Stigmergic Service Composition and Adaptation in Mobile Environments

Andrei Palade, Christian Cabrera, Gary White, Siobhán Clarke

School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland
{paladea,cabrerac,whiteg5,siobhan.clarke}@scss.tcd.ie

Abstract. Users within a limited geographic area can form service-sharing communities using the services deployed on their mobile devices. Creating Quality of Service (QoS) optimal service compositions in such decentralised and dynamic environments is challenging because of the service providers' mobility and the inherent dynamism in the available services. Existing proposals for mobile environments either use template-matching composition or require a-priori knowledge about the QoS objectives' weights, which limits the composition's flexibility in such environments. This paper presents a stigmergic-based approach to model the decentralised, flexible and dynamic service interactions of providers in a mobile environment. A nature-inspired optimisation mechanism is used to approximate the set of QoS optimal compositions that result from these interactions. To facilitate adaptation of the composite during execution, we introduce a procedure that encourages the exploration of service composition configurations that emerge as a result of providers' mobility. We evaluate the performance of the proposed approach with a no-adaptation variant, a Dijkstra-based, a Greedy and a Random approach. The results show that the proposed approach can obtain superior solutions compared with current optimisation methods for flexible service composition in mobile environments at the cost of increased overhead.

Keywords: Stigmergic, Flexible, QoS-Aware Service Composition

1 Introduction

Recent developments in wireless communications technologies allow the mobile users within a limited geographical area to share services deployed on their mobile devices to form a service-sharing community, which can enable new service-based applications. Such applications are generally distributed among several devices since a single device is not capable of executing an entire application. For example, a smart route planner for users in a mall can be opportunistically provisioned by exploiting the ad-hoc interactions of a personal-shopper's phone, a nearby car's satellite navigator, and the mall's information kiosk to enable the requested functionality [8, 17]. Apart from the functional requirements, this composition process should create solutions with the best possible QoS. In the envisioned environment this is challenging because services may fail, their qualities may decline because of the mobility of devices, or functionally similar services with better QoS may appear during the execution of the composition.

Most service composition proposals use a template-matching approach to provision QoS optimal configurations. This approach relies on functionally-equivalent services, but with different QoS, to be optimally assigned to a predefined workflow of tasks. Such an exactly-defined request affects the composition' flexibility when the environment is dynamic [8]. The services can be combined in an input-output dependency graph, where each node corresponds to one service, and an edge is a matching between two connected services. Planning-based algorithms can be used to find paths in this graph using goal-driven approaches [8]. Without QoS consideration, the shortest path is normally identified. With QoS considerations, finding an optimal path is difficult. The shortest path might not be optimal, since a longer path may have better QoS. The providers' mobility and the dynamism in services deployed on their devices may affect the search [30].

The existing QoS optimisation mechanisms for planning-based service composition in mobile environments require a-priori knowledge about QoS objectives' weights, and use exact or heuristics methods to find one optimal solution. This requirement reduces the composition' flexibility, and does not allow for different quality trade-offs to be thoroughly investigated. Also, the exact or heuristics methods trade optimality for computational efficiency, or vice-versa [7]. Nature-inspired metaheuristics address these limitations by constructing one or more solutions and gradually improve these through an iterative process. However, the existing proposals are limited to template-matching composition [16] or require a centralised perspective [26, 28]. Mobile environments need flexible interactions, which may benefit from decentralised processing.

Stigmergic coordination has been used successfully in finding QoS optimal routes in networks with frequent link disconnections, and rapid topology changes such as Mobile (Vehicular) Ad-Hoc Networks [25]. However, the existing stigmergy-based proposals are limited to template-matching composition [16]. An efficient and effective QoS optimisation method for planning-based service composition that can quickly find a set of QoS optimal compositions in a mobile environment is required to provide higher utility with acceptable overhead.

This paper presents a stigmergic-based approach to model the decentralised, flexible service interactions in a mobile environment. New compositions (with better QoS) may emerge as a result of providers' mobility. The main contributions of this paper are: (1) a nature-inspired optimisation method to approximate the set of QoS optimal compositions; (2) to facilitate composition adaptation during execution, a procedure that encourages the exploration of new service composition configurations that emerge as a result of providers' mobility. We evaluate the performance of the proposed approach and compare the results with a no-adaptation variant, a Greedy, a Dijkstra-based and a Random approach. The proposed approach can obtain superior solutions compared with existing optimisation methods for flexible service composition in mobile environments.

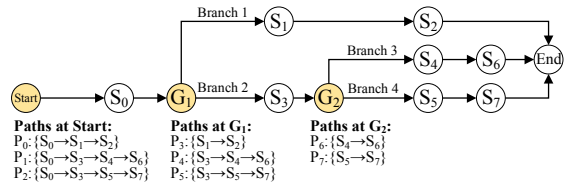
The paper defines the QoS optimisation problem in planning-based composition in Section 2, presents the stigmergic modelling of composition, the proposed QoS optimisation and adaptation procedure in Section 3, the implementation and evaluation of these methods in Section 5, and the evaluation results in Section 6. Section 7 presents the related work, and Section 8 concludes the paper.

