Stigmergic QoS Optimisation for Flexible Service Composition in Mobile Environments

by

Andrei Palade, M.Sci.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Doctor of Philosophy

University of Dublin, Trinity College

January 2019

Declaration

Andrei Palade

May 2, 2019

Permission to Lend and/or Copy

Acknowledgments

This work was made possible through the collective effort of a group of people. First and foremost, I would like to thank my supervisor Prof. Siobhán Clarke for giving me the opportunity to do a PhD, and for her guidance, expertise, and continuous support during this process. I would like to thank to my colleagues in the SURF project - Christian Cabrera, Fan Li and Gary White. I have learned many things from each of you, and thank you for your help over these past years. I would like to thank to my colleagues from office 2.15. I would also like to acknowledge Vivek Nallur for his time and for his help, especially in the incipient phase of this work. I thank my parents and my sister Raluca, for their continuous support over these past years. I dedicate this thesis to them. Finally, I am grateful for the financial support from Science Foundation Ireland, which has offered me the opportunity to a PhD in Ireland. Would like to acknowledge the Trinity Centre for High Performance Computing (TCHPC) department funded by eINIS, for providing me with computational resources to run the experiments for this work.

Andrei Palade

University of Dublin, Trinity College

January 2019

Abstract

With the increasing number of resource-rich handsets equipped with diverse wireless communication technologies, users within a limited geographical area can share the services deployed on their mobile devices to form service-sharing communities. By leveraging the computing resources on nearby devices, new service-based applications can be developed to expand users' service options. Many applications from multiple domains have the potential for improvement with flexible, dynamic service composition, including automotive (e.g., real-time hazard warnings), energy demand-side management (e.g., communities maximising use of renewable energy while catering to individual home needs), and FinTech (e.g., fast insurance response).

Automatic planning, with adaptive composition recovery mechanisms, has been used to tackle complex service provisioning in such dynamic environments. Existing service composition proposals generate a service dependency graph based on the interoperable relationships between available services, and use goal-driven techniques to discover paths that can functionally satisfy user requests. Apart from functional requirements, though, Quality of Service (QoS) such as execution reliability and latency are also major concerns that impact users' satisfaction. Finding paths in this graph that can functionally satisfy a user's request while simultaneously guaranteeing user-acceptable QoS levels is difficult in mobile environments. Given mobile devices' limited communication ranges, the frequent network topology changes, and services with time-dependent QoS, the existing proposals for QoS-optimal service composition trade-off computational efficiency for optimality to provide only best-effort QoS. The existing mechanisms also use a-priori articulation of the QoS objectives' weights, which does not allow for the exploration of different QoS trade-offs. These relative weights might differ at runtime, and the constant enquiry for user's preferences can inhibit the development of automatic, planning-based service composition.

This thesis presents SBOTI (Stigmergic-Based OpTImisation), a decentralised, QoS optimisation mechanism for automatic, planning-based service composition. SBOTI uses a community of homogeneous, mobile software agents, which share the same goal, to effectively and efficiently approximate the set of QoS-optimal service composition configurations available in a geographically-limited, mobile environment. The proposed mechanism uses an iterative, reinforcement-based approach to control the trade-off between computational efficiency and the optimality of the identified service composition solutions. SBOTI incorporates a non-dominated sorting technique to identify the Pareto-optimal set solutions, which allows the user to explore various QoS trade-offs. To control the diversity of the solutions in this set, SBOTI globally updates both dominated and non-dominated solutions using digital pheromones. To allow for exploration of new service composition configurations that may emerge as a result of providers' mobility, SBOTI uses an adaptation procedure that limits the amount of pheromone on previously identified solutions. SBOTI also engages multiple communities, with diverse properties, to collaboratively address the computational efficiency and optimality concerns introduced by a single community of homogeneous agents.

SBOTI is evaluated using simulations under various dynamic conditions. The evaluation metrics are the size of the dominated space, which indicates the optimality of the identified set of solutions, and communication overhead. Baselines for comparison are SimDijkstra, GoCoMo and a Random approach. Also, an utility metric is used to compare the performance of SBOTI with the baselines that require a-priori articulation of user preferences. The evaluation results illustrate both the strengths and the limitations of SBOTI in a mobile environment, under different network densities and mobility speeds.

Contents

List of Tables

List of Figures

List of Abbreviations

ACO	Ant Colony Optimisation
AI	Artificial Intelligence
AIS	Artificial Immune System
BBO	Biogeography-Based Optimization
BCO	Bee Colony Optimisation
BFO	Bacterial Foraging Optimisation
BLE	Bluetooth
BPEL	Business Process Execution Language
BPML	Business Process Modeling Language
D2D	Device-to-Device
DCON	Dynamic Composition Overlay Network
DP	Dynamic Programming
GA	Genetic Algorithm
GAs	Genetic Algorithms
GP	Genetic Programming
GPS	Global Positioning System
IA	Immune Algorithm
IoT	Internet of Things
MANET	Mobile Ad-hoc Network
MANETs	Mobile Ad-hoc Networks
MAS	Multi Agent System
MIP	Mixed-Integer Programming
MOO	Multi-Objective Optimisation
MOP	Multi-Objective Problem
NFC	Near-Field Communication
OWL-S	Web Ontology Language for Services

PSO	Particle Swarm Optimisation
QoS	Quality of Service
SA	Simulated Annealing
SBOTI	Stigmergic-Based OpTImisation
SCE	Service Composition & Execution Engine
SDE	Service Discovery Engine
SI	Swarm Intelligence
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
SOO	Single-Objective Optimisation
SSON	Service-Specific Overlay Network
SURF	Service-centric network for URban-scale Feedback systems
UML	Unified Modeling Language
WLAN	Wireless Local Area Network
WS-BPEL	Web Services Business Process Execution Language
WSNs	Wireless Sensor Networks

List of Publications

This thesis is based on the following peer-reviewed publications:

- **Journal Articles:**

  1. **Andrei Palade**, and Siobhán Clarke. "Collaborative Agent Communities for Resilient Service Composition in Mobile Environments. **submitted** to IEEE Transactions on Service Computing, (submitted January 2019)
  2. **Andrei Palade**, Christian Cabrera, Fan Li, Gary White, M. A. Razzaque, and Siobhán Clarke. "Middleware for Internet of Things: An Evaluation in a Small-Scale IoT Environment." Journal of Reliable Intelligent Environments, no. 1, pp. 3-23, Springer, 2018.

- **Conference Proceedings:**

  1. **Andrei Palade**, Christian Cabrera, Gary White, and Siobhán Clarke. "Stigmergic Service Composition and Adaptation in Mobile Environments." In International Conference on Service-Oriented Computing (ICSOC 2018), pp. 618-633. Springer, 2018.
  2. **Andrei Palade**, and Siobhán Clarke. "Stigmergy-Based QoS Optimisation for Flexible Service Composition in Mobile Communities." In 2018 IEEE World Congress on Services (IEEE SERVICES 2018), pp. 27-28. IEEE, 2018.
  3. **Andrei Palade**, Christian Cabrera, Gary White, M. A. Razzaque, and Siobhán Clarke. "Middleware for Internet of Things: A Quantitative Evaluation in Small Scale". In 2017 IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services (IEEE IoTSoS 2017), pp. 1-6. IEEE, 2017.

- Parts of this work have appeared in the following peer-reviewed publications:

  1. Gary White, **Andrei Palade**, Christian Cabrera, and Siobhán Clarke. "Autoencoders for QoS Prediction at the Edge". In Proceedings of 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom 2019), IEEE, 2019.
  2. Gary White, **Andrei Palade**, and Siobhán Clarke. "Forecasting QoS Attributes Using LSTM Networks". In Proceedings of 2018 International Joint Conference on Neural Networks (IJCNN 2018), IEEE, 2018.
  3. Gary White, **Andrei Palade**, and Siobhán Clarke. "QoS Prediction for Reliable Service Composition in IoT". In International Conference on Service-Oriented Computing, Workshop (ISyCC), pp. 149-160. Springer, Cham, 2017.
  4. Christian Cabrera, Fan Li, Vivek Nallur, **Andrei Palade**, M. A. Razzaque, Gary White, and Siobhán Clarke. "Implementing Heterogeneous, Autonomous, and Resilient Services in IoT: an Experience Report". In 2017 IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services (IEEE IoTSoS 2017), pp. 1-6. IEEE, 2017.
  5. Mohammad Abdur Razzaque, Marija Milojevic-Jevric, **Andrei Palade**, and Siobhán Clarke. "Middleware for Internet of Things: A Survey". IEEE Internet of Things Journal, no. 1, pp. 70-95. IEEE, 2016.

Chapter 1

Introduction

In 2017, 175 billion mobile applications were downloaded [Sydow and Cheney, 2018]. Also, it is estimated that the number of people connected to mobile services surpassed 5 billion globally [gsm, 2018]. The number of personal user devices (e.g., smartphones, tables, wearables, etc.) and Internet of Things (IoT) devices (e.g., Internet-enabled sensors) in the physical environment and the amount of data they generate is expected to grow exponentially over the next few years [Mascitti et al., 2018]. To process this huge amount of data, cloud platforms are typically used, with mobile devices acting as mere data generators and consumers of the output of data elaboration. However, this approach may not be a solution working well in all scenarios. Even next generation cellular technologies might not be able to scale up the capacity to the levels required by predicted traffic demands [Mascitti et al., 2018]. Also, during environment disasters or disaster relief efforts, the network infrastructure that is used to access these cloud-based platforms can be severely damaged or completely destroyed, which affects the potential of mobile applications relying on cloud-based services [Le and Kwon, 2017].

As emerging mobile computing-based techniques become more prevalent, the way services are provided and consumed is evolving, which highlights the potential to bring great opportunities for traditional service computing in mobile environments [Deng et al., 2016, Cicirelli et al., 2018, Olaniyan et al., 2018]. By leveraging the computing resources on nearby devices, new service-based applications can be developed to expand users' service options [Georgantas, 2018, Palade et al., 2018b]. Users within a limited geographical area can share the services deployed on their mobile devices to form service-sharing communities. The capability of flexibly and dynamically composing services in a limited geographic area from mobile, resource-constrained and heterogeneous devices opens up exciting possibilities in a large number of domains [Deng et al., 2017]. For example, indoor location-based service providers have to conduct effort-intensive and time-consuming business negotiations with building owners or operators to collect the floor plans or wait for them to voluntarily upload such data, which is not conducive to large-scale coverage in short time. However, by leveraging mobile data via commodity smartphones to construct the floor plans of complex indoor environments, intensive efforts and time overhead in the business negotiation process for service providers is avoided, which opens up the possibility of fast and scalable floor plan reconstruction in many indoor environments such as airports, train stations, shopping malls, museums and hospitals [Gao et al., 2018]. This leads to increasing attention to mobility issues in pervasive computing research because most of the existing (wired-based) service composition architectures were not designed with mobility taken into consideration. The seamless composition of distributed service components into more complex applications in a dynamic environment is a laborious process, especially when considering the possibility of facing disruptions caused by movement of users and service providers [Groba and Clarke, 2014, Chen et al., 2018].

Apart from functional requirements, Quality of Service (QoS) such as execution reliability and latency of such applications are also major concerns that impact users' satisfaction [Wagner et al., 2012]. Finding service composition configurations that can functionally satisfy a user's request while simultaneously guaranteeing user-acceptable QoS levels is difficult in mobile environments [Liu et al., 2017, Cervantes et al., 2017]. Given mobile devices' limited communication ranges, the frequent network topology changes, the resources variability and services with time-dependent QoS, the existing proposals for QoS-optimal service composition trade-off computational efficiency for optimality to provide only best-effort QoS [Karmouch and Nayak, 2012].

1.1 Service Composition in Mobile Environments

Service computing has become an important computing paradigm to develop distributed information systems through the composition of multiple loosely-coupled, autonomous, self-described, reusable, and portable components encapsulated as services. Such software artefacts have become the basic units for building rapid, low-cost, secure, and reliable applications [Erl, 2005, Baryannis et al., 2008]. With the rapid emergence of smart, resource-rich handsets equipped with diverse wireless communication technologies, services are no longer limited to traditional contexts and platforms [Perera et al., 2013]. They can be deployed on mobile devices and can be delivered over wireless networks [Jiang et al., 2007]. Mobile devices can play the roles of consumer and provider simultaneously. Mobile services delivered through mobile techniques are now emerging as a promising means to extend traditional service computing [Efstathiou et al., 2014, Deng et al., 2017].

1.1.1 Service Composition Process

A single service may not always be available to satisfy a user request, and may need to be provisioned by dynamically combining existing services [Urbieta et al., 2008, Teixeira et al., 2011, Lemos et al., 2016]. For example, a wind-chill service can be created by combining the data flows of a temperature service and a wind-speed service [Hachem et al., 2014]. The service composition process, in its simplest definition, can be defined in two parts:

- Addressing Functional Requirements: Given a set of atomic services available, the goal is to find a sub-set of these services, that can satisfy user's request [van Der Aalst et al., 2003]. This sub-set of services becomes a composite service, and each service in this set is a *composite participant*. In addition to these components, a composite service follows a *composition pattern*, which represents the execution order (data dependency) of the services in the composite. Such a composite can be *strictly-defined* or *loosely-defined* [Wang, 2011]. In a strictly-defined service composition request the services are invoked (executed) in a specific order (e.g., $\{S_1 \rightarrow S_2 \rightarrow S_3\}$). In a loosely-defined service composition request, multiple feasible service invocation orders are possible (e.g., $\{S_1 \rightarrow S_2 \rightarrow S_3\}$ or $\{S_1 \rightarrow S_3 \rightarrow S_2\}$). This work deals with only the strictly-defined service composition request case because it captures the interoperable relationship among available services in the environment. Common composition patterns are: sequential, parallel, conditional and loop [Zheng et al., 2016]. This work deals with only QoS concerns, and, for simplicity, it assumes that the available composites follow only sequential patterns. QoS aggregation formulae can reduce the parallel, conditional and loop structures to either a single or a set of QoS values, which can then be used within a composite that follows a sequential pattern [Cardoso et al., 2004, Jaeger et al., 2004].

- Addressing Non-functional Requirements: These are the constraints and objectives on a service's QoS attributes. Examples of QoS attributes for services are availability, reliability, execution cost, reputation, throughput or response time [Strunk, 2010]. The values of these QoS attributes can be either collected from service providers directly (e.g., price), recorded from previous executions (e.g., response time), or from user feedbacks (e.g., reputation) [Liu et al., 2004]. Also, recent advances in QoS prediction has used collaborative filtering approaches based on similar users invoking similar services to estimate the value of a QoS attribute, thereby avoiding expensive invocation of the service [White et al., 2017a, White et al., 2017b, White et al., 2018b, White et al., 2018c, White et al., 2019]. The *global* QoS of a composite service is determined by the *local* QoS of its composite participants and the composition pattern of the composite service [Ma et al., 2015]. The set of QoS attributes can be divided into *positive* and *negative* QoS attributes. The values of positive attributes need to be maximised (e.g., throughput and availability), whereas the values of negative attributes need to be minimised (e.g., price and response time) [Alrifai and Risse, 2009]. These attributes can also be categorised into *deterministic* and *non-deterministic*. Deterministic attributes are those where their values are known before the service invocations (e.g., price of using a service or security properties). Non-deterministic attributes are the attributes whose values are unknown at service invocation time (e.g., response time or availability) [Liu et al., 2004].

1.1.2 QoS Optimisation

The dynamic nature of mobile systems can negatively impact the quality of the composition [Liu et al., 2017], which, as a consequence, impacts user satisfaction with the provided composition [O'Sullivan et al., 2002, Wagner et al., 2012]. A large number of services can be available for the users to use in a mobile environment. The quality of these services may vary with time. In other words, alternatives that previously had a degraded QoS, can later improve their QoS. For example, the response time of a service improves as the workload required by that service has been reduced, or criteria that were important at a particular point in time are now irrelevant. Optimisation generally deals with finding, among many alternatives, a best (or good enough) solution to a given problem. However, these type of problems can become intractable when the scale increases, and usually require high-dimensional and complex computer algorithms to resolve. Finding the most optimal subset of these services becomes very hard. Such a Multi-Objective Problem (MOP) is generally solved by existing approaches by using *a priori* articulation of user preferences, and enumerating through each possible service composition configuration to find the one with the highest utility. A "simple" method to deal with multi-objective problems is to aggregate multiple objectives using a *weighted average* approach [Cruz et al., 2011]. Equation 1.1 shows how this can be performed, where parameter w_i is a user preference provided as a weight, and x is a proposed solution.

$$min(F(x)),\ where\ F(x) = \sum_{i=1}^{m} w_i * f_i(x),\ and \sum_{i=1}^{m} w_i = 1 \tag{1.1}$$

The concept of optimum with several objective functions changes because in multi-objective problems the aim is to find good compromise solutions (or trade-offs) rather than a single solution as in a global optimization problem. The notion of *optimum* was originally proposed by Francis Ysidro Edgeworth in 1881. This notion was later generalised by Vilfredo Pareto (in 1896), and the commonly accepted term is *Pareto-optimum* [Sindhya, 2011, Chiandussi et al., 2012, Talbi et al., 2012]. A vector of decision variables $\vec{x}^* \in F$ is *Pareto optimal* if there does not exist another $\vec{x} \in F$ such that $f_i(\vec{x}) \leq f_i(\vec{x}^*)$ for all $i = 1, \ldots, k$ and $f_j(\vec{x}) < f_j(\vec{x}^*)$ for at least one j. The vector $\vec{x}^*$ is Pareto optimal if there exists no feasible vector of decision variables $\vec{x} \in F$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion [Deb, 2014]. This concept almost always results in a set of solutions called the *Pareto optimal* set. The vectors $\vec{x}^*$ corresponding to the solutions included in the Pareto optimal set are called *non-dominated.* The plot (mapping) of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the *Pareto front* [Talbi et al., 2012].

1.1.3 Challenges

Service composition in a mobile environment is a challenging task. The time available for service selection and composition is limited. The service providers rely on mobile devices to deliver their services, and their services may become unavailable due to their mobility. The composition fails if a provider becomes unavailable when it is being used, and a new composition is needed. Also, other service providers (with better QoS) may appear in the environment. To avoid frequent re-composition, a set of service providers with a long available time for service is required so that the composition can exist as long as possible [Wang, 2011]. The computation complexity and the timeliness for providing a solution introduced by the optimisation mechanism makes the problem of finding QoS optimal service composition configurations challenging.

Failures can come from different sources such as connections may be intermittent and inconsistent in quality, devices may be switched off, and wireless connections may be of low bandwidth or shared among multiple devices. Such failures in service providers reduces the reliability of a composed service when they are participating in the composition [Wang, 2011]. The fluctuating usage load of these composition participants can reduce the performance of running applications. The large number of users and a potentially large number of applications running on this infrastructure, can impose capacity constraints on the underlying devices such as limited bandwidth [Deng et al., 2017]. The main challenges for enabling service-based applications in a mobile environment are:

(Ch1) **Mobility.** Mobile devices frequently change network operators. They are expected to experience frequent link failures during handover, where services they provide may become temporarily unreachable [Jiang et al., 2007]. Also, the mobile devices may move entirely out range, and become completely unreachable. These present major challenges for providing reliable service compositions in highly dynamic mobile wireless environments [Wang, 2011].

(Ch2) **Limited Resources.** Mobile devices (smartphones in particular) are still recognised as constrained computing devices. This is because of the limited battery power, which remains a major challenge for potential growth of mobile computing [Al Ridhawi and Karmouch, 2015, Deng et al., 2017]. For instance, too many service invocations by mobile device, can result in excessive computation, network traffic and energy consumption, which in turn increases the delay in receiving services [Park and Shin, 2008, Sadiq et al., 2015].

(Ch3) **Scalability.** Given the limited resources of a mobile device, the services deployed on such devices do not scale when many users are expected to concurrently access them [Deng et al., 2017]. The qualities of a service may evolve relatively frequently, either because of internal changes or workload fluctuations [Ardagna and Pernici, 2007].

Existing wired-based service composition architectures were not designed with mobility taken into consideration. The seamless composition of distributed service components into more complex applications in a dynamic environment is a laborious process, especially when considering the possibility of facing disruptions caused by movement of users and service providers [Razzaque et al., 2016, Chen et al., 2018]. Additionally, the heterogeneity of devices, resource variability and service reliability is highly unpredictable in mobile networks. Furthermore, the presence of multiple service providers offering the same functionality exacerbates this problem [Sadiq et al., 2015].

1.2 Problem Definition

As wireless technologies such as Wi-Fi and BLE are increasingly adopted for device-to-device communication [Gubbi et al., 2013], users can share their services deployed on their mobile devices to enable new applications [Deng et al., 2017, Chen et al., 2018, White et al., 2018a]. This section illustrates the characteristics of an environment through a motivating example. The scenario presents an adaptive, multi-modal route planner for a smart city [Palade et al., 2017, Palade et al., 2018a].

Applications for mobile users from multiple domains can be improved with flexible, dynamic service composition, including automotive (e.g., real-time hazard warnings), energy demand-side management (e.g., communities maximising use of renewable energy while catering to individual home needs), FinTech (e.g., fast insurance response) and many more [Gai et al., 2018]. Traditional service composition models rely on cloud-computing resources to deliver applications to users. With variable latencies associated with the Internet, accessing distant compute resources is not suitable for time-critical control or delivering stringent QoS to mobile applications [Mascitti et al., 2018]. In recent years, *resource-sharing* has emerged as a collaborative service consumption model [Owyang et al., 2014, Nielsen, 2014]. Users in a limited physical area (e.g., an office or a region of a city) can leverage the unused capacity of the devices they own or services they can provide [Deng et al., 2017]. Driven by convenience, better price and quality of available services (Figure 1.1a), user attitudes towards this collaborative economy concept are growing more positive (Figure 1.1b).

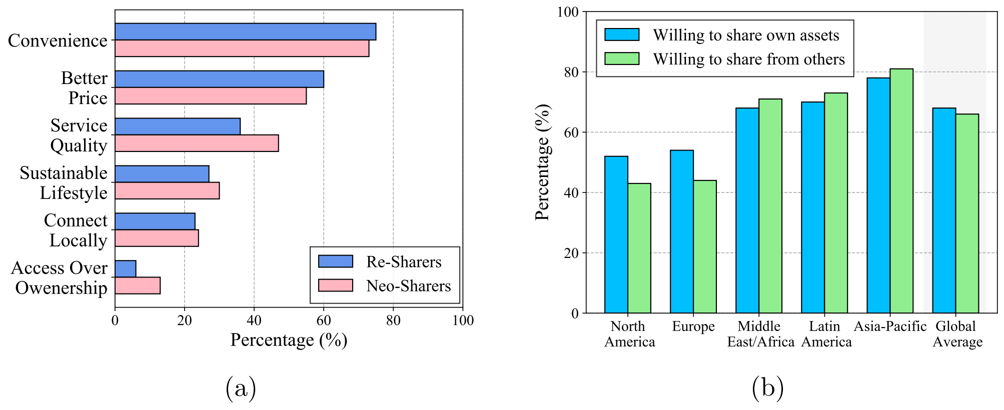


Figure 1.1: Figure 1.1a shows the reasons why users engage in resource-sharing communities. Re-shares are users that already buy/sell services using traditional platforms, and neo-sharers are users that consume emerging sharing services. Data collected and reported by Owyang et al. [Owyang et al., 2014]. Figure 1.1b shows the percentage of users willing to participate in resource-sharing communities based on the answers of 30,000 Internet respondents in 60 countries. Data collected and reported by Nielsen [Nielsen, 2014].

To illustrate the opportunities and challenges, consider a small tourist application example. Tourists and locals move toward different destinations in busy cities, such as Dublin, and hold various mobile devices such as smart phones, tablets, fitness trackers, GPS navigators, and other wearable sensors [Deng et al., 2017]. Because of their geographical position, the services offered by these devices commonly intertwine with services offered by the city's public transport or other road-side units. Buses and taxis supplied with navigators can offer transport services to users. Road-side units can provide tourist information or information about real-time obstacles or current local special offers. This myriad of services can form a virtual-service sharing community, which provides opportunities for complex service requests to be opportunistically solved. For instance, a tourist who is at the Science Gallery, and wants to find the fastest route to The Spire (a landmark in Dublin, Ireland), can use this service-sharing community to provision a smart, multi-modal, service-based route planner application [Palade et al., 2017]. Such an application is processed in a distributed way such that each mobile device executes their service according to a specific flow [Groba and Clarke, 2014, Chen et al., 2018]. The set of flows that can satisfy a user's request can be represented as a service dependency graph [Leung et al., 2010, Feng et al., 2013]. This process is explained in Section 2.1.2.

Figure 1.2 shows an example of a possible service dependency graph for such an application. Such a graph can be enabled by clustering the available services deployed on mobile devices within a limited geographic area in a Service-Specific

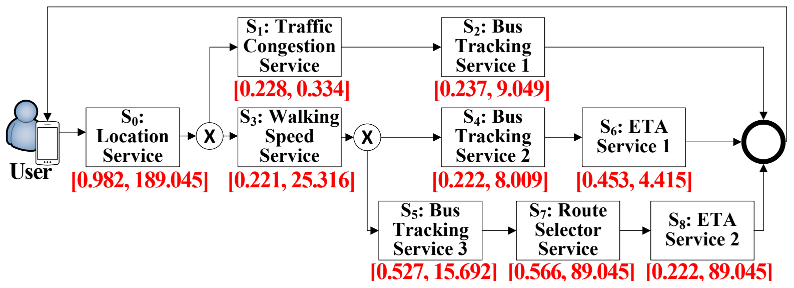


Figure 1.2: An example of a service dependency graph. Each node represents a service and an edge between two nodes represents a syntactic matching between the two services. The direction represents the execution order. The red labels under each node represents the QoS associated with that service, and has the format [Response Time (**RT**), Throughput (**Th**)].

P_i (Available Paths)	RT ($\sum RT_i$)	Th ($\min(Th_i)$)
P_0:{S_0→S_1→S_2}	1.447	0.334
P_1:{S_0→S_3→S_4→S_6}	1.878	4.415
P_2:{S_0→S_3→S_5→S_7→S_8}	2.518	15.692

Figure 1.3: Aggregated QoS of each service composition configuration (path) in the service dependency graph presented in Fig. 1.2.

Overlay Network (SSON) [Al-Oqily and Karmouch, 2011]. Goal-driven service composition using planning-based algorithms can automatically solve a user's request using the available services in these clusters [Chen et al., 2018]. This graph captures the *data-flow* and *control-flow* relations between the available services in the environment, where the former describes the input-output data dependencies between services, and the later how the execution order is performed [D'Mello et al., 2011]. In this example, three service composition configurations are available, which are represented as paths in this graph: P_0: $\{S_0 \rightarrow S_1 \rightarrow S_2\}$, P_1: $\{S_0 \rightarrow S_3 \rightarrow S_4 \rightarrow S_6\}$ and P_2: $\{S_0 \rightarrow S_3 \rightarrow S_5 \rightarrow S_7 \rightarrow S_8\}$. The output of a location service S_0 can be used as input to either a traffic congestion service S_1 (enabled by the local city council and deployed on stationary edge devices installed in bus stations) or a walking speed service S_3 (deployed on a fitness tracker). The output of S_1 can be used as input by the bus tracking service S_2 (enabled by a bus station or by buses equipped with an on-board computer, GPS navigation system and a radio which allows them to report their position [dub, 2018]) to estimate the arrival time of the next bus. The output of S_3 can be used as input to S_4 or S_5, which are another bus tracking services. These can be applications installed on users' mobile devices and offered as services to service consumers. The output flow of S_4 can then be used as input by S_6, an estimated time of arrival service deployed on a nearby mobile device. Similarly, the flow of S_5 is used by S_7 and then S_8. Exclusive choice structures such as guideposts [Chen et al., 2018] allow switching between different services (or paths in this graph) at runtime (e.g., switching between S_1 and S_3 after S_0).

