SPSC: Efficient Composition of Semantic Services in Unstructured P2P Networks

Xiaoqi Cao(✉), Patrick Kapahnke, and Matthias Klusch

German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
{Xiaoqi.Cao,Patrick.Kapahnke,Klusch}@dfki.de

Abstract. The problem of automated semantic peer-to-peer (P2P) service composition has been addressed in cross-disciplinary research of semantic web and P2P computing. Solutions for semantic web service composition in structured P2P networks benefit from the underlying distributed global index but at the cost of network traffic overhead for its maintenance. Current solutions to service composition in unstructured P2P networks with selective flooding can be more robust against changes but suffer from redundant messaging, lack of efficient semantics-empowered search heuristics and proven soundness. In this paper, we present a novel approach, called SPSC, for efficient semantic service composition planning in unstructured P2P networks. SPSC peers conduct a guarded heuristics-based composition to jointly plan complex workflows of semantic services in OWL-S. The semantic service query branching method based on local observations by peers about the semantic overlay alleviates the problem of reaching dead-ends in the not fully observable and heuristically pruned search space. We theoretically prove that the SPSC approach is sound and provide a lower bound of its completeness. Finally, our experimental evaluation shows that SPSC achieves high cumulative recall with relatively low traffic overhead.

Keywords: Semantic services · Workflow composition · P2P computing

1 Introduction

In the past decade, the challenge of automated centralized and decentralized composition of semantic web services in OWL-S, SAWSDL or WSML[1] has attracted considerable interest and development of various solutions in the semantic web and P2P community [21]. In fact, there are quite sophisticated AI planning based tools for centralized composition of semantic services such as OWLS-Xplan [14] for OWL-S services [19]. Unlike web service composition, the automated composition of semantic web services by use of AI planning techniques is inherently supported by their formally grounded semantic descriptions. However, these semantic service composition planners cannot be used for a distributed composition of semantic services for collaborative applications in P2P settings. In these cases, any service composition approach has to cope with the

[1] For an introduction to semantic web services, we refer to, for example, [16].

DOI: 10.1007/978-3-319-18818-8_28

lack of a global service directory or dynamic changes of the set of service prosumers and the availability of semantic services to be found and composed for jointly accomplishing a given task.

For example, approaches to semantic web service composition in structured or hybrid P2P networks such as [9,20,24] benefit from a distributed, semantics-empowered index, but at the cost of traffic overhead for its maintenance in dynamic environments. On the other hand, current solutions for semantic service composition in unstructured P2P networks can be more robust against changes but suffer from redundant messaging, lack of efficient semantics-empowered search heuristics and proven soundness of the distributed composition by the peers. There are a few solutions for this problem. For example, PM4SWS [10,11] applies classical flooding which causes heavy network traffic for on-line query answering. Relying on state transition gossiping and query/network status analysis, SCComp [6–8] enables selective flooding, but still has the risk of one peer receiving duplicated messages with the same sub-goal. AntAgt [3,4,23] yields less network traffic by applying a walker-based query routing strategy, but suffers from its dependence on user specified query plan templates. It does not perform fully automated service composition. Besides, none of the current approaches also takes non-functional factors such as quality of service (QoS) and composition plan length into account for the automated semantic services composition.

To this end, we present SPSC (Semantic P2P Service Composition Planning) for automated and efficient QoS-aware composition of OWL-S services in unstructured P2P networks. The joint generation of complex service workflows by SPSC peers basically relies on (a) the local matching of the semantic input/output/preconditions/effects (IOPE) of OWL-S services with variable bindings, and (b) the memorization of potentially useful services. As a result, SPSC peers jointly explore a heuristically pruned search space using a walker-based query branching strategy, which mitigates the risk of failure due to dead-ends. SPSC is robust against network and service dynamics. In contrast to other approaches to the same problem, we also theoretically prove that SPSC is sound and has a reasonable lower bound of completeness with respect to the solution existence. Finally, our preliminary experimental evaluation revealed that SPSC achieves high cumulative recall with low traffic overhead.

In Sect. 2, we provide preliminaries required to understand the SPSC approach which is detailed in Sect. 3. We then analyze the completeness of SPSC and prove its soundness in Sect. 4, followed by experimental results in Sect. 5. A discussion of related work is in Sect. 6 before we conclude in Sect. 7.

2 Preliminaries

In unstructured P2P networks, peers have no global view on network topology or services provided by other peers. A peer p maintains its limited domain knowledge in its local knowledge base, including an OWL ontology O_p and a set of predicates A_p, based on a shared primitive term vocabulary V. Each $\alpha \in A_p$ is the first order logic interpretation of a concept or property in O_p [1]. Each peer can provide atomic OWL-S services. Besides input (I), output (O), precondition (P) and effect (E), each service has its provider peer id *pid* and a QoS value $qos \in [0, 1]$, indicating the overall service availability. Each IO parameter contains a variable $?x$ and its concept type $X \in O_p$. P/E is a CNF formula over predicates in A_p and IO variables. Denote $\mathcal{S}_p$ the set of services known by p.

Example 1: Service $S9$ (cf. Fig. 1) represents an industrial production process, which consumes some $Material\ ?m$ and $Softener\ ?so$ to produce $Product\ ?pro$. Its precondition $P_9 = tempLargerThan\ (?m, 60) \wedge qualityNotBad(?m) \wedge speedEq(?so, 5)$, requires the temperature of $?m$ to be larger than 60, an adequate quality of $?m$ and the softener $?so$ to be added at speed 5. The effect $E_9 = shaped(?pro)$ ensures that, after the execution, $?pro$ is shaped. ■

In context of SPSC, a request R is defined analogously to a service, containing IOPE, but without QoS and *pid*. We assume that a request can typically not be solved by one atomic service, but a composition, i.e. a *workflow*. Such a workflow includes parameter bindings to make up data flows.

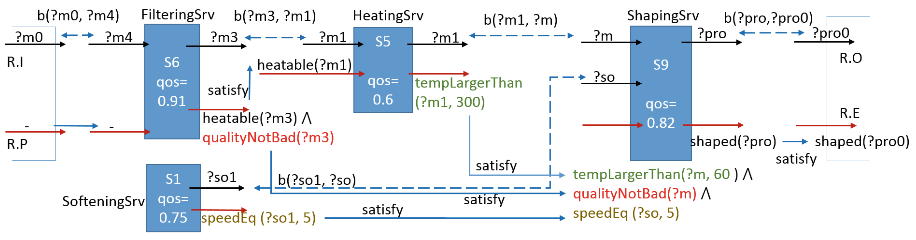


Fig. 1. An example workflow with parameter bindings (Color figure online).

