreement is achieved (Lines 5 and 6). Otherwise, v_j generates a counter-quote $o' = \langle pr', pe'\rangle$ (Line 9). The calculation of pr' is similar to Equation 2 except that provider v_j does not have a deadline for its service, while the calculation of pe' is the same as Equation 3. Then, Lines 10-12, if v_i accepts the counter-quote, a temporary agreement is achieved. Otherwise, the negotiation goes into the next round.

At the end of this stage, for each service request s_k, consumer v_i may have achieved a set of temporary agreements, where each temporary agreement is signed with an individual provider. Consumer v_i selects a provider v_j to provide service s_k for task execution. The temporary agreement between v_i and v_j now becomes a final agreement. Consumer v_i also selects several providers for backup and the temporary agreements between v_i and these backup providers become backup agreements. To achieve a backup agreement, v_i has to pay the corresponding backup provider a non-refundable deposit. The amount of deposit can be set as equal to the basic penalty. Finally, v_i cancels the rest of temporary agreements.

F. *Stage 4: task execution*

As described above, after Stage 3, consumer v_i has achieved a set of temporary agreements for each service request and v_i has to select one provider from the temporary agreements for each service request. In this stage, an Integer Programming (IP) based method [41], [23] is employed by consumer v_i to select one provider for each service request for task execution. For an IP problem, there are three inputs: a set of variables, a set of constraints and an objective function. Both the objective function and the constraints must be linear. The aim of IP is to maximise or minimise the objective function by adjusting the values of the variables under the given constraints. The output of an IP problem is the maximum (or minimum) value of the objective function and the values of variables along with that maximum (or minimum) value.

The problem of selecting a provider for each service request can be modeled as an IP problem as follows. We first introduce a binary indicator, x_{jk}, where $x_{jk} = 1$ if provider v_j is selected for service request s_k and $x_{jk} = 0$ otherwise. We also introduce another binary indicator, y_k, where $y_k = 1$ if service request s_k is critical and $y_k = 0$ otherwise. Each service request, s_k, has a start time, $start_k$. A provider, v_j, needs execution duration time, dur_{jk}, to execute service request s_k with price pr_{jk} and reliability $reli_{jk}$. The constraints are formally expressed as follows.

$$\sum_{1 \leq j \leq |\mathcal{A}^{\mathcal{T}}(v_i)|} x_{jk} = 1, \forall k \in [0, m] \quad (5)$$

Constraint 1 (Equation 5) means that for each service request s_k, consumer selects one and only one provider, v_j, to provide the service for task execution.

$$start_l - (dur_k + start_k) \leq 0, \forall s_k \rightarrow s_l \wedge k, l \in [0, m] \quad (6)$$

Constraint 2 (Inequality 6) indicates that if service request s_l is a direct successor of s_k, i.e., $s_k \rightarrow s_l$, service request s_l must be executed after the execution of s_k.

$$start_l = start_k, \forall s_k \leftrightarrow s_l \wedge k, l \in [0, m] \quad (7)$$

Constraint 3 (Equation 7) indicates that service requests s_l and s_k have to be executed concurrently.

$$\prod_{1 \leq k \leq m} reli_{jk}^{y_k} \geq D_1, 0 < D_1 < 1 \quad (8)$$

Constraint 4 (Inequality 8) indicates that the overall reliability of the combined service should be larger than or equal to a threshold D_1. This constraint is not linear but it can be converted to a linear one by using the logarithm function ln: $\sum_{1 \leq k \leq m} y_k \cdot ln(reli_{jk}) \geq ln(D_1)$.

$$\sum_{1 \leq k \leq m} dur_k \leq D_2 \quad (9)$$

Constraint 5 (Inequality 9) indicates that the overall duration of task execution should be less than or equal to a threshold D_2.

$$\sum_{1 \leq k \leq m} pr_{jk} \leq D_3 \quad (10)$$

Constraint 6 (Inequality 10) indicates that the overall price to execute the task should be less than or equal to a threshold D_3. The values of thresholds, D_1, D_2 and D_3, are associated with specific tasks.

The objective function is then given in the Max function (11) which implies that the overall reliability should be maximised while the overall execution duration time and the total price should be minimised. w_1, w_2 and w_3 are the weights of reliability, execution duration and price, respectively, where $0 < w_i < 1$ and $\sum_{1 \leq i \leq 3} w_i = 1$. The values of weights, w_1, w_2 and w_3, are associated with specific consumers.

$$Max(w_1 \sum_{k=1}^{m} y_k \cdot ln(reli_{jk}) - w_2 \sum_{k=1}^{m} dur_k - w_3 \sum_{k=1}^{m} pr_{jk}) \quad (11)$$

G. Stage 5: network evolution

After Stage 4, when the task execution is completed, consumer v_i may want to modify relationships with its acquaintances. v_i may remove less efficient acquaintances or may add efficient participants as its new acquaintances. Also, consumer v_i may buy extra services if v_i has often used them, or v_i may sell some of existing services if v_i or other participants have not used them for a long time. The aim of removing less efficient acquaintances is that v_i can save communication overhead in the future, because v_i does not need to communicate with these less efficient acquaintances any more. Similarly, the aim of adding efficient participants as new acquaintances is that in the future v_i can avoid brokers and directly communicate with these new acquaintances for service composition. Thus, communication overhead can be saved. The aim of buying/selling services is to save v_i's service usage/service management cost so as to improve v_i's overall utility.