In addition to the requested functionality, any consumer is likely to want a service along specific performance dimensions - for example, within a given cost level, and a minimum time delay, but with high availability. Dimensional qualities may conflict with one another in the real world. For instance, some service providers may charge more (monetary units) to provide services with low response time or high availability [Moustafa et al., 2016]. Each service in the service dependency graph from Figure 1.2 is initialised with two QoS values: response time (RT) and throughput (Th). If the QoS of the composite is not considered, then the optimal path is the shortest path (P_0 in this example). However, if the values of each QoS attribute for each service in the service dependency graph are considered, then P_0 is not the optimal path. Figure 1.3 shows the QoS of each possible path in the service dependency graph presented in Figure 1.2. Path P_0 has the best response time but the worst throughput, whereas, path P_2 has the worst response time, but the highest throughput. It is difficult to say which path is the optimal path without *a priori* preference articulation. Instead, both paths can be presented to the user who can explore the trade-offs between the QoS objectives. *A priori* articulation of user preferences may affect the flexibility of the composition and developing mechanisms for automatic service composition when the environment is dynamic, because it is difficult to analyse the trade-offs between multiple objectives and automatically adjust these weights at runtime [Chen and Bahsoon, 2017]. A *progressive* preference articulation approach through user feedbacks would also not be appropriate because of the increased resolution time in delivering the application to the user. In this context, techniques that minimise or avoid human interaction are preferred. In the *a posteriori* case, the decision maker is presented with a set of Pareto-optimal candidate solutions and chooses a solution from that set [Veldhuizen and Lamont, 2000]. The process of preference articulation with the user is outside the scope of this work.

While the QoS values of each service in the graph can generally be estimated using existing QoS prediction mechanisms [White et al., 2017b], finding valid paths in this dynamic graph is challenging. The QoS values of services may be time-dependent, or services may become un-available if service providers move out of range. In contrast to optimisation towards a static optimum, the goal in a dynamic environment is to track the dynamically changing optima as closely as possible [Blum and Li, 2008]. To describe the adaptivity in changing conditions, Mehboob et al. use the term "landscape" both as a metaphor where an entity needs to climb a landscape in pursuit of the highest peak, and as a mathematical object in which the value of a function maps to the elevation of the landscape [Mehboob et al., 2016]. This metaphor is represented through two landscape models: *rugged landscapes* and *dancing landscapes* [Page, 2010]. A rugged landscape is a landscape with many peaks, valleys and troughs. Such a landscape assumes that the fitness levels do not change. In a dancing landscape, local peaks may change, making a solution that was earlier optimal no longer a peak. Prior QoS optimisation methods for flexible service composition in mobile environments assume a static and well-known environment and are not suited for dynamic environments. In this work, the focus is more *satisficing*, which is the focus on finding sufficiently good solutions that satisfies one's purposes, rather than finding the best solutions.

1.3 Existing Solutions

The QoS-aware service composition problem is NP-hard [Strunk, 2010]. Given the user QoS objectives and constraints, the combinatorial explosion of services and devices in the environment that can execute those services introduces a large solutions space, and searching for the optimal plan is time-consuming. To address this problem, the QoS optimisation community has reduced the service composition process to a *QoS-aware service selection* problem [Yan et al., 2012, Trummer et al., 2014] where services are grouped in classes of equivalent functionality and are mapped to a predefined set of abstract tasks (i.e., user's request) [Jiang et al., 2014]. Each task corresponds to a class of services, and, for each task, a concrete service with optimal QoS is selected to implement the task [Zeng et al., 2004, Canfora et al., 2005a, Ardagna and Pernici, 2007, Calinescu et al., 2011, Deng et al., 2014]. Such an exactly-defined task request affects the flexibility of service composition because services (with potential better QoS) that are not outlined in the predefined abstract set of tasks, but could contribute to the user's request, are not used during composition [Chen et al., 2018].

Partial matching and planning-based techniques can be combined to address this functional limitation. Service composition includes searching the service dependency graph generated according to input/output relationship between services [Geyik et al., 2013, Jiang et al., 2014, Zou et al., 2014, Rodriguez-Mier et al., 2017, Chen et al., 2018]. To address the challenges associated with the environment's openness and dynamism, the existing methods employ flexible service composition for services maintained by different hosts and adaptable composites to create a new service composition when the network topology changes the QoS of services in the environment [Al-Oqily and Karmouch, 2011, Zou et al., 2014, Groba and Clarke, 2014, Chen et al., 2018]. The limitations of the existing representative solutions in the literature, are classified as follows: Single-Objective vs. Multi-Objective Optimisation, and Computational Efficiency and Optimality Trade-off.

1.3.1 Single-Objective and Multi-Objective Optimisation

The existing QoS-aware, flexible service composition proposals transform the multi-objective optimisation problem into a single-objective optimisation problem [Palade and Clarke, 2018, Palade et al., 2018b] by using a weighted average method (Section 1.1.2). This approach is restrictive in dealing with the QoS optimisation problem because (i) it is difficult to observe the trade-offs between multiple QoS objectives to allow the user to make compromises between the available solutions in the environment [Trummer et al., 2014]; (ii) the values of some of the QoS attributes can not be simultaneously optimised (e.g., improving the availability of a service composition configuration may imply an increase in cost) [Liang et al., 2018]; (iii) the *a priori* knowledge requirement about the QoS objectives' weights, can inhibit the development of automatic, planning-based service composition [Palade and Clarke, 2018, Palade et al., 2018b].

1.3.2 Computational Efficiency and Optimality Trade-off

The existing flexible service composition models have used two types of QoS optimisation mechanisms: exact and heuristics [Gabrel et al., 2018]. The exact approach has high computational complexity, and requires a complete (accurate) state of the environment to make decisions. The heuristic algorithms are generally fast but do not offer any optimality guarantees [Trummer et al., 2014, Gabrel et al., 2018]. Metaheuristics algorithms have been introduced to control the trade-off between computational efficiency and optimality. This type of algorithms generally use agent-based approach to explore the solutions space. However, several challenges still need to be considered. For instance, the current approaches require a centralised perspective, or have been implemented for template-matching composition (Section 1.3.1).

1.3.3 Research Gaps and Observations

Based on the service composition challenges introduced by a mobile environment (Section 1.1 and Section 1.2) and the limitations of the existing solutions to address these challenges (Section 1.3), the following research gap is observed. A large number of service composition proposals have been proposed for open and dynamic environments, however, these approaches are based on template-matching composition (introduced in Section 1.3.1), which is not flexible in dynamic environments. Dynamic and opportunistic environments can contain a large number of services and a common problem the lack of interoperability between the services. By imposing signature constraints between the abstract tasks provided by the user and the available services in the environment, many services that could be useful during the composition (with potentially better QoS) are not used, because they do not have the specified signature. The existing flexible service composition proposals do not allow the user to make compromises between multiple QoS objectives, and use optimisation methods that trade-off computational efficiency for optimality, or vice-versa.

1.3.4 Research Questions

In the context of the research gaps and observations of the limitation of existing research presented in Section 1.3.3, this thesis explores the question of how to search for service compositions options that can functionally satisfy a user's request while simultaneously guaranteeing user-acceptable QoS levels in mobile environments. This question can be decomposed into:

Q1 *Multi-objective optimisation.* For service compositions in mobile environments, to what extent can the trade-off between computational efficiency and optimality be controlled, such that a broad range of composition options can be presented to users?

Q2 *Adaptation.* To what extent can service providers' mobility, potentially resulting in new service options appearing in the environment, be leveraged through an adaptive approach to exploring new service compositions?

Q3 *Efficient Exploration.* To what extent will more search agents improve the optimality and diversity of identified service compositions?

The next section will introduce the approach used to answer these questions.

1.4 Thesis Approach

This thesis presents SBOTI (**S**tigmergic-**B**ased **OpTI**misation), a decentralised, nature-inspired optimisation mechanism designed to find a set of QoS near-optimal service composition configurations that may emerge in mobile environments as a result of service providers' mobility.

Assumptions. The following assumptions are made about the mobile environment and the service composition process performed in such an environment:

(A1) The physical environment is a limited geographic area such as a mall [Wang, 2011, Sadiq et al., 2015, Chen et al., 2018] or university campus [Deng et al., 2017] where users can form service-sharing communities using the services deployed on their mobile devices. Such communities are formed in highly dynamic environments using existing ad-hoc (infrastructure-less) wireless technologies (e.g., WLAN, BLE, NFC [Deng et al., 2016]) and do not require dedicated infrastructure to exchange services deployed on the participating mobile devices [Karmouch and Nayak, 2012, Cervantes et al., 2017, Liu et al., 2017, Mascitti et al., 2018]. A container with micro-service oriented architecture operates on each device and provides specific services [Liu et al., 2017]. Stationary, resource-rich entities such as the ones used for web services may relieve service providers from some of their duties, may not be available or accessible at all times [Chakraborty et al., 2005, Wang, 2011, Groba and Clarke, 2014]. The users are willing to share the services available on their mobile devices to participate as *helpers* in the composition process [Al-Oqily and Karmouch, 2011, Hachem et al., 2014, Georgantas, 2018, Peng et al., 2018]. These users consume their own resources such as battery and computing power. Incentive mechanisms can provide participants with enough rewards for their participation costs [Zhao et al., 2014, Wang and Du, 2016]. Such mechanisms are outside the scope of this work, and will not be discussed any further.

(A2) The available services are clustered based on their syntactic or semantic similarity using a Service-Specific Overlay Network (SSON) [Al-Oqily and Karmouch, 2011, Cabrera et al., 2017b, Cabrera et al., 2017a]. Goal-driven service composition using planning-based algorithms are used to automatically solve a user's request using the available services in these clusters [Chen et al., 2018, Cabrera et al., 2018b, Cabrera et al., 2018a]. The available service compositions are merged into a service dependency graph (e.g., Figure 1.2), and are modelled as a service dependency graph [Gu et al., 2008, Jiang et al., 2014]. This thesis assumes that the generated service compositions guarantee to provide the expected behaviour from a functional perspective. Several tools such as static analysis or model checking can be used to prove the functional correctness of these services [Hull and Su, 2005]. *Probability-free* or *probabilistic* techniques may be used to model the uncertainty in providers' mobility. A probability-free approach assumes that each service provider reports a time-window $[a_i, b_i]$ during which they may leave. In the probabilistic approach, *a priori* information about each service provider's estimated available time in the current environment is available and it is modelled using a random variable X_i [Wang, 2011].

(A3) There is no central repository that can offer access to the QoS of each service available in the environment. However, this information can be collected by each device that provides a service. More precisely, if a device hosts one or more services, the device can estimate the QoS of these services using techniques such as *sampling-based* methods through active execution monitoring [Cardoso et al., 2004, Wiesemann et al., 2008], or *collaborative filtering* procedures based on feedbacks from similar users [White et al., 2017b]. This work assumes that the service providers that rely on such devices have the mechanisms to advertise the values of the QoS attributes in a device-enabled, local registry [Liu et al., 2004]. This work deals with deterministic QoS values. Also, this work assumes that the QoS of available services is likely to change faster than the functional structure (i.e., service dependency graph) built on top of a service-sharing community to satisfy a user's request.

(A4) There is no *a priori* knowledge about the preference ranking of the QoS objectives. More precisely, there is no specification with regards to the weights of each QoS objective. For instance, if there are two QoS objectives, to minimise the response time and to maximise the reliability of the composition, there is no scalar to quantify how the "minimise the response time" objective relates to the "maximise the reliability" one [Trummer et al., 2014]. The elicitation of these preferences will happen after the user is presented with the set of identified solutions. This elicitation process will not discussed in this work. However, a set of user QoS objective weights will be used to allow performance comparison with the existing baseline proposals [Palade and Clarke, 2018, Palade et al., 2018b], which use a single-objective optimisation approach (Section 1.3.1).

Observations. The key driver for this research is the observation that limited attention has been paid to the existing QoS optimisation mechanisms for flexible service composition in mobile environments [Karmouch and Nayak, 2012]. The existing approaches are limited to best effort QoS. The mechanisms use either exact algorithms or heuristic-based approaches. Exact algorithms can find the optimal solution, but lack scalability because of their exponential complexity. These mechanisms generally use (greedy-based) heuristic approaches, which have low computational cost, 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 [Trummer et al., 2014]. While this is generally difficult in an open and dynamic setting such as the mobile environment, an approximate set of solutions that is as close as possible to the real set of optimal solutions can be provided to the user to allow him to make various compromises between the QoS values. The existing proposals require a centralised perspective, which, in a mobile environment, is a single point of failure or has the potential to cause processing and communication bottlenecks because of frequent state updates.

Hypothesis. This thesis investigates the QoS optimisation problem for flexible service composition in mobile environments. As opposed to the existing heuristic-based, best effort approaches, this thesis investigates whether an optimisation mechanism can be developed to be able to find service composition configurations with better QoS. The hypothesis is framed as follows: By using a decentralised, iterative, stigmergic-based mechanism that can control the trade-off between the computational efficiency of the execution and optimality of identified solutions, in a mobile environment, service composition configurations with higher utility than the ones produced by the existing heuristic-based proposals can be identified.

Basic Idea. The concept of opportunistic computing combined with flexible service composition in a mobile environment can improve the resilience of the service composition. Previous works such as Groba and Clarke [Groba and Clarke, 2014] and Chen and Clarke [Chen et al., 2018] focused on investigating failure rate of the composition by proposing various service discovery and composition mechanisms. These works focused on minimising the interactions with the service providers. The proposed models defer all interactions with the service providers until they are indispensable for the service composition to make progress in the execution. However, the existing investigations are limited to a functional level. These works have used (greedy-based) heuristics approaches for service selection, and make only best-effort guarantees. The model proposed in this thesis is a nature-inspired metaheuristic that can control the trade-off between computational efficiency and QoS, to optimise the QoS that can be achieved. While the main objective of flexible service composition in dynamic, opportunistic environments is to reduce failures, the proposed approach aims to improve the QoS that can be achieved at an acceptable communication overhead.

Objectives. Existing proposals for flexible service composition in mobile environments (e.g., Groba and Clarke [Groba and Clarke, 2014] or Chen and Clarke [Chen et al., 2018]) focused on functional requirements and single-objective optimisation. This work's high level research research objective is to investigate a QoS optimisation mechanism that can address the limitations of these systems as follows:

(O1) All currently available services should be considered when searching for the most appropriate service compositions. A user's request should be functionally solved using multi-granular, goal-driven service composition approaches. Syntactic, partial matching can automatically identify the available compositions in the environment.

(O2) Optimal utilisation of devices should be enabled. Services are likely to be deployed on battery powered devices. The overhead introduced by the model should be kept to an acceptable level to increase the network lifetime.

(O3) Device mobility should be supported. Provider devices may move out of range, which may cause an executing composition to fail. Service provisioning should remain seamless to the consumer, even under those conditions.

(O4) Multiple possible service composition configurations should be produced, to allow for exploration of QoS trade-offs, and any preference articulation process with the user should be done after the optimisation process.

1.5 Thesis Contribution

This thesis investigates the QoS optimisation problem for flexible service composition in a dynamic ad hoc environment while accommodating for changes in the operating environment. This research contributes to the body of knowledge by providing:

(C1) **Stigmergic-based QoS Optimisation Mechanism.** Finding QoS optimal service compositions in mobile-based, service-sharing communities is challenging because of the inherent dynamism in services deployed on mobile devices, and in the underlying physical network used to enable these services. Existing service composition proposals for such environments require *a priori* knowledge either about the service composition structure, or about the QoS objectives' weights, which limits the composition flexibility and does not allow for compromises to be made between multiple QoS objectives. This thesis proposes SBOTI, a nature-inspired, decentralised, iterative, QoS optimisation mechanism for service composition in mobile environments. SBOTI uses a community of homogeneous, mobile software agents, which share the same goal, to effectively and efficiently approximate the set of QoS-optimal service composition configurations available in a geographically-limited, mobile environment. The proposed mechanism uses a reinforcement-based approach to control the trade-offs between computational efficiency and the optimality of the identified service composition solutions. The proposed mechanism is inspired by existing swarm-based algorithms, which use the collective intelligence of a group of agents distributed in the environment to efficiently and effectively explore the solution space. SBOTI incorporates a non-dominated sorting technique to identify the Pareto-optimal set solutions, which allows the user to explore various QoS trade-offs. To control the diversity of the solutions in this set, SBOTI globally updates both dominated and non-dominated solutions using digital pheromones. This contribution is focused on answering research question **Q1**.

(C2) **Adaptation Mechanism.** New service composition configurations may emerge (with potentially better QoS) as a result of service providers' mobility. Most 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 and does not allow for compromises to be made between multiple QoS objectives. Also, the existing stigmergic-based service composition proposals do not allow for efficient exploration of the solution space. The digital pheromone mechanism used by such approaches may affect the exploration of new service composition configuration that may emerge as a result of service providers mobility. This thesis proposes an adaptation procedure to deal with the dynamic topology of the service providers deployed on mobile devices. To allow for exploration of new service composition configurations that may emerge as a result of providers' mobility, SBOTI uses an adaptation procedure that limits the amount of pheromone on previously identified solutions. This contribution is focused on answering research question **Q2**.

(C3) Collaborative Agent Communities. Given the dynamic nature of the environment under consideration for this work, re-optimisation needs to be performed to closely track any changes in the available service composition configurations. SBOTI's optimisation process requires a number of hyper-parameters to be configured, such as the number of mobile agents required to explore the search space, the initial pheromone level τ associated with each service agent, the evaporation frequency ρ parameter that allows new paths to be explored. An important limitation of metaheuristic-based algorithms is that these hyper-parameters need to be tuned, because a universally optimal parameter values set does not exist [Talbi, 2009]. A tuning process allows for a larger flexibility and robustness, but it is difficult to perform since it may require problem specific information, or *a priori* knowledge about the environment [Hussein and Saadawi, 2003]. This thesis proposes a collaborative approach to engage multiple communities of agents for provisioning QoS-optimal service compositions in mobile environments. New service compositions (with better QoS) can emerge from local decisions and interactions with agents from diverse communities. Several communities can independently tackle the optimisation problem in parallel, and maximise the explored/exploited search space. Each community searches and converges into different areas in the search space. When a dynamic change occurs, the multi-community approach has knowledge from a set of previously good solutions whereas a single community approach knows about only a single solution. This contribution is focused on answering research question **Q3**.

1.5.1 Evaluation

The performance and communication overhead of the proposed contributions is evaluated in Simonstrator [Richerzhagen et al., 2015], and the results are compared with baseline variants that are Dijkstra-based, (Greedy) Heuristic-based and Random. A prototype case study is proposed to demonstrate the feasibility of the proposed approach on simulated mobile devices. The metrics used for evaluation are the size of dominated space, the spread, the utility of the identified service composition configurations and the introduced communication overhead.

1.6 Thesis Scope

The goal of this thesis is to provide an optimisation method for approximating the set of QoS-optimal service composition configurations in a mobile environment. To achieve this, this thesis introduces SBOTI. The aim of this thesis is threefold. First, SBOTI uses stigmergy, a form of indirect communication through the environment to deal with the challenges introduced by the studied environment, and uses a multi-agent approach to deal perform efficient search of the solutions space. Second, this thesis proposes an optimisation for flexible service composition that can optimise multiple user objectives and produce a number of solutions to allow the user to make various QoS compromises. Third, the mechanism should allow for control of the computational efficiency and optimality trade-off. This should allow for more optimal or more diversified solutions to be identified if the user is willing to wait for a particular amount of time.

This thesis does not deal with the functional requirements of the composition. SBOTI takes as input a service dependency graph, which represents a set of solutions available that can functionally satisfy a user's request (e.g., Figure 1.2). Such a graph is built using a service discovery component [Cabrera et al., 2018a]. The available services are combined in an input-output service dependency graph, based on the (syntactic/semantic) interoperable relationship among available services. Each node in the graph corresponds to one service, and an edge is a syntactic matching between two connected services [Kalasapur et al., 2007, Jiang et al., 2014]. Planning-based algorithms are generally used to find paths in this graph using goal-driven approaches [Chen et al., 2018] that provide the required flexibility as to which services currently in the environment can be used in a service composition, and contribute to multiple possible configurations. This work assumes that such a component exists, and, once it initialises SBOTI with a new graph, this component can perform in parallel to SBOTI to continuously adapt this functional structure relative to the changing environment. The adaptation process is required because service providers are mobile. New services may become available at runtime, and existing services may become unavailable. Examples of how such a component can be built have been previously outlined in Chen and Clarke [Chen et al., 2018] or Groba and Clarke [Groba and Clarke, 2014]. Mascitti et al. [Mascitti et al., 2018] made a similar assumption. Previous work has investigated whether user feedback can improve the accuracy of the search results during the planning process [Cabrera et al., 2018a].

The incentives for mobile service providers to share their resources in service-sharing communities are not studied in this thesis. Different methods to encourage collaboration between available providers can be found in the literature (e.g., Li and Shen [Li and Shen, 2012]). Also, this thesis does not deal with the recourse value, which is the observed difference between the pre-invocation time and after-invocation time value of a certain QoS attribute. This issue is left as future work and will be discussed in the last section of this work.

1.7 Thesis Structure

The rest of this thesis is structured as follows:

- **Chapter 2 State of the Art** analyses how the state of the art optimisation mechanisms meet the challenges of mobile environments. In particular, the chapter explores the common types of performing service composition, the existing proposals for multi-objective optimisation, and how the existing approaches deal with the computational efficiency concerns.

- **Chapter 3 Design** returns to the challenges of QoS-aware service composition in mobile environments outlined in Chapter 1 and describes the design objectives, and design decisions of this thesis. Thereafter the chapter outlines how the proposed optimisation mechanism addresses the design decisions in detail.

- **Chapter 4 Implementation** outlines the implementation of the optimisation mechanism for QoS-aware, flexible service composition. The chapter describes how the integration with the Simonstrator enables mobility for composite participants. Then the chapter highlights the implementation details of the optimisation mechanism, and specifies the interaction between the actors used during the optimisation.

- **Chapter 5 Evaluation** evaluates how well SBOTI achieves its objective of finding QoS optimal service composition solutions compared to baseline approaches. The chapter first describes the experimental setup of the simulation-based study. The second part of the chapter presents and analyses the results showing that the proposed optimisation mechanism is a suitable alternative to existing optimisation mechanism for flexible service composition in mobile environments.

- **Chapter 6 Discussion and Conclusion** summarises the thesis and its achievements. The chapter then discusses the findings with regards to the proposed optimisation mechanism and highlights potential areas for future work. A final remark provides a succinct wrap-up of this work.

1.8 Chapter Summary

This chapter introduced the context of this research, together with the limitations of the existing works, the thesis approach and the thesis contributions. 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. Also, these proposals do not allow for efficient exploration of various QoS trade-offs between multiple service composition configurations available in the environment.

This thesis presents SBOTI, a decentralised, QoS optimisation mechanism for automatic, planning-based service composition. SBOTI uses a community of homogeneous, mobile software agents, which share the same goal, to effectively and efficiently approximate the set of QoS-optimal service composition configurations available in a geographically-limited, mobile environment. The proposed mechanism uses an iterative, reinforcement-based approach to control the trade-off between computational efficiency and the optimality of the identified service composition solutions. SBOTI incorporates a non-dominated sorting technique to identify the Pareto-optimal set solutions, which allows the user to explore various QoS trade-offs. To control the diversity of the solutions in this set, SBOTI globally updates both dominated and non-dominated solutions using digital pheromones. To allow for exploration of new service composition configurations that may emerge as a result of providers' mobility, SBOTI uses an adaptation procedure that limits the amount of pheromone on previously identified solutions. SBOTI also engages multiple communities, with diverse properties, to collaboratively address the computational efficiency and optimality concerns introduced by a single community of homogeneous agents. The performance of the proposed approach is compared with baseline variants, which are Dijkstra-based, (Greedy) Heuristic-based and Random.

Chapter 2

State of the Art

This chapter presents the background of this work and provides a review of the relevant literature. Existing surveys of service composition illustrate its maturity as an approach to building new applications in web [Lemos et al., 2016], cloud [Vakili and Navimipour, 2017, Hayyolalam and Kazem, 2018], and dynamic environments [Urbieta et al., 2008, Stavropoulos et al., 2013, Immonen and Pakkala, 2014]. Two methods are generally used to functionally solve a user's request: *Template-Matching* and *Flexible Service Composition.* In addition to the functional requirements, a service composition process needs to satisfy non-functional (QoS) requirements such response time or throughput of the final service composition configuration. Searching for service compositions that satisfy these requirements when many functionally-equivalent services are available can be formulated as a multi-objective optimisation problem, where each objective is associated with the value of a QoS attribute [Trummer et al., 2014].

This chapter is structured as follows: Section 2.1 defines the key SOA concepts, and Section 2.2 introduces the representative works for QoS optimisation in mobile environments. The discussion moves towards metaheuristic based proposals, and Section 2.2.5 introduces the stigmergic-coordination mechanism, which 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. This chapter culminates with a summary of the existing works, and an overview of the research gaps in Section 2.3.

2.1 Service Computing

Services computing is a design paradigm that provides a conceptual foundation for building software applications by re-using loosely-coupled software entities called *services* [Erl, 2005]. Services are self-contained software modules that perform pre-defined tasks. A service takes a set of inputs, performs a certain task, and produces a set of outputs. A set of non-functional Quality of Service (QoS) parameters (e.g., response time, throughput, reliability, availability) are associated with a service, which determine the performance of the service [Georgakopoulos and Papazoglou, 2008]. A single service may not be able to satisfy a user's request, and service composition is required. During service composition, a set of services are combined in a specific order based on their syntactic/semantic dependencies to produce the required outputs given a set of inputs [Groba and Clarke, 2014, Chen et al., 2018]. While providing a solution to a user's request, it is also necessary to satisfy user's end-to-end QoS requirements, which is the main challenge in QoS-aware service composition [Gu et al., 2003, Xiao and Boutaba, 2005, Estévez-Ayres et al., 2009, Liang et al., 2009, Wada et al., 2012, Ma et al., 2013, Zou et al., 2014, Chen et al., 2015b, Trummer et al., 2014, Jiang et al., 2014, Mostafa and Zhang, 2015, Wang and Du, 2016, Hashmi et al., 2016, Lemos et al., 2016, Wang et al., 2017, Rodriguez-Mier et al., 2017, Chattopadhyay and Banerjee, 2017, Mascitti et al., 2018, Gabrel et al., 2018].

This section extends the description of the key SOA concepts presented briefly previously in Section 1.1.1. A comprehensive list of the key principles of this paradigm have been presented in previous works such as Zhang et al. [Zhang et al., 2007], Georgakopoulos and Papazoglou [Georgakopoulos and Papazoglou, 2008], Papazoglou et al. [Papazoglou et al., 2008], O'Sullivan et al. [O'Sullivan et al., 2002], and Huhns and Singh [Huhns and Singh, 2005]. This work focuses on the two general types for performing service composition, which are *template-matching* and *flexible service composition* [Baryannis et al., 2008].

2.1.1 Key Concepts

Service. A service is a self-contained software unit that encapsulates a well-defined processing logic, which can be described by a set of functional and non-functional properties [Srinivasan and Treadwell, 2005]. A service consists of well-published, implementation agnostic interfaces that are loosely-coupled, promote re-usability and concepts such as location transparency. *Service providers* are entities offering services to *service consumers*. In the context of this work, heterogeneous resources available on devices can be abstracted as services to simplify the access and have platform independence [Sadiq et al., 2015]. A service is deployed and executes on a mobile device [Groba and Clarke, 2014, Chen et al., 2018, Mascitti et al., 2018].