Definition 1: *Service parameter binding* $b(?x, ?y)$.
A binding $b(?x, ?y)$ of service parameters $?x$ and $?y$ of services S and respectively S' is a tuple $\langle ?x, ?y, \varphi \rangle$ where φ is a substitution $\{?x \mapsto ?z, ?y \mapsto ?z\}$. ■

That is, if an output $?x$ of $S.O$ is bound to an input $?y$ of $S'.I$, the data of $?x$ can be transmitted to $?y$ and used by S' as input, which is modeled by introducing the common substitute $?z$.

Definition 2: *Workflow* wf.
A workflow is an orchestration of semantic services constructed to fulfill a request R. It consists of a set of services and a set of parameter bindings defining the data flow among them. Start and end of wf are defined according to R. The side starting with $R.I$ and $R.P$ (ending with $R.O$ and $R.E$) is called the left side $L(wf)$ (right side $R(wf)$) of wf. wf is *correct* wrt. R (satisfies R), iff:

(i) All inputs of services $S \in wf$ are bound to outputs of other services in wf or $R.I$.
(ii) The overall IO signature of wf plugs into request R:

(a) $\forall S \in wf, ?x : X \in S.I : \exists S' \in (wf \backslash \{S\}) \cup \{R\}, ?x' : X' \in S'.I : X' \sqsubseteq X$;
(b) $\forall ?y' : Y' \in R.O : \exists S \in wf, ?y : Y \in S.O : Y \sqsubseteq Y'$.

(iii) With parameter bindings, no conflicting literals is in $(\bigcup_{S \text{ in } wf} S.E) \cup R.P$;
(iv) With parameter bindings, preconditions of all services in wf are satisfied: $\forall S \in wf : \exists \mathbb{S} \subseteq wf : (\bigwedge_{S' \in \mathbb{S}} S'.E) \wedge R.P \implies S.P$;
(v) $R.E$ can be implied by wf: $\bigwedge_{S \text{ in } wf} S.E \implies R.E$. ■

Example 2: Figure 1 shows a correct workflow wf wrt. (satisfying) R: $R.I = \{?m0 : Material\}$, $R.O = \{?pro0 : Product\}$, $R.P = true$ and $R.E = shaped(?pro0)$. R asks for some service that produces a shaped product using its source material. IO (PE) are illustrated with black (red) arrows. Parameter bindings (implications) are shown as blue dashed (solid) arrows. In brief, wf specifies the following procedure: a source material $?m0$ is applied to a filtering process to assure some quality requirements. Then it is heated and shaped with additional softener into a product. wf is correct wrt. R because: (i) each input of any service in wf has been bound to an output of another service in wf. E.g. the binding between $?m4$ and $?m0$; (ii) wf plugs into R, because the type $Material$ of $?m0$ is equal to the type of $?m4$, and the type $Product$ of $?pro$ in $S9$ is equal to the type of $?pro0$; (iii) there are no conflicting literals in $(\bigcup_{S \text{ in } wf} S.E) \cup R.P$, given the bindings; (iv) all preconditions in wf hold: $S9.P$ is implied by $S6.E$, $S5.E$ and $S1.E$ with $b(?m3, ?m1)$, $b(?m1, ?m)$ and $b(?so1, ?so)$; $S5.P$ is implied by $S6.E$ with $b(?m3, ?m1)$. (v) $R.E$ is implied by $S9.E$ with $b(?pro, ?pro0)$. ■

While the former definitions focus on the workflows and its composition, the remainder of the section will elaborate on finding such workflows in unstructured P2P networks under assumptions stated above.

Definition 3: *Distributed stateless semantic service composition problem.*
The distributed stateless semantic service composition problem is a tuple $\langle \mathcal{N}, \mathcal{S}, R, wf \rangle$. Given request R, the goal is to construct a correct workflow wf satisfying R. wf is collaboratively composed by peers in an unstructured P2P network $\mathcal{N}$ based on their services $\mathcal{S}$. ■

In SPSC, a request R to the network is delegated to a walker-based query. Besides R, a query contains auxiliary fields required by the proposed algorithm.

Definition 4: *Semantic query (abbr. query) q for a request R.*
$q = \langle R, path, psug, TTL, wf, Tb, h \rangle$, where R is the request; $path$ is a sequence of peer IDs that q has traversed; $psug$ is a path suggestion for this query; TTL is the time-to-live value of this query; wf is the workflow (initialized as empty) answering to R; Tb is the memo table (initialized as empty), which will be used by the memorization strategy (cf. Sect. 3.2); h ($h \in [0, 1]$) is the current guard value of wf (initialized as 0). ■

3 Distributed Semantic Service Composition

SPSC mainly builds on three aspects: local observations of peers, guarded composition and query routing.

Local Observations of Peers: In SPSC, any peer p is allowed to observe the entire content of a received query q while it is backtracking. Any unknown service in $q.wf$ is added to $\mathcal{S}_p$ for later use. Once knowing about (not providing) S, p is called a *service signature maintainer of* S. Besides, p updates its local view on the network topology based on the observed query path and path suggestion.

Guarded Composition: A query q in SPSC is an epidemic walker with TTL limitation. A workflow wf is built collaboratively by peers on the query path using a bidirectional chaining approach. On receiving q on its forward journey, a peer p executes the local composition process (cf. Algorithm 1) based on its local knowledge about services. Once wf is correct or TTL=0, p makes q backtrack.

For composition, p locally checks whether each $S \in \mathcal{S}_p$ can be chained to the left or right side of wf. A chaining score (cf. Sect. 3.1) is computed to measure the chaining quality. If it is larger than a threshold (cf. Sect. 3.2), p considers S to be potentially useful and temporarily adds it to wf, yielding a new workflow wf'. In order to protect wf from arbitrary augmentation, a guard value h' wrt. wf' is computed. If h' is larger than the original guard value recorded in q, S is treated as a useful service wrt. wf, and wf' will be fixed. Then, p replaces $q.h$ with h'. In cases that S can **(a)** be chained but without incrementing the guard value, or **(b)** not be chained into a workflow at all, p applies memorization strategy for carrying the potentially useful service in q. For **(a)**, p adds S into the memo table of q; for **(b)**, p adds S into the memo table with a rate r_m (cf. Eq. 2). If an alternative service S' (cf. Sect. 3.1) of a candidate service $S \in wf$ is found, p issues a sub-query with a new workflow wf^* using S' instead of S.