2 Problem Description

The services deployed on mobile devices in a geographic area such as a mall [8] or an university campus [9] can be used to create new service-based applications. To enable such applications, services can be clustered based on their syntactic or semantic similarity using a Service-Specific Overlay Network [1]. Goal-driven service composition using planning-based algorithms can automatically solve a user's request using the available services in these clusters [8].

Figure 1 shows the result to a user's request, which is a service dependency graph with three possible service composition solutions (paths): P_0, P_1 and P_2. In this example, the inputs of service S_2 can be satisfied by the outputs of service S_1. When two or more nodes (services) in the graph share the same input, an execution guidepost is created [8]. An execution guidepost $G_i = <R_{id}, D>$ is a split-choice control element in the composition process that maintains a set of execution directions (branches) D for a composition request R_{id}. Each element in the set D is defined as $d_j = <id, w>$, where j $\leq |D|$, w represents the set services in the branch and id represents the identifier of the branch. For example, at G_2 two paths can complete the execution: P_6: $\{S_4 \rightarrow S_6\}$ and $\{P_6 \rightarrow S_7\}$. In the envisioned environment, new service compositions can emerge as a result of providers' mobility. New nodes can be attached at runtime to the graph if they can provide the required inputs/outputs [8, 9].

Apart from the functional requirements, a composite service needs to satisfy non-functional requirements. Given a set of QoS objectives, the challenge is how to efficiently construct a composite service that maximises the satisfaction of these QoS requirements. Such multi-objective optimisation problems are characterised by several objectives that have to be optimised simultaneously. When a user's preferences about each objective weight are known, the task is to find the



S_i	RT	Th
0	5.982	0.334
1	0.228	17.543
2	0.237	25.316
3	0.221	9.049
4	0.222	9.009
5	0.527	5.692
6	0.453	4.415
7	0.566	189.045

Services Table

P_i	RT	Th
P_0: {S_0→S_1→S_2}	6.447	0.334
P_1: {S_0→S_3→S_4→S_6}	6.878	0.334
P_2: {S_0→S_3→S_5→S_7}	7.296	0.334
P_3: {S_1→S_2}	0.465	17.543
P_4: {S_3→S_4→S_6}	0.896	4.415
P_5: {S_3→S_5→S_7}	1.314	5.692
P_6: {S_4→S_6}	0.675	13.424
P_7: {S_5→S_7}	1.093	5.692

Paths Table

Legend
P_i – i-th path
S_i – i-th service
G_i – i-th guidepost
RT – Response Time
Th – Throughput

Fig. 1. A service dependency graph with 3 available service compositions (paths): P_0, P_1 and P_2. Paths P_3, P_4, P_5, P_6, P_7 are alternative paths that can be selected at execution time. Services Table shows the QoS for each service i in the graph, and Path Table shows the aggregated QoS for each path. These values are aggregated using the formulas in Table 1.

Table 1. QoS aggregation formulas.

Parameter	Aggregation Formula	Description
RT (Response Time)	$\sum_{i=1}^{n} rt_i$	rt - response time of i-th service
Th (Throughput)	$min(th_i)$	th - throughput of i-th service

service combination that can optimise a global utility function. However, when this information is not available, compromises among multiple objectives need to be made to identify the set of Pareto-optimal solutions. A Pareto-optimal solution is a solution that is not dominated by any other solution in the set. The population of such solutions form the Pareto-optimal set (Pareto-front) [14, 23], and are generally identified using non-dominated sorting techniques [15].

While the QoS values of each service in the graph can generally be estimated using existing QoS prediction mechanisms [27], finding valid paths in this dynamic graph is challenging. The QoS values of services may be time-dependent, or services may become un-available because of the mobility of service providers. In contrast to optimisation towards a static optimum, the goal in a dynamic environment is to track as closely as possible the dynamically changing optima [4]. Prior optimisation methods for planning-based service composition in mobile environments used exact or heuristics algorithms to address this problem. Exact algorithms can find the optimal solution, but lack scalability because of their exponential complexity. Heuristics have polynomial complexity, but do not offer any worst-case guarantees on how close the QoS values of the returned solutions come to the QoS values of the Pareto-optimal ones [23]. Also, the existing proposals require a centralised perspective, which, in the envisioned environment, is a single point of failure or has the potential to cause processing and communication bottlenecks because of frequent state updates. Mobile environments need flexible interactions, which may benefit from decentralised processing.

3 QoS Optimisation Mechanism

Stigmergic coordination, exhibited by the social insects to coordinate their activities, can be used to enable service-based applications in mobile environments. A set of mobile software agents interact with the environment by encoding application-specific information as pheromone to achieve certain tasks. These agents achieve collaboration and self-organisation by exchanging pheromone and performing several pheromone operations. A service is composed by a group of service agents that can form an organisation in order to collaborate. We use this abstraction to model the decentralised, flexible service interactions in a mobile environment. We are extending the notation used by Moustafa et al. [16] to account for the forward-backward motion of the mobile agents, which is used to reinforce the optimal paths [21].