Service Composition. Service composition facilitates seamless and flexible integration of applications from different providers [Cervantes et al., 2017]. Users do no have prior knowledge about the available services in the environment. Service composition promotes the quick creation of high level applications by aggregating already existing software services to provide complex functionalities that none of the services could provide by itself. Previous research efforts have undertaken two orthogonal directions: manual or automatic [Ko et al., 2008]. Manual service composition is performed by a developer which chooses the outsourced services that are relevant to the user request, and programs the interaction logic between the services using using a programming language such as BPEL. This approach is ad-hoc and time consuming, and does not scale with the number of service composition, or with the number of service composition requests. The automated approach has been introduced to address this limitation. This is combination done through the combination of syntactic/semantic matching and planning-based algorithm. Goal-driven service composition is used to identify the available service composition configurations [Chen et al., 2018].

Quality of Service. Non-functional requirements (or QoS) can be considered as constraints over the functionality of a service and can be used to differentiate between multiple functionally-similar services [O'Sullivan et al., 2002]. Service quality is used to define a contract between a service consumer and a service provider to guarantee that their expectations are met [Kritikos et al., 2013]. This contract is used by a service management system to assess whether the requested quality levels during service execution are achieved, and to enforce appropriate adaptation actions if these cannot be satisfied by the existing service providers [Wada et al., 2012]. Such actions could be to increase the underlying service resources, to substitute or recompose the faulty service, and, where a Service Level Agreement (SLA) needs to be enforced, to determine which settlement actions apply based on the actual quality values delivered [Duan et al., 2003]. These could be penalties to be paid by the service provider, or re-negotiation for contract termination [Yan et al., 2007, Kritikos et al., 2013, Lemos et al., 2016]. Table 2.1 shows examples of some of the widely proposed QoS attributes and their aggregation formulae for sequential service composition configurations [Canfora et al., 2005b, Cardoso et al., 2004].

QoS Attribute	**Unit**	**Sequence Pattern**
Response Time (q_{rt})	msec	$\sum_{i=1}^{n} q_{rt}(f_i)$
Latency (q_l)	msec	$\sum_{i=1}^{n} q_l(f_i)$
Price (q_p)	per invocation	$\sum_{i=1}^{n} q_p(f_i)$
Availability (q_{av})	percent	$\prod_{i=1}^{n} q_{av}(f_i)$
Reliability (q_r)	percent	$\prod_{i=1}^{n} q_r(f_i)$
Accuracy (q_{ac})	percent	$\prod_{i=1}^{n} q_{ac}(f_i)$
Throughput (q_{tp})	invocation/sec	$min(f_i, \ldots, f_n)$

Table 2.1: QoS aggregation formulae for service composition configurations using Sequence Pattern, which is the main service composition pattern considered in this work (Section 1.1.1).

As introduced in Section 1.1.1, QoS attributes can be *positive* or *negative*, where the value of a positive attribute should be maximised (e.g. throughput and availability), and the value of a negative attribute should be minimised (e.g. price and response time) [Alrifai and Risse, 2009]. These attributes can also be categorised into *deterministic* and *non-deterministic*. Deterministic attributes are those for which values are known before the service invocations (e.g., price of using a service or other security properties). Non-deterministic attributes are those whose values are unknown at service invocation time (e.g., response time or availability) [Liu et al., 2004]. This description is extended in Section 2.1.3.

2.1.2 Methods for Addressing Functional Requirements

Service-oriented architectures have been used for designing and deploying mobile and pervasive computing environments, to facilitate user tasks that require a number of resources that may be spread over the networked environment. If the user requirements are known a priori, the required resources can be tuned for user access when desired. However, such a static design of systems limits their possible usage by other user tasks. Because of the growing applications of pervasive computing, there is a need to provide support to user tasks in the face of dynamic challenges such as heterogeneity, resource restrictions, and mobility. Service-oriented environments promise flexibility in terms of user support, as well as better resource utilisation. By modelling the available resources as services and by designing a mechanism to access the available services, a pervasive computing environment can be effectively transformed into a service- oriented environment. Further, by creating access mechanisms that can dynamically use the available services in an efficient way to provide the best possible support for user tasks, guaranteed results can be delivered [Kalasapur et al., 2007].

One of the most promising features of the service-oriented paradigm is *run-*

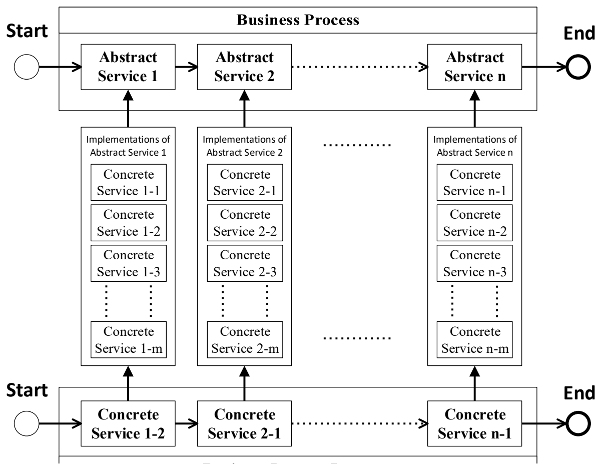


Figure 2.1: Template-matching service composition example.

time service discovery and *late-binding* [Canfora et al., 2005b]. Runtime service discovery implies that a workflow or program contains a list of abstract tasks that need to be matched to *concrete services.* Late-binding mechanisms select those services that best contribute to maximise a set of *local* and *global* objectives. Local objectives refer to single service selection. Global objectives refer to composite level selection. From a functional perspective of how the users' requests are provisioned, the general approaches for performing service composition can be divided in *template-based* and *flexible service composition* [Wang et al., 2017].

Template-based

Service composition selects and interconnects services offered by different service providers on the basis of a specified business process. Such a business process is generally represented using a workflow language for web services such as WS-BPEL [Weerawarana et al., 2005, Oasis, 2007], OWL-S [Martin et al., 2004] or BPML [Model, 2011] to define the data dependency and control flow between tasks. As example, Cardoso et al. [Cardoso et al., 2004] developed METEOR-S system, to compose services. The users' queries are used to logically build the service composition description using a BPEL-based language. This description does not contain any invocable service, as this process is left for execution time, where services are matched to abstract tasks.

Template-based service composition is based on an abstract composite service, consisting of a set of abstract services controlled through workflow patterns. This is abstract composite is instantiated and executed at runtime by binding abstract services to concrete services. As mentioned earlier, the dynamic binding ensures a loose coupling of services [Wu et al., 2016]. The optimisation process is transformed into a template-matching approach where the QoS optimal services are allocated to each abstract service. Figure 2.1 shows an example of template-matching service composition. In this example, the business process, which represents the formalised functional requirements of the application, is specified as a set of abstract services (or tasks). This specification of the set of abstract tasks is provided *a priori* by the user [Kalasapur et al., 2007]. Examples of works that use this approach are Zeng et al. [Zeng et al., 2004], Canfora et al. [Canfora et al., 2005a], Ardagna and Pernici [Ardagna and Pernici, 2007], Calinescu et al. [Calinescu et al., 2011], Deng et al. [Deng et al., 2014], Al Ridhawi and Karmouch [Al Ridhawi and Karmouch, 2015], Mostafa and Zhang [Mostafa and Zhang, 2015], and Wu et al. [Wu et al., 2016]. The description of these works is extended in the next section from a non-functional perspective. The common lifecycle of the template-matching service composition approach consists of three phases:

- *Composition Phase*: This phase deals with synthesising the service composition schema [Baryannis et al., 2008]. Given a user's request, the service composition schema designer formalises this request to build the service composition schema. This represents the description of the *business process* that needs to be performed [Zeng et al., 2004]. This business process consists of a set of *abstract tasks.* This schema also contains the data-flow and the control-flow constructs to specify how the order of the abstract tasks should be executed. This composition schema is generally manually constructed.

- *Selection Phase*: For each task from the set of abstract tasks, a set of *concrete services* is identified from the service registry after the schema is created. The selection phase finds and matches the advertised composite service specification to the available service implementations [Jiang et al., 2014]. This follows either a *static* or a *dynamic* procedure. In a static approach, the specific services are known in advance, and are matched to the service composition specification at design-time. In a dynamic approach, the service composition specification is matched with the available services at runtime. The static approach is preferred when the business process and the available services rarely change. However, when the service composition requirements change, or the environment is continuously changing, dynamic approaches are preferred [Chen et al., 2015a].

- *Execution Phase*: In this phase an executable instance of the composite service is created. The composite service instance is then invoked by the end user using a service execution engine. The common tasks performed by such an engine are logging, execution monitoring, performance measuring and exception handling [Baryannis et al., 2008].

An important limitation of this approach is that it requires *a priori* knowledge about user tasks or about the environment. In a mobile environment, this knowledge may not be available. Because of the mobility of service providers, services that may be available during the discovery phase, may not be available at runtime, or vice-versa. Also, this approach requires that the user has complete knowledge about the specification of available services. Such an exactly-defined request reduces the flexibility of service composition and limits the capability of developing automatic service composition approaches to cope with the challenges introduced by the environment [Kalasapur et al., 2007].

Flexible Service Composition

Open and dynamic environments such as mobile environments require flexible and automatic service composition approaches. A number of surveys capture the approaches that can support automatic service composition such as Rao and Su [Rao and Su, 2004], Küster et al. [Küster et al., 2005], and Alférez et al. [Alférez and Pelechano, 2017]. This work summaries the general approach of how this is performed. These approaches combined partial matching and AI planning to identify the service composition configurations.

An AND/OR graph structure can be used to represent the set of possible alternative service composition configurations. Alternative configuration routes are denoted by an OR-subgraph to indicate that only one of the alternative paths needs to be visited. A dummy node is used to merge all the service composition configurations [Leung et al., 2010]. In this work, this structure is called a service dependency graph [Palade and Clarke, 2018, Palade et al., 2018b]. Examples of works that present how such a structure is built are Callaway et al. [Callaway et al., 2010], Groba and Clarke [Groba and Clarke, 2014], Lv et al. [Lv et al., 2015], GoCoMo [Chen et al., 2018] and Mascitti et al. [Mascitti et al., 2018]. These works use distributed service discovery mechanisms that use the local registries deployed on mobile devices to build this service dependency graph structure. Section 2.2 discusses such approaches in detail, and presents the existing QoS optimisation mechanisms used to find QoS optimal solutions in such proposals.

2.1.3 QoS Requirements

QoS requirements can be considered as constraints over the functionality of a service or a composed service. These non-functional properties are then used to differentiate between such multiple candidate services. The notion of QoS-awareness is being aware of the user-required QoS and the QoS required by various resources at the user's location and time. This may require having full visibility of QoS variations, and being able to perform adaptation countermeasures to deliver the required QoS requirements.

Multi-Objective Optimisation (MOO) is defined as the simultaneous optimisation of a set of objective functions [Deb, 2014]. A simple example is commuting between one's home and workplace. Given such a task, a user's solution might need to meet several objectives to maximise the level user's satisfaction relating to his choice. The first objective is to minimise monetary cost (e.g., fuel use, bus fare, tram fare, etc.). A second objective is to minimise the travel time. Additional objectives may be related to the level of comfort, safety or environmental impact. For many situations there is no simple feasible solution which simultaneously optimises all objectives and a trade-off or compromise solution must be selected. When constraints imposed, the set of feasible solutions might be null or have only a small number of elements. When no constraints are imposed, the set of possible solutions increases significantly [Angus and Woodward, 2009]. With respect to the preference elicitation of the decision maker, multi-objective optimisation problems are generally approached in three ways in the literature:

- *A priori* preference articulation: The user specifies a set of weights, which represent his preferences for each objective. This process is done through a utility function. The system identifies the configurations that maximises the value of this utility function (e.g., weighted-sum approach or lexicographic approach [Freitas, 2004]).
- *Progressive* preference articulation: The user and the system work, at run-time, through a number of iterations and user feedbacks, to identify the user preferences.
- *A posteriori* preference articulation: The system computes a list of Pareto

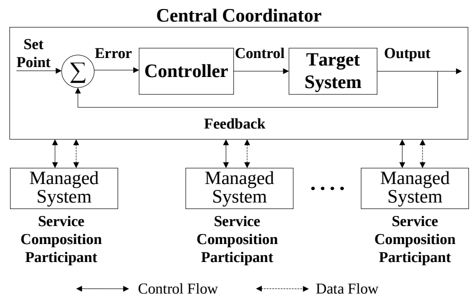
optimal solutions and presents it to the user. The user can use this list to explore various compromises between the objectives (e.g., Pareto approach [Trummer et al., 2014]).

Services operate autonomously in variable environments, and, as a consequence, their QoS continuously changes (e.g., higher system loads due to large number of requests). Composition participants may change their QoS, others may become unavailable, or others may emerge [Zeng et al., 2004]. This is the case especially in a mobile environment, where service providers rely on mobile devices to provide their services. Such an environment transforms the QoS-optimisation into a dynamic optimisation problem. Dynamic optimisation problems generally involve changes in the parameters of objective functions, domain variables, and the number of considered variables, and also whether the changes are cyclic/recurrent in the search space or not. Yang et al. [Yang et al., 2013] provide a complete list of parameters that characterises such a problem.

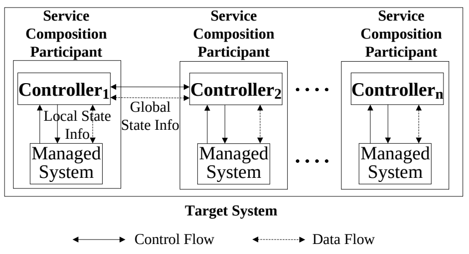
From a functional perspective, most existing service composition proposals focused on how to offer support for automatic service composition to create dynamic service composition configurations using the available services deployed on mobile devices with minimal or no human intervention. However, the existing proposals have paid little or no attention to the quality of service aspect. Section 2.2 presents how the existing proposals addressed this type of requirements.

2.2 QoS-Aware Service Composition

QoS-aware service composition problem can be modelled using a combinatorial model or using a graph model [Yu et al., 2007]. In the combinatorial approach, the optimisation problem is formulated using as a Multi-Choice 0-1 Knapsack Problem (MMKP). In the graph model case, the problem is formulated as a Multi-Constraint Optimal Path. QoS-aware service composition has used either centralised or decentralised service composition models to facilitate the adaptation of the service configurations at runtime [Ye et al., 2016]. Centralised service composition models generally use a feedback loop controller to measure, at regular intervals, the utility of the current service composition configuration [Chen et al., 2015a]. Changes to this configuration are made according to how this utility value changes over time. Trade-off analysis is used to measure which service compositions configurations satisfy user requirements. Decentralised service composition models use a network of decentralised controllers to collect and



(a)



(b)

Figure 2.2: Figure 2.2a shows a centralised approach to service composition process. Figure 2.2b shows a decentralised approach to service composition process.

disseminate information about the available service providers [Cardellini et al., 2012]. These decentralised approaches use global information for service composition. For example, a service consumer is allowed to select services in the whole environment, which implies that a service consumer knows all the service providers in the environment. The use of global information is infeasible in large environments. To avoid using global information, some decentralised approaches were developed in networked environments, where each participant has a set of acquaintances and each participant has knowledge only about these acquaintances [Ye et al., 2016].

Adaptation decisions are managed using a decision mechanism, which is responsible for adapting a service configuration based on the conditions of the environment. These type of mechanisms have two parts: the managed system and the managing system. In a service composition system context, the managed system deals with the composition functionality, and the managing system deals with the adaptation of the managed system to achieve particular (quality) objectives. The managing systems are implemented using a controller [De Lemos et al., 2013] and can be divided in two categories based on their distribution: centralised and decentralised [Patikirikorala et al., 2012]:

1. *Centralised Controller*: This type of controller uses a central coordinator which has a global view of the entire composed system to make adaptation decisions. The most common type of controller is a feedback control loop, which uses a fixed-gain control scheme. The main controller uses the difference between the output of the system and the set point, in order to adjust the target system (Figure 2.2a) [Chen et al., 2015a]. QoSMOS [Calinescu et al., 2011] use this type of controller to find another service mapping for the current process configuration or to find another process configuration

when the available service providers can not provide the required QoS at the process level.

2. *Decentralised Controller*: This type of control is achieved through the coordination of service providers controllers (Figure 2.2b). This coordination is achieved by propagating the global state information of the composition between the composition participants. GoCoMo [Chen et al., 2018] and Mabrouk et al. [Mabrouk et al., 2009] use this type control to improve the resilience of composed services in dynamic and uncertain environments.

The rest of this work focuses on the graph model, as the combinatorial problem is can be considered as an extension to this model. The combinatorial model assumes that the service dependency graph (e.g., Figure 1.2) is created, and the available services are to be matched to services available in the environment [Jiang et al., 2014].

2.2.1 Exact Algorithms-Based Proposals

Dijkstra's algorithm is a graph-based search algorithm that solves the single source shortest path problem for a graph with non negative edge costs. The algorithm maintains the length d of the shortest path from the source node s to another node v in the graph and the predecessor $p(v)$ of v on the path for every node in the graph. The algorithm can be summarised as follows: initially $d(s) = 0$, $d(v) = \infty$ for all other vertices and $p(v) = null$ for all v. A queue of unvisited vertices with finite d values is maintained. The algorithm extracts the minimum valued node from the queue and scans it. If v is the queued node with smallest distance, and $d_e(v, u)$ the distance from v to a neighbour u, for each neighbour u of v the algorithm looks at all edges $(v, u) \in E$ and if $d(u) < d(v) + d_e(v, u)$, sets $d(u) = d(v) + d_e(v, u)$ and $p(v) = u$. The algorithm terminates when the target t is extracted. The bidirectional version of Dijkstra's algorithm is similar, running a forward search from s and a reverse search from t. If an edge (v, u) is scanned by the forward search and u has already been scanned by the reverse search, the concatenation of paths $s - v$ and $u - t$ is a new path P from s to t. Dijkstra's original algorithm has high computational cost and runs in $O(|V^2|)$. This algorithm commonly implemented in link-state routing protocols.

GraphPlan [Yan et al., 2012], SimDijkstra [Jiang et al., 2014], and Rodriquez-Mier et al. [Rodriguez-Mier et al., 2017] presented a Dijkstra-based algorithm to find composition paths that satisfy users' QoS objectives, subject to constraints. To improve the resilience of the composition, Jiang et al. [Jiang et al., 2014] return the top-k optimal paths. However, these works do not address the dynamic nature of a mobile environment. They assume that the services are static. This means that the service dependency graph does not change over time. To address this limitation, SimDijkstra-CQ, an extension of the SimDijkstra model, that uses a continuous quary algorithm to keep track of the functional structure of the service dependency graph [Lv et al., 2015].

While the Dijkstra-based algorithms may have polynomial time to search a graph, building and maintaining an up-to-date view of the service components of the service dependency graph requires a high communication overhead [Gabrel et al., 2018].

2.2.2 Heuristics-Based Proposals

GoCoMo [Chen et al., 2018] and Kalasapur et al. [Kalasapur et al., 2007] 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. Also, Kalasapur et al. [Kalasapur et al., 2007] used A*-search (A*-prune) to identify a service service composition configuration that satisfies the constraints. An important limitation of the existing flexible service composition proposals is that they use a utility function to identify a single service composition configuration that maximises user satisfaction. As mentioned in Section 2.1.3, such an approach requires *a priori* preference elicitation of a user's preferences to produce a single solution. This approach does not allow the user to perform trade-offs between multiple QoS objectives. In addition to this, such an approach does not support automatic service composition. The relative weights presented by the user may need to be adjusted at runtime based on the available solutions in the environment [Chen and Bahsoon, 2017].

2.2.3 Metaheuristics-Based Proposals

A mobile environment introduces a number of constraints on the available service composition solutions. Because of the mobility of service providers, an algorithm that needs a global (complete) view of the environment to search for solutions may introduce considerable overhead. Also, the QoS of services available is continuously changing, which exacerbates this problem. Most proposals focus on the functional requirements on the composition and consider only best-effort solutions in terms of QoS. However, this work explores whether better solutions can be obtained.

Metaheuristics have been designed to address challenges that are common in large-scale, dynamic and decentralised networks [Di Caro et al., 2008]. 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. As opposed to the exact (Section 2.2.1) and heuristic-based (Section 2.2.2) proposals, these algorithms can balance the trade-off between computation efficiency and optimality [Sim and Sun, 2003], and can obtain a set of solutions having two important properties: good convergence and diversity in solutions.

A multi-objective metaheuristic contains three main search features [Talbi, 2009]: fitness assignment, diversity preservation and elitism. Fitness assignment is a search procedure that is used to guide a search toward Pareto-optimal solutions for a better convergence. Diversity preservation is used to ensure the diverse set of Pareto solutions in the objective and/or decision space. Elitism is the preservation and use of only elite solutions (e.g., Pareto-optimal solutions), which allows for a robust, fast, and monotonically increasing performance of the search. The fundamental properties by which metaheuristics are characterised [Alba, 2005] are:

- The metaheuristics are used as strategies to "guide" the search process with the goal to efficiently explore the search space for (near-) optimal solutions.
- The metaheuristics algorithms are approximate and usually non-deterministic.
- The metaheuristics may employ mechanisms to avoid getting trapped in confined areas of the search space.
- The metaheuristics are not problem-specific.
- The metaheuristics make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
- Search experience (embodied in some form of memory) is used to guide the search.

A metaheuristic is successful on a given optimisation problem if it can provide a balance between the exploration (diversification) and the exploitation (intensification) phases. In the exploration phase, the parts of the search space with high quality solutions are identified. Exploitation is important to intensify the search in some promising areas from the accumulated search experience. The main differences between the existing metaheuristics concern the way in which they try to achieve this balance [Alba, 2005, BoussaïD et al., 2013].

Nature-inspired approaches have been introduced to address the challenges that are common to large-scale networks [Talbi, 2009]. Such networks are characterised by complex and heterogeneous architectures, a dynamic and self-organizing nature, resource constraints, and the absence of a centralised control and infrastructure [Dressler and Akan, 2010]. In this context, swarm-based networking and communication algorithms such as Ant Colony Optimisation [Stutzle and Dorigo, 2002] mimic the laws and dynamics that govern biological systems. The following features of swarm-based algorithms make such mechanisms appealing for the development of optimisation mechanisms in challenging environments [Bello and Zeadally, 2016]:

1. *Adaptability to changing environmental conditions.* These systems can learn and evolve when the properties of the environment change. They can deal with the issue of unpredictability of service providers or the available resources that can be used at runtime [Blackwell and Branke, 2006, Wang and Shen, 2017]. These systems facilitate collaboration and self-organisation among participating agents, without using a central coordinator.

2. *Efficient management of resource-constrained devices.* The foraging process used in Ant Colony Optimisation techniques has inspired many resource efficient techniques that report high resource efficiency [Dressler and Akan, 2010]. In the context of this work, service providers rely on mobile devices to deliver their services, and require a resource-efficient mechanism to distribute information about their services because they are battery-powered.

Examples of nature-inspired algorithms include evolutionary-based algorithms such as Genetic Algorithms (GAs) [Wada et al., 2012] or swarm-based algorithms such as Particle Swarm Optimisation (PSO) [Hossain et al., 2016] or Ant Colony Optimisation (ACO) [Moustafa et al., 2016]. The GAs belong to the larger class of evolutionary algorithms, which generate solutions to optimisation problems using techniques such as *selection*, *cross-over* and *mutation*. Swarm Intelligence (SI) is based on the observation of the collective behaviour of decentralised and self-organized systems such as ant colonies, flocks of fishes, or swarms of bees or birds [Di Caro et al., 2004].

The main indicator to evaluate the performance of an exact optimisation method is the efficiency in terms of search time as they guarantee global optimal solutions. To evaluate the effectiveness of metaheuristic-based methods, the following groups of quality indicators are generally used: quality of solutions, computational effort, and robustness. In addition to these groups, qualitative criteria such as the development cost, simplicity, ease of use, flexibility, and maintainability may be used [Talbi, 2009].

Genetic Algorithms-Based Approaches

A Genetic Algorithm (GA) is a nature-inspired metaheuristic computational method that imitates the robust procedures used by various biological organisms to adapt as part of their natural evolution [Mehboob et al., 2016]. The adaptation is done through processes such as natural selection, survival-of-the-fittest, reproduction, mutation, competition and symbiosis [Kulkarni et al., 2011]. Genetic Algorithms (GAs) algorithms have been successfully used in fields such as chip design, computer animation, telecommunications and financial markets [Goldberg, 2006]. In the context of QoS optimisation for service composition, a number of mechanisms have been proposed.

Canfora et al. [Canfora et al., 2005b] introduced GAs for solving the QoS-aware service composition problem by finding a set of services that satisfy QoS constraints and optimise the QoS of the composite. Each service composition configuration is encoded as a chromosome, and a fitness value is calculated based on the aggregated QoS of the composite service. GAs are iterative. In each iteration, a new generation of service composition configuration is generated. The fitness value increases from generation to generation, and the fittest chromosome in each generation represents the QoS-optimal service composition configuration. An extension of this work compares the results of this approach with MIP [Canfora et al., 2005a]. Also, Jaeger et al. [Jaeger and Mühl, 2007] reproduce these works and propose a different set of parameters to initialise the algorithm. They also extend the baselines evaluations to highlight the performance of their proposed approach.

SanGA [Klein et al., 2014] use a genetic algorithm to find the best candidates for a set of abstract tasks. The authors use a template-based service composition approach. They also integrate the network QoS by using the shortest distance between two network location points. Wu et al. [Wu et al., 2016] developed a QoS-aware multi-granularity service composition model, and use Genetic Programming (GP) to encode the available service composition configurations. These procedure require a centralised perspective of the environment. The composition follows a template-based approach.

Swarm Intelligence-Based Approaches

Swarm Intelligence (SI) is a distributed intelligence paradigm for solving optimisation problems that takes inspiration from the collective behaviour of a group of insect colonies or animal societies [Saleem et al., 2011, Karaboga et al., 2014]. SI systems are made up of a population of simple agents (an entity capable of performing/executing certain operations) interacting with one another and with the environment. These entities with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. Although there is no centralised control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global and self-organised behaviour.

Several optimisation algorithms inspired by the metaphors of swarming behaviour in nature have been proposed: Ant Colony Optimisation (ACO) [Dorigo and Stützle, 2004], Particle Swarm Optimisation (PSO) [Yin et al., 2014], Bacterial Foraging Optimisation (BFO) [Das et al., 2009], Bee Colony Optimisation (BCO) [Karaboga and Basturk, 2007], Artificial Immune System (AIS) [Hofmeyr and Forrest, 2000] and Biogeography-Based Optimization (BBO) are examples to this effect [BoussaïD et al., 2013, Gui et al., 2016, Patnaik et al., 2017]. With respect to a mobile, dynamic environment, the ACO-based algorithms present several advantages compared to the other approaches [Cobo et al., 2010]:

- ACO is fully distributed with no single point of failure.
- The operations, which need to be performed at each node, are simple. Limited storage is required, which is suitable for resource-constrained devices.
- The algorithm is based on agents' asynchronous and autonomous interactions.

Algorithm 1 General ACO Procedure

```
1: return Best solution found
2: while termination conditions are not met do
3:     ConstructSolutions()
4:     UpdatePheromone()
5:     DaemonActions() {Optional}
6: end while
```

- It is self-organising and resilient. Path recovery algorithms are generally not required.
- It inherently adapts to network traffic without requiring complex, and inflexible metrics.
- It adapts to all kinds of long-term variations in topology and traffic demand, which are difficult to consider in deterministic approaches.

Compared to other metaheuristics, an important aspect in ACO is the differential path length (DPL) effect of ants, which means that ants that travel on the shorter paths can deposit pheromone before ants travelling on longer paths [Dorigo and Stützle, 2001]. This allows faster convergence on the optimal path. The decentralised and asynchronous nature of ants is important in solving distributed problems where there is no global view of the objective function. Decisions must be taken under a local view of the problem [Talbi, 2009]. ACO is a metaheuristic approach that was inspired by the foraging behaviour of ants in an ant colony. This behaviour enables ants to find the shortest paths between their food sources and their nests. This is enabled by stigmergy, which is a form of indirect communication between ants through the environment, realised by depositing a substance called a pheromone on the path. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide the other ants to the food source. This functionality of real ant colonies has been exploited in artificial ant colonies in order to solve multi-objective optimisation problems [Alba, 2005, Farooq and Di Caro, 2008]. Algorithm 1 shows the basic steps of the ACO procedure.