Query Routing: p routes q, after the local composition process. Instead of an immediate neighbor, p suggests a path for routing q, based on its local knowledge. The suggested path traverses multiple key peers and its total inverse importance score per traffic cost is minimized, under TTL limit (cf. Eq. 3). Once formed, such a path is set to $q.psug$ (cf. Def.4) and q is routed to the first peer on $q.psug$.

3.1 Chaining Between Two Services

On receiving a query, each peer checks whether a known service can be chained to the left or right side of the current workflow. In particular, to chain service S' after S, the following has to be considered: (i) to what extent can the variables of $S.O$ be accepted by $S'.I$, and (ii) to what extent can the effect $S.E$ imply $S'.P$. For this, the peer computes chaining scores $ch_{IO}(S,S')$ at IO and $ch_{PE}(S,S')$ at PE levels. On top of this, the overall score $ch(S,S')$ is computed:

$$\begin{aligned} ch(S,S') &= \tfrac{1}{2}(ch_{IO}(S,S') + ch_{PE}(S,S')) \cdot df(ch_{IO}(S,S'), ch_{PE}(S,S')); \\ df(t,t') &= min\{\tfrac{t}{t'}, \tfrac{t'}{t}\}, t, t' \in (0,1]. \end{aligned} \tag{1}$$

This valuation considers IO and PE equally, while it further adjusts the overall outcome by including a difference factor $df(\cdot,\cdot)$. This ensures that low quality results with large discrepancy between $ch_{IO}(S,S')$ and $ch_{PE}(S,S')$ are downgraded and possibly filtered out later. To compute $ch(S,S')$, $ch_{IO}(S,S')$ is considered first.

This step determines which parameter in $S.O$ can be used by which parameter in $S'.I$ and yields a set of parameter bindings. By applying the substitutions $\{?z \mapsto ?x,\ ?z \mapsto ?y\}$ of bound parameters $?x \in S.O$ and $?y \in S'.I$, the $S.E$ and $S'.P$ formulas are adapted, in order to compute $ch_{PE}(S, S')$ afterwards. SPSC concerns IO before PE, because the latter with all possible parameter bindings would introduce a large computational overhead.

IO Chaining Score. For each concept Y of each variable $?y \in S'.I$, this process tries to find its best subsumee from the set of variable/concept pairs $(?x : X)$ of $S.O$. Such a subsumee yields the largest concept subsumption similarity score (such as [17]), compared to the others in $S.O$ and additionally exceeds a binding threshold $\beta \in (0, 1]$. Once found, a binding is created. Given the bindings, $ch_{IO}(S, S') = \frac{|M|}{|S'.I|}$, where $M \subseteq S'.I$ is the set of bound parameters in $S'.I$.

PE Chaining Score. The first step is to apply the substitutions of the bindings above to $S.E$ and $S'.P$. Subsequently, the implication $S.E \rightarrow cl$ is checked using θ-subsumption [18], for each clause cl in $S'.P$ that does not contain unbound variables. Let SC be the set of satisfied clauses in $S'.P$, $|S'.P|$ the total number of clauses $S'.P$: $ch_{PE}(S, S') = \frac{|SC|}{|S'.P|}$. If there exists a contradiction in $(\bigcup_{S \text{ in } wf} S.E) \cup R.P$ given the variable substitutions, $ch_{PE}(S, S')$ is set to 0.

Algorithm 1. *queryProcess*(q). **Input**: query q; **Output**: void.

```
1:  if $q$ is being forwarded
2:      $q.TTL \leftarrow q.TTL - 1$;
3:      for each $S \in \mathcal{S}_p$ do {
4:          $ch \leftarrow$ bidirectional chaining $S$ into $q.wf$;
5:          compute the new guard value $h'$;
6:          if $h' > q.h$
7:              if the workflow of $q$ is correct, break;    else update $q.Tb$;
8:              $q.h \leftarrow h'$; goto line 3;
9:          else if ($ch > \theta$ and $h' \leq q.h$) add $S$ into memo table $q.Tb$; goto line 3;
10:         else add $S$ into $q.Tb$ with the rate $r_m(S)$; }
11:     if $q.wf$ is correct or $q.TTL = 0$, make $q$ backtrack;
12:     else, make a path suggestion for $q$ and route $q$;
13:     if $q.wf$ is not correct and $q.TTL > 0$
14:         for each candidate service $S$ in $q.wf$ do {
15:             $\mathcal{S}_{pred} \leftarrow findAlternativePredecessorServices(S,\ wf)$;
16:             for each $S' \in \mathcal{S}_{pred}$ do {
17:                 $q' \leftarrow createSubQuery(q,\ \{S'\})$;  if $q'.wf$ is new, $queryProcess(q')$;}
18:             $\mathcal{S}_{succ} \leftarrow findAlternativeSuccessorServices(S,\ wf)$;
19:             for each subset $\mathcal{S}'_{succ} \in 2^{(\mathcal{S}_{succ} \cup S)} \backslash \{S\}$ do {
20:               $q' \leftarrow createSubQuery(q,\ \mathcal{S}'_{succ})$;  if $q'.wf$ is new, $queryProcess(q')$;} }
21:     endif
22: else $q$ is backtracking, update the local observation;
23:     if $p$ is not the requester peer of $q$, force $q$ to backtrack;
```

3.2 Semantic Composition of Services

Guarded Composition: Each query q in SPSC is a TTL-restricted epidemic walker starting from the requester peer. The workflow is collaboratively constructed by peers on the query path by means of bidirectional chaining. On receiving q, each peer p executes $queryProcess(q)$ (cf. Algorithm 1). Workflow constructions takes places while q is being forwarded. For each $S \in \mathcal{S}_p$, p considers to chain S to both $L(wf)$ and $R(wf)$ (line 4). For this, it computes $ch(L(wf), S)$ and $ch(S, R(wf))$, where the output of $L(wf)$ (input of $R(wf)$) contains all unbound outputs (inputs) of services currently in $L(wf)$ ($R(wf)$). If either $ch(L(wf), S)$ or $ch(S, R(wf))$ is larger than the chaining threshold $\theta \in (0, 1]$, S will be regarded as *candidate service*. p temporarily makes the hypothesis that S has been chained to wf, which results in a new hypothetical workflow wf'. Subsequently, p computes the guard value h' (line 5), which is the chaining score (cf. Eq. 1) of $L(wf')$ and $R(wf')$: $h' = ch(L(wf'),\ R(wf'))$. If $h' > q.h$, S is treated as *useful service* for constructing the workflow and wf' is fixed. $q.h$ is replaced with h' (line 8). In case that two observed services have the same IOPE signature but different QoS values, the one with higher quality is used.