Definition 1. ***Service Agent (sa)****. A tuple* $sa = <id, F>$*, where id is the agent's identifier and F is the pheromone store used during composition.*

Definition 2. ***Pheromone Store (F)****. A tuple* $F = <id, f>$*, where id is the identifier of the direction and f is the pheromone value associated with that direction.*

Definition 3. ***Memory Store (M)***. *A set* $M = \{sa_0, \ldots, sa_n\}$ *where each element in the set is a service agent.*
Definition 4. ***Mobile Agent (ma)***. *A tuple* $ma = <id, M>$, *where id is the identifier of each agent, and M is the memory store associated with that agent.*

The proposed QoS optimisation method is inspired by Ant Colony Optimisation [25]. The mechanism uses a decentralised architecture, which allows service composition configurations to adapt in mobile environments. A service composition request is satisfied by a requester service agent. Using a service discovery component, a user's request can be functionally modelled as a service dependency graph (Figure 1). Each service in the service dependency graph is modelled using a service agent. A node in the graph is a service agent, and can function as a source, destination or intermediate node. The source and destination node can be the service requester agent. The intermediate nodes are the composition participants. Each intermediate node (service agent) stores a list of addresses of the service agents that can provide the next service in the graph. A pheromone level is associated with each service agent in the list. The quantity of pheromone is associated with the quality of the service provided by that service agent.

For each service composition optimisation task, a set of mobile agents is used. The mobile agents communicate through the pheromone store of each service agent. The mobile agents modify this value, to communicate the quality of solution to other mobile agents that visit the node. The number of mobile agents is task dependent. For simplicity, this number is set to the number of nodes in the service dependency graph. This number can affect the overhead, as well as the performance of the optimisation method. In this work we focus on the impact of the environment variables on the optimisation mechanism.

The mobile agents can iteratively traverse the graph in a forward-backward motion. The forward moving mobile agents use the intermediate nodes' probability routing tables to advance in the graph:

$$P_d = \frac{\tau_d^\alpha}{\sum_i^N \tau_i^\alpha} \tag{1}$$

where d is a branch in the set of branches D (as defined in Section 2), and τ_i is the pheromone level associated with the first node (service agent) from branch d. As they travel towards the destination node, the forward agents collect paths' QoS information. This information contains the QoS of each service component in the path and is added to the agent's memory store and aggregated. When all the forward agents reach the destination node, the identified paths are filtered using a non-dominated sorting technique [15] to identify the Pareto-optimal paths (optimal solutions). At the destination node, forward agents become backward agents only if their identified path is Pareto-optimal. As backward agents move in the reverse path, the intermediate nodes modify their pheromone store using: $\tau_i = \tau_i + \tau_i * (Q/l_i)$ (positive reinforcement) where τ_i is the pheromone level, Q is a constant and l is the length of the i-th path. To allow for new paths to emerge, an evaporation procedure $\tau_i = \tau_i * (1 - \rho)$ is used. Heuristic information is not used in this work to avoid expensive network state dissemination.

4 Adaptation Procedure

To facilitate composition adaptation during execution, a procedure that encourages the exploration of new service composition configurations that emerge as a result of providers' mobility is required. The stigmergic optimisation mechanism can converge to the optimal solution by selecting the path with the highest pheromone value. However, the envisioned environment introduces two additional challenges to finding the optimal configuration: (1) services in the service dependency graph may join or leave at any time because of the mobility of the providers; (2) the QoS of the services offered by such providers may vary in time. Therefore adaptation procedure must be promoted to effectively address these issues.

To facilitate composition adaptation during execution, and allow other parts of the solution space to be explored, we use a pheromone smoothing scheme. This scheme allows identified optimal paths to be reinforced with lesser pheromone [22]. This allows for other (some previously marked as non-optimal) paths to be explored. Each element in the pheromone store is updated as follows:

$$\tau_{ij'} = \tau_{ij} + \delta * (\tau_{max} - \tau_{ij}) \tag{2}$$

where $\tau_{ij'}$ is the new pheromone level, τ_{ij} is the previous pheromone level, δ is a smoothness coefficient and τ_{max} is the maximum pheromone level that can be associated with this path.

While evaporation adopts a uniform discount rate for every node, the pheromone smoothing technique places a greater reduction in the reinforcement of pheromone concentration on the optimal path(s), by reducing the rate of reinforcement of dominant paths [21]. Previous research showed that such technique can prevent the generation of dominant paths. Stuzle and Hoos [22] introduced a similar mechanism to solve the travelling salesman problem, but in a stationary environment. Also, this approach showed promising results in a dynamic, but stationary Wireless Sensor Networks environment [5].

In the envisioned environment, the topology of the service dependency graph can change continuously. The devices on which the services are deployed may move out of range, or may be power depleted. If these services are part of the optimal service composition paths, when the environment changes, other paths in the service dependency graph may become optimal during the execution of the composition. Also, new service composition may emerge because of the mobility of users, which may have more optimal QoS.