The three problems, namely modifying relationships, buying services and selling services, can be modeled as a decision making problem, i.e., whether or not to do some thing based on some observation. The decision making process of v_i modifying relationship with v_j is described as follows. The decision making process of other problems is similar.

Suppose that v_j is one of v_i's acquaintances. Every time, when v_i's task execution is completed, v_i has two options, i.e., whether it should keep v_j as an acquaintance or not. Let p_0 be the probability of keeping v_j as an acquaintance and p_1 be the probability of removing v_j from v_i's acquaintance list. Also, let $Q(ob(t), 0)$ be the expected reward of keeping v_j as an acquaintance and $Q(ob(t), 1)$ be the expected reward of removing v_j from v_i's acquaintance list, where $ob(t)$ is v_i's observation to v_j till time step t. Initially, $p_0 = 1$, $p_1 = 0$, $Q(ob(t), 0) = 0$ and $Q(ob(t), 1) = 0$. Then, after a task execution, $t+1$, $Q(ob(t+1), 0)$ is updated using Equation 12, where $\mathcal{R}_0$ is immediate reward obtained by v_i for keeping v_j as an acquaintance during this task execution, and γ, $0 < \gamma < 1$, is a constant learning rate.

$$Q(ob(t+1), 0) = Q(ob(t), 0) + p_0 \cdot \gamma \cdot (\mathcal{R}_0 - Q(ob(t), 0)) \quad (12)$$

Here, immediate reward, $\mathcal{R}_0$, is determined based on 1) the number of services that v_j provides v_i and 2) the number of proper providers that v_j has found for v_i during this task execution. Then, after the calculation of $Q(ob(t+1), 0)$ using Equation 12, p_0 and p_1 are adjusted using Equation 13, where ϵ, $0 < \epsilon < 1$, is a greedy coefficient.

$$\begin{cases} p_0 = p_0 + \epsilon \cdot Q(ob(t+1), 0) \\ p_1 = 1 - p_0 \end{cases} \quad (13)$$

After the probability adjustment, function $Normalise()$ (**Algorithm 2**) is again used to project p_0 and p_1 to a valid probability distribution: $0 \leq p_0, p_1 \leq 1$ and $p_0 + p_1 = 1$. Finally, v_i selects an option based on the probability distribution, p_0 and p_1. If v_i selects 1, namely that v_i decides to remove v_j from its acquaintance list, the above process terminates.

For the two other problems, buying/selling services, the process is similar to the above process. If v_i decides to buy or sell a service, v_i can use the methods proposed in Stages 1, 2 and 3, where in Stage 1, v_i searches potential sellers/buyers; then in Stage 2, v_i uses the trust model to select trustworthy sellers/buyers; finally, in Stage 3, v_i negotiates with these candidates and buys/sells services from/to the candidates which can maximise v_i's utility.

As our approach is developed in a general environment, the application of it in a specific environment, e.g., sensor networks, is not studied in this paper. The application of our approach in specific environments needs corresponding modification. For example, if the approach is applied in sensor networks, as a sensor network is a resource-constrained environment, buying/selling services among sensors must be realised by using energy-efficient resource sharing techniques [42].

H. Discussion

In the proposed approach, it is assumed that for each service type, there is only one candidate service. Generally, for each service type, a provider may have different services which may have different property values, e.g., different response times and different prices. In this paper, as the approach consists of five stages, for the ease of presentation and understanding, it is assumed that for each service type, there is only one candidate service and each provider may or may not offer the candidate service. However, the prices of the candidate services provided by different providers may be different, because the prices of the candidate services are negotiable (refer to Section IV-E). Moreover, in the proposed approach, it is not assumed that the quality of the candidate services offered by different providers is identical. Thus, the negotiable prices imply that the quality of the candidate service offered by different providers may be different. Therefore, the proposed approach can accommodate multiple candidate services, although in the current version, the multiple candidate services are not taken into account. The evaluation of the proposed approach with multiple candidate services is one of our future studies.

V. Experiment and Analysis

In this section, the proposed integrated self-evolving service composition approach, named as *Self-Evo*, is evaluated[4] in comparison with three other related and representative approaches which are described as follows.

1) Mixed Integer Programming (*MIP*) [1]: MIP is an efficient and well-known approach for service composition in a centralised manner [23], [1]. Service composition problem can be mapped to an MIP problem by setting a set of constraints, defining a set of variables and establishing an objective function. The MIP problem is then solved by a central manager. In this experiment, the central manager is a simulated entity which is not a participant of the network. The central manager can directly communicate with any participants in the network. As the simulated network is a general network, specific communication medium is not considered.

2) Multi-Agent Reinforcement Learning (*MARL*) [12], [26]: MARL is a subdiscipline of multi-agent techniques. In MARL, each agent learns by trial-and-error interaction with a dynamic environment. Agents can autonomously adjust their behaviour based on the feedback from the environment so as to achieve better results in the future. Thus, using MARL, systems inherently have good self-adaptability. In existing MARL based service composition approaches, users are modeled as agents and services are modeled as actions, where in different states, users take different actions and obtain different rewards. Based on the rewards, users adjust the probability distribution over their actions, i.e., services, for service selection in the future. However, although agents can make decisions individually without central control, agents still need global information.