Memorization Strategy: A query q can encounter some service S that **(a)** can be chained to one side but without increasing the guard value or **(b)** can not be temporarily chained to any side at all. S potentially would work as a key predecessor/successor of other useful services at later steps. Please note, that this situation is considerably different from what is typically assumed for centralized AI planners such as [14] which can fully observe the problem space. In the P2P setting considered for SPSC however, this case may appear frequently. To avoid missing S, a memorization strategy is introduced to carry over information about those potentially useful services. For this, the memo table Tb (cf. Definition 4) is used. Each row in Tb corresponds to a candidate service. It contains 3 entries: (1) the profile of S; (2) a side flag in $\{L, R, null\}$, which indicates whether S can be chained at the left, right or none of the both sides of the workflow; (3) a pointer that references another service S' in this table, if S can be chained to S' as a direct predecessor or successor. The pointer is *null*, if the service of this row can not be chained to any side or its direct predecessor/successor has not been determined yet. The memorization strategy is as follows: In case **(a)**, p adds S to $q.Tb$; sets the side flag; and sets the pointer to the direct predecessor/successor (line 9). In case **(b)**, p adds S to $q.Tb$ based on the usefulness rate $r_m(S)$ of S ($r_m(S) \in [0, 1]$, cf. Eq. 2) (line 10).

Apart from the cases (a) and (b), when the chaining of a useful service S^* leads to an increment of guard value, the memo table can also be updated by removing predecessor and successor services of S^*. This can happen, when they do not have unbound inputs or unsatisfied preconditions, due to their chaining with S^*. p checks this and updates the memo table if needed (line 8).

To estimate the potential usefulness of an un-chainable service S wrt. wf, $r_m(S)$ is computed by each peer locally, based on a set Q_S of observed queries in the past. Let $a(S)$ ($a'(S)$) be the number of (correct) workflows that use S; $b(S)$ ($b'(S)$) be the number of (correct) workflows that do not use S.

$$r_m(S) = \begin{cases} \omega & \textbf{, if } a'(S) = 0; \\ \frac{a'(S)}{a(S)+1} \cdot (1 - \frac{b'(S)}{b(S)+1}), & \textbf{otherwise}. \end{cases} \tag{2}$$

$\frac{a'(S)}{a(S)+1}$ ($\frac{b'(S)}{b(S)+1}$) is the statistical positive (negative) influence of treating (ignoring) S as candidate. ω ($\omega \in [0, 1], \omega \in \mathbb{R}$) is the default memorization rate. To choose Q_S, one option is to collect all the observed queries. It is easy to be applied, but rather inaccurate due to irrelevant queries. Another option is to consider only the queries similar to q by applying service matchmakers, like iSeM [15]. It yields better accuracy, but some computational overhead. A compromise is to consider the relevant queries observed in a time window.

Query Branching with Alternative Service: Let S_1 and S_2 be two different services. If they can be chained to the same side of another service S with the bindings that contain the same subset of variables in $S.O$ ($S.I$), S_1 and S_2 are called *alternative successors (predecessors)* wrt. S. For example, $S10$ (*ShapingSrv1*) is an alternative predecessor service to the *shapingSrv* (cf. Fig. 1) wrt. $R.O$: $I_{10} = (Material\ ?m)$; $O_{10} = (Product\ ?pro)$; $P_{10} = tempLargerThan(?m,$ $200) \wedge qualityNotBad\ (?m)$; $E_{10} = shaped(?pro)$. Both of $S9$ and $S10$ can be chained to $R.O$ with bindings on the same subset $\{Product\ ?pro0\}$ of variables.

If a peer p can not find a correct solution locally, p tries to find the alternative predecessor ($\mathcal{S}_{pred}$) / successor ($\mathcal{S}_{succ}$) services (lines 15 and 18) for each hypothetically chained candidate service S in memo table. On top of this, p determines the possible sub-queries. For each $S' \in \mathcal{S}_{pred}$, p creates a sub-query q', in order to investigate the possible workflow with S' (line 17): p initializes q' as a copy of q. Then, it replaces S with S' in $q'.Tb$ and unchains those candidate services that depend on S. Further, p computes $q'.h$ by Eq. 1. Finally, p executes $queryProcess(q')$, if $q'.wf$ has not been processed by p before. If there exists a non-empty set $\mathcal{S}_{succ}$ of alternative successor services of S wrt. a service S^* in wf, the services in any subset of $\mathcal{S}_{succ}$ can be chained to $S^*.O$ at the same time by sharing the data of bound variables. Namely, after the execution of S^*, all services in $\mathcal{S}_{succ}$ have chance to be executed in one workflow. In this case, p will issue (line 20) one sub-query for each element in the power set $2^{\mathcal{S}_{succ} \cup S} \setminus \{S\}$. E.g. if $\mathcal{S}_{succ} = \{S'\}$, p issues the sub-queries for $\{S'\}$ and $\{S, S'\}$.