In this paper, we are proposing this approach as an adaptation handling procedure for planning-based service composition in mobile environments. Each service agent is allocated with two pheromone update rules, which will be executed periodically. These rules are the evaporation and pheromone smoothing of each element in the pheromone store.

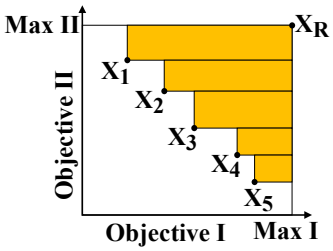


Fig. 2. PS (A) metric. Example of space dominated (in orange) by a given Pareto set (points X_1, X_2, X_3, X_4 and X_5) when two objectives are minimised.

5 Implementation and Evaluation

The proposed QoS optimisation method is implemented and evaluated using the Simonstrator platform [19]. The proposed QoS optimisation algorithm, both with and without the adaptation handling, produces a set of solutions for each traversal of the service dependency graph. First, we evaluate how well the proposed QoS optimisation mechanism approximates the set of Pareto-optimal solutions. We perform the evaluation with and without adaptation handling. Second, we measure utility of the produced solutions and the introduced overhead of the proposed QoS optimisation mechanism, with and without adaptation handling. We compare the results with a Dijkstra-based, a Greedy and a Random approach. The two objectives considered in this evaluation are the response time and the throughput, but multiple objectives can be considered such as battery lifetime of devices and service availability. We assume that each node in the service dependency graph stores the estimated QoS value.

5.1 Performance metrics

PS (A) - Size of dominated space: This metric was introduced by Zitzler and Thiele [35] and indicates how well the Pareto-optimal set is approximated by the set of solutions A. Here, the set A contains only non-dominated solutions. The greater the size of the space dominated, the closer the solutions are to the Pareto-optimal set [26]. This metric is calculated as follows:

$$\begin{gathered} PS(A) = \frac{S(A)}{Max\ I * Max\ II}, \text{and} \\ S(A) = (Max\ I - x_1) * (Max\ II - y_1) + \sum_{i=2}^{n} (Max\ I - x_i) * (y_{i-1} - y_i) \end{gathered} \tag{3}$$

where $Max\ I$ and $Max\ II$ are the maximum values of Objective I and Objective II. These two values are used to create the Reference Point X_R. As an example, Figure 2 shows the composition of set A, which is a Pareto front made of 5 points. The dominated space is given by the orange surface.

Utility This metric indicates the overall satisfaction of QoS requirements of a service composition configuration [2]. Certain QoS values such as response time are considered negative criteria and need to be minimised to increase user satisfaction, whereas others such as throughput are considered positive criteria and need to be maximised.

$$U_imin = \begin{cases} \frac{Q_i^{max} - Q_i}{Q_i^{max} - Q_i^{min}} \\ 1 \end{cases} \quad U_imax = \begin{cases} \frac{Q_i - Q_i^{min}}{Q_i^{max} - Q_i^{min}} & if Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & if Q_i^{max} - Q_i^{min} = 0 \end{cases} \tag{4}$$

where Q_i is a QoS value, and Q_i^{min} and Q_i^{max} are the minimum/maximum QoS values available for Q_i. Equation 5 computes the utility of each service composition configuration as follows:

$$Utility_g = \sum_{1}^{n} U_i * W_i \tag{5}$$

where W_i represents the importance weights assigned to the i-th metric. Each weight is a number in the range [0, 1] and the sum of all weights is equal to 1. In this evaluation, all objectives carry an equal weight.

Overhead The overhead measures the number of exchanged messages between the nodes used during the composition. This metric includes all the types of messages that are exchanged, including retransmission or probing messages.

5.2 Evaluated Algorithms

We chose several QoS optimisation mechanisms designed for planning-based service composition to compare their utility and overhead with our proposed model. Each algorithm was implemented in the simulator. Also, two versions of proposed QoS optimisation were formalised:

1. **ACO**: the implementation of the QoS optimisation mechanism described in Section 3, without the adaptation procedure.
2. **E-ACO**: an extension to **ACO**, which uses the proposed adaptation procedure introduced in Section 4.

An important limitation of the selected baseline algorithms is that they require user's QoS preferences and they produce a single solution, whereas the proposed QoS optimisation mechanism can does not have a requirement for user input (for QoS weights) and may produce a set of solutions. We use the utility metric to select a single solution from the set of Pareto-optimal solutions produced by ACO and E-ACO algorithms. We compare the utility values and overhead with the existing proposals for QoS optimal service composition in mobile environments:

1. **SimDijkstra**: an exact algorithm that uses a Dijkstra-based algorithm to find the shortest path in the service dependency graph [11]. The execution of the algorithm finishes when the end node is reached.
2. **GoCoMo**: a heuristics-based algorithm that uses a Greedy-based approach to select the path with the highest utility value to the user. It uses monitors to collect QoS of potential paths. Probes are sent regularly to update the QoS information from the leaf nodes to each guidepost in the service dependency graph [8].

3. **Random**: randomly selects the next service in the service dependency graph. This algorithm shows the expected value of a random solution and provides a baseline that the specialised algorithms should easily outperform [13].