3) Service Capability Table based approach (*SCT*) [10]: SCT based approach does not need central control or global information. Each agent has a service capability table which records a list of acquainted agents and their service capabilities, which implies that each agent has limited knowledge of the environment. Consumer agents request services via broker agents which have two SCTs that record information about service provider agents and other broker agents. When a broker agent receives a request to carry out a service composition, the broker agent contracts services with proper provider agents in its SCT if possible, otherwise the broker agent subcontracts to other broker agents in its SCT to fulfill the unresolved service requirements.

In this experiment, the performance of the four approaches, i.e., our *Self-Evo* and three above-mentioned representative approaches, is evaluated by measuring three quantitative metrics: success rate, communication overhead and time consumption.

1) Success rate is the percentage of successful service composition. As success rate is a ratio between the number of successful service composition and the total number of attempted service composition, it does not have a unit.

2) Communication overhead is the average number of messages used for a successful service composition. The size of a message is neglected, as it is the same in all the four evaluated approaches in the experiment. Thus, communication overhead does not have a unit.

3) Time consumption [5] is the average time consumed for a successful service composition. The unit of time consumption is millisecond (ms).

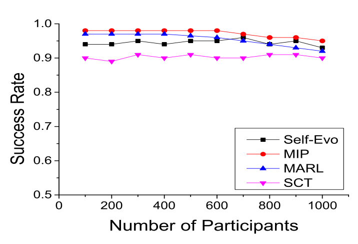
A. Experimental Setup

The experiment was conducted in both static and dynamic environments. The network structure is randomly generated, where every two participants are acquaintances of each other with a specific probability. The average number of neighbours of each agent is set to 8. [6] In the static environment, the number of participants in the environment is fixed, while in the dynamic environment, participants can join and leave the environment dynamically. In the static environment, the experiment was conducted in three scenarios. In the first scenario, it is measured that how the performance of the four approaches changes with increase of network size. In the second scenario, it is measured that how the performance of the approaches changes with the variation of task introduction probability. Task introduction probability means that at each time step, with which probability a task is introduced into the environment. In the third scenario, it is measured that how the performance of the approaches changes with the variation of service allocation probability. Service allocation probability means that with which probability a service is allocated to a participant. In the dynamic environment, in addition to the above three scenarios, there is the fourth scenario, where it is measured that how the performance of the approaches changes with the variation of the probability of participant leaving and joining. This experiment is simulated by using a general programming language, Java. Then, 'agent' is initially programmed as a class and then all agents are the objects of this class. Services are simply represented as integers and each agent is randomly allocated some of the services. A task is programmed as an array. Each element in the array is an integer which represents the required service. The number of services required by a task is randomly generated. Moreover, the experiment is run on an Intel i5-2450m 2.5GHz Windows

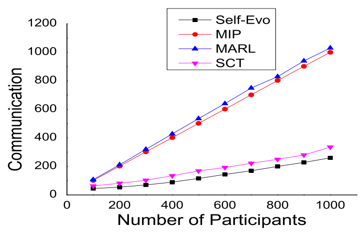
[4]Although target tracking in WSNs is used as an example in this paper, the proposed approach will not be evaluated in WSNs. This is because our approach is developed in a general environment and should not be limited in WSNs only. Instead, we evaluate our approach in a general environment to ensure that the experimental results are representative in most operating environments.

[5]In service computing, runtime has specific meaning: the running or execution of service-based systems. Thus, in the experiment, in order to avoid confusion, we use "time consumption" instead of "runtime".

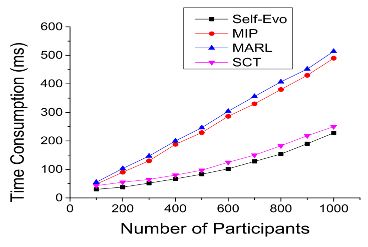
[6]Different average numbers of neighbours, e.g., 4, 6, 8, 10 and 12, have been attempted in the experiment. Those different average numbers of neighbours, however, do not affect the trend of the performance, i.e., relative performance, of the four approaches. We thus use the middle number, 8, in the experiment.



(a) Success rate

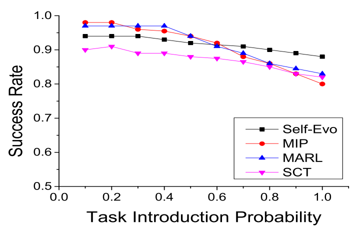


(b) Communication overhead

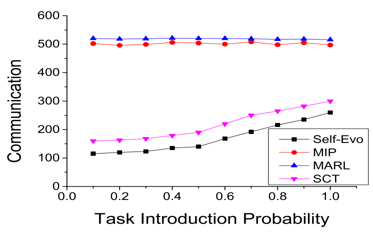


(c) Time consumption

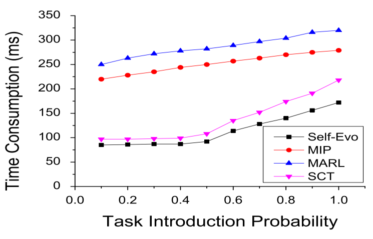
Fig. 3. Performance of the four approaches in different static network scales



(a) Success rate



(b) Communication overhead



(c) Time consumption

Fig. 4. Performance of the four approaches with different task introduction probabilities in a static network

10 PC with 10GB RAM. The experimental results are obtained by averaging 100 runs. Here, we use Java instead of a specific agent platform to implement the experiment, because 1) this paper mainly studies service composition instead of agent technologies which are used as development methods; 2) the aim of the experiment is to evaluate the effectiveness of the proposed approach; and 3) using specific agent platforms is certainly feasible for the implementation of agents but will not affect the effectiveness of the proposed approach. Therefore, specific agent platforms are not necessary in the experiment.