3.3 Query Routing

A query q is forwarded, if its workflow wf is *not correct* and $q.TTL > 0$. For this, p computes a path suggestion (PS) containing a sequence of key peers, for which the total inverse importance score per traffic cost is minimized, under the TTL limit. (line 13 in Algorithm 1). A key peer is either (1) the provider of a memorized candidate service S in wf, or (2) the signature maintainer peer p_m of S. From p_m, a peer $p^* \in q.path$ got to know S and p^* is the first one (compared with the others in $q.path$) that uses S for the building of wf. The reason to consider key peers is that they have higher chances of knowing about other services chain-able to S. The creation of PS is modeled as a relaxed variant of the travelling salesman problem (Eq. 3): (i) no distinct return journey is needed; (ii) a peer can be traversed by a (sub-)query multiple times, if needed to reach key peers.

$$\begin{array}{ll} \textbf{minimize}: \sum_{p' \in P_{key}} w(p'', q, p'); & \textbf{s.t.}: \sum_{p'_i,\ p'_{i+1}} L(p'_i, p'_{i+1}) \leq q.TTL. \\ w(p'', q, p') = \frac{L(p'', p')}{sr(p', S)}; & sr(p', S) = r_m(S) \cdot \frac{qos(S)}{|wf_H| + 1 - dep(S)}. \end{array} \tag{3}$$

where P_{key} is the set of key peers and $p'' \in \{p\} \cup P_{key}$; $w(p'', q, p')$ is the inverse importance score per traffic cost of a key peer p', if q is suggested to reach p' from p''; $L(p'_i, p'_{i+1})$ is the length of the shortest path between the i-the and the $(i+1)$-th key peers in PS; $sr(p', S)$ is the importance score of S wrt. wf, which is determined by the historical usefulness rate value $r_m(S)$ and the stability factor of wf. The latter is estimated based on the service quality and the dependency relations between S and its predecessors or successors. $|wf_H|$ is the total number of hypothetically chained candidate services; the dependency factor $dep(S)$ is the number of the necessary predecessors/successors of S in Tb.

Inspired by the closest neighbor heuristics [13], p computes an approximately optimal PS in a greedy manner. Based on p's local knowledge about the network topology, p iteratively selects the current best key peer p'_{best} that yields the minimal $w(p''_{last}, q, p'_{best})$. p''_{last} is either the last key peer or p in current PS. After each iteration, p concatenates the shortest path $p''_{last} \rightarrow p'_{best}$ to the tail of current PS, considering TTL limitation. When p receives q and $q.psug \neq empty$, p recomputes it if p has made contribution to the building of workflow, i.e. $q.Tb$ has been updated by p; otherwise, p routes q according to $q.psug$. p routes q to a random neighbor, if $q.psug$ is empty and p is not able to suggest a path for q.

4 Theoretical Analysis

Lower Bound of Completeness wrt. Plan Existence. Let $F \in \mathbb{N}^+$ be the initial TTL value of each query. In unstructured P2P, approaches for solving a query q can only be incomplete in any case, given that there is no guarantee that peers knowing about services required for the solution can be traversed before TTL is exceeded. Therefore, the following analysis is focused on solvable cases, which are characterized as follows: in a connected unstructured P2P network with N peers, all services $\mathcal{S}$ required for the correct solution to $q.R$ are known by a set P ($|P| \leq F$) of peers that can be traversed in F hops from q's requester. The collaborative composition process is modeled by a finite Markov process. Let v $(1-v)$ be the probability of q being (not) forwarded to a peer in P. Once $q.wf$ is correct (final state), q is not forwarded anymore with probability 1. If $|P| \leq F$, the probability Pr of reaching the final state within F hops is: $Pr = \sum_{j=|P|+1}^{F} \binom{j-1}{|P|-1} \cdot v^{|P|} \cdot (1-v)^{j-|P|} + v^{|P|}$; otherwise, $Pr = 0$.

In the worst case, $|P| = F$ and each $p \in P$ knows at least one service in $\mathcal{S}$. This yields the generic lower bound $v^{|P|}$. By memorization strategy and peers observation, a request then can be solved by less than $|P|$ peers. Namely, $v = \frac{n_S^{(t)} \cdot dg}{N} \cdot \frac{1}{dg} = \frac{n_S^{(t)}}{N}$ increases over queries. $n_S^{(t)}$ is the total number of peers that know S after the t-th query; dg is the maximum peer connectivity; ω^* is the average memorization rate wrt. a service $S \in \mathcal{S}$. The propagation of the signature of S can be modeled by a recursive function (cf. Eq. 4) with $n_S^{(0)} = 1$:

$$n_S^{(t)} = n_S^{(t-1)} + \frac{F^2}{N^2}\omega^* \cdot n_S^{(t-1)} \cdot (N - n_S^{(t-1)}), \tag{4}$$

We investigate the following: (**1**) Will all N peers eventually know about S or not? (**2**) How fast will the epidemic process converge? For (**1**), it holds that $n_S^{(\infty)} = N$ and moreover $v \rightarrow 1$ for $t \rightarrow \infty$; For (**2**), the right-hand part $(N - n_S^{(t-1)})$ of Eq. 4 will eventually reach some fixed $\varepsilon > 0$, allowing for the following substitution: $n_S^{(t)} = n_S^{(0)} \cdot (1 + \frac{F^2\omega^*\varepsilon}{N^2})^t$. This indicates that $n_S^{(t)}$ converges *sub-linearly* to N with the rate $(1 + \frac{F^2\omega^*\varepsilon}{N^2})$.

Proof of Correctness. The correctness of SPSC consists of two aspects: (1) correctness of the guard heuristics and (2) correctness of the collaborative composition. Intuitively, (1) means that a correct workflow is achieved when its guard value equals to 1 (Theorem 4.1), while (2) indicates that the guard value of a workflow is monotonically increasing during the distributed composition process and reaches 1 within a lower bound (Theorem 4.2).

Theorem 4.1. *A workflow wf is correct, if $h = 1$.*

Proof: We prove this by contradiction. Assuming that $h = 1$, but wf is not correct, it follows that at least one criteria in Def. 4 is not satisfied. By Eq. 1, the violation of any criterion leads to $h < 1$. Contradiction. ∎

Lemma 4.1. *The guard value of a (sub-)query q is monotonically increasing during the entire query processing on all its traversed peers.*

Proof: According to Algorithm 1, no peer decreases the guard value by its local composition process. During query routing, no process changes the guard value. ∎

Inspired by the Floyd-Hoare theory [2] and Polyhedral Compilation Foundations lecture notes[2] of the UCLA, we reduce the joint composition of SPSC into a loop algorithm, and prove the correctness of it. Consider the whole P2P network as a huge computer containing lots of processing units (peers). A query is a task that is processed by peers in turn until TTL=0 or it is resolved (correct workflow composed). Each unit (peer) executes Algorithm 1 on receiving the query. This corresponds to an iteration. Following [2], we prove the preservation of evidencing invariants: (**a**): h of each (sub-)query q is monotonically increasing. This means that the joint composition process leads any intermediate partial solution (workflow) strictly towards a better follow-up step; (**b**): No alternative branch is missed. This indicates that all possible workflow options will be investigated.