5.3 Test case generation

For the purpose of evaluation, we use a scenario based on an adaptive route planner application [8]. Such an application can be built using services provided by the available (mobile) devices in the environment. Many services can provide similar functionality and more than one service composition configuration may be available. The composition with the optimal QoS should be selected to maximise user satisfaction of non-functional requirements.

The response time and throughput of the final composite service were measured. These values are used to compute the utility of the final composition plan. For ACO and E-ACO, we also measure the size of the dominated space based on a reference point fixed using the best QoS values for response time and throughput. In this evaluation, the reference point was set to the highest throughput plus 5 units and the lowest response time minus 0.1 units. For each algorithm presented in Section 5.2 we also measured the ovearhed. To minimise the impact of external factors on our results, the presented algorithms were executed 100 times per problem instance to compute the averages.

5.4 Environment Setup

The mobile devices in the environment move at a speed of 7.5 to 13.5 m/s. The other environment's properties are: size - 1000*1000 m, communication range - 250 m, and movement model - Gauss Markov. Each device is set to offer one service. Each service has QoS values. We use a QoS dataset to initialise the QoS of each service participant, which consists of a matrix of response time and throughput values for 339 users by 5,825 services [34]. To address the dynamism in the services available in the environment, these QoS values are randomly changed after every service iteration by multiplying every QoS value with a random number in the interval [0.9, 1.1]. For simplicity, only QoS discrete numerical values are considered in this work.

In ACO and E-ACO a number of parameters need to be set before running the algorithm such as the number of mobile agents, the factor α, parameter Q, the pheromone coefficient ρ, and the initial pheromone level in each node. A larger number of mobile agents improves the convergence rate, however, overhead is introduced. We set the number of mobile agents to be equal to the number of nodes in the service dependency graph. The other parameters are initialised as follows: $\alpha = 0.9$, $\rho = 0.01$, $\tau_{initial} = 10.0$, and $Q = 10$. The adaptation procedure is initialised using $\tau_{max} = 20.0$ and $\delta = 0.5$. The smoothing rate is set to (evaporation rate) * 0.5.

6 Results

6.1 Size of dominated space

Figure 3 shows how the size of the dominated space achieved by ACO and E-ACO evolves as the number of iterations is increased. The size of the dominated

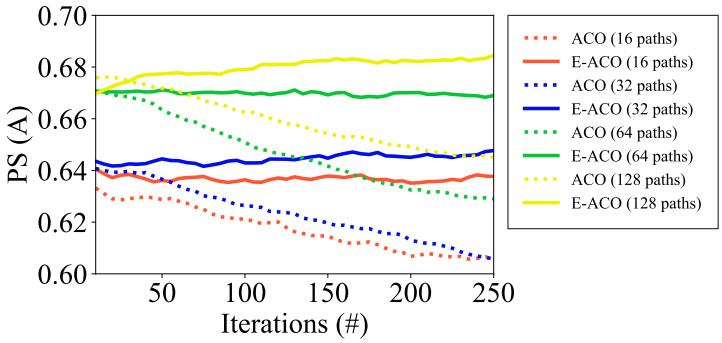


Fig. 3. Evolution of PS (A) metric (defined in Section 5.1) after 250 iterations for different number of paths (higher is better).

space indicates how well the Pareto-optimal set is approximated by the identified solutions. The greater the size of the space dominated, the closer the solutions are to the Pareto-optimal set. This metric was defined in Section 5.1. We show the results of this evaluation for the cases when the number of available paths (size of the solution space) is 16, 32, 64, and 128. When 16 paths are available, the size of the dominated space for ACO is 63.32% after 10 iterations and decreases to 60.54% after 250 iterations, whereas for E-ACO maintains a level of 64%. When the number of paths increases to 32, ACO obtains 64.07% after 10 iterations and decreases to 60.56% after 250 iterations, whereas E-ACO slowly increases from 64.35% to 64.76%.

Both ACO and E-ACO increase the size of their dominated space as the number of available paths increases. For example, when 128 paths are available ACO achieves 67.59% and E-ACO achieves 66.96%. However, ACO maintains the same decreasing slope and achieves 64.51% after 250 iterations, whereas E-ACO achieves 68.45% after 250 iterations. We also report that when the size of the solution space is 64, ACO achieves 67.05% after 10 iterations and 62.90% after 250 iterations, whereas E-ACO achieves 66.96% after 10 iterations and 68.45% after 250 iterations.

6.2 Utility

Figure 4 shows the results of the utility evaluation as the number of available paths (service composition configurations) is increased. We show the results of this evaluation for the cases when the number of available paths (size of the solution space) is 16 (Figure 4a), 32 (Figure 4b), 64 (Figure 4c), and 128 (Figure 4d). The utility is calculated using the configurations produced by ACO and E-ACO in the last 5 iterations, and the configurations produced by SimDijkstra, GoCoMo and Random. This metric was defined in Section 5.1. Both objectives have equal weights.