The values and meanings of the parameters used in the experiment are listed in Table I. These values were chosen experimentally to provide the best results.

TABLE I
PARAMETERS SETTING

Parameters	Values	Explanations
α, ζ, γ	0.2, 0.3, 0.2	Learning rates
ρ	0.7	Threshold for candidate selection
ϵ	0.1	Greedy coefficient

B. *Experimental Results: Static Environment*

1) The first scenario: Fig. 3 demonstrates the performance of the four approaches in different network scales. In this scenario, the task introduction probability is fixed at 0.3 and the service allocation probability is fixed at 0.5.

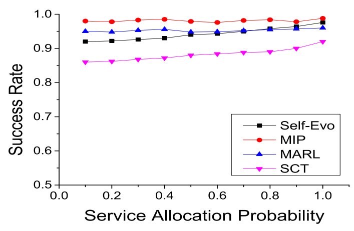
In Fig. 3(a), it can be seen that with the increase of network scale, the success rates achieved by *MIP* and *MARL* decrease gradually, while the success rates achieved by *SCT* and our *Self-Evo* keep relatively steady. This is because both *MIP* and *MARL* need information of all the participants in the network and when the number of participants in the network increases, the amount of required information also rises. Thus, the computation overhead under *MIP* and *MARL* becomes huge. Due to the limitation of processing capability, some tasks cannot be processed on time, so the success rate decreases. For *SCT* and *Self-Evo*, no global information is required and each participant requests services only from its acquaintances, so the increase of network scale does not have much impact on *SCT* and *Self-Evo*. It should be noted that the success rate achieved by our *Self-Evo* is higher than that achieved by *SCT*. This is because under *Self-Evo*, network structures evolve over time, where each participant keeps only useful acquaintances and services after evolution. Therefore, a service composition for a task can be completed faster. In other words, more service compositions can be completed in a period by using *Self-Evo* than using *SCT*.

In Figs. 3(b) and 3(c), it can be seen that with the increase of network scale, both the communication overhead and the time consumption under the four approaches rise. For *MIP* and *MARL*, as the two approaches need global information, the central manager in *MIP* and the focal consumer agent[7] in *MARL* has to request information from every participant in the network. Thus, as the number of participants in the network increases, both *MIP* and *MARL* need more communication and time to obtain the global information of the network. For *SCT* and *Self-Evo*, with the increase of the number of participants, the average number of acquaintances of each participant also increases, which then incurs the increase of the communication overhead and time consumption under *SCT* and *Self-Evo*. Again, as the network structure can be optimised under *Self-Evo*, *Self-Evo* uses less communication and time than *SCT*.

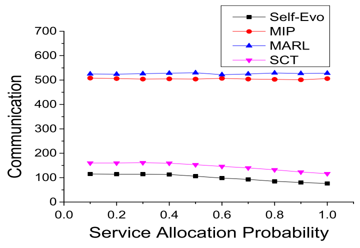
2) The second scenario: Fig. 4 displays the performance of the four approaches with different number of tasks in the network. In this scenario, the number of participants in the network is fixed at 500 and the service allocation probability is fixed at 0.5.

In Fig. 4(a), with the increase of task introduction probability, more and more tasks exist in the network. As each task has a deadline, some tasks may not be processed on time. Thus, the success rates achieved by all the four approaches

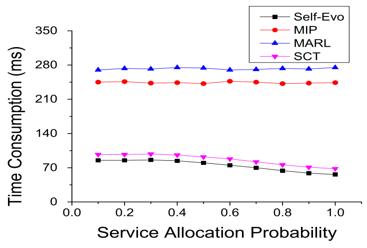
[7]Here, a focal consumer agent is a consumer agent which initialises a service composition.



(a) Success rate



(b) Communication overhead



(c) Time consumption

Fig. 5. Performance of the four approaches with different service allocation probabilities in a static network

decrease. It should be noted that when the task introduction probability is larger than 0.7, *Self-Evo* performs better than the other three approaches, while *MIP* performs the worst. This is because with the increase of the number of tasks, as in *MIP* all the processing is based on a single central manager, tasks will overwhelm the central manager given that the processing capability of the central manager is finite. In *Self-Evo*, as participants can adjust their behaviour with change of the environment, the pressure of a large number of tasks can be alleviated to some extent. For example, if a consumer has a number of tasks to do, it will attempt to buy proper services from other participants and it will establish acquaintance relationship with those participants who have the required services. Such adjust can relax the task pressure.