Previous ACO-based proposals have addressed multi-objective optimisation problem either through a single pheromone matrix or with multiple pheromone matrices [Angus and Woodward, 2009]:

1. *Single.* One pheromone matrix associated with each ant colony. Based on the qualities of a solution, the entry in this matrix, that is associated with this solution is reinforced.

2. *Multiple.* Several pheromone matrices, each associated with one objective. This model is associated with multiple ant colonies. Each colony is focused on optimising a particular objective.

There are different types of ant colony optimisation methods: Ant System (AS), Elitist Ant System (EAS - each ant that finds a better solution has the chance to deposit more pheromone), Ant-Q (where the deposited pheromone amount is directly proportional to the quality of the found solution), Max-Min Ant System, Rank-Based Ant System. In Max-Min System the pheromone level is kept between τ_{max} and τ_{min}. In MMAS, only the best ant in iteration or the best ant so far is permitted to deposit the pheromone. Initially, before the search begins, the pheromone intensity of all the ways be maximum value and updated only through a pheromone evaporation operation. In Ant System (AS), the ants use a local updating policy for the amount of pheromone when they construct a tour (solution). After all the ants have finished their tour, a global updating policy is applied to modify the pheromone level on the edges that belong to the best ant tour [Angus and Woodward, 2009].

Like all the evolutionary approaches, ACO provides the means of exploring parts of the solutions space that offer optimal/near optimal solutions [Chitty and Hernandez, 2004]. This procedure is performed by creating a trade-off between the conflicting aims of *exploration* and *exploration.* Exploration is defined as the search through the space of possible solutions to find optimal or near-optimal solutions. Exploitation is the fusion of information regarding the quality of identified solutions at a particular moment to focus on promising areas of the search space. This process is enforced through a reinforcement phase where the pheromone trail is updated according to the generated solutions. Three different strategies may be applied [Talbi, 2009]:

1. Online step-by-step pheromone update: The pheromone trail τ_{ij} is updated by each ant, at each step of the solution construction.

2. Online delayed pheromone update: The pheromone update τ is applied after the solution is constructed.

3. Offline pheromone update: The pheromone update τ is applied after *all* ants generate a complete solution. Different strategies are based on this approach: quality-based pheromone update (updates the pheromone value associated with the best found solution among all ants), rank-based approach, worst pheromone update, elitist pheromone update.

2.2.4 Other Proposals

Yu and Lin [Yu et al., 2007] propose two models for service composition: a combinatorial model and a graph model. In each model, two algorithms are introduced to find the optimal service composition configuration. All the four algorithms are compared and an usage context for each algorithm is suggested.

Zeng et al. [Zeng et al., 2004] propose two optimisation models: local optimisation and global optimisation. The local optimisation is required to satisfy local QoS constraints, but cannot guarantee global optimality. The global optimisation mechanism is used to seek globally-optimal components for service composition by using a Mixed-Integer Programming (MIP) approach. However, this approach suffers from poor scalability because of its exponential computational complexity and is impractical for real-time and dynamic applications [Wu and Zhu, 2013]. To address the scalability challenges, Alrifai et al. [Alrifai and Risse, 2009] propose a preprocessing step to delete low quality service candidates. The proposed mechanism combines global optimisation with local selection techniques to adapt in distributed environments. MIP is also employed to find the best decomposition of global QoS constraints into local QoS constraints and then local selection is performed to find the best services satisfying these local constraints for each abstract task in the user's request. The optimisation problem is modelled as an extended flexible constraint satisfaction framework and a heuristic algorithm based on branch and bound is proposed to solve it. However, these approaches require a template-matching service composition model.

Liu et al. [Liu et al., 2017] propose a three-staged distributed approach for finding service composition configurations in mobile environments. The service dependency graph is decomposed into multiple Independent Function Flows, and each sub-solution is explored in a distributed manner. The authors use Dynamic Programming (DP), and propose a branch-and-bound approach to search for the optimal service composition configuration. However, the authors transform the multi-objective optimisation problem into a single objective optimisation problem.

2.2.5 Stigmergic QoS Optimisation

Given the dynamic nature of the mobile environment under consideration for this work, re-optimisation needs to be performed to closely track any changes in the service composition configurations. Multi Agent System (MAS) techniques have been applied to the service composition problem [Singh et al., 2005, Temglit et al., 2017, Ye et al., 2016]. A multi-agent system provides a way to solve a software problem by decomposing the system into a number of autonomous entities embedded in the environment to achieve the functional and non-functional requirements of the system [Weyns et al., 2013].

In a MAS system several agents operating in the system cooperate to achieve a particular system goal. A coordination mechanism is used to ensure that the agents act in a coherent manner, to prevent chaos in the system. Since no single agent has a global view of the system, it is possible that an agent actions will conflict with other agent's actions. Another reason to coordinate agent's actions is to meet global constraints, which should be met by the system as a whole.

García-Martínez et al. [García-Martínez et al., 2007], Angus and Woodward [Angus and Woodward, 2009] and López-Ibánez and Stutzle [Lopez-Ibanez and Stutzle, 2012] discuss stigmergic-based proposals from a generic perspective. In a mobile environment, stigmergic-based algorithms have been successfully used for QoS routing in MANETs (e.g., AntNet [Di Caro and Dorigo, 1998], ARAMA [Hussein and Saadawi, 2003], AntHocNet [Di Caro et al., 2005], EARA-QoS [Liu et al., 2005], SAMP-DSR [Khosrowshahi-Asl et al., 2011], QAMR [Krishna et al., 2012], QoRA [Al-Ani and Seitz, 2016], and WSNs (e.g., AntSensNet [Cobo et al., 2010]). Zhang et al. [Zhang et al., 2017] presented a comprehensive list of ACO routing protocols.

Moustafa et al. [Moustafa et al., 2016] propose a decentralised, stigmergic-based service composition mechanism. The mechanism offers support for uncertain and changing environments. The proposed mechanism does not consider the QoS of the network, and the description of the architecture is limited.

MO_ACO [Zhang et al., 2010] introduces an ACO for template-based service composition. The authors use a reference-point function (e.g., the reference function in VIKOR [Opricovic and Tzeng, 2004]) to make compromises between different objectives when selecting a concrete service to be mapped to an abstract service. This reference point function is used because the user's preferences are not specified *a priori* and can quantify the utility of a candidate service in the abstract composition graph. This number is used to update the pheromone matrix. This algorithm is evaluated against a genetic algorithm and the results show that the proposed algorithm has a lower convergence time and produces higher quality solutions.

Wu and Zhu [Wu and Zhu, 2013] focus on the transactional properties of services and how to compose individual services in a transactional manner, and then formulate the problem of transactional and QoS-aware dynamic service composition. Transactional support is used to guarantee consistent outcome and correct execution.

AACO [Wang et al., 2014] uses mechanism to generate Pareto-optimal solutions and introduces an adaptive model. The proposed mechanism uses a trust-based model to dynamically control the pheromone evaporation procedure. This procedure is used to adjust the routing behaviour of ants according to the dynamic change in the environment.

MOACS [Wang et al., 2015] propose a multi-objective ACO for data-intensive service composition with global QoS constraints. The composition model follow a template-matching approach. The authors compare the performance of the proposed approach with a multi-objective GA. The authors conclude that when a large number of concrete services are available for each abstract service, a multi-objective GA is more efficient. However, when a small number of concrete services is available for each abstract service, the proposed ACO-based approach is preferred.

C-MMAS [Wang et al., 2011] propose a ACO protocol for template-based service composition. They combine a Max-Min ant model with a culture algorithm. A generic QoS model and domain QoS model are used to updated the values of the pheromone. DACO [Xia et al., 2008] introduce a dynamic ACO-based mechanism for finding a list of service composition configurations. The solution with the highest utility is reinforced.

MOCACO [Li and Yan-Xiang, 2010] is a mechanism for finding a set of QoS-optimal configurations in template-matching service composition. The mechanism uses a chaos-operator and chaos-variable based on Logistic Mapping to control the diversity of solutions. The performance of the mechanism is compared to a multi-objective GA and a multi-objective ACO. The authors measure only the number of solutions produce and the running time of the mechanism, and show that the proposed mechanism can outperform the baseline algorithms.

MOACO4WS [Qiqing et al., 2009] presents an ACO-based protocol for template-based service composition. For each abstract task in the abstract service composition task, the authors select the non-dominated instances which are Pareto-optimal. The QoS value of each service instance is used to calculate the evaporation rate. These values are also used to calculate the heuristic used to select the next service (i.e., node) to visit. In the evaluation, the authors showed that

Proposal	**Environment**	**Service Composition Constructs**	**Optimisation Approach**	**Service Composition Process**	**Reinforcement Strategy**	**Adaptation Procedure**	**Population**
ACAGA_WSC [Yang et al., 2010] ACS-based [Zheng et al., 2007] DACO [Xia et al., 2008] Shanshan [Shanshan et al., 2012] DGQoS [Wang et al., 2011] ACO-WSC [Yu et al., 2015] Wu & Zhu [Wu and Zhu, 2013] MOACO4WS [Qiqing et al., 2009] Chaos-ACA [Yunwu, 2009]	Web Services	Template-Matching	Single-Objective	Centralised	Optimal Solution	N\A	Single
MOACS [Wang et al., 2015] AACO [Wang et al., 2014] MO_ACO [Zhang et al., 2010]	Web Services	Template-Matching	Multi-Objective	Centralised	Non-dominated	N\A	Single
Pop et al. [Pop et al., 2010]	Web Services	Flexible Service Composition	Single-Objective	Centralised	Optimal Solution	N\A	Single
Li [Li and Yan-Xiang, 2010]	Web Services	Template-Matching	Multi-Objective	Centralised	Non-dominated	Chaos Variable	Single
Moustafa [Moustafa et al., 2016]	Dynamic Environment	Template-Matching	Single-Objective	Decentralised	Optimal Solution	N\A	Single
Legend	**1. Web Services 2. Dynamic**	**1. Template-Matching 2. Flexible Service Composition**	**1. Single-Objective 2. Multiple-Objective**	**1. Centralised 2. Decentralised**	**1. Optimal Solution 2. Non-dominated Solutions**		**1. Single 2. Multiple**

Table 2.2: Summary of stigmergic-based service composition proposals.

their implementation outperforms a genetic variant.

Other ACO-based algorithms for Web Services Environments are ACO-WSC [Yu et al., 2015], Shanshan et al. [Shanshan et al., 2012], ACS-Based [Zheng et al., 2007], Yunwu [Yunwu, 2009], Pop et al. [Pop et al., 2010] and ACAGA_WSC [Yang et al., 2010]. These optimisation mechanisms propose an ACO-based algorithm for identifying the QoS-optimal service composition configuration in a web services environment. ACO-WSC [Yu et al., 2015] is mechanism for selecting the cloud combination that contains a minimum number of clouds. Yunwu proposes an ACO mechanism that uses chaos-search to improve the diversity of identified solutions. The performance evaluation shows that the proposed approach can obtain more optimal solutions than GA and SA. Pop et al. [Pop et al., 2010] propose a mechanism to support flexible service composition, however the evaluation not compared with any baselines. ACAGA_WSC use Genetic Algorithm to find the optimal parameters for ACO. However, this mechanism produces a single solution, and no adaptation produce is used to deal with the variation in the QoS values. The performance of this approach is compared against a GA and an IA, and the results show that the proposed mechanism can identify more optimal solutions.

Table 2.2 summarises the existing stigmergic-based service composition proposals. Most of these proposals have been developed for web services environment where services are generally deployed on resource-rich, stationary devices. Also, most of the existing proposals follow a template-matching service composition. This table shows that none of the proposals consider multi-objective QoS optimisation for decentralised, flexible service composition open and dynamic environments such as the mobile environments, with reinforcement strategies and adaptation procedures to ensure efficient exploration of the solution space and diversity of identified solutions. None of the existing proposals consider the collaboration of multiple population of agents in the optimisation process.

2.3 Chapter Summary

This chapter reviewed the state of the art of QoS optimisation processes for service composition from a perspective of openness and dynamism. The existing service composition proposals have been generally categorised as template-matching or flexible service composition. The template-matching proposals require *a priori* knowledge about the environment and require an *a priori* description in the form of a set of abstract tasks. The QoS optimisation process is used to find an optimal allocation of available services in the environment to the set of abstract tasks. To perform this allocation, a number of approaches have been proposed, many based on Mixed-Integer Programming (MIP), or Genetic Algorithms (GAs). However, the template-matching approach is not flexible in open and dynamic environments such as the mobile environments. Services that can be used during composition (with potentially better QoS) are not used because they do not meet the required signature specification.

Flexible service composition has been introduced to address this limitation by combining partial syntactic/semantic matching and planning based algorithms, and using goal-driven techniques to identify the service composition configurations that can satisfy users' goals. However, in terms of QoS support, the existing proposals for flexible service composition in mobile environments have only used (greedy-based) techniques to provide only best effort QoS. In addition to trading off computational efficiency for optimality of solutions, the existing proposals transform the multi-objective QoS optimisation problem into a single-objective optimisation problem by using a utility function which requires a user's preferences. This limits their support for automatic service composition, which reduces their applicability in mobile environments. Also, this approach does not allow users to explore various compromises between the QoS attributes.

Nature-inspired metaheuristics, and in particular swarm-based algorithms using stigmergy-based mechanisms have been successfully used for QoS routing in Mobile Ad-Hoc Networks. A number of stigmergic-based service composition mechanisms have been proposed for finding QoS optimal service compositions in web services environments. However, the existing approaches do not consider the mobility of service providers, and most of the existing approaches have been developed for template-matching service composition and transform the multi-objective optimisation problem into a single-objective optimisation problem to find a single QoS-optimal service composition configuration.

Figure 2.3 shows how the most related QoS optimisation mechanisms for flexible service composition address to the following reference criteria, and how these address the requirements established in Chapter 1:

- *Service Composition Procedure*: This criterion refers to the method to address functional requirements used by the proposal (Section 2.1.2). Liu (Liu et al. [Liu et al., 2017]), GoCoMo (GoCoMo [Chen et al., 2018]), SimDijkstra (Jiang et al. [Jiang et al., 2014] follow a flexible service composition

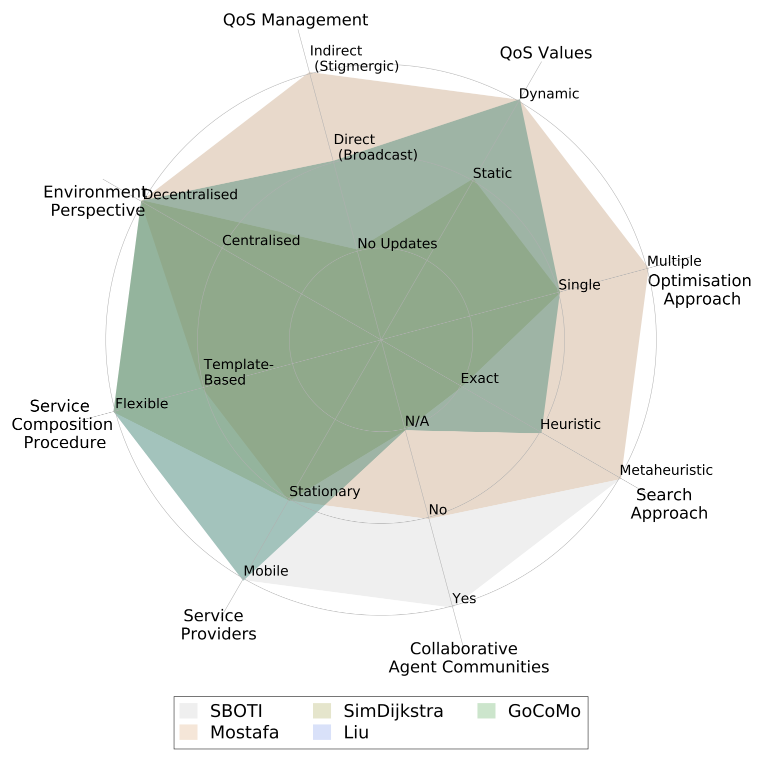


Figure 2.3: The State of the Art review shows how the most related solutions address the eight reference criteria. These solutons are: SBOTI (current work), Mostafa (Mostafa et al. [Moustafa et al., 2016]), Liu (Liu et al. [Liu et al., 2017]), GoCoMo (GoCoMo [Chen et al., 2018]), SimDijkstra (Jiang et al. [Jiang et al., 2014])

approach, and Mostafa et al. [Moustafa et al., 2016], uses a template-matching service composition approach.

- *Service Providers*: This criterion refers to the type of environment considered by the proposals. Services providers may rely on stationary (static) or mobile devices to deliver their services. Liu (Liu et al. [Liu et al., 2017]), GoCoMo (GoCoMo [Chen et al., 2018]), have been developed for mobile environments, whereas SimDijkstra (Jiang et al. [Jiang et al., 2014] and Mostafa et al. [Moustafa et al., 2016], which do not consider the mobility of service providers.

- *Environment Perspective*: This criterion refers to the type of the composition approach (Section 2.2). All proposals used for comparisons follow a decentralised approach. Even if all the works address this criterion with the highest value, this was intentionally added here to outline why other popular approaches are not used.

- *QoS Management*: This criterion refers to the type of QoS management performed by the service composition proposals. More precisely, how the information about the new QoS values is distributed between composition participants. This can be done through either direct communication (i.e, broadcasts) or through indirect communication (i.e., stigmergic). Only Mostafa et al. [Moustafa et al., 2016] uses the concept of stigmergy, whereas the other methods assume a mechanism that continuously broadcasts the QoS of the services across the service providers.

- *QoS Values*: This criterion considers if the proposals considers static or dynamic values. Most of the approaches assume that the QoS values fluctuate at runtime and do not maintain the same values, with the exception of SimDijkstra (Jiang et al. [Jiang et al., 2014].
- *Optimisation Approach*: This criterion refers to if the proposals produces a single or multiple set of solutions. Only Mostafa et al. [Moustafa et al., 2016] produces a set of multiple (Pareto-optimal) solutions. The other approaches transform the multi-objective optimisation problem into a single-optimisation problem using the weighted sum approach.
- *Solution Approach*: This criterion refers to the type of optimisation approach (Section 2.2). Only Mostafa et al. [Moustafa et al., 2016] is implemented based on a metaheuristic algorithm. The other approaches are either heuristic or exact approaches.
- *Collaborative Agent Communities*: This criterion is only for agent-based proposals, and informs if the proposal uses the information from multiple agent communities to improve the exploration of the solution space (Section 2.2.5). Only Mostafa et al. [Moustafa et al., 2016] considers using a group of agents to efficiently explore the solutions space. However, their solution does not extend to multiple collaborative communities.

Chapter 3

Design

The review of the state-of-the-art, presented in Chapter 2, has identified a number of limitations in current QoS optimisation mechanisms for flexible service composition in mobile, dynamic environments. Open issues with current research on QoS-aware service composition are that: (i) existing proposals are not sufficiently flexible to cope with finding QoS-optimal composites mobile environments; (ii) the existing mechanisms have limited support for efficient exploration of the solutions space; (iii) the existing flexible service composition proposals do not allow for exploration of various QoS trade-offs.

This chapter introduces SBOTI (**S**tigmergic-**B**ased **O**p**TI**misation). Section 3.1 re-iterates over the requirements for a QoS optimisation mechanism for flexible service composition in mobile environments introduced in Chapter 1, and outlines the required features for such a mechanism. Section 3.2 defines the system model, and highlights how the optimisation mechanism is mapped to the physical environment. Section 3.3 introduces the design decisions that were made to address the required features presented in Section 3.1. Then Section 3.4 presents SBOTI, and shows how the design addresses the requirements. Section 3.5 outlines functional limitations due to design decisions. Finally, Section 3.6 provides an in-depth discussion of the solution and the contribution.

3.1 Design Objectives and Required Features

As introduced in Chapter 1, the main goal of this thesis is to design a QoS-optimisation mechanism for flexible service composition in mobile environment. The proposed mechanism must deal with the challenges introduced by the environment, which are (i) the frequent link failures during handover caused by the mobility of service providers [Wang, 2011]; (ii) the difficulties in achieving an up-to-date global view of the environment because the service providers rely on devices with limited resources to deliver their services [Sadiq et al., 2015]; and (iii) the large number of potential service consumers that may want access to the available services, and with the qualities of these services evolving with the internal changes or workload fluctuations [Ardagna and Pernici, 2007, Deng et al., 2017]. To address these challenges, this work proposes SBOTI, a QoS-optimisation mechanism for flexible service composition that supports the following required features:

(F1) **Flexible Service Composition Support**: All currently available services should be considered when searching for the most appropriate service compositions. A user's request should be functionally solved using multi-granular, goal-driven service composition approaches. Syntactic, partial matching can be used to automatically identify the available compositions in the environment. SBOTI should be able to cope with service composition configurations built with flexible service composition approaches (Section 2.1.2).

(F2) **Resource-Aware**: Optimal utilisation of devices should be enabled. Services are likely to be deployed on battery-powered devices. The overhead introduced by the model should be kept to an acceptable level to increase the network lifetime. SBOTI should consider the resource limitations of devices, and the methods employed should avoid resource-intensive procedures that rely on a large number of exchanged messages.

(F3) **Mobility-Aware**: Device mobility should be supported. Provider devices may move out of range, which may cause an executing composition to fail. Service provisioning should remain seamless to the consumer, even under those conditions. SBOTI should consider the mobility of service providers, and the intermittent availability or unavailability of services as a consequence of the mobility.

(F4) **Pareto-Optimal Solutions Support**: Multiple possible service composition configurations should be produced, to allow for exploration of QoS trade-offs, and any preference articulation process with the user should be done after the optimisation process. SBOTI should allow for *a posteriori* preference articulation after analysing all optimal solutions, to simplify the operation of users and to avoid deviations caused by unreasonable setting

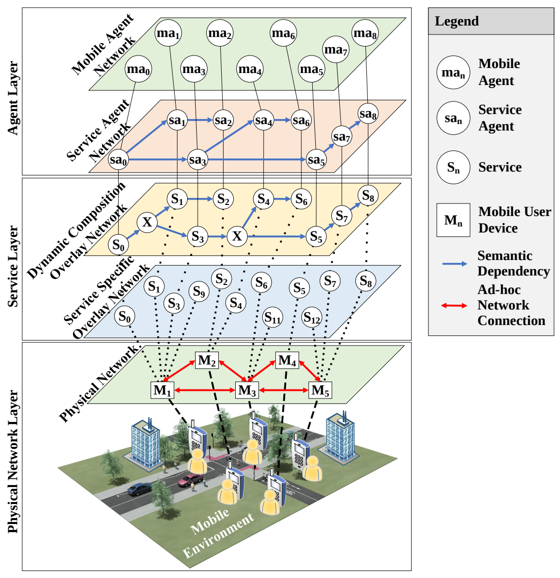


Figure 3.1: Mapping of the agents in the Agent Layer to the physical mobile devices in the Physical Network Layer, through the Service Layer (Section 3.4 introduces the *Service Agent* and *Mobile Agent* concepts). Adding the Agent Layer on top of the Service Layer created using a Service-Specific Overlay Network (SSON) and a Dynamic Composition Overlay Network (DCON) network is a contribution of this work [Palade and Clarke, 2018, Palade et al., 2018b]. All the contributions of this thesis are made at this Agent Layer.

of QoS objectives' weights. The optimal solutions is a set of Pareto-optimal solutions, where dominated solutions are removed and the available solutions are diverse to allow exploration of various compromises between the QoS criteria (Section 2.1.3).

3.2 System Environment

The services deployed on mobile devices in a limited geographic area such as a mall [Chen et al., 2018] or an university campus [Deng et al., 2017] can be used to create new service-based applications. These devices are generally connected through a Mobile Ad-hoc Network (MANET). To enable service-based applications on top of such a network, services can be clustered based on their syntactic or semantic similarity using a Service-Specific Overlay Network [Al-Oqily and Karmouch, 2011]. Goal-driven service composition using planning-based algorithms can automatically solve a user's request using the available services in these clusters [Chen et al., 2018]. The optimisation process is modelled using a multi-agent system approach. This approach is presented in Section 3.4. Figure 3.1 shows the main layers of the system model, which are:

- **Physical Network Layer**: This layer represents the physical network infrastructure made of mobile devices in the environment [Park and Shin, 2008]. Generally, MANET-based systems are used to create the communication links between these devices. Applying SOA principles in a mobile environment is fundamentally different from the traditionally web-based systems [Halonen and Ojala, 2006]. The distributed nature, limited bandwidth, network reliability, heterogeneity, and energy constraints factor in the dynamism introduced by the mobility of the service providers [Efstathiou et al., 2014].

- **Service Layer**: A composite service contains a set of services and the data and control flow between them, and the set of composite services can be represented as a service dependency graph, in which the nodes of the graph correspond to services and the directed edges between the nodes reflect the syntactic/semantic interoperability [Park and Shin, 2008]. This thesis assumes that composites do not contain repetitive parts and models the set of available service composition configurations that can satisfy a user's request as a service dependency graph [Feng et al., 2013]. A service dependency graph can be enabled by clustering the available services deployed on mobile devices within a limited geographic area in a Service-Specific Overlay Network (SSON) [Al-Oqily and Karmouch, 2011]. Goal-driven service composition using planning-based algorithms can automatically solve a user's request using the available services in these clusters [Chen et al., 2018]. The result of this process is a service dependency graph, which uses a Dynamic Composition Overlay Network (DCON) to perform the administrative tasks (e.g., service invocation). A DCON organises the service providers that are currently participating in the current composition. This overlay is temporary, and only exists when a composition is executed in the environment [Chen et al., 2018].

- **Agent Layer**: A decentralised approach to service composition allows a service consumer to autonomously organise the service composition process without assuming a full knowledge of the environment. Each service in the DCON is equipped with an intelligent agent, which provides access to the service [Ye et al., 2016]. Then, various multi-agent technologies can be used to perform the optimisation process. This procedure is detailed in Section 3.4.

3.3 Design Decisions

QoS-aware service composition in mobile environments can be performed in different ways. The following sub-sections outline the decision decisions made to address the required features identified in Section 3.1.

3.3.1 Flexible Service Composition in Mobile Environments

The service providers available in a mobile environment rely on mobile devices to deliver their services. These entities are autonomous, and they can leave the composition process at any time. The service composition process can be designed from a centralised or a decentralised perspective. Most of the existing service composition mechanisms have used centralised approaches, with a few decentralised proposals. In the centralised approach, a central controller has full knowledge of the environments and is responsible to organise service composition for consumers in environments. However, such approach may incur a performance bottleneck as the number of service providers increases. Also, this controller represents a single point of failure for the system. Decentralised approaches distribute the decision process relating to service composition across multiple service composition participants. Thus, computation bottlenecks and a single point of failure are avoided.

Design Decision 1: Distributed Execution

SBOTI decentralises the challenges relating to service execution, using a swarm-based approach to deal with the challenges introduced by the mobile environment.

The decentralised approaches either assume that each service consumer has full knowledge about the environment, or use a modeling approach where each service composition participant has knowledge only about its neighbours. Having a service consumer with full knowledge about the environment may not be feasible in large-scale environments. The proposed modeling approach overlooks the evolution of the environment, where participants may change their acquaintance relationship between each other over time [Ye et al., 2016]. Such evolution has been explored previously in the context of multi-agent systems. In the context of service composition, previous approaches have shown that multi-agent systems can be used to engineer the coordinated activities of a multitude of decentralised autonomous components [Castelli et al., 2015].