[2] http://www.cs.ucla.edu/~pouchet/lectures/doc/888.11.algo.6.pdf.

Theorem 4.2. *SPSC is correct: Given a query, SPSC returns a correct solution with lower bound probability, if that solution exists within F hops.*

Proof: Initialization: $q.h$ is initialized with 0. Before the composition starts, h is not decreasing. The workflow container is empty. Hence, no alternative can be missed. **Maintenance:** By Lemma 4.1, $q.h$ is monotonically increasing during local composition. Moreover, Algorithm 1 ensures that it checks all alternative services for each service in wf. No alternative service is missed at p. **Termination:** The entire process terminates, if $q.TTL = 0$ or $h = 1$. At this time, h is not smaller than its initial value. No alternative service for each service in wf has been missed, as no one was missed in each iteration. Overall, $q.h$ is not decreasing and has a lower bound probability to reach 1 ($q.wf$ is correct). ■

Complexities. Denote **N** (**E**): the total number of peers (edges) in an unstructured P2P network; $\mathbf{m_1}$: the number of primitive terms in V; $\mathbf{m_2}$ ($\mathbf{m_3}$): the maximum number of PE predicates (IO parameters) in a service; $\mathbf{l_{ch}}$: the complexity for computing service chaining score: $l_{ch} = l_{bp} + l_{sat}$, where $l_{bp} = \mathcal{O}(m_3^2 \cdot m_1^{m_1})$ is the complexity for determining parameter bindings; $l_{sat} = \mathcal{O}(|O_p|^{m_3} \cdot m_2)$ is the complexity for checking whether a service effect is satisfied [18], where $|O_p|$ is the number of concepts in a peer's local ontology. $\mathcal{O}(m_1^{m_1})$ is the cost for measuring concept similarity; **n**: the number of services a peer can know about; **F**: the initial value of query TTL; $\mathbf{l_{sp}} = \mathcal{O}(E + NlogN)$: the cost for computing a shortest path [5]; **L**: the number of services in a workflow.

Computational complexity. In Algorithm 1, p attempts to chain local services exhaustively to the current workflow. This yields the worst case complexity $\mathcal{O}(n^n \cdot l_{ch})$. p also checks alternative services for each candidate service in wf yielding the process of up to $L \cdot 2^n$ sub-queries. Thus, the complexity of local composition is $\mathcal{O}(L \cdot 2^n \cdot n^n \cdot l_{ch} + n^n \cdot l_{ch}) \sim \mathcal{O}(L \cdot (2n)^n \cdot l_{ch})$. To suggest a routing path, p computes the $w(\cdot,\cdot,\cdot)$ value (cf. Eq. 3) for each candidate service S. $r_m(S)$ is computed in $\mathcal{O}(1)$ incrementally. The actual workflow size and dependencies can be computed in $\mathcal{O}(L)$. For augmenting a suggested path, p selects the best key peer. This costs at most $\mathcal{O}(L^2 \cdot l_{sp})$. Further, there can be at most F augmentations. Overall, the computation complexity for path suggestion at p is $\mathcal{O}(L^2 \cdot F \cdot l_{sp})$.

Traffic complexity. For a query, p issues $\mathcal{O}(L \cdot 2^n)$ sub-queries at most, of which each inherits the current TTL value. Thus, the total number of forwarding messages of a query is $2 \cdot \sum_{j=0}^{F-1}((L \cdot 2^n + 1)^j) \sim \mathcal{O}((L \cdot 2^n + 1)^F)$.

Robustness. Unstructured P2P networks are ad hoc environments. SPSC handles the dynamics of the network topology and services. The arrival of a new service provider or the addition of a new service S matters, if the provider peer p in the meanwhile processes a query q. In this case, p performs the local composition process against S. If q is backtracking, p issues a sub-query q from itself when S can work as an alternative service. The departure of a service provider or the deletion of a service can cause in incorrect path suggestions. In SPSC, peers react passively in this situation, without extra message exchange. If a departure event is detected (messaging timeout) by another peer p' with routing q, p'

deletes the reference of S from $\mathcal{S}_{p'}$ and $q.wf$. Subsequently, p' recomputes $q.psug$. Service signature update is treated as a sequence of deletion and addition.

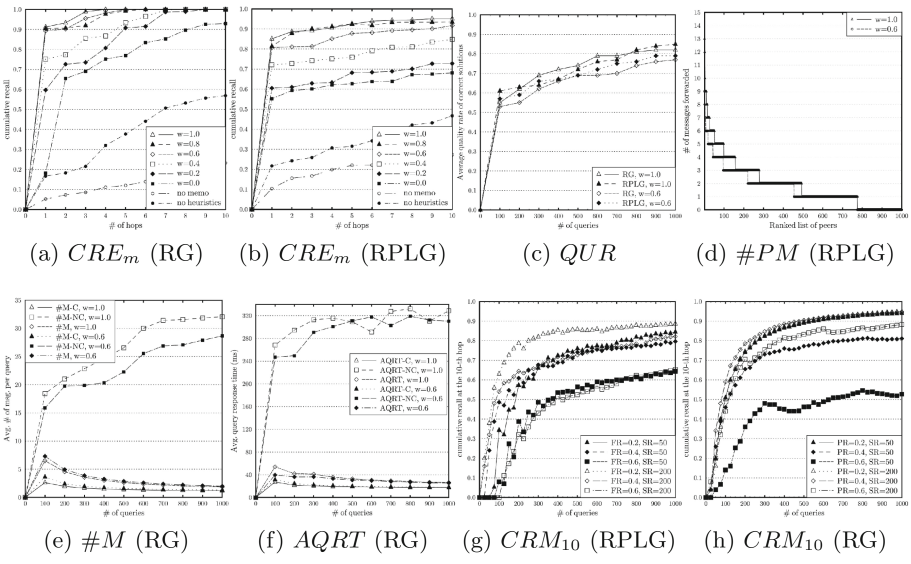
5 Experimental Evaluation

Settings. Based on our P2P framework[3], we simulated random graph (RG) and random power law graph (RPLG) based unstructured P2P networks with 1000 peers. Their average connectivity degrees are 10.295 and 4.457, respectively. To enable large scale evaluation, we disabled peer IP-based communication and simulated this by using global data structures and function calls. The initial query TTL is 10. $\theta = 0.3$ and $\beta = 1.0$. To the best of our knowledge, no test collection is suited for stateless composition of semantic services with IOPE. The IPC 2011[4] test collection is well-known in stateful AI planning. However, the factual representations of initial and goal states of IPC queries are not suitable for applying SPSC. The WSC[5] test bed supports only IO but not PE. Therefore, we developed a preliminary test collection[6] with 40 IOPE semantic services and 7 requests. Each request is labeled with one or two correct solutions with different groups of services. The service and query distributions are random. The experiments has been done on a PC with Core i7 CPU (2.80 GHz), 8 GB RAM.