In Figs. 4(b) and 4(c), it can be seen that with the increase of the number of tasks, the communication overhead under *MIP* and *MARL* keeps relatively steady while the time consumption under *MIP* and *MARL* increases gradually. In *MIP* and *MARL*, the communication overhead to complete a service composition for a task is independent of the number of tasks, as only successfully completed tasks are counted in. To successfully complete a service composition for a task, the communication overhead for *MIP* and *MARL* is almost fixed, i.e., the central manager or focal consumer agent broadcasts a request message in the network, receives response messages and then calculates a solution. The time consumption under *MIP* and *MARL* increases gradually, because with the increase of the number of tasks, service compositions for some tasks may not be completed at once and have to stay in the waiting list for a while. Thus, the average time used to complete a service composition rises.

For *SCT* and *Self-Evo*, the communication overhead and time consumption rise with the increase of the number of tasks in the network. In *SCT* and *Self-Evo*, communication happens between service consumers, brokers and providers. When the number of tasks increases, available services are more and more difficult to obtain, which implies that consumers have to find and negotiate with more providers to achieve a service agreement. Thus, the communication overhead and time consumption to complete a service composition rise.

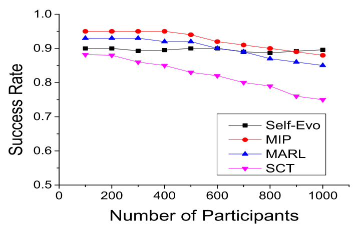
3) The third scenario: Fig. 5 displays the performance of the four approaches with different service allocation probabilities. In this scenario, the number of participants in the network is fixed at 500 and the task introduction probability is fixed at 0.3.

In Figs. 5(a), 5(b) and 5(c), with the increase of service allocation probability, the success rates achieved by *SCT* and *Self-Evo* increase gradually, while the communication and time used by *SCT* and *Self-Evo* decrease gradually. With the increase of service allocation probability, each participant possesses more services. Thus, each consumer can complete a service composition more quickly and easily, as it needs less search and negotiation for a service composition. For *MIP* and *MARL*, with the increase of service allocation probability, the success rates and communication overhead keep relatively steady, while the time consumption decreases gradually. The service composition under *MIP* and *MARL* is, to some extent[8], independent of the number of services that each participant has, as in *MIP* and *MARL*, the central manager and the focal consumer agent can utilise any available services in the environment no matter where the services are. Thus, the success rates and communication overhead keep almost unchanged. The decrease of time consumption is explained as follows. When each participant has few services, tasks have to wait for a relatively long time to be served. Thus, the average time consumption of each service composition is relatively long. This situation, however, is alleviated when each participant has more services, so the time consumption decreases with the increase of service allocation probability.

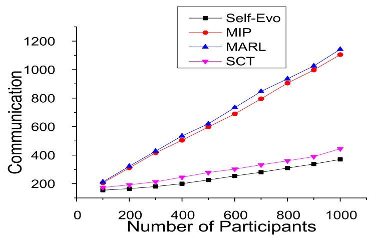
C. Experimental Results: Dynamic Environment

The experiment was also conducted in dynamic environments, where participants may leave or join the environment dynamically. In the first, second and third scenarios, it is set that after execution of every ten time steps, 5% existing participants leave the network and 5% new agents join the network with a specific probability. Figs. 6, 7 and 8 demonstrate a similar trend on performance variation in dynamic environments compared to static environments. In general, the performance of the four approaches are less efficient in dynamic environments than the performance in static environments. This is because for example, in dynamic environments, consumers may leave the environments before their tasks are completed. These tasks may then become unsuccessful tasks. Thus, success rates will decrease. Moreover, if providers leave the environments, consumers may have to find other providers, so the communication overhead and time consumption will increase. Moreover, our *Self-Evo* has to face the cold-start problem due to the use of the trust model. The cold-start problem means that when new agents join the network, their neighbours do not have information about their performance. Thus, the trust against these new agents has to be built up

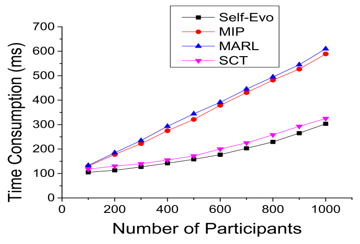
[8]When the task introduction probability is high, e.g., 0.7, the service composition under *MIP* and *MARL* is not independent of the number of services that each participant has. This is because when the number of tasks in the environment is large and each participant has only a few services, many tasks may not be completed on time due to the lack of enough services in the environment.



(a) Success rate

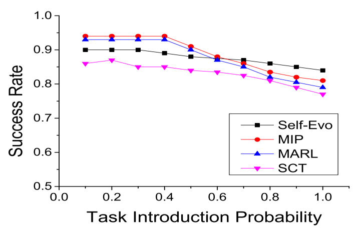


(b) Communication overhead

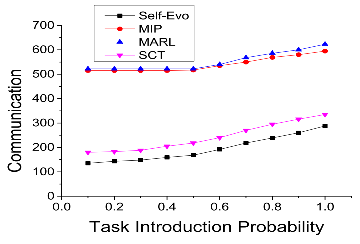


(c) Time consumption

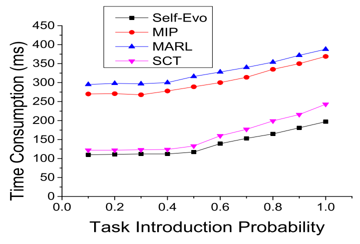
Fig. 6. Performance of the four approaches in different dynamic network scales