Design Decision 2: Agent-Based

SBOTI uses a multi-agent approach to model the QoS optimisation problem for flexible service composition in a mobile environment. The proposed solution uses an agent-layer deployed on top of the service layer to perform the optimisation process. More precisely, this network is created on top of the DCON (see Figure 3.1).

Two of the challenges introduced by mobile environments are *mobility* and *scalability* of the service providers (Section 1.1.3). As previously mentioned, these providers rely on mobile devices to deliver their services. In large and decentralised environments whose components belong to different stakeholders, service composition configurations solutions that are able to maximise the satisfaction of QoS requirements, and have a self-adaptive behaviour without centralised control strategies [Castelli et al., 2015]. Natural systems promote the capability of emergence of self-organised patterns of coordinated behaviours which are suitable in the dynamic environments.

Natural systems (e.g., ant colonies) introduce coordination mechanisms to identify the requirements of the environment spontaneously and efficiently [Omicini, 2013]. For example, ants use stigmergic interactions, which is a form of indirect communication through the environment by depositing and locally capturing pheromones in the environment. This type of interactions decouple interacting agents and let interactions take place spontaneously [Parunak, 1997]. Also, pheromones can promote simple forms of situation-aware interactions, by expressing some fact/event/information that has occurred in a particular portion of the environment. As a result of overall activities of ants in building complex distributed pheromones structures, and in reacting to the presence and shape of such structures, globally coordinated behaviours emerge in the colony, supporting self-adaptation and self-organization [Dorigo and Gambardella, 1997, Castelli et al., 2015].

Design Decision 3: Indirect Communication

The mobile environment is dynamic and a coordination mechanism that can deal with this is needed. SBOTI uses a stigmergic-based mechanism to coordinate the behaviour of the agents that are used in the optimisation process. This mechanism uses an indirect communication approach through the environment.

3.3.2 Multi-Objective Optimisation

Most of the existing service composition approaches have formulated QoS optimisation as a single-objective objective optimisation problem. This makes it difficult to analyse trade-offs for various QoS attributes.

> **Design Decision 4: A Posteriori Preference Articulation**
>
> Because of the decentralised and dynamic environment, users are unfamiliar with the QoS of available services, and so it is not easy to decide in advance what should be the precise weights on their QoS criteria. SBOTI allows users to explore various compromises between the available QoS attributes by presenting them with a set of solutions, each of which may have different trade-offs across the QoS criteria. The preference elicitation process is performed after the user is presented with the possible set of solutions.

The goal of Multi-Objective Optimisation (MOO) algorithms is to find a set of Pareto-optimal solutions, which have been identified using the concept of *dominance* [Zitzler and Thiele, 1999, Wang et al., 2015]. The definition of this process was presented in Section 1.1.2.

> **Design Decision 5: Non-dominated sorting**
>
> SBOTI uses non-dominated sorting to identify solutions that are not dominated. These solutions will be reinforced throughout the optimisation process and returned to the user at the end of the optimisation process.

New service composition configurations may emerge (with potentially better QoS) as a result of service providers' mobility. Therefore, to adapt to such evolving environment, it is important to develop a service composition approach which can self-evolve over time [Ye et al., 2016]. The digital pheromone mechanism used by such approaches may affect the exploration of new service composition configuration that may emerge as a result of service providers mobility.

> **Design Decision 6: Adaptation Support**
>
> To allow for exploration of new service composition configurations that may emerge as a result of providers' mobility, SBOTI uses an adaptation procedure that limits the amount of pheromone on previously identified solutions.

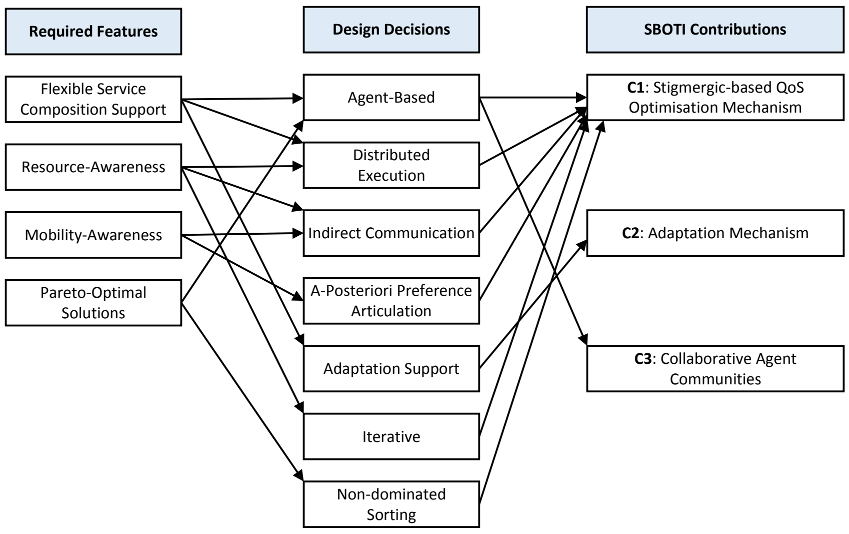


Figure 3.2: Mapping of the required features (Section 3.1), the design decisions made to implement these features presented in this section, and the SBOTI contributions (Section 3.4.1) outlined as **C1** (Section 3.4.2), **C2** (Section 3.4.4), **C3** (Section 3.4.5)

3.3.3 Computational Effort and Optimality Trade-off

Centralised approaches can achieve a global optimum, but are a single point of failure and a potential performance bottleneck during composition. Decentralised approaches make decisions based on their local view of the environment, but inferring a global optimum from local calculations can be difficult and expensive in a mobile environment. Previous decentralised methods relied on direct communication through broadcasts. However, in a mobile environment this can increase the failure rate (Section 2.2).

Design Decision 7: Iterative

SBOTI uses an iterative approach to continuously (and selectively) explore the environment. This design decision is made to avoid the need for a complete knowledge about the environment, which may require a continuous broadcast of each participant state. This design decision is made to address a resource-awareness requirement of mobile devices. The trade-off between computational efficiency and optimality of identified service composition configurations is achieved by the continuous sampling of the environment. This number reflects the time a user is willing to wait for a particular optimisation request.

Figure 3.2 summarises the design decisions that constitute the novel QoS optimisation mechanism to address the required features discussed in Section 3.1, and maps these decisions to the contributions made by the proposed solution detailed in Section 3.4.2, Section 3.4.4, and Section 3.4.5. The next section provides the details about the proposed solution and explains how the design decisions materialise in SBOTI.

3.4 Proposed Solution: SBOTI

SBOTI is a nature-inspired, metaheuristic-based, distributed optimisation mechanism for finding QoS-optimal service composition configurations in a mobile environment. SBOTI uses the concept of *stigmergy*, a (digital) pheromone-based approach for coordinating and controlling swarming agents. Stigmergic-based systems can generate resilient, complex, intelligent behaviours at the system level even when the individual agents only possess limited or no intelligence [Theraulaz and Bonabeau, 1999]. Such multi-agent systems can cope with continuously changing environments through the decentralised control of agents distributed in the environment. Unlike existing optimisation approaches for flexible service composition in mobile environments, SBOTI produces a set of Pareto-optimal solutions, and does not require a global (complete) view of the search space. SBOTI uses a probabilistic approach to efficiently explore different parts of the solution space. SBOTI is inspired by Ant Colony Optimisation [Dorigo and Gambardella, 1997], and is based on a model originating from biological collective organisms, and inherits the following principles observed in swarm-based systems [Parunak, 1997]:

- **Simplicity**. Swarm-based entities do not perform any deep reasoning and rely on a small set of simple rules during their lifecycle. The execution of these rules leads to the emergence of complex behaviour. The active entities used by SBOTI obey this principle of simplicity.

- **Dynamism**. Natural swarms perform self-organisation procedures to adapt to dynamically changing environments. SBOTI can adapt to the changing environment and identify a different set of solutions, if previous service composition configurations are rendered unavailable because of the environment's dynamics.

- **Locality**. Swarm-based entities observe their direct neighbourhood and take decisions based on their local view. A key design decision in SBOTI is that the active entities perform only local searches and communicate only with direct neighbours.

3.4.1 Stigmergy System Model

The users within a limited geographic area can create a service-sharing community using the services deployed on their mobile devices [Deng et al., 2017]. Stigmergic coordination, exhibited by social insects to coordinate their activities, can be used to find QoS-optimal service composition configurations in such mobile environments. Each (mobile) service can be modelled as a *service agent*, and a service-sharing community as a multi-agent system [Moustafa et al., 2016]. A set of *mobile agents*, which form an *agent community*, interact with the service agents in the environment by encoding application-specific information as a pheromone to achieve certain tasks [Parunak, 1997]. These agents achieve collaboration and self-organisation by exchanging pheromones and performing several pheromone procedures, such as positive/negative reinforcement of optimal/non-optimal solutions [Palade and Clarke, 2018]. This section defines these terms, and uses this abstraction to model the decentralised, flexible service provider interactions in a mobile environment.

A service agent is a stationary software entity responsible for the interactions with a service deployed on a physical device. A service represents a single unit of functionality. The set of interactions include the execution of the service, and QoS performance monitoring of the service. Given the graph from Figure 1.2 as an example, each service s_i is associated with a service agent. In this example, at runtime, after s_0 is invoked, the execution can follow three directions: towards P_0, P_1 or P_2. The service agent associated with s_0 has two service agent neighbours (associated with s_1 and s_3). A *Service Agent* is defined as follows:

Definition 1. ***Service Agent*** *(sa). A service agent is modelled as a tuple $sa = <id, F>$, where id is the identifier of the service for which this service agent is responsible in the given service dependency graph, and F is the pheromone store used to facilitate the service composition process.*

Definition 2. ***Pheromone Store*** *(F). A pheromone store is modelled as a set of tuples $<id, \tau>$, where id is the identifier of a direction (to an outgoing node in the service dependency graph of the current node) and τ is the pheromone level associated with that direction. τ pheromone level is a value that indicates the quality of a direction and is proportional with the quality associated with that direction. When initialised, each direction is associated with the same initial*

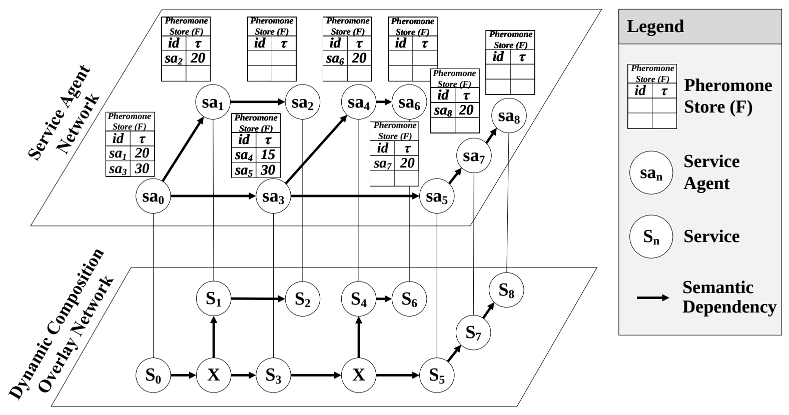


Figure 3.3: Mapping of Service Agent Network to Dynamic Composition Overlay Network (DCON), and showing the *Pheromone Store* associated with each *Service Agent.*

pheromone level, $\tau_{initial}$, which is set before the start of the optimisation process. In case a direction becomes invalid due to a link disconnection between two service agents, the tuple associated with that direction is removed from the store.

Figure 3.3 shows the mapping of Service Agent Network to the Dynamic Composition Overlay Network (DCON). This figure shows also shows the *Pheromone Store* elements associated with each service agent sa_j. For example, the *Pheromone Store* associated with sa_0, has two entries sa_1 and sa_3, where sa_1 is associated with pheromone level $\tau = 20$, sa_3 is associated with pheromone level $\tau = 30$. The mobility of service providers may cause link disconnections between service agents. In this case, the pheromone level τ associated with the disconnected direction (node) is removed from the table. If a new direction becomes available at runtime, a tuple with the $<id, \tau_{initial}>$ is added to the store, where id is the identifier of the direction and $\tau_{initial}$ is the initial pheromone level.

A mobile agent is a non-stationary software entity capable of autonomously moving between the networked devices [Satoh, 2010]. Service agents are associated with a service in the service dependency graph. Such services may reside on separate physical devices. These (distributed) service agents (e.g., sa_1 and sa_2) need to interact through a (unreliable) network of mobile devices. A mobile agent has several advantages as opposed to continuous communication between service agents:

- **Reduced Communication Costs:** A mobile agent at sa_1 can include execution logic and the data collected (i.e., QoS values of the service) from sa_1, and migrate to service agent sa_2 instead of requiring continuous communication between sa_1 and sa_2.

- **Asynchronous Execution:** After migrating to sa_2, the mobile agent does not have to interact sa_1. The mobile agent can continue processing at sa_2 even when sa_1 is disconnected from sa_2, which is useful in environments with unstable communications.

- **Direct Manipulation:** A mobile agent is locally executed on the service agent that it is visiting. A mobile agent can directly interact with a service agent if they reside on the same physical device. This may be helpful in service management, in particular in improving the resilience of the network.

- **Easy Development of Distributed Applications:** A mobile agent can easily interact with other service agents using some standard messaging protocol. A mobile agent can carry information between service agents, and a service agent provides the necessary abstractions for the interaction of a mobile agent with the service.

A mobile agent can query and update the pheromone store of the service agent. A mobile agent ma_i queries all the values stored in the pheromone store F of service agent sa_j. Agent ma_i uses these values to decide which neighbour of sa_j will be next explored. Agent ma_i can also update the values in F. Each update changes only one value in F with a positive or negative offset given by the quality of the identified solution (see in detail in Section 3.4.2). The query and update interactions are possible only when the mobile agent resides on the service agent reside on the same overlay network node (i.e., DCON). A *Mobile Agent* is defined as follows:

Definition 3. ***Mobile Agent*** *(ma). A mobile agent is modelled as a tuple* $ma = <id, M>$*, where id is the agent's identifier, and M is the memory store associated with that agent.*

Definition 4. ***Memory Store*** *(M). A memory store is modelled as an ordered set of tuples* $<sa_i, q>$ *represented as a stack where each element* sa_i *is a service agent and q is service specific information provided by that service agent to facilitate the optimisation process (e.g., QoS of the service). This set represents the path traversed by a mobile agent in the given service dependency graph.*

Figure 3.4 shows the mapping of Mobile Agent Network to Service Agent Network. This figure shows also shows the *Memory Store* elements associated with each mobile agent ma_i. This example shows two mobile agents: ma_0 and

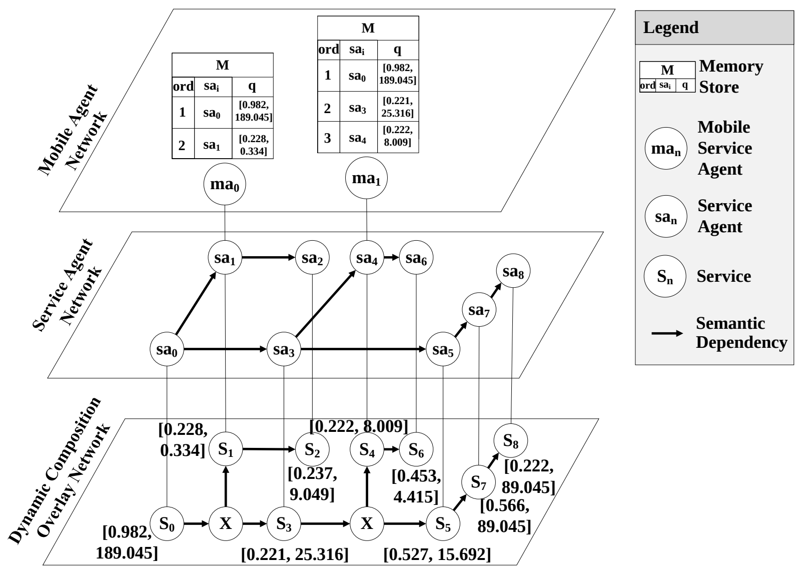


Figure 3.4: Mapping of Mobile Agent Network to Service Agent Network, and showing the *Memory Store* associated with each *Mobile Agent*. The *Pheromone Store* structure used by service agents is omitted from this figure to highlight the *Memory Store* used by the mobile agents.

ma_1. The *Memory Store* associated with ma_0, has two entries sa_0 and sa_1, where sa_0 is associated with QoS values $[0.982, 189.045]$, sa_1 is associated with QoS values $[0.228, 0.334]$. This means that mobile agent ma_0 has first visited sa_0 and recorded in its memory store the QoS associated with service S_0 (i.e., service specific information) and the *id* of sa_0. Afterwards, agent ma_1 traversed to sa_1 where it performed a similar procedure. The mobility of service providers may cause link disconnections between service agents. In this case, the mobile agent will continue its traversal. If the links to any of the service agents registered in the memory store of a ma_i become unavailable, then the mobile agent will not update the pheromone store associated with that service agent or the reminder of service agents in that memory store.

An optimisation process performed by SBOTI will require a group of mobile agents to explore the available services in the environment. This group can be formed based on similarity between agents. A set of mobile agents can form an agent community to share information about the quality of identified solutions. An *Agent Community* is defined as follows:

Definition 5. ***Agent Community*** *(AC). An agent community is modelled as a tuple $AC = \{id, S\}$, where $S = \{ma_0, \ldots, ma_n\}$ is the set of mobile agents in the community, and id is the identifier of the community.*

The number of mobile agents in a community is task dependent. For simplicity, this number can be 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 [Moustafa et al., 2016].

Two procedures are defined to facilitate the coordination of service and mobile agents: pheromone evaporation and pheromone deposit:

Definition 6. ***Pheromone Evaporation.*** *Pheromone evaporation is a procedure associated with the pheromone store F of each service agent sa_i. This procedure periodically decreases each pheromone level in the pheromone store according to a defined rule. The period T is given by the evaporation frequency. The rule is defined in Section 3.4.2.*

Definition 7. ***Pheromone Deposit.*** *A pheromone deposit is a procedure associated with a service agent sa_i, and is triggered by a mobile agent ma_i. During this procedure, a quantity of pheromone is added or subtracted from the pheromone level associated with a particular neighbour from the pheromone store F.*

Different communities of agents can form based on their similarity. Each community may have their own properties, and during the optimisation process one community or multiple communities may be available. We assume that the agents within a single community are homogeneous, are willing to collaborate and do not have any conflicting goals. The set of available communities that can perform the optimisation process can then be categorised as follows:

1. *Homogeneous.* The communities share the same properties: each community has the same number of mobile agents; each mobile agent in each community deposits the same pheromone scalar Q_1; each community has the same initial pheromone level τ; and each community uses the same evaporation coefficient ρ. Each community searches in the same part of the solution space.

2. *Heterogeneous.* The communities have different properties: each community decides how many mobile agents it has; each community has its own Q_1 value, its own initial deposited pheromone on each service agent, and its own evaporation coefficient ρ. Each community may search in different parts of the solution space.

The SBOTI model for a single agent community [Palade and Clarke, 2018, Palade et al., 2018b] is introduced in Section 3.4.2. An extension of this model for multiple, collaborative agent communities is presented in Section 3.4.5. Each individual agent community (even in a multiple community configuration) behaves as described in Section 3.4.2.

3.4.2 Single Agent Community

SBOTI takes as input a service dependency graph, which represents a set of available composites that can functionally satisfy a user's request (e.g., Figure 1.2). Such a graph is built using a service discovery component. This work assumes that such a component exists, and, once it initialises SBOTI with a new graph, this component can perform in parallel to SBOTI to continuously adapt this functional structure relative to the changing environment.

Examples of how such a component can be built have been previously outlined in Chen and Clarke [Chen et al., 2018] or Groba and Clarke [Groba and Clarke, 2014]. Local knowledge bases available on each mobile device, built by each node using the information from nodes they encounter, can be used to find all the composites that can satisfy user's request [Mascitti et al., 2018]. During the encountering phase, two nodes will exchange information about their expected contact time and the services deployed on each node. The contact time is the duration of being in the proximity of each other, and is used to maintain each node's local knowledge base. Information about the services stored by each node can be removed when the node associated with those services is no longer in contact. User feedback can be used to improve the accuracy of the search results during the discovery process [Cabrera et al., 2018a].

Within an agent community, a service agent sa_j can have *source*, *destination* or *intermediate* roles. As the source, sa_j is responsible for the initialisation of the optimisation process. The agent receives the service dependency graph from a service discovery component, initialises a set of mobile agents based on the properties of the graph, and forwards each mobile agent ma_i towards the first node (root) in the graph (Algorithm 2). This node is identified using the syntactic/semantic dependency relations between services in the service dependency graph. After this, the source service agent transits to a *Listening* state where it waits for the mobile agents to return after their forward/backward traversal of the graph. When a mobile agent ma_i returns, sa_j transits to the *Enqueuing* state where it adds the received mobile agent to a local queue. If all the mobile agents are received, sa_j transits to an *Re-optimisation* state where it checks if the number of iterations $iter$ is below the requested threshold N to start a new

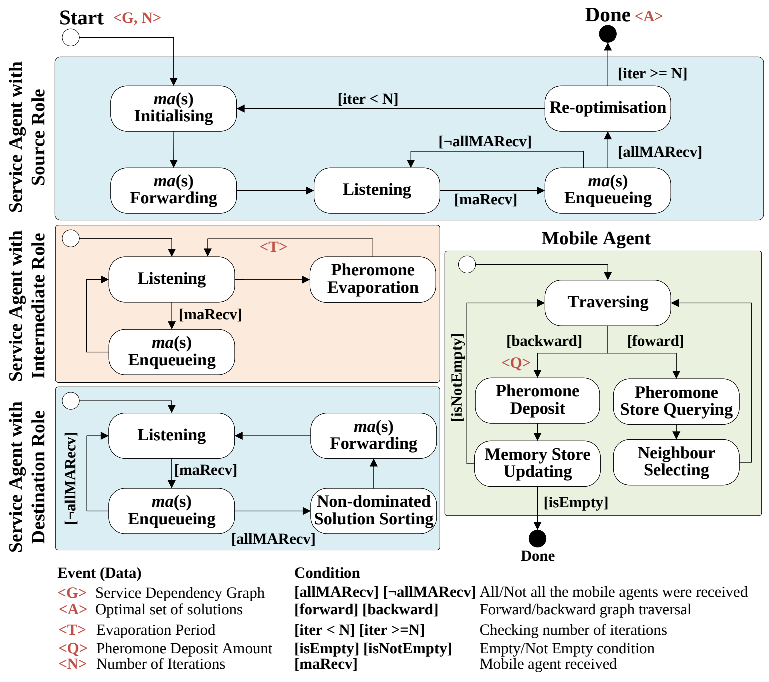


Figure 3.5: Overview of participating agents in the stigmergic service composition process, which captures the state of each agent in different roles, and the events that triggered the state transition. Algorithm 2, Algorithm 3, Algorithm 4, and Algorithm 5 describe the flow of the events that trigger the state transition.

traversal of the graph. This number is provided by the user as part of the composition request, and may be used to control the user's time willing to wait for the optimisation process.

As the destination, sa_j is responsible for collecting the mobile agents after they finish traversing the service dependency graph, sorting the solutions which are the paths in the graph traversed by the mobile agents, and deciding which ma_i will be sent in the backward direction to reinforce the path used in the forward direction, as each ma_i records the *id* of each visited graph node in its memory store. sa_j starts in a *Listening* state where it waits for all ma_i to arrive. Metadata can be piggybacked on each ma_j to transfer information such as the number of expected mobile agents. When a ma_i arrives, sa_j transits to an *Enqueueing* state where it adds the received ma_i to a local queue. When all have been received, sa_j extracts the memorised path from each ma_i, uses a non-dominated sorting solution technique [Mishra and Harit, 2010] to identify which paths are associated with non-dominated solutions, and forwards each ma_i in the backwards direction (Algorithm 3). After this it transits back to the *Listening* state. Based on how dynamic the environment is, a decision can be

Algorithm 2 Initialisation of optimisation process by the service agent with source role

```
Require: $G$ {Service Dependency Graph}
Require: $N$ {Total number of iterations}
 1: $iter \leftarrow 0$ {Current Iteration}
 2: $R \leftarrow extract\_address(G)$ {Root of G address}
 3: while $iter < N$ do
 4:    $n \leftarrow size(G)$ {Number of services}
 5:    $AC \leftarrow \{\}$ {Empty queue to store $ma$(s)}
 6:    for i ← 1 to n do
 7:       Initialise $ma_i, M_i \leftarrow \{\}$
 8:       $M_i \leftarrow M_i \cup R$ {Add root address to M for this $ma$}
 9:       $d \leftarrow forward$ {Set forward direction for this $ma$}
10:    end for
11:    for i ← 1 to n do
12:       Forward $ma_i$ to {Trigger traversal}
13:    end for
14:    allMARecv ← false
15:    while ¬allMARecv do
16:       *Listening()* {Wait for *all* mobile agents to return}
17:       if maRecv then
18:          $AC \leftarrow AC \cup ma_i$ {$ma$ Enqueueing}
19:       end if
20:       if $n = size(AC)$ then
21:          allMARecv ← true
22:       end if
23:    end while
24:    $iter \leftarrow iter + 1$
25: end while
```

made to configure SBOTI to wait for only a fraction of mobile agents in the destination or source roles, as waiting for all the mobile agents may affect the resolution time. This can be used to cater for any traversal failures that may be caused by network failures. A requester service agent may play both the source and destination roles. As an intermediate, sa_j facilitates ma_i's movement and access to its pheromone store (Algorithm 4). To allow for new paths to emerge, sa_j runs a *Pheromone Evaporation* procedure (Definition 6) periodically, with period T using (Equation 3.1):

$$\tau_i = \tau_i * (1 - \rho) \tag{3.1}$$

where τ_i is the pheromone scalar associated with an entry in the pheromone store M of a service agent sa_j, and ρ is the pheromone evaporation coefficient.

The mobile agents iteratively traverse the graph in a *forward* and *backward* motion, and communicate with each other through the pheromone store of service agents. In the forward direction, a mobile agent ma_i collects the QoS of the

Algorithm 3 Procedure used by the service agent with destination role

```
Require: $n$ {Initial number of mobile agents}
1: $AC \leftarrow \{\}$ {Empty queue to store $ma$(s)}
2: allMARecv ← false
3: while ¬allMARecv do
4:    *Listening()* {Wait for *all* mobile agents to return}
5:    if maRecv then
6:       $AC \leftarrow AC \cup ma_i$ {$ma$ Enqueueing}
7:    end if
8:    if $n = size(AC)$ then
9:       allMARecv ← true
10:   end if
11: end while
12: $P \leftarrow extract\_solutions(AC)$
13: $ND \leftarrow non\_dominated\_solution\_sorting(P)$
14: for i ← 1 to n do
15:    if $ma_i \in ND$ then
16:       $set\_non\_dominated(ma_i)$
17:    end if
18:    $d \leftarrow backward$ {Set backward direction for this $ma$}
19:    Forward $ma_i$ {Trigger traversal}
20: end for
```

service maintained by the explored service agent through a *Pheromone Store Querying* procedure to decide which neighbour will be next explored. Based on these values, a probabilistic approach is used to select the next neighbour (direction), using equation:

$$P_d = \frac{[\tau_d]^\alpha}{\sum_i^N [\tau_i]^\alpha} \tag{3.2}$$

where d is the index of the neighbour, τ_i is the pheromone level associated with that neighbour, and P_d is the probability of selecting neighbour d. While heuristic information is generally used in existing stigmergic-based systems for service composition (e.g., Mostafa and Zhang [Moustafa et al., 2016]), SBOTI does not assume such *a priori* knowledge about the environment. When a mobile agent ma_i arrives at a service agent that has no neighbours (no directions), then the mobile agent moves to the destination agent. This work assumes that such a node exists and its address does not change throughout the *forward* traversal of the mobile agents, or if it does, there is a method or a procedure pre-defined that can update this address in every mobile agent that is used in the current optimisation process.

