Stigmergy-Based QoS Optimisation for Flexible Service Composition in Mobile Communities

Andrei Palade, Siobhán Clarke
School of Computer Science and Statistics, Trinity College Dublin, Ireland, {paladea,siobhan.clarke}@scss.tcd.ie

***Abstract*—Mobile users can form a service-sharing community within a geographic area by using their mobile devices. Finding Quality of Service (QoS) optimal service compositions in such mobile environments is challenging because of the inherent dynamism in services deployed on mobile devices. Existing service composition proposals for mobile environments either use template-matching composition or require a-priori knowledge about the QoS objectives' weights, which limits the composition flexibility in such environments. This paper introduces a QoS optimisation mechanism for planning-based service composition in mobile environments, where mobile software agents use stigmergic coordination to iteratively explore parts of the distributed service composition space to approximate a set of QoS optimal configurations. We present a mechanism that minimises the exploration of previously identified non-optimal solutions to encourage exploration of different parts of the service space. We evaluate the performance of the proposed approach and compare the results with a baseline variant, a Dijkstra-based, a Greedy and a Random approach. The results show that the proposed approach can achieve higher utility compared to the evaluated proposals at the cost of increased overhead.**

***Index Terms*—Mobile, QoS-Aware Service Composition**

I. Introduction

In mobile environments, how to efficiently construct QoS optimal service compositions is a challenging task. Composition participants may fail or may provide different QoS levels [1]. Most composition proposals solve the QoS optimal service composition problem using template-matching, where functionally-equivalent services, but with different QoS, are optimally assigned to a predefined workflow of tasks. Such an exactly-defined request affects the flexibility of composition when the environment is dynamic. The available services can be dynamically combined in an input-output dependency graph, where each node corresponds to one service, and an edge represents a syntactic matching between the two connected services. Planning-based algorithms can find a path in this graph from the initial state to a goal state [2]. 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. Also, the inherent dynamism in services deployed on mobile devices may affect the search [2].

Existing planning-based proposals require a-priori knowledge about the QoS objectives' weights. These proposals use exact or heuristics methods to find QoS optimal compositions, with computational efficiency and optimality implications [3]. Metaheuristics can balance the trade-off between computation efficiency and optimality, but existing proposals either are limited to template-matching composition or require a centralised perspective [4]. However, 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 [5]. However, the existing stigmergy-based composition mechanisms are limited to template-matching composition. An efficient and effective QoS optimisation method for planning-based service composition that can quickly find a set of QoS optimal service configurations in a mobile environment is required.

The contributions of this paper are: (1) an approximate, stigmergy-based QoS optimisation mechanism for decentralised, planning-based service composition; (2) a mechanism that minimises the exploration of previously identified non-optimal solutions to encourage exploration of different service composition configurations.

II. Proposed Mechanism

The proposed QoS optimisation method is inspired by the Ant Colony Optimisation [5]. The mechanism uses a decentralised architecture, which allows service composition configurations to adapt in mobile environments. Both service requesters and providers are mobile. Using a service discovery component, a user's request can be functionally modelled as a service dependency graph. A node in the graph can function as a source, destination or intermediate node. The source and destination node can be the service requester node. The intermediate nodes are the composition participants. Each intermediate node stores a list of addresses of the service providers that can provide the next service in the graph. A pheromone level is associated with each service provider in the list. The amount of pheromone is associated with the quality of the service. The agents modify this value, to communicate the quality of solution to other agents that visit the node.

Forward agents and backward agents are used to iteratively traverse the graph. The forward agents move in the graph using the intermediate nodes' probability routing tables: $P_D = \frac{\tau_D^\alpha}{\sum_i^N \tau_i^\alpha}$, where τ_i is the pheromone level and D represents the service nodes, which are following the current node in the graph. 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 and aggregated. When all the forward agents reach the destination node, the identified paths are filtered using a non-dominated sorting technique

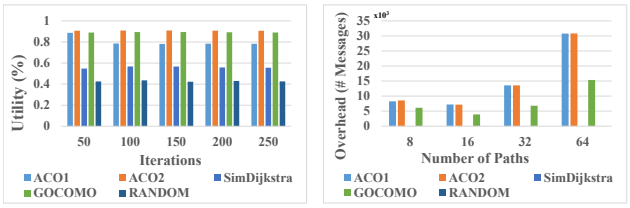
to identify the Pareto-optimal paths (optimal solutions). At the destination node, forward agents become backward agents only if their identified path is Pareto-optimal. As backward agents move in the reverse path, the intermediate nodes modify their pheromone matrix using: $\tau_i = \tau_i + \tau_i * (Q_1/l_i)$ (positive reinforcement) where τ_i is the pheromone level, Q_1 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. To minimise the exploration of previously identified non-optimal service configurations, a new pheromone update mechanism is introduced. This allows the backward agents associated with non-optimal solutions to update the pheromone levels on the intermediate nodes. This is done through $\tau_i = \tau_i - \tau_i * (Q_2/l_i)$ (negative reinforcement) combined with a $[\tau_{min}, \tau_{max}]$ reset strategy, where τ_{min} and τ_{max} are lower/upper bounds for pheromone level and Q_2 is a constant, l is the length of the path.