(a) Success rate

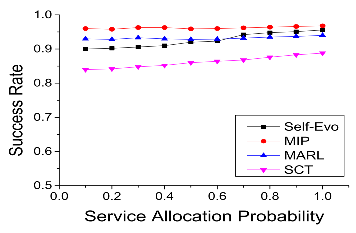


(b) Communication overhead

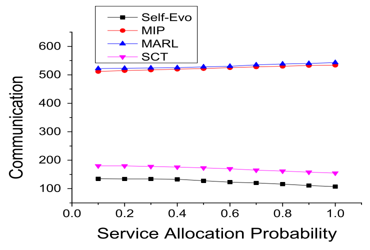


(c) Time consumption

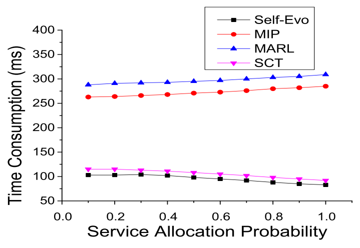
Fig. 7. Performance of the four approaches with different task introduction probabilities in a dynamic network



(a) Success rate

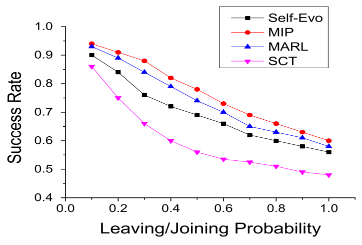


(b) Communication overhead

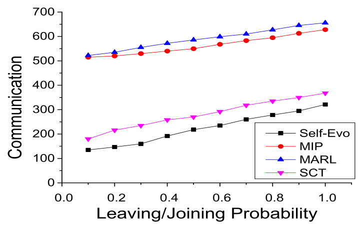


(c) Time consumption

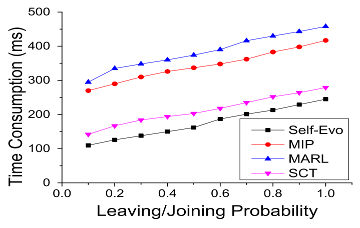
Fig. 8. Performance of the four approaches with different service allocation probabilities in a dynamic network



(a) Success rate



(b) Communication overhead



(c) Time consumption

Fig. 9. Performance of the four approaches with different leaving/joining probabilities in a dynamic network

anew. Such cold-start problem has a negative impact on the performance of our approach, because along with the cold-start problem, the trust model in *Self-Evo* cannot effectively help participants select efficient candidates.

Comparing Fig. 6(a) with Fig. 3(a), with the increase of network scale, the success rate of *SCT* decreases in the dynamic environment, while the success rate of *SCT* keeps relatively steady in the static environment. In *SCT*, service composition is carried out via participants' neighbours. In the dynamic environment, a participant's neighbours may leave the environment before a task is completed. Thus, the success rate is affected in a negative way. This situation becomes worse in a large scale network, as the more participants in the network, the more they may leave (recalling above that 5% existing participants leave the network with a specific probability). When a participant loses most of its neighbours, it is difficult for the participant to complete any complex tasks. Although new agents may join the network, they are randomly scattered in the network and do not have to be the neighbours of that participant. Similar to *SCT*, in our *Self-Evo*, service composition is also conducted through participants' neighbours. However, in our *Self-Evo*, when a participant loses its neighbours, the participant can establish new neighbourship with other participants and also the participant can buy required services from other participants. Thus, the negative impact of the dynamism on *Self-Evo* can be alleviated to some extent via self-evolution. As *MIP* and *MARL* utilise global information, the dynamism does not have much impact on them.

Comparing Fig. 7(b) with Fig. 4(b), in the dynamic environment, when the number of tasks increases, the communication overhead of *MIP* and *MARL* also increases, while in the static environment, when the number of tasks increases, the communication overhead of *MIP* and *MARL* keeps relatively steady. As described above, if providers leave the environment, consumers may have to find other providers, which incurs extra communication and time consumption. This situation will be worse when the number of tasks increases. This is because when the number of tasks increases, a provider may serve more tasks simultaneously. If this provider leaves, more tasks will be affected and thus, more communication and time are needed.

In the exclusive scenario for the dynamic environment, Fig. 9 demonstrates the performance change of the four approaches with the variation of the probability of participant leaving and joining. For the four approaches, with the increase of the probability of participant leaving and joining, the success rates decrease while the communication overhead and time consumption increase. This means that the increase of dynamism level of the environment has negative impact on the performance of the four approaches. Since the leaving of participants implies the failure of participants, consumers or the central manager have to spend extra communication and time to counter such situation, which is certainly harmful for the overall performance. The joining of new participants can also offset such situation to some extent, but the usefulness of the new participants is limited. For *MIP* and *MARL*, if providers leave the environment, the joining of new participants can increase the probability of completing the service compositions which are left by those leaving providers. However, the joining of new participants cannot avoid the communication and time used by the central manager or focal consumers to find new providers. For *SCT* and *Self-Evo*, the usefulness of the new participants is further limited. In *SCT* and *Self-Evo*, consumers can only ask acquaintances for help. However, these new participants are randomly spread in the network, so the consumers, which need new providers for service composition, may not find these new participants.