As mobile agents traverse the graph in the backwards direction using the path stored in their memory store, the intermediate service agents associated with this

Algorithm 4 Procedure used by service agents

```
Require: $s_{id}$ {A service in service dependency graph G}
Require: $NG$ {The set of successor neighbours of $s_{id}$ in G}
Require: $T$ {*Pheromone Evaporation* procedure period}
Require: $\tau_{initial}$ {Initial pheromone level}
1: Initialise pheromone store $F \leftarrow \{\}$
2: for i $\leftarrow$ 1 to $size(NG)$ do
3:    $\tau_i \leftarrow \tau_{initial}$
4:    $ng_i \leftarrow extract\_index(NG, i)$ {$i$-th element in set $NG$}
5:    $F \leftarrow F \cup (ng_i, \tau_i)$
6: end for
7: while true do
8:    for i $\leftarrow$ 1 to $size(NG)$ do
9:       $update(\tau_i)$ {*Pheromone Evaporation* procedure}
10:   end for
11:   $wait(T)$ {Listening for other events}
12: end while
```

Algorithm 5 Procedure used by mobile agents

```
Require: Parameters $\alpha, Q_1$
Require: $s_{id} \leftarrow current\_service\_agent\_id$
1: while $isEmpty(M)$ = false do
2:    $d \leftarrow getTraversalDirection$
3:    if $d = forward$ then
4:       $F \leftarrow queryPheromoneStore(s_{id})$
5:       $ng \leftarrow getNeighbour(F)$ {Using Equation 3.2}
6:       $M \leftarrow M \cup ng$ {Append address to M}
7:    else if $d = backward$ then
8:       $ng \leftarrow removeLast(M)$ {Remove last item from memory store M}
9:       $i \leftarrow extract\_index(F, ng)$
10:      $update(\tau_i)$ {*Pheromone Deposit* procedure}
11:   end if
12:   $traverse(ma_i, ng)$
13: end while
```

path will modify their pheromone store (through positive reinforcement) using the following *Pheromone Deposit* procedure (Definition 7):

$$\tau_i = \tau_i + \tau_i * (Q_1/l_i) \tag{3.3}$$

where τ_i is the pheromone level, Q_1 is a constant and l is the length of the i-th path (Algorithm 5). After this ma_i transits to a *Memory Store Updating* state where it removes the top element from its M. If M is empty then the lifecycle of this mobile agent is over, otherwise it will move in the *Traversing* state. When all the mobile agents arrive at the source service agent, they are discarded.

3.4.3 Solution Diversity Strategy

A reinforcement strategy where both dominated and non-dominated solutions are reinforced is used. The non-dominated solutions are reinforced with a positive offset, whereas the dominated solutions are reinforced with a negative offset [Palade and Clarke, 2018]. This is done through Equation 3.4.

$$\tau_i = \tau_i - \tau_i * (Q_2/l_i) \tag{3.4}$$

where τ_i is the pheromone scalar, Q_2 is a constant and l is the length of the i-th path (Algorithm 5).

To deal with the dynamic environments, a $[\tau_{min}, \tau_{max}]$ pheromone reset strategy can be used, where τ_{min} and τ_{max} are lower/upper bounds for pheromone level. When the pheromone scalar is below the lower bound or above upper bound, the pheromone scalar is reset to the initial $\tau_{initial}$ value.

3.4.4 Adaptation Support

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, a mobile 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. An adaptation procedure is required to effectively address these issues [Palade et al., 2018b].

A pheromone smoothing scheme is used to facilitate composition adaptation during execution, and allow other parts of the solution space to be explored. Identified optimal paths are reinforced with less pheromone [Stützle and Hoos, 2000], which allows for other (some previously marked as non-optimal) paths to be explored again. Each element in the pheromone store is updated as follows:

$$\tau_{ij'} = \tau_{ij} + \delta * (\tau_{max} - \tau_{ij}) \tag{3.5}$$

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 emphasis on the reinforcement of pheromone concentration on the optimal path(s), by reducing the rate of reinforcement of dominant paths [Sim and Sun, 2003]. Previous research showed that such a technique can prevent the generation of dominant paths. Stuzle and Hoos [Stützle and Hoos, 2000] 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 [Cai et al., 2006].

In a mobile environment, the devices on which the services are deployed may move out of range, or may be power depleted. If these services are part of 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 compositions may emerge, which may have more optimal QoS.

The pheromone smoothing approach is used as an adaptation handling procedure for planning-based service composition in mobile environments. Each service agent is allocated with two pheromone update rules, the evaporation and pheromone smoothing of each element in the pheromone store.

3.4.5 Collaborative Agent Communities

Given the dynamic nature of the environment under consideration for this work, re-optimisation needs to be performed to closely track any changes. SBOTI's optimisation process requires a number of the previously-defined hyper-parameters to be configured, in particular the number of mobile agents required to explore the search space, the initial pheromone level $\tau_{initial}$ associated with each element in M of each service agent (Definition 2), the evaporation frequency ρ parameter that allows new paths to be explored (Definition 6), the parameter α used during neighbour selection (Equation 3.2). An important limitation of metaheuristic-based algorithms is that hyper-parameters need to be tuned, because a universally optimal parameter values set does not exist [Talbi, 2009]. The ACO metaheuristic used in SBOTI requires a number of parameters to be set, and the selected values can affect the performance of the method [Engelbrecht, 2006, Fulcher, 2008, Chitty and Hernandez, 2004, Wang and Shen, 2017]:

- n_k, number of ants: affects the exploration ability of the algorithm. A large number increases the time until the pheromone levels on good links increase to higher levels than they do on inferior links. A small number affects the level of information about the environment. Also, this number directly affects the communication overhead as the number exchanged messages increases with the number of ants.

- n_t, number of iterations: The quality of the solutions depends on the space explored by the ants. If the number of iterations is small, then the ants may not have time to explore and settle on a single path, whereas a large number of iterations may introduce unnecessary communication overhead through repetitive exploration of the same paths.

- $\tau_{initial}$, initial pheromone level: During the initialisation step, all the links are initialised with either a constant value, $\tau_{initial}$, or to random values in the range $[0, \tau_0]$. The random values may cause a bias towards the links with initial large pheromone concentrations, and links with small pheromone concentrations may be neglected as components in the final solution.

- The pheromone evaporation period T allows for the trade-off between the exploration and exploitation to be controlled. If T has a large value, the pheromone matrix is highly dependent on good solutions from previous generation, which leads to search around these good solutions. The smaller the value of T, the greater the contribution of good solutions from all the previous generations and the greater the diversity of search space through the solution space.

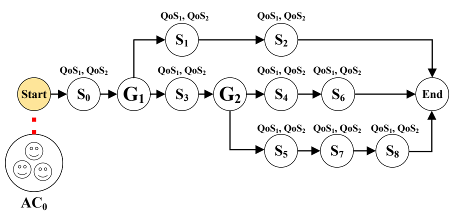
- Influence of α: If $\alpha = 0$, the services with the best QoS attributes are more likely to be selected: this corresponds to a stochastic greedy algorithm. The smaller values of α lead to the slow convergence and local optimum, whereas the larger values lead to a strong emphasis on initial solution, random fluctuations and bad algorithm behaviour. If $\beta = 0$, only the pheromone information is used, which generally leads to poor results. The larger values lead to suboptimal solutions.

- Pheromone evaporation coefficient ρ: The larger values affect the global search ability, and the smaller values reduce the convergence speed.

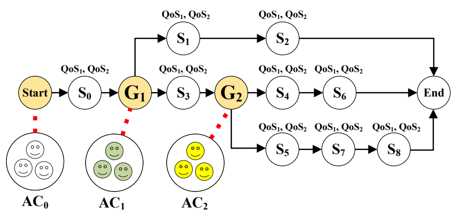
Selecting the best values for the control parameters is a difficult task. A tuning process allows for larger flexibility and robustness. Previous empirical studies have investigated these parameters, suggested values and heuristics to calculate the values. However, it is difficult to maximise the efficiency of such algorithms since these parameters have to be optimised for the specific problem being solved, which requires problem specific information, or *a priori* knowledge about the environment [Hussein and Saadawi, 2003]. Previous works used empirical analysis or through learning. An additional algorithm can be used to "learn" the best values of the parameters for the given algorithm and problem. For instance, Botee and Bonabeau [Botee and Bonabeau, 1998] used a GA mechanism to explore the possible parameters for the implementation to address the slow convergence, tendency to stagnancy [Xia et al., 2008].

Multiple, collaborative agent communities can improve the diversity and optimality of identified solutions (a shared goal), without an optimal, pre-defined hyper-parameters set or *a priori* knowledge about the environment. Several communities can independently tackle the optimisation problem in parallel, and maximise the explored/exploited search space. Each community searches and converges into different areas in the search space. When a dynamic change occurs, the multi-community approach has knowledge from a set of previously good solutions whereas a single community approach knows about only a single solution. Agent collaboration for multi-agent systems [Ferber and Weiss, 1999] has been shown to benefit the accomplishment of complex tasks, sharing constrained resources, or achieving individual or shared goals. A successful collaboration process requires agents to form an effective collaboration community. This includes agents that are willing to support other community members by providing useful domain information and taking actions towards a shared goal [Huebscher and McCann, 2008]. The rest of section presents how such an agent community is formed, explain the reasoning behind the collaboration process, and introduce the mechanism used to exchange information about the solution space between multiple communities of agents.

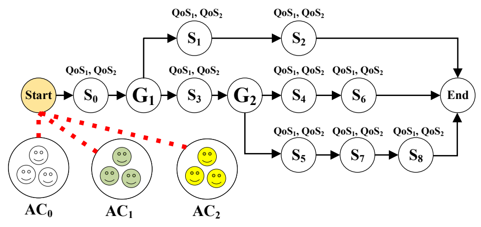
As the mobile and service agents interact on behalf of users seeking services to be composed subject to certain QoS constraints and objectives, communities of interest begin to emerge. These agent communities are formed using agents who share the same interests, and may overlap by sharing goals or available resources in the environment (e.g., the services and devices available in the service-sharing communities). For finding QoS-optimal service composition configurations, these interests could be exploring solutions bound to only a single region of the Pareto-optimal set of solutions, or optimising only a partition of the service dependency



(a) Single agent community.



(b) Multiple agent communities each optimising a partition of the graph.



(c) Multiple agent communities each optimising the entire graph.

Figure 3.6: Overview of the optimisation process using single and multiple agent communities, where AC_0, AC_1 and AC_2 are agent communities, S_i is a service agent, and G_i is an exclusive choice control element (i.e., Guidepost [Chen et al., 2018]). Sub-figure 3.6a shows the optimisation process starting from the Start (root) node of the given service dependency graph. Sub-figure 3.6b shows three colonies AC_0, AC_1 and AC_2, where AC_0 starts exploring the service dependency graph from the *Start* node (root), and AC_1 and AC_2 start from the guideposts G_1, respectively G_2. The process in sub-figure 3.6c is similar to the optimisation process in sub-figure 3.6c, but all the agent communities start traversing the graph from the *Start* node.

graph. As more services are composed, the mobile agents become more efficient and effective by interacting with the agents in the communities most likely to be able to provide them with required service components [Shen et al., 2011]. This can be locating services that maximise the overall QoS.

An agent community can choose to search for optimal solutions in only a partition of the service dependency graph. For example, in Figure 3.6b three agent communities are available: AC_0, AC_1, AC_2. In this case, AC_0 may start from the *Start* node, AC_1 may start from the exclusive choice control element (guidepost) G_1, and AC_2 may start from guidepost G_2. Heuristic-based methods can be used to select the starting service agents, or the agent communities may overlap with between several concurrent service compositions [Shen et al., 2011]. The *Start*, *G1* and *G2* also represent the source and destination of the mobile agents. If an agent community chooses to search for solutions in the entire service dependency graph, then the optimisation process will start from the *Start* node (e.g., Fig. 3.6a, Fig. 3.6b, Fig. 3.6c). With regards to the destination, all available communities can arrive at the same destination service agent, or they may arrive at different destinations. This work assumes that all the communities start and end at service agents available in the given service dependency graph.

The proposed information exchange mechanism uses the pheromone values from multiple communities for the same direction (neighbour) to efficiently explore the solution space. This work assumes that the transfer happens when a mobile agent ma_i needs to select a neighbour to explore. Equation 3.6 replaces Equation 3.2 for calculating P_d, which is the probability of choosing the neighbour d (adjusted from Equation 3.2). More precisely, a mobile agent ma_i will use Equation 3.6 instead of Equation 3.2 for selecting a neighbour (direction) to explore. The pheromone information for neighbour d from all the agent communities having information about that node is aggregated, averaged, and then used to select the next node as follows:

$$P_d = \frac{[\tau]^{\alpha}[avg(\tau)^{\lambda} + var(\tau)^{1-\lambda}]^{\beta}}{\sum_i^N [\tau_i]^{\alpha}[avg(\tau)^{\lambda} + var(\tau)^{1-\lambda}]^{\beta}} \tag{3.6}$$

where $avg(\tau)$ is the average of the pheromone level from all the communities except the community of the current mobile agent making the selection, $var(\tau)$ is the variance of this value, λ is a parameter used to select the preference (weight of) the average and variance values, and β represents the preference for external information (knowledge) from other communities.

The use of multiple collaborating communities allows SBOTI to use more knowledge to efficiently explore the solution space, with Equation 3.6 replacing Equation 3.2 (Algorithm 5). In Figure 3.5, the *Pheromone Store Querying* and *Neighbour Selecting* states use the information provided by all the communities available in the system.

3.5 Limitations Due to Design Decisions

SBOTI is inspired by a SI system, and two limitations are inherited because of this. This work outlines these limitations and presents how they are addressed to mitigate these limitations. A detailed evaluation of the performance and feasibility of SBOTI is presented in Chapter 5.

(**L1**) *Time Critical Applications*: The optimal service composition configurations are neither predefined nor pre-programmed. These configurations emerge from the environment through continuous exploration. Because of this, SBOTI may not be suitable for time-critical applications that require hard-QoS [White et al., 2017b]. However, this work explores if SBOTI can provide satisfactory solutions within restrictive time-frames.

(**L2**) *Parameter Tuning*: This is known as a general drawback of the SI-based systems. This work proposes using a collaborative approach to address this limitation. By using multiple agent communities with diverse properties, this thesis explorers whether the utility of identified solutions can be improved.

(**L3**) *Stagnation*: This is another known general drawback of the SI-based systems. Because of lack of a central coordinator, SI-based systems may converge to a local optimum. For example, in ACO, stagnation occurs when all the ants follow the same path. In the context of this work, this means that the same service composition configuration is identified in every iteration. This work proposes an adaptation procedure to avoid stagnation on sub-optimal path.

3.6 Chapter Summary

This chapter introduced SBOTI, a QoS optimisation mechanism for flexible service composition in mobile environments. To address the challenges of a mobile environment, SBOTI is offers support for flexible service composition, is resource-aware and mobility-aware, and can produce a set of Pareto-optimal solutions to allow users to make compromises between the service composition configurations available in a mobile environment. Section 3.1 introduced the required features for a QoS optimisation mechanism for flexible service composition in a mobile environment, and Section 3.2 introduced how the proposed agent-based approach is mapped to the physical environment. Section 3.3 presented the design decisions made to implement the required features presented in Section 3.1 in the mobile environment.

Section 3.4.1 introduces SBOTI, a decentralised, QoS optimisation mechanism for automatic, flexible service composition. SBOTI uses a community of homogeneous, mobile software agents, which share the same goal, to effectively and efficiently approximate the set of QoS-optimal service composition configurations available in a geographically-limited, mobile environment. The proposed mechanism uses an iterative, reinforcement-based approach to control the trade-offs between computational efficiency and the optimality of the identified service composition solutions. Furthermore, it uses stigmergic-modelling (presented in Section 3.4.1) to deal with the mobility of service providers. SBOTI incorporates a non-dominated sorting technique to identify the Pareto-optimal set solutions, which allows the user to explore various QoS trade-offs (Section 3.4.2). To control the diversity of the solutions in this set, SBOTI globally updates both dominated and non-dominated solutions using digital pheromones.

New service composition configurations may emerge (with potentially better QoS) as a result of service providers mobility. The digital pheromone mechanism used by such approaches may affect the exploration of new service composition configuration that may emerge as a result of service providers mobility. To allow for exploration of new service composition configurations that may emerge as a result of providers' mobility, SBOTI uses an adaptation procedure that limits the amount of pheromone on previously identified solutions. This procedure is presented in Section 3.4.4.

Given the dynamic nature of the environment under consideration for this work, re-optimisation needs to be performed to closely track any changes in the available service composition configurations. SBOTI's optimisation process requires a number of hyper-parameters to be configured, such as the number of mobile agents required to explore the search space, the initial pheromone level $\tau_{initial}$ associated with each service agent, the evaporation frequency ρ parameter that allows new paths to be explored. A collaborative approach to engage multiple communities of agents for provisioning QoS-optimal service compositions in mobile environments is presented in Section 3.4.5. New service compositions (with better QoS) can emerge from local decisions and interactions with agents from diverse communities. Several communities can independently tackle the optimisation problem in parallel, and maximise the explored/exploited search space. Each community searches and converges into different areas in the search space. When a dynamic change occurs, the multi-community approach has knowledge from a set of previously good solutions whereas a single community approach knows about only a single solution.

Section 3.5 highlights the limitation introduced due to design decisions. These limitations are because of the limitation inherited from SI-based systems. This section also presents how these limitations are addressed in this thesis. The last section summarises this chapter.

Chapter 4

Implementation

The previous chapter described the design of SBOTI, presented the main components of this mechanism, and described how they address the challenges of flexible, QoS-aware service provision in mobile environments. SBOTI is designed and implemented on top of the Service Composition & Execution Engine (SCE) component in the Service-centric network for URban-scale Feedback systems (SURF) middleware [Cabrera et al., 2017a]. The SURF middleware offers the necessary components for service discovery, QoS-aware service composition and optimisation, QoS monitoring and prediction, runtime Service Level Agreement (SLA) between service consumers and service providers, and re-negotiation support in case these agreements can not be enforced.

This chapter is structured as follows. Section 4.1 introduces the SURF middleware, enumerates the main components of the middleware, and describes their interaction to provision QoS-aware service compositions in dynamic IoT environments. Section 4.2 presents the architecture and implementation details of SBOTI in the context of the Simonstrator platform. Section 4.3 summarises this chapter.

4.1 SURF Middleware

The SURF middleware is a distributed software-layer that can be deployed in a network of IoT gateways [Cabrera et al., 2017a]. Figure 4.1 shows the main components of this middleware and their interaction. The main components are Service Discovery Engine (SDE), Service Discovery Engine (SDE), QoS Monitoring Engine and QoS Prediction Engine, SLA Negotiator and SLA Manager. Two utility components are included to facilitate the QoS-aware service provisioning process: a Request Handler and a Service Registry. This middleware is designed

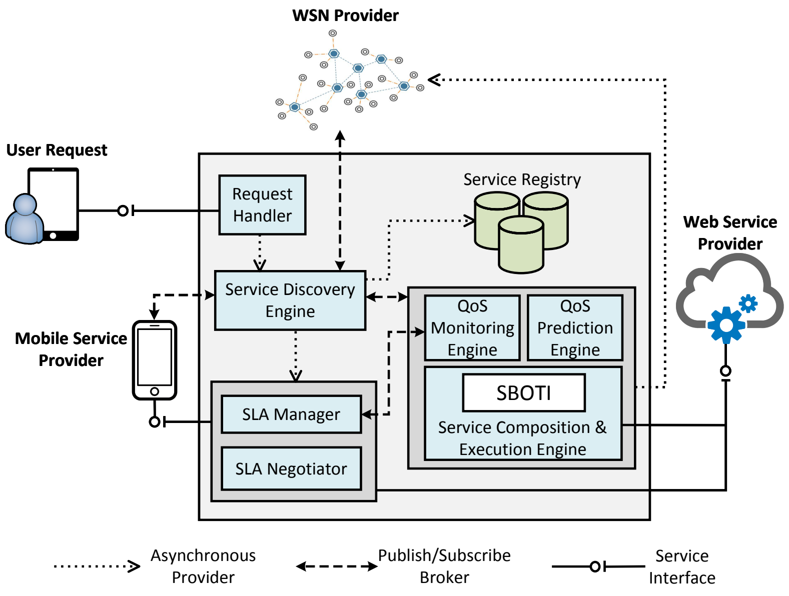


Figure 4.1: Architecture showing the main components of the SURF middleware.

to be deployed on IoT gateways and to manage services from different service providers (i.e., web services, WSN services, and mobile service providers). Depending on the capabilities of the physical device, an IoT gateway can be a fixed (embedded) or a mobile (e.g., a smartphone) [Aloi et al., 2017]. The reminder of this section briefly describes each component.

4.1.1 Request Handler

Service consumers such as smart city applications or software developers access the available services through (service) requests. In this work, a request is defined as $r = \langle I, O, QoS \rangle$, consisting of the request inputs, outputs and QoS requirements. The Request Handler is a utility module that obtains user's request and formalises this request in a format that can be used by the Service Discovery Engine (SDE) to find the service composition plans that can functionally satisfy the composition request, and by the Service Composition & Execution Engine (SCE) to perform the runtime service composition procedures (i.e., QoS optimisation, and service execution and adaptation). The Request Handler exposes an interface that receives requests, establishes the communication channel with the application and with the user, and sends requests to other components in the architecture.

4.1.2 Service Registry

The Service Registry is a distributed component that can store the descriptions of services. This component facilitates the service discovery process. Each available service has a description $s_{desc} = \langle id, I, O, D \rangle$, consisting of a service identifier, inputs, outputs, and domains. The registration can be pro-active, re-active or both. When a new service provider becomes available, this provider can register its services with the gateway. Also, the service provider can announce its presence and push the details of the stored services to this gateway.

4.1.3 Service Discovery Engine

The Service Discovery Engine (SDE) is a distributed component used to perform the discovery of service composition configurations that can satisfy users' requests. This component is notified of users' request by the Request Handler component (Section 4.1.1). When such a notification is received, the SDE searches for services in the distributed registry. The output of this search is a set of service composition configurations (or service plans) that satisfy the request from the functional perspective (i.e., input and outputs matching). Each service plan can have one or more services. In addition to the service components, each service plan includes the data-flow and control-flow relations between these service components (Section 1.2). The set of service plans are passed to the Service Composition & Execution Engine (SCE) that chooses the best plan according to the non-functional requirements (i.e., QoS parameters).

4.1.4 QoS Monitoring Engine and QoS Prediction Engine

IoT applications can be used in a range of domains that have different QoS requirements based on the sensitivity and criticality of the application. IoT application QoS can typically be categorised as best-effort (no QoS), differentiated services (soft QoS) and guaranteed services (hard QoS) [White et al., 2017b]. To maintain Soft QoS guarantees for IoT applications, service execution require continuous monitoring to ensure QoS is maintained, with processes required for corresponding real-time negotiation and adaptation to cater for perturbations in the environment. The choice for monitor placement in related work is either on the service execution server [Baresi et al., 2004, Artaiam and Senivongse, 2008, Cabrera and Franch, 2012] or designed to overhear events generated by services [Mahbub and Spanoudakis, 2004, Baresi and Guinea, 2005, Barbon et al., 2006] or to monitor at the client side [Jurca et al., 2007]. Existing monitoring frameworks include Wetzstein et al. [Wetzstein et al., 2009], who provide a framework for dependency analysis, using machine-learning techniques to analyse the main factors that influence performance, but does not allow for distributed execution of services and assumes a single workflow engine, and Ameller and Franch [Ameller and Franch Gutiérrez, 2008], who describe a framework with a monitor, an analyser that checks for SLA violations and a decision-maker that provides treatment for violations, but requires instrumentation of the servers that are running the services. In an urban scale environment, requiring every service provider to be instrumented is an unreasonable assumption. To address these limitations, the QoS Monitoring Engine and QoS Prediction Engine are introduced in the SURF middleware.

The QoS Monitoring Engine captures the user side QoS and stores the values in a distributed registry. The communication links for invoking these services are diverse, which may affect the personal QoS experience of users, so they are monitored using the monitoring engine. The QoS Prediction Engine focuses on forecasting future values of currently executing services to identify if they may be about to fail and makes predictions for component services that the middleware may select when composing an application based on other users in the environment. Each service consumer contributes with QoS information to enable collaborative filtering. The values reported by the users are then input for the IoTPredict algorithm [White et al., 2018b] to make QoS predictions for candidate services. This algorithm can achieve higher prediction accuracy compared to the state-of-the-art approaches, while maintaining a low overhead.

4.1.5 Service Composition & Execution Engine

The Service Composition & Execution Engine (SCE) searches for the optimal service composition configurations using the QoS information that the QoS Monitor Engine and QoS Prediction Engine provide (Section 4.1.4). The SCE is initialised using the set of service plans provided by the SDE (Section 4.1.3). This component merges all these plans into as a service dependency graph using an AND/OR graph structure introduced in Section 2.1.2.

In a dynamic environment, service components may have intermittent availability, or, if the devices that provide these service components are mobile, the service components may become permanently unavailable. Because of the frequent changes, finding a QoS-optimal service composition configuration (service plan) may be difficult. By the time the search process for such a configuration finishes, the service components may become unavailable or the QoS information may be out-of-date. Also, a user may want to explore various trade-offs between the QoS attributes.

SBOTI is introduced to address these requirements. The optimisation process is distributed over the available gateways. Service component failures caused by the intermittent availability of the nodes are managed by requesting a replacement from the SDE. Also, SBOTI uses a stigmergic-based mechanism to cope with this dynamic environment. The architecture and design of SBOTI is presented in Section 3.4. Also, the implementation details of this design is presented in Section 4.2.

4.1.6 SLA Negotiator and SLA Manager

Best-effort services are not sufficient for mission-critical applications like transportation, healthcare, and emergency response [Islam et al., 2015]. To provide a certain level of control to service consumers and deliver services with quality guarantees, a Service Level Agreement (SLA) is used as a contract-like concept to formalise the obligations of the involved parties [Ludwig et al., 2003]. This agreement is the result of an SLA negotiation process, which is performed to tailor service properties (e.g., QoS) before the actual service delivery [Faniyi and Bahsoon, 2016].

The SURF middleware contains two components to deal with the SLA negotiation and compliance process: SLA Negotiator and SLA Manager. In particular, the SLA Negotiator analyses the agreement templates submitted by service providers, selects the candidate service providers that have potential to satisfy the requirements, and negotiates on behalf of service consumers to solve possible conflicts between service providers and consumers. The SLA Manager ensures compliance with the agreed terms at runtime (i.e., during service execution time). This component subscribes to performance deviation notifications from the QoS Monitoring Engine (Section 4.1.4) through a publish/subscribe broker. The SLA Manager subscribes to the QoS Monitoring Engine for performance deviation notifications, to trigger SLA negotiation in case the current performance does not comply with the guarantee terms and measurement metrics specified in the negotiated SLA agreement [Palade et al., 2018a].

4.2 SBOTI Implementation

The SBOTI optimisation process performs in the context of the SURF middleware. SBOTI is part of the SCE component. This section shows the implementation of SBOTI's main components by following the design specifications presented in Section 3.4. SBOTI is implemented on the Simonstrator platform [Richerzhagen et al., 2015], which is an event-based simulator implemented in Java for mobile applications. The Simonstrator platform has been selected because of the portability of the implementation. The Simonstrator platform can provide a runtime environment, which enables systems to run on network simulators such as OMNeT++, commodity PCs, or Android mobile devices [Richerzhagen et al., 2015]. The simulator facilitates the performance evaluation of service-based applications in mobile environments.

Figure 4.3 shows how SBOTI was implemented in Simonstrator. The figure can be split into two parts. The first shows the main components of Simonstrator. The second shows SBOTI's components, their interaction and how these are connected to Simonstrator.