By increasing the number of iterations, two trends can be observed for ACO and E-ACO: (1) the utility of both increases as the number of paths increases,

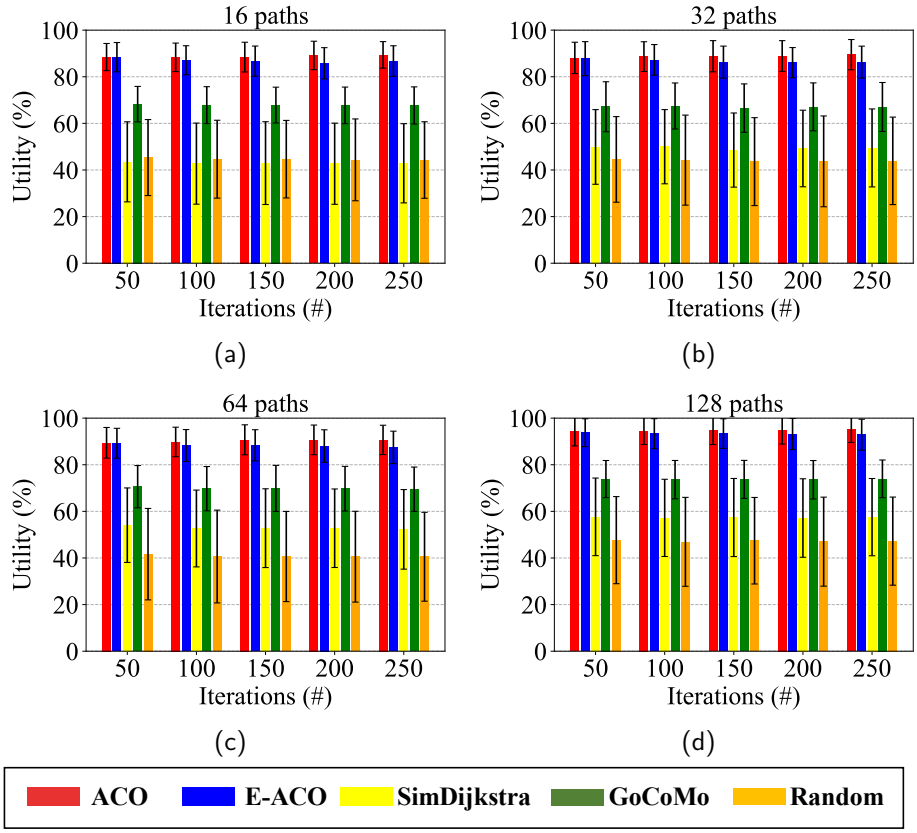


Fig. 4. The utility of solutions produced by the evaluated algorithms (higher is better) after 250 iterations for different number of paths. The results are averaged over 100 executions.

and (2) the utility of ACO is higher than the utility of E-ACO after 100 iterations. This difference decreases as the number of paths increases. When 16 paths are available, ACO achieves an utility of 86.58% and E-ACO achieves an utility of 80.38%. This decreases to 93.26% for ACO and 88.52% for E-ACO, when 128 paths are available. The SimDijkstra and Random algorithms have the lowest utility for almost all runs. GoCoMo performs comparably to ACO and E-ACO.

6.3 Overhead

Figure 5 shows the overhead (number of exchanged messages) after 250 iterations as the number of available paths in the service dependency graph varies. In GoCoMo, a global state mechanism periodically disseminates network state to all participating nodes, which explains the high overhead. This mechanism is not used in SimDijkstra and Random and the overhead introduced by these two algorithms is insignificant. However, the utility of solutions of these algorithms is

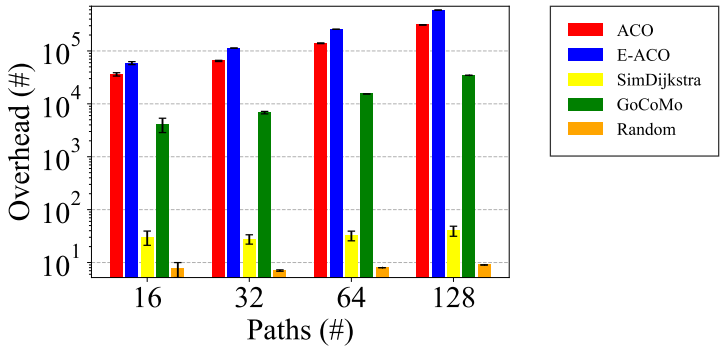


Fig. 5. The overhead when various number of paths are available (lower is better). The results are averaged over 100 executions.

considerably lower than the utility produced by ACO and E-ACO and GoCoMo. ACO and E-ACO introduce a higher overhead than the other approaches because of the large number of agent messages. This number is task dependent, and, for simplicity, it was set to the number of nodes in the service dependency graph. Tuning this parameter would likely reduce the overhead, though this needs to be verified.

7 Related Work

Most QoS-aware service composition mechanisms use a template-matching approach where the user's request is a predefined set of abstract tasks and the goal is to find an assignment of services to pre-defined abstract tasks to maximises a QoS-related user benefit. Such approaches used exhaustive search, genetic algorithms [6], linear and integer programming [32, 3, 2] to select, for each task, an optimal service candidate from a functionally-equivalent list of candidates.