D. Discussion

In overall terms, according to the experimental results, the proposed approach, *Self-Evo*, can achieve about 96% of the success rates of the centralised approach *MIP* and the global information based approach *MARL*, but uses about 75% less communication and 65% less time than them due to the decentralised features of *Self-Evo*. This achievement is significant considering that a large amount of communication and time are saved. Such huge communication and time consumption saving is very important in some real world systems, where the energy of each participant is limited[9] and data has to be processed in a timely manner. The proposed approach, *Self-Evo*, may be computation-intensive, because computation is carried out in each of the five stages of the approach. Such intensive computation may imply a high time consumption in some cases. However, in *Self-Evo*, the computation is carried out by each participant concurrently in a distributed manner. Thus, the time consumption is not excessive. Moreover, the proposed approach does not require global information. Thus, the time for information acquisition can be saved. Therefore, the time consumption of *Self-Evo* stays at a relatively low level compared to the other approaches.

[9]The proposed approach is developed in a general environment rather than for specific systems or networks. Thus, we did not evaluate the energy consumption in the experiment, as energy consumption needs to be evaluated within specific networks, e.g., WSNs or mobile ad hoc networks. We, however, have evaluated the communication overhead which is an indicator of energy consumption. As communication usually consumes energy, especially in WSNs, low communication overhead implies low energy consumption.

Moreover, *Self-Evo* is better than *SCT* in various circumstances. This is mainly because *Self-Evo* allows participants to evolve over time, where only useful acquaintances and services are kept by each participant. Such evolution can significantly reduce communication and time used to find proper providers for service composition.

VI. Conclusion and Future Work

This paper proposes an agent-based integrated self-evolving service composition approach. This approach is decentralised and self-evolvable. Also, unlike most existing approaches which study only some of the five stages of service composition, the proposed approach consists of five stages of service composition: service discovery, candidate selection, service negotiation, task execution and self-evolution. Such joint optimisation can make the proposed approach more efficient than those which optimise only some stages of service composition.

The proposed approach studies multiple stages of service composition. Each stage has attracted research efforts. Compared to those research efforts, the methods used in the proposed approach for each stage are effective though somewhat simple. Thus, we will focus on the improvement and enhancement of these methods so as to achieve better performance. First, as described in previous sections, there are two limitations in our approach: 1) service presentation is simplified, where for each service type, there is only one candidate service; 2) the approach is relatively computation-intensive in some systems, especially where the computation resource is scarce. These two limitations need to be solved in the future before the proposed approach can be applied in specific systems. Then, as stated in Section V-C, the trust model used in our approach has the cold-start problem which is a common issue of recommender systems in a network environment, especially in a dynamic network environment. Currently, there have been a number of algorithms developed to handle the cold-start problem, which have been summarised in [43]. The development of a new algorithm to address the cold-start issue, which is suitable for the service composition problem, is one of our future studies. Moreover, in the proposed approach, we do not consider specific control logic or workflows, because workflows in service composition depend on specific application domains [5], [20] while our work is carried out in general environments. Thus, when our approach is applied to specific systems, the workflows have to be considered as shown in Section III. Finally, energy consumption was not evaluated in this paper, as it needs to be evaluated within specific networks. The detailed evaluation of energy consumption, however, has to be done when the proposed approach is applied to specific networks in the future.

References

[1] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, pp. 369–384, 2007.

[2] S. Kalasapur, M. Kumar, and B. A. Shirazi, "Dynamic service composition in pervasive computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, pp. 907–918, 2007.

[3] K. M. Sim, "Agent-based cloud computing," *IEEE Transactions on Services Computing*, vol. 5, pp. 564–577, 2012.

[4] P. Papadopoulos, H. Tianfield, D. Moffat, and P. Barrie, "Decentralized multi-agent service composition," *Multiagent and Grid Systems*, vol. 9, pp. 45–100, 2013.

[5] N. Chen and S. Clarke, "A dynamic service composition model for adaptive systems in moblie computing environments," in *Proc. of ICSOC'14*, 2014, pp. 93–107.

[6] S. C. Geyik, B. K. Szymanski, and P. Zerfos, "Robust dynamic service composition in sensor networks," *IEEE Transactions on Services Computing*, vol. 6, pp. 560–572, 2013.

[7] Q. Z. Sheng and et al., "Web services composition: A decades overview," *Information Sciences*, vol. 280, pp. 218–238, 2014.

[8] P. Bartalos and M. Bielikova, "Automatic dynamic web service composition: A survey and problem formalization," *Computing and Informatics*, vol. 30, pp. 793–827, 2011.

[9] A. Jula, E. Sundararajan, and Z. Othman, "Cloud computing service composition: A systematic literature review," *Expert Systems with Applications*, vol. 41, pp. 3809–3824, 2014.

[10] J. O. Gutierrez-Garcia and K. M. Sim, "Agent-based cloud service composition," *Applied Intelligence*, vol. 38, pp. 436–464, 2013.

[11] A. Moustafa, M. Zhang, and Q. Bai, "Trustworthy stigmergic service composition and adaptation in decentralized environments," *IEEE Transactions on Services Computing*, p. 10.1109/TSC.2014.2298873, 2014.

[12] A. Moustafa and M. Zhang, "Multi-objective service composition using reinforcement learning," in *Proc. of ICSOC'13*, 2013, pp. 298–312.

[13] W. Chen and I. Paik, "Toward better quality of service composition based on a global social service network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 1466–1476, 2015.