Evaluation Measures. Let Q be the set of issued queries. $EC_{m,q} \in \{0,1\}$ means whether (or not) there exists a set of services at remote peers reachable within m hops from the requester. $C_{m,q} \in \{0,1\}$ means whether (or not) a correct solution of q has been composed within m hops. We check: **(1) $\mathbf{CRE_m}$**: average cumulative recall within distance m: $CRE_m = \frac{1}{|Q|}\sum_{q\in Q}\frac{C_{m,q}}{EC_{m,q}}$. $\frac{C_{m,q}}{EC_{m,q}}$ is 0, if $EC_{m,q} = 0$. **(2) QUR**: average QoS rate of resolved queries. Besides the services in the test collection with their pre-defined QoS, another copy of them with 50 % lower QoS has been deployed. Let $rtqu_q$ be the run time quality of a resolved query q, defined as the average quality of all services used in the result workflow, and $exqu_q$ the average quality of the optimal solution wrt. service quality (100 % QoS). $QUR = \frac{1}{|Q|}\sum_{q\in Q}\frac{rtqu_q}{exqu_q}$. **(3) #M**: average number of forwarded messages per query. **(4)** average traffic load size (in KB) of query. **(5)** total number of forward messages of each peer. **(6) AQRT**: average query response time.

Cumulative Recall and Workflow Quality. We compare the average cumulative recall after 1000 queries using different memorization rates ω, in RG and RPLG networks (cf Fig. 2a and b). Baseline results for composition without memorization and without guard value mechanism are also shown. As can be seen, SPSC with memorization largely outperforms the baselines, which either are not able to keep track of potentially useful services, or perform arbitrary chaining. In addition, SPSC achieves 10 % $\sim$ 20 % higher cumulative recall in RG compared to RPLG, as the latter may contain islands, while the RG does not.

[3] http://sourceforge.net/p/mymedia-peer/code/HEAD/tree/trunk/.
[4] http://www.plg.inf.uc3m.es/ipc2011-deterministic/Resources.html.
[5] http://www.ws-challenge.org/.
[6] http://sourceforge.net/projects/mymedia-peer/files/.



(a) CRE_m (RG) (b) CRE_m (RPLG) (c) QUR (d) $\#PM$ (RPLG)

(e) $\#M$ (RG) (f) $AQRT$ (RG) (g) CRM_{10} (RPLG) (h) CRM_{10} (RG)

Fig. 2. Experimental evaluation results of SPSC with different settings.

For both configurations, it holds that higher ω values yield better recall. In RG, more than 90 % of queries are resolved before their TTL limit is reached. Particularly, the correct solutions for at least 90 % of the queries are composed at early hops already when the memorization rate was relatively high ($\omega \geq 0.6$). This indicates that necessary services for resolving requests are propagated effectively and path suggestions support proper routing. Further, we check the system evolution speed with the CRE_{10} value (CRE at 10th hop) over time. After about (200) 500 queries, more than (60 %) 80 % requests are correctly resolved. This evidences the effectiveness of peer local observation and memorization mechanism. SPSC query routing also effectively considers QoS (cf. Fig. 2c) as a criterium for path suggestions, ultimately leading to an increase of QUR over time.

Network Traffic Overhead. The average number of messages per query $\#M$ decreases as the number of queries increases, and converges to a value less than 3 for RG (cf. Fig. 2e) and 2 for RPLG. The number of messages for successful queries $\#M_C$ also decreases similarly, since the knowledge about services from observations at peers is enriched over time. The number of messages for unresolved queries $\#M_{NC}$ is large and increases over time. However, the $\#M$ values imply that query branching occurs more rarely, since queries are solved in a few hops. Overall, SPSC in RG costs larger network traffic than with RPLG, since peers in RG have similar connectivity, while in RPLG, some "backbone" peers are better connected. They are easier to observe queries and hence obtain more knowledge to solve queries. Investigation of the traffic load size per query shows results in line with the previous observations. The average size of resolved queries was 60 KB (RPLG) and 75 KB (RG), while for unresolved queries, messages of about 200 KB (RPLG) and 620 KB (RG) size have been sent on average. Fig. 2d depicts the number of messages received at each peer of the RPLG network in descending order. The overall maximum is only 13 without bottleneck problem, even with RPLG, as peers learn over time and resolve queries in few hops.

Query Response Time. Similar to $\#M$, the overall AQRT in RG (cf. Fig. 2f) and RPLG decreases over queries, due to the peers increasingly observed knowledge, helping to resolve queries in few hops. Less messaging and query processing decrease AQRT in the long run for RG and RPLG.

Robustness. We test CRE_{10} of SPSC in RG and RPLG based configurations with network topology dynamics. For this, we programmatically force, after each SR queries (called a stable round), a percentage (FR) of randomly selected peers to leave off and the departed peers in the last round to re-join the network. System starts with all peers online and no dynamics event occurs during a stable round. The results with RPLG (cf. Fig. 2g) and RG networks show that the system can recover given the network dynamics, since peer observation helps to repair the semantic overlay. More frequent (SR) or heavier (FR) dynamics yields stronger impact, but SPSC still gives acceptable performance, as the chance of losing "backbone" peers in RPLG is relatively small given the random selection. To test SPSC performance with service dynamics, we programmatically force a percentage (PR) of randomly selected peers to forget all their knowledge about services after each stable round. The results with RG (cf. Fig. 2h) and RPLG show that the system can recover under service dynamics. The impact is stronger, if dynamics is more frequent or heavier.

6 Related Work

PM4SWS [10,11] performs IO-level semantic service composition. Each peer records observed compositions in a lookup table. Given a query, if no correct solution found in the table, it tries to chain a known service to the current workflow. Using the classic flooding, it can cause in immense network traffics. In contrast, SPSC peer issues sub-queries for only the new partial solutions.