Planning-based service composition was introduced to address the functional limitation of the template-matching model. However, the existing proposals have limited support for non-functional requirements in mobile environments. They use exact and heuristic approaches to find an optimal QoS composition plan. GraphPlan [30], SimDijkstra [11], and Rodriquez et al. [20] presented a Dijkstra-based algorithm to find compositions paths that satisfy users' QoS objectives subject to constraints. To improve resilience of the composition, Jiang et al. [11] return the top-k optimal paths. However, these works do not address the dynamic nature of a mobile environment. GoCoMo [8] and Kalasapur et al. [12] use a heuristics approach based on greedy selection to find the service composition configuration with the highest utility to the user. These planning-based service composition approaches require a-priori knowledge about the user's objectives, which reduces the flexibility of composition in a dynamic environment.

Nature-inspired metaheuristics have been designed to address challenges that are common in large-scale, dynamic and decentralised networks [10]. Such algorithms can support self-organisation, self-configuration, and collaboration in

changing environmental conditions. They can dynamically adapt to ensure end-to-end communication between devices and provide efficient management of available resources. These algorithms can balance the trade-off between computation efficiency and optimality [21], and can obtain a set of solutions having two important properties: good convergence and diversity in solutions. The convergence represents the distance to the set of Pareto-optimal solutions. The diversity property refers to obtaining a uniform-spaced set of solutions to indicate a good exploration of the search space without losing any valuable information [14]. Such methods are approximate because they start from constructing one or more complete solutions and gradually improve these solutions during the iterative process. This makes it possible to track the quality of the solutions found during each iteration, and stop as soon as acceptable results are identified.

The existing stigmergic-based proposals for QoS-aware composition used ACO-based algorithms to identify the set of optimal solutions. MO_ACO [33] use an ideal vector with the best QoS value to guide the exploration of solution space and identify the path with the highest utility. Wu and Zhu [29] combine a transaction-aware service composition and a QoS-aware service composition model to find the optimal candidate for each abstract task. AACO [24] adjusts the pheromone evaporation factor at runtime based on the dynamics of the environment. ACO-WSC [31] present an ant-colony optimisation to minimise the length of the composition plans when composition participants are cloud services. However, these proposals do not maintain populations of non-dominated solutions. MOACO4WS [18] uses a reset strategy to address adaptation in dynamic environments. The existing proposals either are limited to template-matching composition [28, 16] or require a centralised perspective [26].

8 Conclusion and Future Work

The main contributions of this work are a nature-inspired optimisation method to approximate the set of QoS optimal compositions when composing services deployed on mobile devices, and an adaptation procedure to facilitate the exploration of new service composition configurations (with potentially better QoS) that emerge as a result of providers' mobility. The existing proposals for service composition in such environments are limited to template-matching composition or require a-priori knowledge about the QoS objectives' weights. Previous work in service composition showed that template-matching composition can limit the flexibility of composition when the environment is dynamic. Also, new services may emerge as a results of mobility of providers, and the requirement of user input may affect the time of providing the user with new service composition configurations. The proposed mechanism is designed for planning-based service composition, and does not require user input. The mechanism, with and without using the adaptation procedure, is evaluated in a dynamic scenario where various number of compositions can be created. In our evaluation, we control this number to measure the performance of the proposed QoS optimisation mechanism, and compare the results with GoCoMo, SimDijkstra and a Random approach.

The results show that the proposed QoS optimisation mechanism can achieve a higher utility at the cost of increased overhead compared to the existing baseline proposals for planning-based service composition for mobile environments. Also, the results show that the proposed adaptation procedure improves the approximation of the set of QoS optimal service composition configurations that emerge as a results of providers mobility. The decrease in utility of solutions when the adaptation procedure is used can be motivated by the selection of weights that are used to evaluate this metric. However, the adaptation procedure can achieve a better approximation of the Pareto-optimal solutions than the no-adaptation variant. In our future work will explore the case when different utility weights are used. Also, in our future work we will investigate how the number of agents used in ACO and E-ACO affects the overhead.

Acknowledgment

This work was funded by Science Foundation Ireland (SFI) under grant 13/IA/1885.