In Simonstrator, all individual devices are modelled as *Hosts*. This means that all the active entities that are part of SBOTI (e.g., service agents) are modelled using the *Host* interface. All the Simonstrator's internal features (e.g., *Overlay*, *Network*, *Service*, *Sensor*) are exposed as components. These components are accessed through a *Host* object. In addition to these, Simonstrator offers a *Scheduling* component and an *Instrumentation* component. The *Scheduling* allows for operations (events) to be generated. The *Instrumentation* offers logging features.

4.2.1 Agent Design

SBOTI uses a multi-agent, decentralised architecture, which allows service composition configurations to adapt in decentralised environments. As introduced in Chapter 3, there are two types of agents that are used by this optimisation mechanism: *Service Agent* and *Mobile Agent*. SBOTI uses the stigmergic coordination of these agents to perform the optimisation process. Figure 4.2 shows SBOTI's class diagram. The main components of SBOTI are:

- **Service Agent**. A *Service Agent* is associated with each service in the service dependency graph, and it has the following features:

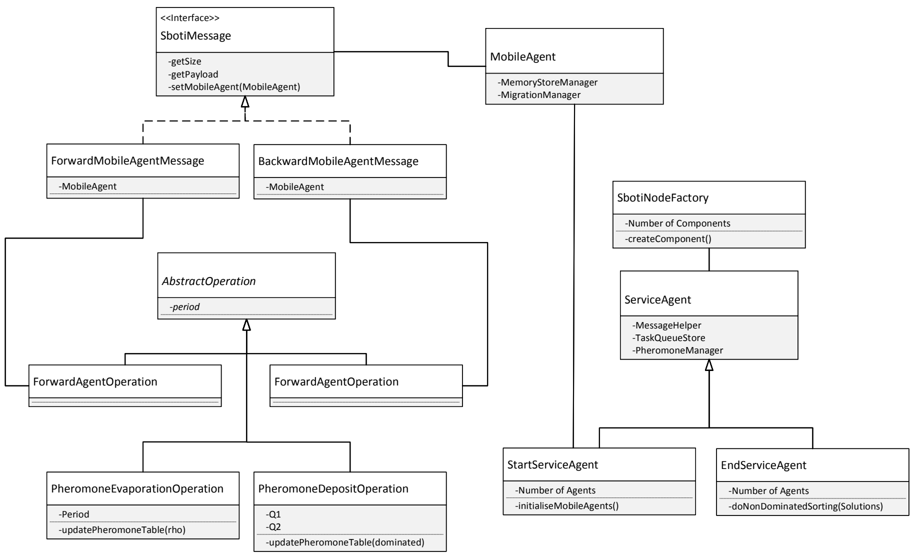


Figure 4.2: Class Diagram.

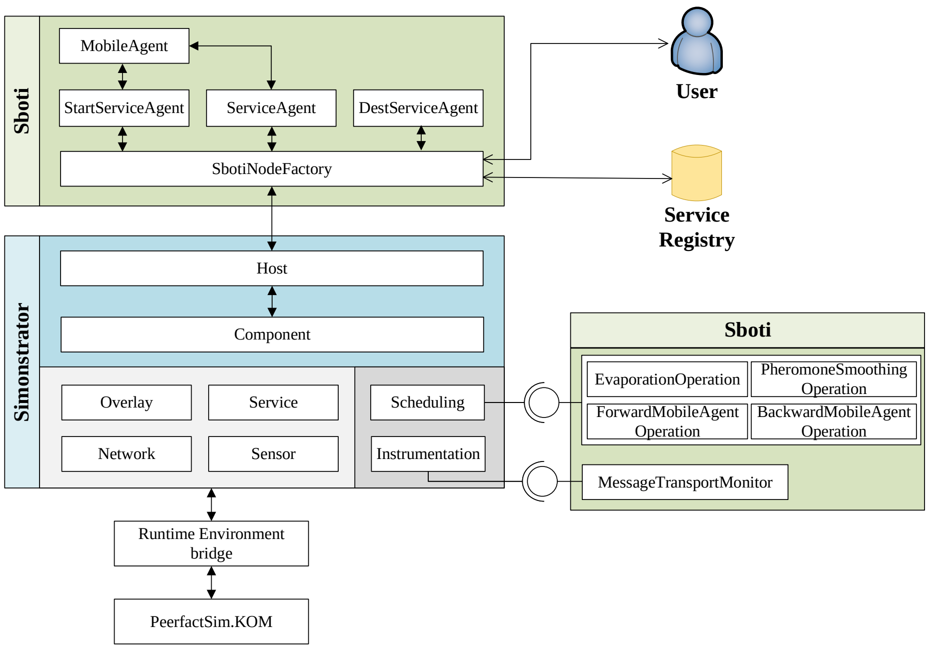


Figure 4.3: SBOTI prototype implemented in Simonstrator [Richerzhagen et al., 2015].

– **Message Helper**. This component deals with serialisation and deserialisation of the incoming and outgoing messages. When a message is received, this component retrieves the message and extracts the header and body of the message, and processes the body of the message based on the type of task extracted from the header. Also, this component creates the messages that are to be sent to the next service agents. Mobile agents are wrapped in messages when they migrate between service agents.

  - **Task Queue Store**: This structure stores the tasks that are performed by the agent. Two types of tasks are stored in this queue, which are associated with the incoming messages and the outgoing messages. In case of an incoming message, which represents the arrival of a mobile agent at a service agent, a task is created to include the details of the mobile agent. In case of an outgoing message, a task which includes the details of the mobile agent is stored in this queue, until the message has been acknowledged by the receiving node. Using a separate sending/receiving queue for processing tasks facilitates parallel execution of tasks, and the resilience of the system. This features allows for re-transmissions in case a message delivery times out. These transmissions are performed through *Forward Mobile Agent Operation* and *Backward Mobile Agent Operation* components.
  - **Pheromone Store Manager**: This structure is the implementation of the *Pheromone Store* concept presented in Section 3.4. The component uses a hash map data structure to associate each successor neighbour in the service dependency graph with a pheromone scalar. This component facilitates the periodic execution of the *Pheromone Evaporation* and *Pheromone Deposit* operations. This component also performs the pheromone smoothing procedure presented in Section 3.4.4. The *Pheromone Evaporation* operation is performed through the *Evaporation Operation* component. The pheromone smoothing procedure is performed through the *Pheromone Smoothing Operation* component.

- **Start Service Agent**. This component is responsible for the initialisation of the set of mobile agents that are used during the optimisation process. It initialises and configures each mobile agent, and forwards each mobile agent towards the service agent responsible for the first service in the service dependency graph. Also, this component is responsible for collecting all the mobile agents when they return from traversing the service dependency graph.

- **End Service Agent**. This component is responsible for storing the identified solutions by the mobile agents. After the solutions are stored, it uses a non-dominated sorting technique to identify the Pareto-optimal solutions. This component is also selects the reinforcement strategy that will be used during the execution of the mechanism (Section 3.4.2).

- **Sboti Node Factory**. This component initialises the *Service Agent* components and allocates a *Start Service Agent* and a *End Service Agent*. A user (developer) specifies the properties of the service dependency graph. Then, the this component initialises the *Service Agent* components using the properties of this graph. Also, this component contacts a *Service Registry* to extract a set of QoS values, which will be used to initialise the services in the service dependency graph.

- **Mobile Agent**. This entity is used in the optimisation process. This component traverses the service dependency graph, and collects the QoS of each service in the path. The *Start Service Agent* component initialises a set of *Mobile Agents*, and are forwarded as messages towards the *Service Agent* associated with the first node in the service dependency graph. A *Mobile Agent* provides two features:

  – **Memory Store Manager**: This component is the implementation of the memory store as described in the design chapter. It is implemented as a list using a stack data structure. This component provides the necessary procedures to update the memory store. These procedures are the insertion of new service agent address in the memory store when the mobile agent traverses in the forward direction, and dequeuing when the mobile agent traverses in the backwards direction. When the memory store is empty, the mobile agent is transferred to the *Start Service Agent*. The logical address of the *Start Service Agent* can be embedded into the memory of a *Mobile Agent*.

  – **Migration Manager**: This component is responsible with the traversal procedures of the mobile agent. In the forward direction, this component queries the *Pheromone Store Manager* associated with the current service agent for the neighbours of the current service, and the pheromone values associated with each neighbour. A probabilistic procedure (as explained in Section 3.4.2) is used to select which neighbour will be visited next. In the backward direction, this component is responsible for updating the *Memory Store Manager*.

- **Message Transport Monitor**. This is an utility component that records the messages sent between the *Host* components in Simonstrator. This

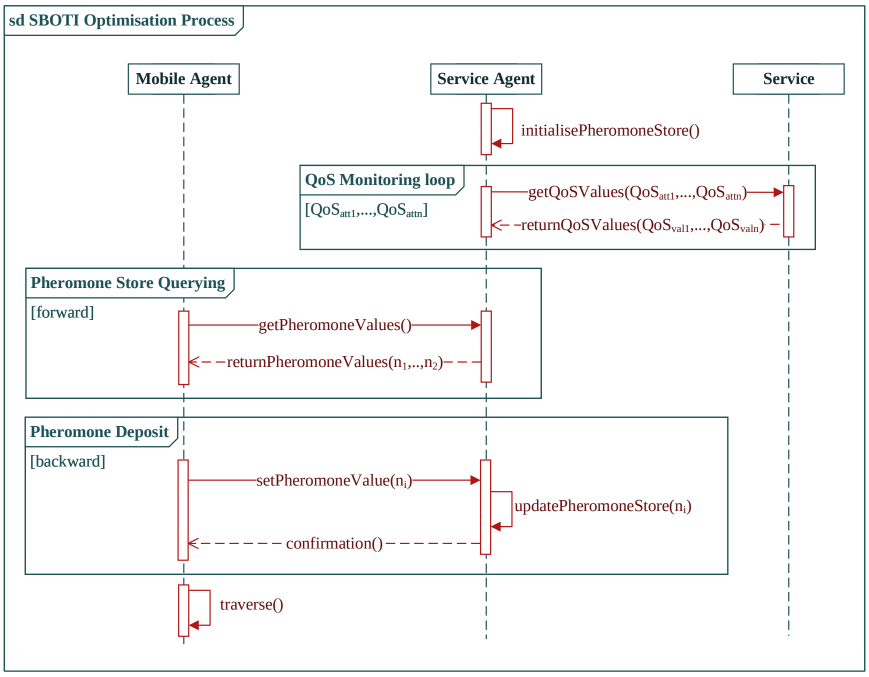


Figure 4.4: Sequence diagram showing the interactions between the main actors of the system: *Mobile Agent*, *Service Agent* and *Service*. This figure complements Figure 3.5 by showing the messages exchanged between these actors.

component is used to measure the communication overhead.

4.2.2 Agents Interaction

Figure 4.4 shows the interactions between the main actors of the SBOTI mechanism: *Mobile Agent*, *Service Agent* and *Service*. The Mobile Agent and Service Agent concepts were introduced in Section 3.4.1. The Service is a node in the service dependency graph, which is maintained in the DCON network (Section 3.2). The UML 2.0 specification was used to generate this sequence diagram[1]. Figure 4.4 complements Figure 3.5 by showing the messages exchanged between these actors. The self-messages (events) that trigger the state changes are shown in Figure 3.5.

Figure 4.4 shows the SBOTI optimisation process. The *Service Agent* sends an *initialisePheromoneStore()* request to initialise its pheromone store, based on the successor neighbours in the service dependency graph. This agent then starts a *QoS Monitoring* loop to retrieve the values of each QoS attribute of user' interests. These values are retrieved from a local registry that is deployed on the same device as the service. The monitoring component samples the QoS of the *Service*, and stores these values in a local registry for further analysis.

[1]https://www.ibm.com/developerworks/rational/library/3101-pdf.pdf

Alternatively to this sampling procedure, the QoS Monitoring loop could use the QoS values reported by previous users to make forecast the future values of these QoS attributes. This procedure is performed by the QoS Monitoring and Prediction Engine (Section 4.1.4).

The *Mobile Agent* performs two procedures: *Pheromone Store Querying* and *Pheromone Deposit* (Section 3.4.2). In the *Pheromone Store Querying* procedure, the *Mobile Agent* sends a synchronous message request *getPheromoneValues()* to the *Service Agent* to retrieve the pheromone values associated with each neighbour in the pheromone store of the *Service Agent*. The *Service Agent* replies to this messages with the contents of the pheromone store. This procedure is triggered only when the *Mobile Agent* moves in the *forward* direction in the service dependency graph. In the *Pheromone Deposit* procedure, the *Mobile Agent* sends a synchronous message request *setPheromoneValue(n_i)* to the *Service Agent* to set the pheromone value for i-th element in the pheromone store. When this request is received, the *Service Agent* generates an *updatePheromoneStore(n_i)* self-message to update the i-th element in its pheromone store. The *Service Agent* sends a confirmation message to the *Mobile Agent* to inform the end of the *Pheromone Deposit* procedure. The *Mobile Agent* produces a self-message to continue the traversal of the service dependency graph.

4.3 Chapter Summary

SBOTI is implemented the SURF Middleware, and performs in the context of the Service Discovery Engine (SDE) component. Section 4.1 introduces the SURF Middleware, which is a distributed software-layer deployed in a network of IoT gateways. The gateways may be fixed or mobile. The main components are Service Discovery Engine (SDE), Service Discovery Engine (SDE), QoS Monitoring and Prediction Engine, SLA Negotiator and SLA Manager, and two utility components Request Handler and Service Registry. The SURF middleware can manage services from different service providers (i.e., web services, WSN services, and mobile service providers). Section 4.2 described the implementation of SBOTI based on the design description presented in Chapter 3.

Chapter 5

Evaluation

This chapter evaluates SBOTI's performance in a mobile environment, and compares the results with GoCoMo [Chen et al., 2018], SimDijkstra [Jiang et al., 2014], and Random approach. Three evaluation objectives are realised in this chapter: (i) to determine whether SBOTI can outperform the existing heuristic-based proposals for flexible, QoS-aware service composition in mobile environments; (ii) to determine whether SBOTI, while adapting to the mobile environments, can maintain or further improve the diversity and optimality of identified solutions; (iii) to determine whether SBOTI can improve the optimality and diversity of identified solutions by increasing the communication overhead, and by re-using knowledge about the solution space from other agents.

The chapter is structured as follows: Section 5.1 maps the evaluation objectives to the research questions, and introduces the performance metrics measured during the evaluation. Section 5.2 presents the statistical analysis performed to investigate whether the observed differences in results are statistically significant. Section 5.3 presents the evaluation setup. Section 5.4 presents the evaluated baseline algorithms and Section 5.5 the test cases proposed to perform the evaluation. Three studies are presented in Section 5.6, Section 5.7 and Section 5.8 to evaluate the limitations of SBOTI under various dynamic conditions. Section 5.10 summarises this chapter.

5.1 Evaluation Objectives and Performance Metrics

The existing proposals for flexible, QoS-aware service composition in mobile environments are generally based on best-effort approaches. Such approaches trade-off optimality for computational efficiency. Further more, these approaches have been developed to address the optimisation problem from a single-objective perspective. This means that they only return one solution. This section evaluates the performance of SBOTI to measure to what extent can the trade-off between computational efficiency and optimality be controlled, such that a broad range of composition configurations can be presented to users (research question **Q1**), to what extent the service providers' mobility can be leverage through an adaptive approach to exploring new service compositions (research question **Q2**). Also, this section measures to what extent will more search agents improve the optimality and diversity of identified service compositions (research question **Q3**). Table 5.1 shows how these evaluation objectives are mapped to the research questions.

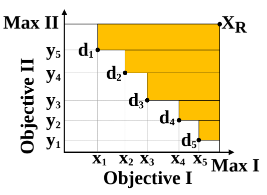
To be able to evaluate SBOTI's performance, and compare the results with baseline algorithms for QoS optimisation in mobile environments, a number of metrics are required. In general, the performance of an algorithm is assessed using both the quality of identified solution and the amount of resources required to generate that solution. While this is true for both Single-Objective Optimisation (SOO) and Multi-Objective Optimisation (MOO) problems, there is a substantial difference in the process of estimating the performance of the resulted solution (in the SOO case) or a set of solutions (in the MOO case). In SOO problems a single value is produced by using users' preferences. The larger this value, the better the solution produced. The outcome of a MOO problem is a set of mutually incomparable Pareto-optimal solutions. The size of dominated space metric (Section 5.1.1) and spread metric (Section 5.1.2) are used to measure the performance of this set of solutions. To compare the performance with SOO-based proposals, the utility metric is used (Section 5.1.3). Finally, the communication overhead metric presents the communication cost introduced by each evaluated algorithm (Section 5.1.4).

5.1.1 Size of Dominated Space

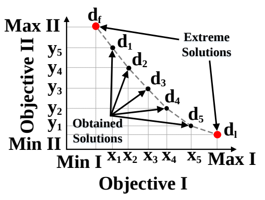
This metric indicates how well the Pareto-optimal set is approximated by the set of solutions [Zitzler and Thiele, 1999]. The greater the size of the space dominated, the closer the solutions are to the Pareto-optimal set [Wang et al., 2015]. The size of dominated space of a set A of solutions, which contains only non-dominated solutions, is calculated as follows:

Research Question	Proposed Study	Performance Metrics
Q1 *Multi-objective optimisation*	Study 1: Stigmergic Optimisation and Diversity Strategy (Section 5.6)	- Size of Dominated Space - Spread - Utility of Solutions - Communication Overhead
Q2 *Adaptation*	Study 2: Adaptation Support (Section 5.7)	- Size of Dominated Space - Spread - Utility of Solutions - Communication Overhead
Q3 *Efficient Exploration*	Study 3: Collaborative Agent Communities (Section5.8)	- Size of Dominated Space - Spread - Utility of Solutions - Communication Overhead

Table 5.1: Mapping of Evaluation Objectives to Research Questions.



(a) Size of Dominated Space.



(b) Spread.

Figure 5.1: Size of dominated space and spread metrics. Figure 5.1a shows the space dominated (in orange) by a given Pareto set (points d_1, d_2, d_3, d_4 and X_5) when two objectives are minimised. Figure 5.1b outlines the Euclidean distance between these points.

$$PS(A) = \frac{(Max\ I - x_1) * (Max\ II - y_1)\ + \sum_{i=2}^{n}(Max\ I - x_i) * (y_{i-1} - y_i)}{Max\ I * Max\ II} \tag{5.1}$$

where $Max\ I$ and $Max\ II$ are the obtained maximum values of two given objectives: Objective I and Objective II. These two values create the Reference Point X_R. Figure 5.1a shows the composition of set A, which is a Pareto front made of 5 points. The size of the dominated space is illustrated by the area covered by the orange surface. In this work, the Reference Point X_R was set to the highest throughput plus 5 units and the lowest response time minus 0.1 units.

5.1.2 Spread

This metric measures the extent of non-uniformity achieved in the obtained solutions [Deb et al., 2002]. The obtained set of solutions should span the entire Pareto-optimal region. The Euclidean distance between consecutive solutions in the obtained non-dominated set of solutions is identified, and then the average of these distance is calculated. The extreme solutions (in the objective space) are identified using all the solutions obtained from the evaluated algorithms. This metric is calculated as follows:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left|d_i - \overline{d}\right|}{d_f + d_l + (N-1)\overline{d}} \tag{5.2}$$

where the parameters d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions in the obtained non-dominated set. The parameter $\overline{d}$ is the average of the distances between all points d_i, given by the N solutions, with $N-1$ distances between these solutions. Figure 5.1b shows the distribution of five points d_1, d_2, d_3, d_4, d_5. A uniformly spread set would make all the distances between these points equal to $\overline{d}$, and the parameter Δ would be zero [Deb et al., 2002].

5.1.3 Utility

This metric indicates the overall satisfaction of QoS requirements of a service composition configuration [Zeng et al., 2004, Alrifai and Risse, 2009, Ye et al., 2016]. 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. The utility metric is calculated as follows:

$$U_i min = \begin{cases} \frac{Q_i^{max} - Q_i}{Q_i^{max} - Q_i^{min}} \\ 1 \end{cases} \quad and \, U_i max = \begin{cases} \frac{Q_i - Q_i^{min}}{Q_i^{max} - Q_i^{min}} & if Q_i^{max} - Q_i^{min} \neq 0 \\ 1 & if Q_i^{max} - Q_i^{min} = 0 \end{cases} \tag{5.3}$$

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 5.3 computes utility of a composite from the perspective of an objective that needs to be minimised or maximised [Alrifai and Risse, 2009]. The utilities are aggregated as follows:

$$Utility_g = \sum_{1}^{i} U_i * W_i \tag{5.4}$$

where W_i represents the importance weights assigned to the i-th utility metric. Each weight is a number in the range [0, 1] and the sum of all weights is equal to 1. Evaluations with various weights are performed to assess the utility of produced solutions, while covering the entire Pareto-front. Table 5.3 shows the values of these weights used in this evaluation.

5.1.4 Communication Overhead

This metric measures the number of exchanged messages between the nodes used during the composition. All the types of messages that are exchanged are included, counting retransmission or probing messages.

5.2 Statistical Analysis

Because this work deals with stochastic algorithms, the observed results must be provided with a certain level of confidence. Also, some aggregation numbers that summarise the average and deviation tendencies must be considered when using a performance indicator (i.e., size of dominated space and spread indicators). To provide this, a statistical analysis procedure is performed to verify whether the observed differences in results of the evaluated algorithms are statistically significant [Talbi, 2009]. First, a Kolmogorov-Smirnov test and a Shapiro–Wilk test are performed to verify whether the obtained results follow a normal (Gaussian) distribution. This step allows to decide between non-parametric and parametric tests. Generally, if results do not follow a normal distribution and the sample size is small, non-parametric tests such as Wilcoxon-Mann-Whitney are performed. Otherwise, parametric tests such as *t-test* are used [Alba, 2005]. This statistical tests were selected because they are the most used statistical tests for dynamic multi-objective optimisation [Bechikh et al., 2016].

In all tests in this work, the confidence level is set to 95% in all statistical tests (p-value under 0.05), which means that the differences are unlikely to have occurred by chance with a probability of 95% [Durillo et al., 2006]. The sample size is set to 50 data points. A data point is the value of the measured performance indicator in the 250-th iteration. If the observed differences are significant, the effect size is recorded [Grissom and Kim, 2005]. The effect size is calculated as follows:

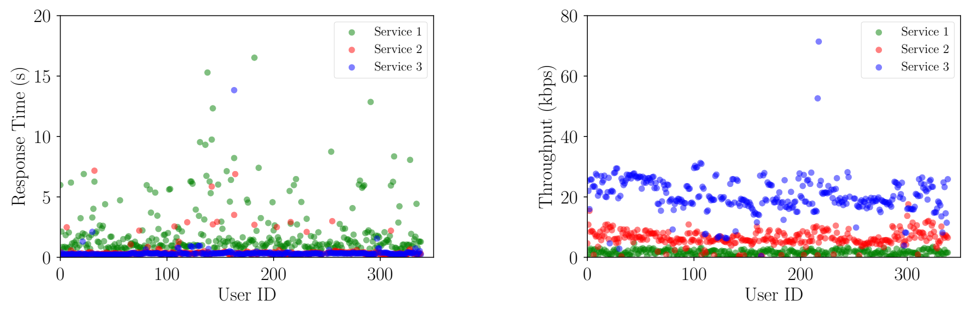
$$\widehat{p}_{a,b} = \frac{U}{n_a n_b} \tag{5.5}$$

where U is the statistic calculated using Wilcoxon-Mann-Whitney test, n_a is the number of samples in group a, respectively group b for n_b. The parameter $\widehat{p}_{a,b}$ is the probability that an observation from group a will be higher than an observation from group b.

5.3 Experimental Setup

SBOTI is implemented on the Simonstrator platform [Richerzhagen et al., 2015], which is an event-based simulator implemented in Java for mobile applications. The experiments were performed on the Lonsdale cluster, which was provided by TCHPC at Trinity College Dublin, Ireland. Each node in the cluster has the following configuration: Linux OS, 2.30GHz AMD Opteron processor, and 16GB of RAM[1]. A mobile environment was simulated where each node represents a mobile device. The environment was designed based on the specifications in

[1]https://www.tchpc.tcd.ie/resources/clusters/lonsdale



(a) (b)

Figure 5.2: Figure 5.2a and Figure 5.2b show the response time and throughput experienced by the users in the WS-DREAM dataset for three different services [Zheng et al., 2014].

Table 5.2.

Table 5.2: Simulator properties.

Simulator Parameters	**Value**
Speed	SLOW (1.5 - 2.5m/s) MEDIUM (2.5 - 7.5m/s) FAST (7.5 - 13.5m/s)
Communication Range	250m
Field Size	1000m^2
Node Placement	Random Position Distribution
Movement Model	Gauss Markov

Each device is set to offer one service, and each service is associated with two QoS attributes: response time and throughput. A QoS dataset is used to initialise the QoS of each service participant. This work uses the WS-DREAM dataset [Zheng et al., 2014], which consists of a matrix of response time and throughput values for 339 users by 5825 services. Figure 5.2a and Figure 5.2b show the response time and throughput experienced by the users in the dataset for three different services. To address the dynamism in the services available in the environment, the QoS values of each service are changed after every iteration by multiplying every value with a random number in the interval [0.5, 1.5] as performed in previous works (e.g., [Alsaryrah et al., 2018]). The formulae in Table 2.1 were used to aggregate the QoS values of composite participants (the services in a service composition). Only service compositions following a sequential pattern were used in this work.

5.4 Baseline Approaches

For the proposes of establishing a baseline against which to compare GoCoMo, this study is using existing proposals for flexible service composition in mobile environments. The baselines for comparison are:

1. **SimDijkstra**: an algorithm that uses a Dijkstra-based algorithm to find the shortest path in the service dependency graph [Jiang et al., 2014]. The execution ends when a leaf node is reached. A leaf node is the last service in a path in the service dependency graph.

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 from the leaf nodes to each exclusive choice structure in the service dependency graph [Chen et al., 2018].

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 [Klein et al., 2014].

These baselines use an utility function to transform the multi-objective optimisation problem into a single-objective optimisation. SBOTI does not require *a priori* articulation of preferences and may produce a set of solutions. To compare the performance of the solutions produced by SBOTI with the baseline single-solution approaches, the utility metric (Section 5.1.3) is used to select a single solution from the set of Pareto-optimal solutions produced SBOTI.

5.5 Test Case Generation

For the purpose of evaluation, a scenario based on an adaptive route planner application [Chen et al., 2018] is used. Such an application can be built using services provided by the available (mobile) devices in the environment. There can be many functionally similar services, 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 service dependency graph is generated using a service discovery component. This component can also perform in parallel to SBOTI (or its variants) to continuously adapt this functional structure relative to the changing environment. As mentioned in Section 1.4, this work assumes that such a component exists.

Four control variables are used in this work. The first two are given by the data structure that is used to represent the service dependency graph, which is a full and complete k-ary tree [Cormen et al., 2009] with the height h and k number of children. Each path in this graph from root to a leaf node represents a service composition configuration. Variable k controls the number of potential service composition configurations, whereas variable h is used to vary the length of the service composition configurations. For simplicity, only sequential service composition configurations are used. Each internal node in the service dependency graph is an exclusive choice control element (i.e., guidepost). The number of alternatives for this structure is given by the number of leaf nodes in this structure. The third control variable is the *speed* of devices in the environment (Section 5.3). The fourth control variable adjusts the weights w_{RT} and w_{TH} for the response time (RT) and throughput (TH) objectives used in this evaluation. These values are used to compute the utility of the final composition plan. These values were adopted by experimenting with various sets of parameters and by considering other researchers' earlier experiments [Wang et al., 2015]. Table 5.3 shows the values used to initialise the control variables.

Table 5.3: Control variables used in the studies.