In [6–8], Furno et al. present the probabilistic flooding-based stateful service composition method SCComp. Peers route queries to a set of selected helpful neighbors to resolve the sub-goals. Once a sub-goal has been resolved, extra transitive messages are sent for re-constructing the overall solution. Despite the selective flooding, network traffic can still be heavy, since a peer still has chance to receive duplicate queries with the same sub-goal. In contrast, SPSC uses walker-based routing with search space pruning and memorization strategy. SCComp needs extra messages to adapt to network dynamics, but SPSC does not.

AntAgt [3,4,23] utilizes an ant-inspired and agend-based approach. Peers maintain co-use matrices that contain pairs of classified reusable services observed in historical plans. A composition task is assumed to be pre-configured in terms of a set of key classes, forming a workflow template. Give a query, a peer selects its local services matching some keys and forward queries to another peer for the remaining keys. SPSC does not depend on plan templates but actually finds workflows at runtime. Similar efforts [12,22] also rely on design phase.

Approaches based on (semi-)structured overlay, e.g. DHT [20,24] or super peers [9], can assure certain completeness. However, to adapt to the network dynamics, approaches in this kind cost large traffics to maintain overlay. Using super peers introduces bottlenecks and single points of failure. In contrast, SPSC operates unstructured and does not require additional coordination effort.

7 Conclusion

The presented SPSC approach solves the problem of efficient and distributed OWL-S service composition planning in unstructured P2P networks. In contrast to related work, it mitigates composition failures caused by dead-ends and prunes the search space for efficient joint composition through heuristics-based semantics-empowered memorization and query branching. SPSC has been theoretically proven to be sound with reasonable lower bound of completeness. The experiments revealed its high cumulative recall with low network traffic overhead, and robustness to dynamic changes.

Acknowledgment. This work is supported by the German ministry for education and research (BMBF) in the projects Collaborate3D and Inversiv under grant numbers 01IW1102 and 01IW14004.

References

1. Baader, F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York (2003)
2. Floyd, R.W.: Assigning meanings to programs. Mathematical Aspects of Computer Science, vol. 1, pp. 19–32. American Mathematical Society, Providence (1967)
3. Forestiero, A., Mastroianni, C., Papadakis, H., Fragopoulou, P., Troisi, A., Zimeo, E.: A scalable architecture for discovery and planning in P2P service networks. In: Gorlatch, S., Fragopoulou, P., Priol, T. (eds.) Grid Computing, pp. 97–108. Springer, New York (2008)
4. Forestiero, A.; Mastroianni, C.; Papuzzo, G.; Spezzano, G.: A proximity-based self-organizing framework for service composition and discovery. In: Proceedings of the 10th International Symposium on Cluster, Cloud and Grid Computing, pp. 428–437. IEEE (2010)
5. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3), 596–615 (1987). ACM
6. Furno, A., Zimeo, E.: Efficient cooperative discovery of service compositions in unstructured P2P networks. In: Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDNBP), pp. 58–67. IEEE (2013)
7. Furno, A., Zimeo, E.: Gossip strategies for service composition. In: Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 27–35. IEEE (2014)

8. Furno, A., Zimeo, E.: Self-scaling cooperative discovery of service compositions in unstructured P2P networks. J. Parallel Distrib. Comput. **74**(10), 2994–3025 (2014). Elsevier
9. Galatopoullos, D.G., Kalofonos, D.N., Manolakos, E.S.: A P2P SOA enabling group collaboration through service composition. In: Proceedings of the 5th International Conference on Pervasive Services, pp. 111–120. ACM (2008)
10. Gharzouli, M., Boufaida, M.: PM4SWS: A P2P model for semantic web services discovery and composition. J. Adv. Inf. Technol. **2**(1), 15–26 (2011). Acadamy Publisher
11. Gharzouli, M., Boufaida, M.: A generic P2P collaborative strategie for discovering and composing semantic web services. In: Proceedings of the 4th International Conference on Internet and Web Applications and Services, pp. 449–454. Venice/Mestre, Italy (2009)
12. Gu, X., Nahrstedt, K.: Distributed multimedia service composition with statistical QoS assurances. IEEE Trans. Multimedia **8**(1), 141–151 (2006). IEEE
13. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: a case study in local optimization. Local Search Comb. Optim. **1**, 215–310 (1997)
14. Klusch, M., Gerber, A., Schmidt, M.: Semantic web service composition planning with OWLS-XPlan. In: AAAI Fall Symposium on Semantic Web and Agents (2005)
15. Klusch, M., Kapahnke, P.: The iSeM matchmaker: a flexible approach for adaptive hybrid semantic service selection. Web Semant. **15**, 1–14 (2012). Elsevier
16. Klusch, M.: Semantic web service description. In: Schumacher, M., Helin, H., Schuldt, H. (eds.) CASCOM - Intelligent Service Coordination in the Semantic Web, Chap. 3, Springer, New York (2008)
17. Li, Y., Bandar, Z.A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. EEE Trans. Knowl. Data Eng. **15**(4), 871–882 (2003). IEEE
18. Maloberti, J., Sebag, M.: Fast theta-subsumption with constraint satisfaction algorithms. Mach. Learn. **22**(2), 137–174 (2004). Springer
19. Martin, D. et al.: OWL-S: Semantic markup for web services (2004). http://www.w3.org/Submission/OWL-S/
20. Qin, P., Liu, R.: Search and combination of semantic web services based on chord. Comput. Knowl. Technol. **6**(28), 7936–7938 (2010)
21. Staab, S., Stuckenschmidt, H.: Semantic Web and Peer-to-Peer. Springer, New York (2006)
22. Tao, F., LaiLi, Y., Xu, L., Zhang, L.: FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans. Industr. Inf. **9**(4), 2023–2033 (2013). IEEE
23. Zimeo, E., Troisi, A., Papadakis, H., Fragopoulou, P., Forestiero, A., Mastroianni, C.: Cooperative self-composition and discovery of grid services in P2P networks. Parallel Process. Lett. **18**(3), 329–346 (2008). World Scientific
24. Zhu, Z., Hu, Y., Lan, R., Wu, W., Li, Z.: A P2P-based semantic web services composition architecture. In: Proceeding of ICEBE, pp. 403–408. IEEE (2009)