III. Implementation and Evaluation

Each service in the graph is initialised with response time and throughput from the WS-Dream [6]. The evaluation is limited to two objectives, but multiple objectives can be considered such as battery lifetime and service availability. The user's request is a route planner application [2], provisioned using the services on the mobile devices in user's proximity. For simplicity, the service dependency graph is represented as a k-ary tree with 8, 16, 32 and 64 paths available. Each node corresponds to one mobile (physical) node. The nodes move at a speed of 7.5 to 13.5 m/s. The environment's properties are: size - 1000*1000 m, communication range - 250 m, movement model - Gauss Markov. Two variants of the proposed QoS optimisation mechanism ACO1 (uses only positive reinforcement) and ACO2 (uses positive and negative reinforcement with the reset strategy) are evaluated using Simonstrator (an event-based simulator for mobile environments). These algorithms produce a set of Pareto-optimal solutions. We use weighted sum to merge all the objectives into a single objective (both objectives have equal weight) to be able to compare the utility [4] and overhead with SimDijkstra [7], GoCoMo [2], and Random. To initialise ACO1 and ACO2 we use the following values: $\alpha = 0.1$, $\rho = 0.01$, $Q_1 = 10$ and $Q_2 = 10$. Initially, τ is 10. The interval $[\tau_{min}, \tau_{max}]$ is set to $[1, 20]$.

IV. Results and Future Work

Figure 1a shows the improvement in utility as the number of iterations of ACO1 and ACO2 increases from 50 to 250 iterations, and number paths available is set to 64. In the full set of experiments (not illustrated for space reasons), we also report the utility values when 8, 16, 32 paths are available. The utility is calculated using the configurations produced by ACO1 and ACO2 in the past 50 iterations, and the single configuration produced by SimDijkstra, GoCoMo and Random. By increasing the number of iterations, ACO2 achieves a higher utility than ACO1. When 64 paths are available, the difference between ACO2 and ACO1 increases from 2% after 50 iterations to 12% after 250 iterations. When



(a) Utility. (b) Overhead.

Fig. 1. Figure 1a shows the utility of solutions produced by the evaluated algorithms when 64 paths are available. Figure 1b shows the overhead when various number of paths. The results are averaged over 30 executions.

16 and 32 paths are available, the utility difference between the two variants increases from 3% at 50 iterations to 10% after 250 iterations. The lowest difference in utility between the two variants is when only 8 paths are available and the number of iterations is 50. However, as the number of iterations is increased to 250 iterations, the difference in utility between ACO2 and ACO1 increases to 3%. The SimDijkstra and Random have the lowest utility for all runs. GoCoMo performs comparably to the ACO variants.

Figure 1b 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 considerably lower than the utility produced by ACO variants and GoCoMo. ACO1 and ACO2 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. The overhead introduced by ACO2 compared to ACO1 is insignificant.

The proposed QoS optimisation mechanism obtains more optimal solutions than the evaluated optimisation approaches for decentralised, planning-based service composition in mobile environments. The proposed reinforcement mechanism can produce service composition configurations of higher utility than positive-only reinforcement in mobile environments.

Acknowledgment

This work was funded by SFI under grant 13/IA/1885.

References

[1] Z. Wang, D. Huang, H. Wu, Y. Deng, A. Aikebaier, and Y. Teranishi, "Qos-constrained sensing task assignment for mobile crowd sensing," in *GLOBECOM*. IEEE, 2014.

[2] N. Chen, N. Cardozo, and S. Clarke, "Goal-Driven Service Composition in Mobile and Pervasive Computing," *IEEE TSC*, vol. 11, no. 1, 2018.

[3] S. Chattopadhyay and A. Banerjee, "Qos constrained large scale web service composition using abstraction refinement," *IEEE TSC*, 2017.

[4] Q. Wu, F. Ishikawa, Q. Zhu, and D.-H. Shin, "QoS-Aware Multigranularity Service Composition: Modeling and Optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 11, 2016.

[5] L. Wang and J. Shen, "A systematic review of bio-inspired service concretization," *IEEE TSC*, vol. 10, no. 4, 2017.

[6] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE TSC*, vol. 7, no. 1, 2014.

[7] W. Jiang, S. Hu, and Z. Liu, "Top K query for qos-aware automatic service composition," *IEEE TSC*, vol. 7, no. 4, 2014.