[14] H. Kurdi, A. Al-Anazi, C. Campbell, and A. Al-Faries, "A combinatorial optimization algorithm for multiple cloud service composition," *Computers and Electrical Engineering*, vol. 42, pp. 107–113, 2015.

[15] D. Ye, M. Zhang, and D. Sutanto, "Self-organization in an agent network: A mechanism and a potential application," *Decision Support Systems*, vol. 53, pp. 406–417, 2012.

[16] D. Ye, M. Zhang, and D. Sutanto, "Cloning, resource exchange, and relation adaptation: An integrative self-organisation mechanism in a distributed agent network," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, pp. 887–897, 2014.

[17] Y. Jiang and J. C. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2743–2759, 2014.

[18] H. Park and M. van der Schaar, "Evolution of resource reciprocation strategies in p2p networks," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1205–1218, 2010.

[19] V. Prinz and et al., "Adaptive and fault-tolerant service composition in peer-to-peer systems," in *Proc. of DAIS*, 2008, pp. 30–43.

[20] C. Groba and S. Clarke, "Opportunistic service composition in dynamic ad hoc environments," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 642–653, 2014.

[21] V. Cardellini and et al., "A decentralized approach to network-aware service composition," in *Proc. of ESOCC*, 34-48, pp. 34–48.

[22] R. Kota, N. Gibbins, and N. Jennings, "Decentralized approaches for self-adaptation in agent organizations," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 7, no. 1, pp. 1–28, 2012.

[23] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, pp. 311–327, 2004.

[24] F. Wagner and et al., "Towards robust service compositions in the context of functionally diverse services," in *Proc. of 21th WWW*, 2012, pp. 969–978.

[25] I. Trummer, B. Faltings, and W. Binder, "Multi-objective quality-driven service selectiona fully polynomial time approximation scheme," *IEEE Transactions on Software Engineering*, vol. 40, pp. 167–191, 2014.

[26] H. Wang, X. Chen, Q. Wu, and Z. Zheng, "Integrating on-policy reinforcement learning with multi-agent techniques for adaptive service composition," in *Proc. of ICSOC'14*, 2014, pp. 154–168.

[27] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 29, pp. 1104–1113, 1980.

[28] W. Sun, Z. Yang, X. Zhang, and Y. Liu, "Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1448–1459, 2014.

[29] T. He and et al., "Achieving real-time target tracking using wireless sensor networks," in *Proceedings of the Twelfth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS06)*, 2006.

[30] H. Tsai, C. Chu, and T. Chen, "Mobile object tracking in wireless sensor networks," *Computer Communications*, vol. 30, pp. 1811–1825, 2007.

[31] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[32] G. Wei, Y. Ling, B. Guo, B. Xiao, and A. V. Vasilakos, "Prediction-based data aggregation in wireless sensor networks: Combining grey model and kalman filter," *Computer Communications*, vol. 34, pp. 793–802, 2011.

[33] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications*, vol. 7, pp. 16–27, 2000.

[34] J. H. Naegl, S. A. Gossage, N. Testi, M. O. Vahle, and J. H. Maestas, "Building networks for the wide and local areas using asynchronous transfer mode switches and synchronous optical network technology," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, pp. 662–672, 1995.

[35] I. Satoh, "Building reusable mobile agents for network management," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 33, no. 4, pp. 350–357, 2003.

[36] R. Dai, J. Maximoff, and M. Mesbahi, "Optimal trajectory generation for establishing connectivity in proximity networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, pp. 1968–1981, 2013.

[37] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, pp. 115–152, 1995.

[38] F. Smarandache and J. Dezert, *Advances and Applications of DSmT for information Fusion*. America Research, 2004.

[39] A. Rubinstein, "Prefect equilibrium in a bargaining model," *Econometrica*, vol. 50, no. 1, pp. 97–109, 1982.

[40] B. An, V. Lesser, and K. M. Sim, "Strategic agents for multi-resource negotiation," *Autonomous Agents and Multi-Agent Systems*, vol. 23, pp. 114–153, 2011.

[41] H. Karloff, *Linear Programming*. Birkhauser, 1991.

[42] B. Fateh, "Resource management algorithms for real-time wireless sensor networks with applications in cyber-physical systems," Ph.D. dissertation, Iowa State University, 2013.

[43] L. H. Son, "Dealing with the new user cold-start problem in recommender systems: A comparative review," *Information Systems*, p. doi:10.1016/j.is.2014.10.001, 2014.

Dayong Ye received his MSc and PhD degrees both from University of Wollongong, Australia, in 2009 and 2013, respectively. Now, he is a research fellow of computer science at Swinburne University of Technology, Australia. His research interests focus on service-oriented computing, self-organisation and multi-agent systems.

Qiang He received the his first Ph. D. degree in information and communication technology from Swinburne University of Technology (SUT), Australia, in 2009. He is now a lecturer at Swinburne University of Technology. His research interests include software engineering, cloud computing and services computing.

Yanchun Wang received his M.Eng. degree in computer science from Beihang University, Beijing, China, in 2006. He is currently a Ph.D. student at Swinburne University of Technology, Australia. His research interests include service computing and cloud computing.

Yun Yang received the PhD degree from the University of Queensland, Australia in 1992. He is a full professor at Swinburne University of Technology. His research interests include software technologies, cloud computing, workflow systems and service-oriented computing.