References

1. Al-Oqily, I., Karmouch, A.: A decentralized self-organizing service composition for autonomic entities. ACM TAAS **6**(1) (2011)
2. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient QoS-aware service composition. In: Proc. of the 18th Int. Conf. on World wide web. ACM (2009)
3. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. on Software Engineering **33**(6) (2007)
4. Blum, C., Li, X.: Swarm intelligence in optimization. In: Swarm Intelligence. Springer (2008)
5. Cai, W., Jin, X., Zhang, Y., Chen, K., Wang, R.: Aco based qos routing algorithm for wireless sensor networks. In: Int. Conf. on Ubiquitous Intelligence and Computing. Springer (2006)
6. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of the 7th annual conference on Genetic and evolutionary computation. pp. 1069–1075. ACM (2005)
7. Chattopadhyay, S., Banerjee, A.: Qos constrained large scale web service composition using abstraction refinement. IEEE Trans. on Services Computing (2017)
8. Chen, N., Cardozo, N., Clarke, S.: Goal-Driven Service Composition in Mobile and Pervasive Computing. IEEE Trans. on Services Computing **11**(1) (2018)
9. Deng, S., Huang, L., Taheri, J., Yin, J., Zhou, M., Zomaya, A.Y.: Mobility-aware service composition in mobile communities. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(3) (2017)
10. Di Caro, G.A., Ducatelle, F., Gambardella, L.M.: Ant colony optimization for routing in mobile ad hoc networks in urban environments. IDSIA (2008)
11. Jiang, W., Hu, S., Liu, Z.: Top K query for qos-aware automatic service composition. IEEE Trans. on Services Computing **7**(4) (2014)
12. Kalasapur, S., Kumar, M., Shirazi, B.A.: Dynamic Service Composition in Pervasive Computing. IEEE Trans. on Parallel and Distributed Systems **18**(7) (2007)
13. Klein, A., Ishikawa, F., Honiden, S.: SanGA: A self-adaptive network-aware approach to service composition. IEEE Trans. on Services Computing **7**(3) (2014)
14. López-Ibánez, M., Stutzle, T.: The automatic design of multiobjective ant colony optimization algorithms. IEEE Trans. on Evolutionary Computation **16**(6) (2012)

15. Mishra, K., Harit, S.: A fast algorithm for finding the non dominated set in multi objective optimization. Int. Journal of Computer Applications **1**(25) (2010)
16. Moustafa, A., Zhang, M., Bai, Q.: Trustworthy stigmergic service composition and adaptation in decentralized environments. IEEE Trans. on Serv. Comp. **9**(2) (2016)
17. Palade, A., Cabrera, C., White, G., Razzaque, M., Clarke, S.: Middleware for Internet of Things: A quantitative evaluation in small scale. In: 6th IEEE Workshop on the IoT: Smart Objects and Services. pp. 1–6. IEEE (2017)
18. Qiqing, F., Xiaoming, P., Qinghua, L., Yahui, H.: A global qos optimizing web services selection algorithm based on moaco for dynamic web service composition. In: IFITA. vol. 1. IEEE (2009)
19. Richerzhagen, B., Stingl, D., Rückert, J., Steinmetz, R.: Simonstrator: Simulation and prototyping platform for distributed mobile applications. In: Proceedings of the 8th International Conference on Simulation Tools and Techniques (2015)
20. Rodriguez-Mier, P., Mucientes, M., Lama, M.: Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition. IEEE Trans. on Serv. Comp. (2015)
21. Sim, K.M., Sun, W.H.: Ant colony optimization for routing and load-balancing: survey and new directions. IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans **33**(5) (2003)
22. Stützle, T., Hoos, H.H.: Max–min ant system. Future generation computer systems **16**(8), 889–914 (2000)
23. Trummer, I., Faltings, B., Binder, W.: Multi-objective quality-driven service selectiona fully polynomial time approximation scheme. IEEE Trans. on Software Engineering **40**(2) (2014)
24. Wang, D., Huang, H., Xie, C.: A novel adaptive web service selection algorithm based on ant colony optimization for dynamic web service composition. In: International Conf. on Algorithms and Architectures for Parallel Processing (2014)
25. Wang, L., Shen, J.: A systematic review of bio-inspired service concretization. IEEE Trans. on Services Computing **10**(4) (2017)
26. Wang, L., Shen, J., Luo, J.: Facilitating an ant colony algorithm for multi-objective data-intensive service provision. Journal of Computer and Syst. Sciences (2015)
27. White, G., Palade, A., Clarke, S.: Qos prediction for reliable service composition in iot. In: ICSOC 2017, Workshop (ISyCC), Proceedings (2017)
28. Wu, Q., Ishikawa, F., Zhu, Q., Shin, D.H.: QoS-Aware Multigranularity Service Composition: Modeling and Optimization. IEEE Trans. on Systems, Man, and Cybernetics: Systems **46**(11) (2016)
29. Wu, Q., Zhu, Q.: Transactional and qos-aware dynamic service composition based on ant colony optimization. Future Generation Computer Systems **29**(5) (2013)
30. Yan, Y., Chen, M., Yang, Y.: Anytime QoS optimization over the PlanGraph for web service composition. In: Proceedings of the 27th Annual ACM SAC (2012)
31. Yu, Q., Chen, L., Li, B.: Ant colony optimization applied to web service compositions in cloud computing. Computers & Electrical Engineering **41**, 18–27 (2015)
32. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-Aware Middleware for Web Services Composition. IEEE Trans. on Software Engineering **30**(5)
33. Zhang, W., Chang, C.K., Feng, T., Jiang, H.y.: Qos-based dynamic web service composition with ant colony optimization. In: COMPSAC, 2010 IEEE 34th Annual
34. Zheng, Z., Zhang, Y., Lyu, M.R.: Investigating QoS of Real-World Web Services. IEEE Trans. on Services Computing **7**(1) (2014)
35. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. on Evolutionary Computation **3**(4) (1999)