Control Variable	**Description**	**Values**
h	Height of the tree	3, 4
k	The number of children in every node	2, 3, 4
Speed	The speed of nodes in the environment	SLOW (1.5 - 2.5 m/s) MEDIUM (2.5 - 7.5 m/s) FAST (7.5 - 13.5 m/s)
w_{RT}, w_{TH}	The weights of the QoS objectives	(0.01, 0.99), (0.25, 0.75) (0.50, 0.50), (0.75, 0.75) (0.99, 0.01)

5.6 Study 1: Stigmergic Optimisation and Diversity Strategy

Finding QoS optimal service compositions in mobile-based, service-sharing communities is challenging because of the inherent dynamism in services deployed on mobile devices, and in the underlying physical network used to enable these services. Existing service composition proposals for such environments require *a priori* knowledge either about the service composition structure, or about the QoS objectives' weights, which limits the composition flexibility and does not allow for compromises to be made between multiple QoS objectives.

This study is focused on answering research question **Q1**. This section investigates to what extent can the trade-off between computational efficiency and optimality be controlled, such that a broad range of composition options can be presented to users and whether SBOTI can outperform the existing baseline approaches. To control the diversity of the solutions in this set, SBOTI uses a diversity strategy that globally updates both dominated and non-dominated solutions using digital pheromones (Section 3.4.3).

5.6.1 Evaluated Algorithms

The following algorithms are evaluated in this study:

1. **SBOTI**: the implementation of the QoS optimisation mechanism using a single community of agents as described in Section 3.4.2 (e.g., Figure 3.6a). The agent colony uses a non-dominated reinforcement strategy approach. Only the solutions that belong to the observed Pareto-front are reinforced.

2. **SBOTI-NDS**: the implementation of the QoS optimisation mechanism using a single community of agents as described in Section 3.4.2 (e.g., Figure 3.6a). The agent colony uses both a non-dominated and a dominated reinforcement approach. The solutions that belong to the observed Pareto-front are reinforced using a positive scalar, whereas the other solutions are reinforced with a negative scalar.

3. **GoCoMo**: presented in Section 5.4.

4. **SimDijkstra**: presented in Section 5.4.

5. **Random**: presented in Section 5.4.

5.6.2 Evaluated Algorithms Settings

A number of parameters need to be set before running SBOTI. These are: the initial pheromone scalar $\tau_{initial}$ associated with each entry in each pheromone store (Definition 2), the parameter α which is used to control the influence of $\tau_{initial}$ (used in Equation 3.2 and Equation 3.6), parameter Q_1 which is the amount of pheromone added during the *Pheromone Deposit* procedure (Definition 7), parameter Q_2 which is the amount of pheromone subtracted during the *Pheromone Deposit* procedure (Section 3.4.3), and the pheromone evaporation coefficient ρ used for the *Pheromone Evaporation* procedure (Definition 6). A pheromone reset strategy is used to allow the previously explored solutions, but depleted by the *Pheromone Evaporation* procedure, to be re-explored. When a pheromone value is outside the interval $[\tau_{min}, \tau_{max}]$, it is reset to $\tau_{initial}$. This strategy is used because the QoS values of each service in each solution are

Table 5.4: Parameter values used to initialise SBOTI and SBOTI-NDS.

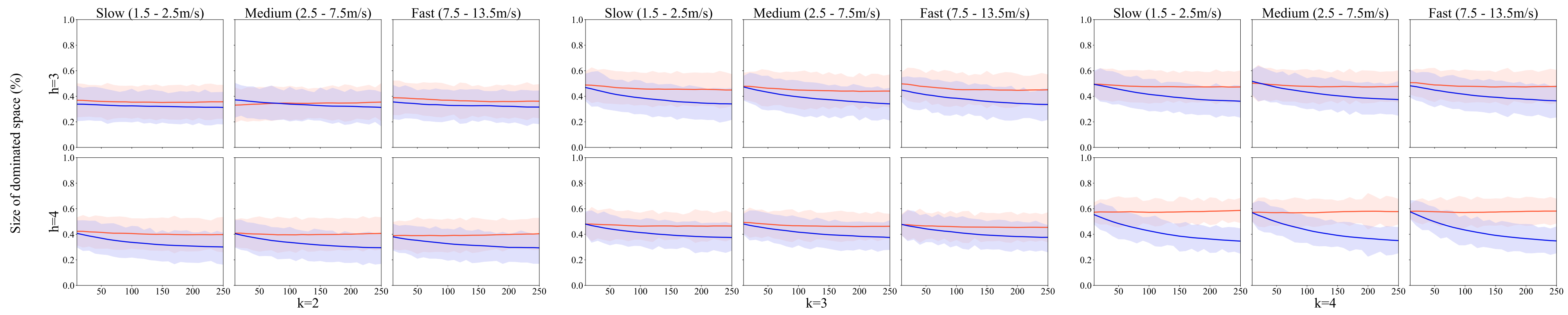
Parameter	**Value**
α	5.0
ρ	0.015
Q_1	55.0
Q_2	25.0
τ_{min}	10.0
τ_{max}	1000.0
$\tau_{initial}$	250.0

continuously changing. Table 5.4 shows the values used to initialise these parameters. The number of mobile agents is set to the initial number of services in the service dependency graph. This number is set based on other researchers' experiments [Palade et al., 2018b].

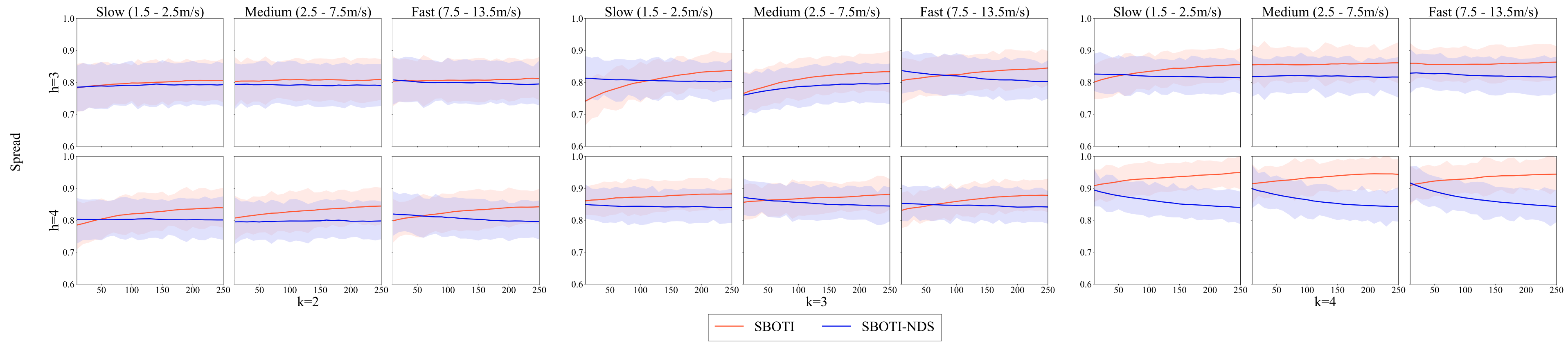
5.6.3 Results

Size of Dominated Space

Figure 5.3a shows the trade-off between the size of the dominated space metric and the number of iterations (i.e., search time), for SBOTI, SBOTI-NDS and SBOTI-PM, where all the mechanisms are initialised using the values from Table 5.4. The figure shows this trade-off as the number and lengths of service compositions configurations available in the environment is varied. In all the evaluated cases, the size of the dominated space of SBOTI-NDS decreases as the iterations number control variable k (the number of service composition configurations in the solutions space) is increased. The size of dominated space of SBOTI is higher than SBOTI-NDS in all the cases. As an example, when h is 4, the speed is Fast, and k is 2, after 10 iterations, the sizes of the dominated space for SBOTI, SBOT-NDS are 43.15%, respectively 41.49%. When k is 3, the results are 50.22%, respectively 49.41%. And when k is 4, the results are 59.4%, respectively 57.4%. SBOTI trades-off exploration time (in number of iterations) for optimality, whereas SBOTI-NDS achieves a lower value for this metric. After 250 iterations, and when k is 2, the sizes of the dominated space are 46.13%, respectively 35.16%. When k is 3, the results are 50.11%, respectively 42.58%. And when k is 4, the results are 61.16%, respectively 39.77%. By increasing the length of the service composition configurations, the sizes of dominated space obtained by both mechanisms increases. As the *speed* is increased, the size of dominated space obtained by SBOTI-NDS decreases significantly compared to the other two approaches. A statistical analysis is performed to measure whether the observed differences are statistically significant by following the procedure



(a) Size of Dominated Space.



(b) Spread.

Figure 5.3: The size of dominated space (higher is better) and the spread (lower is better), as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied.

Table 5.5: Statistical analysis for the size of the dominated space and the spread indicator results. Statistically significant differences (Yes/No) and exact p-value are highlighted with bold. The effect size is calculated in the same cell if the results are statistically significant.

		Slow (1.5 - 2.5 m/s)			**Medium (2.5 - 7.5 m/s)**			**Fast (7.5 - 13.5 m/s)**		
		k=2	**k=3**	**k=4**	**k=2**	**k=3**	**k=4**	**k=2**	**k=3**	**k=4**
	Pair	**Size of Dominated Space**								
h=3	SBOTI vs. SBOTI-NDS	No (0.005)	**Yes (0.000) 0.74**	**Yes (0.000) 0.7356**	No (0.005)	**Yes (0.027) 0.6112**	**Yes (0.000) 0.7064**	No (0.113)	**Yes (0.000) 0.7692**	**Yes (0.000) 0.7656**
h=4	SBOTI vs. SBOTI-NDS	**Yes (0.000) 0.7428**	**Yes (0.000) 0.7776**	**Yes (0.000) 0.9072**	**Yes (0.000) 0.7272**	**Yes (0.000) 0.7408**	**Yes (0.000) 0.9232**	**Yes (0.000) 0.7896**	**Yes (0.001) 0.6704**	**Yes (0.000) 0.8992**
	Pair	**Spread**								
h=3	SBOTI vs. SBOTI-NDS	No (0.917)	**Yes (0.042) 0.6002**	**Yes (0.001) 0.677**	No (0.856)	No (0.098)	**Yes (0.041) 0.6008**	No (0.685)	**Yes (0.006) 0.6436**	**Yes (0.041) 0.6006**
h=4	SBOTI vs. SBOTI-NDS	No (0.294)	**Yes (0.019) 0.62**	**Yes (0.000) 0.7922**	No (0.148)	**Yes (0.008) 0.6388**	**Yes (0.000) 0.8092**	**Yes (0.038) 0.6024**	**Yes (0.011) 0.6318**	**Yes (0.000) 0.718**

presented at the beginning of this section. If the observed differences are significant, the effect size is also measured (Section 5.2).

After applying the Kolmogorov-Smirnov and Shapiro-Wilk tests, this work concludes that the results do not follow a Gaussian distribution. A non-parametric analysis is performed to verify whether the differences observed between the means of the obtained results are statistically significant. The Wilcoxon-Mann-Whitney test by pairs of algorithms is performed. Table 5.5 shows the p-values obtained from this statistical test. The cases in which the differences are statistically significant are highlighted with bold. These tests confirm the observations made about the variable *speed* and the size of the solution space (i.e., number of service composition configurations). The table shows that by increasing the *speed* of devices in the environment and the number of service composition configurations h, SBOTI-NDS achieves inferior results compared to SBOTI. By increasing the length of the service composition configurations, the differences between the results obtained by each mechanism become significant.

Spread

Figure 5.3b shows the trade-off between the spread of identified solutions and the number of iterations (i.e., search time), for SBOTI and SBOTI-NDS. The figure shows this trade-off as the number and the lengths of service compositions configurations available in the environment is varied. A first observation is that the variables *speed* and h significantly influences the spread metric. For example, as variable *speed* is increased the spread values increases from 0.8 when h is 3 to 0.9 when h is 4. Another observation is that SBOTI-NDS achieves has a better spread than SBOTI in all the cases. A statistical analysis is performed to measure whether the observed differences in the obtained spread values are statistically significant by following the procedure presented at the beginning of this section. If the observed differences are significant, the effect size is also measured.

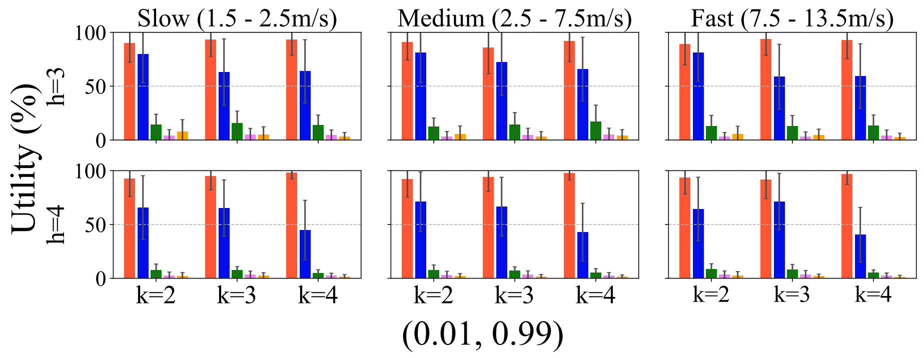
The Kolmogorov-Smirnov and Shapiro-Wilk tests indicate that the results do not follow a Gaussian distribution. A non-parametric analysis is performed to verify whether the differences observed between the means of the obtained results are statistically significant. The Wilcoxon-Mann-Whitney test by pairs of algorithms is performed. Table 5.5 shows the p-values obtained from this statistical test. The cases in which the differences are statistically significant are highlighted. These tests confirm the observations about the spread values obtained by SBOTI-NDS as the *speed* and the size of the solution space is increased. The results show that SBOTI-NDS trades-off the spread of solutions for optimality of solutions.

Utility of Solutions

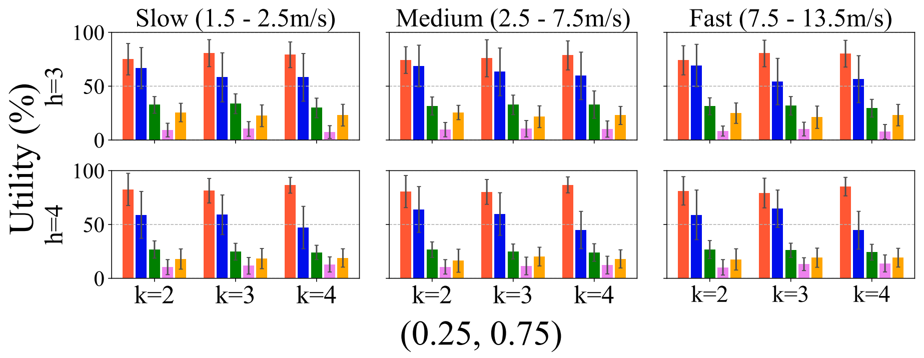
Figure 5.4 shows the results of the utility evaluation as the number of service composition configurations (i.e., variable k), the length of the service composition configurations (i.e., variable h) and the *speed* of devices in the environment is increased. The utility is calculated using the configurations produced by SBOTI and SBOTI-NDS in their 250-th iteration, and the configurations produced by SimDijkstra, GoCoMo and Random. This metric was defined in Section 5.1.3. Two objectives are considered in this evaluation: minimising response time and maximising the throughput of the composition. The weights of these objectives as well as the values of variables k, h and *speed* are varied using the values in Table 5.3. Two trends can be observed for SBOTI and SBOTI-NDS: (1) the utility of these variants confirm their optimality results from Figure 5.3a, and (2) the utility of these variants is higher than the utility of the SimDijkstra, GoCoMo and Random algorithms for a particular portion of the Pareto-front. The utility of SBOTI is higher than SBOTI-NDS in all the cases. This is because these mechanisms focus on exploitation of the information learned from the environment, whereas SBOTI-NDS uses the diversify strategy to improve the exploration of the solution space. Compared to the evaluated baselines, the results show that the evaluated variants achieve a higher utility when the weights (RT, TH) are $\{(0.01, 0.99), (0.25, 0.75), (0.50, 0.50)\}$.

Communication Overhead

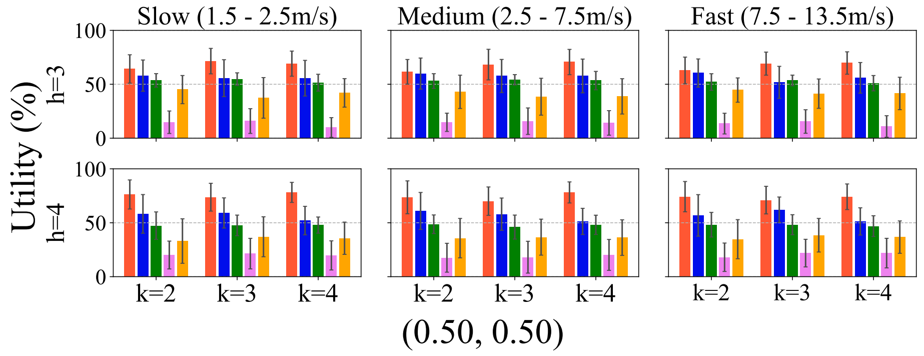
Figure 5.5 shows the communication overhead (number of exchanged messages) after 250 iterations as the size of the solution space varies over the number of ser-



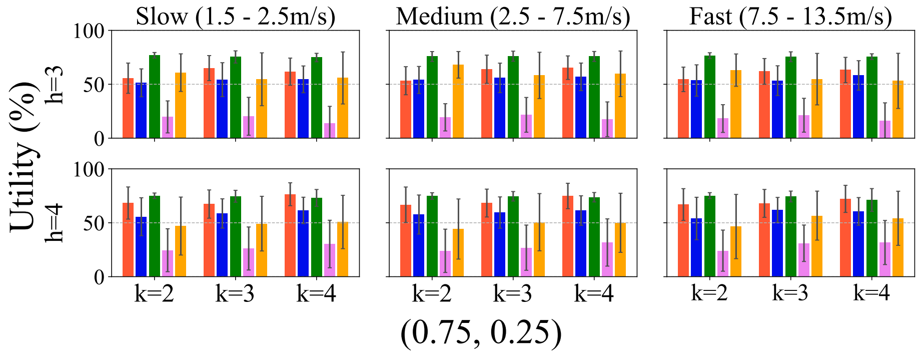
(a) Utility when weights (RT, TH) = (0.01, 0.99).



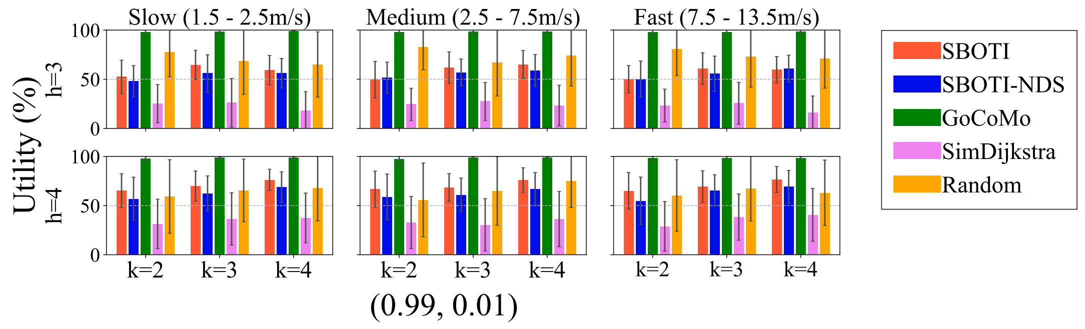
(b) Utility when weights (RT, TH) = (0.25, 0.75).



(c) Utility when weights (RT, TH) = (0.50, 0.50).



(d) Utility when weights (RT, TH) = (0.75, 0.25).



(e) Utility when weights (RT, TH) = (0.99, 0.01).

Figure 5.4: The utility (higher is better) of the evaluated algorithms as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The results are averaged over 50 executions.

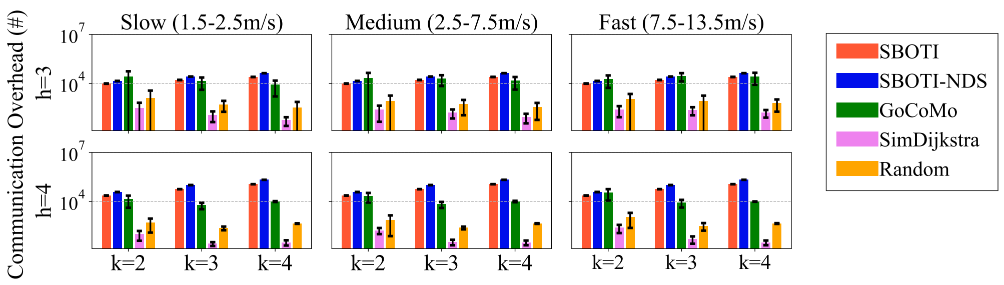


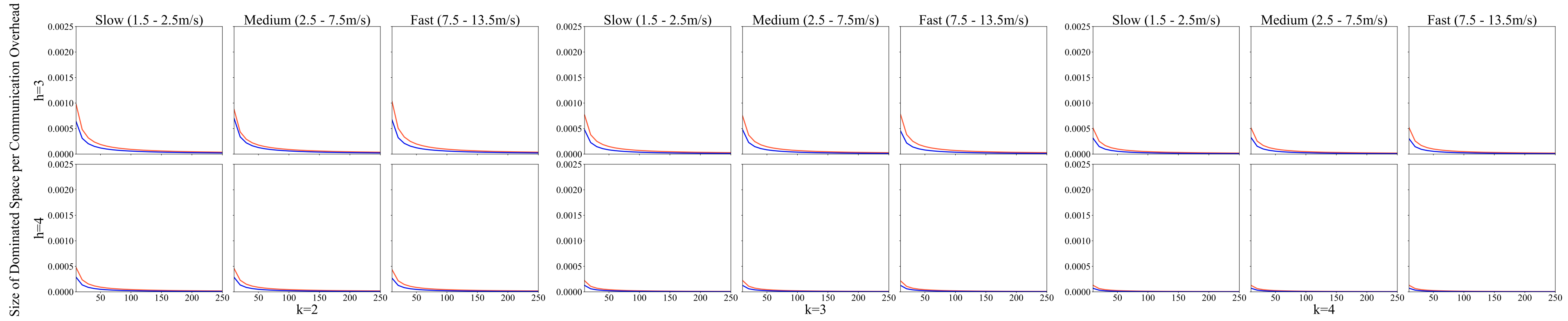
Figure 5.5: The overhead when various number of paths are available (lower is better). The results are averaged over 50 executions.

vice composition configurations, the length of the configurations and the speed of devices. All the control variables influence the overhead. The *speed* introduces communication failures between devices, which require retransmissions. This increases the communication overhead, which can be observed in every presented scenario. The number and the length of the service composition configurations also affect the overhead as follows. 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 communication overhead introduced by these two algorithms is insignificant. The results show that the evaluated agent-based approaches trade-off communication efficiency for optimality and diversity of solutions. SBOTI and SBOTI-NDS introduce a higher overhead than the GoCoMo, SimDijkstra and Random approaches because of the large number of agents that exchanged messages to perform the optimisation process.

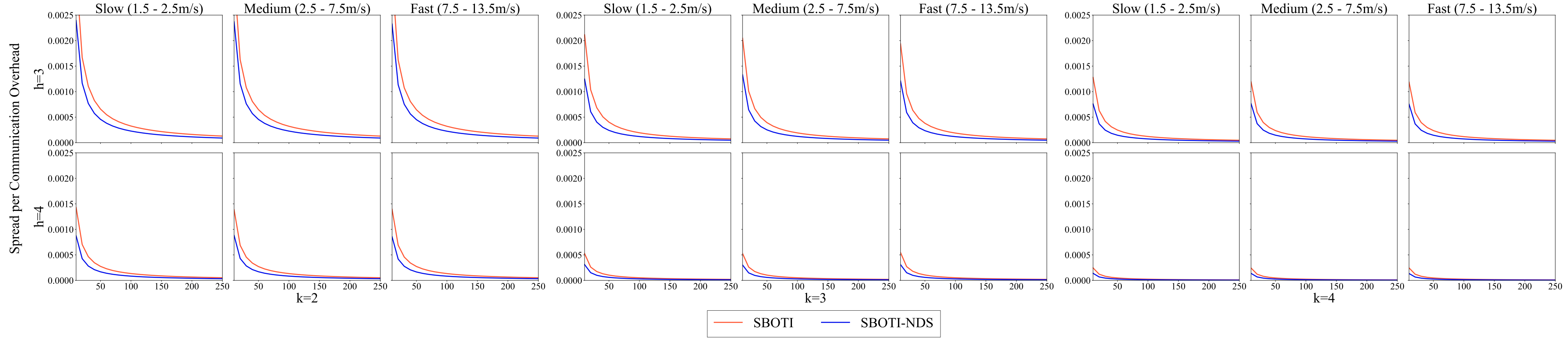
5.6.4 Discussion

The first study evaluated SBOTI and SBOTI-NDS, a proposed variant that uses a solution diversity strategy to improve the diversity of identified solutions (Section 3.4.3). The proposed diversity strategy uses a both positive and negative reinforcement approach to improve the exploration of different parts of the solutions space. The results showed that the proposed diversity strategy, based on positive reinforcement of optimal solutions and negative reinforcement of non-optimal solutions, can improve the diversity of identified solutions. The results of this study addresses research question **Q1** in Section 1.3.4.

Figure 5.6a shows the size of dominate space per communication overhead ratio as number of iterations increases. Figure 5.6b shows the spread per communication overhead ratio as the number of iterations increases. These results



(a) Size of Dominated Space per Communication Overhead.



(b) Spread per Communication Overhead.

Figure 5.6: Figure 5.6a shows the size of dominated space per communication overhead and Figure 5.6b the spread per communication overhead, as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The horizontal axis represents the number of iterations.

are shown for various length and number of service composition configurations. In all the cases SBOTI-NDS has a lower value than SBOTI at the beginning, but as the number of iterations increases the gap between the two is reduced.

5.7 Study 2: Adaptation Support

New service composition configurations may emerge (with potentially better QoS) as a result of service providers' mobility. Most 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 and does not allow for compromises to be made between multiple QoS objectives. The existing stigmergic-based service composition proposals do not allow for efficient exploration of the solution space. The digital pheromone mechanism used by such approaches may affect the exploration of new service composition configuration that may emerge as a result of service providers mobility.

This study is focused on answering research question **Q2**. This section investigates to what extent can service providers' mobility, potentially resulting in new service options appearing in the environment, be leveraged through an adaptive approach to exploring new service compositions. To allow for exploration of new service composition configurations that may emerge as a result of providers' mobility, SBOTI uses an adaptation procedure that limits the amount of pheromone on previously identified solutions. This procedure is implemented in SBOTI. A new variant is defined as SBOTI-PM, which includes this procedure. This variant is compared with SBOTI, SBOTI-NDS and the baseline algorithms.

5.7.1 Evaluated Algorithms

The following algorithms are evaluated in this study:

1. **SBOTI**: the implementation of the QoS optimisation mechanism using a single community of agents as described in Section 3.4.2 (e.g., Figure 3.6a). The agent colony uses a non-dominated reinforcement strategy approach. Only the solutions that belong to the observed Pareto-front are reinforced.

2. **SBOTI-PM**: the implementation of the QoS optimisation mechanism using a single community of agents as described in Section 3.4.2 (e.g., Fig-

Table 5.6: Two sets of parameter values used to initialise the evaluated algorithms as follows: *Set 1* initialised SBOTI, SBOTI-NDS and SBOTI-PM, and *Set 2* initialises SBOTI and SBOTI-PM.

	Set 1	**Set 2**
Parameter	**Value**	**Value**
α	5.0	0.9
ρ	0.015	0.01
Q_1	55.0	10
Q_2	25.0	-
τ_{min}	10.0	0
τ_{max}	1000.0	20
$\tau_{initial}$	250.0	10
δ	0.5	0.5

ure 3.6a). This variant also include the adaptation mechanism presented in Section 3.4.4.

3. **SBOTI-NDS**: presented in Section 5.6.1.

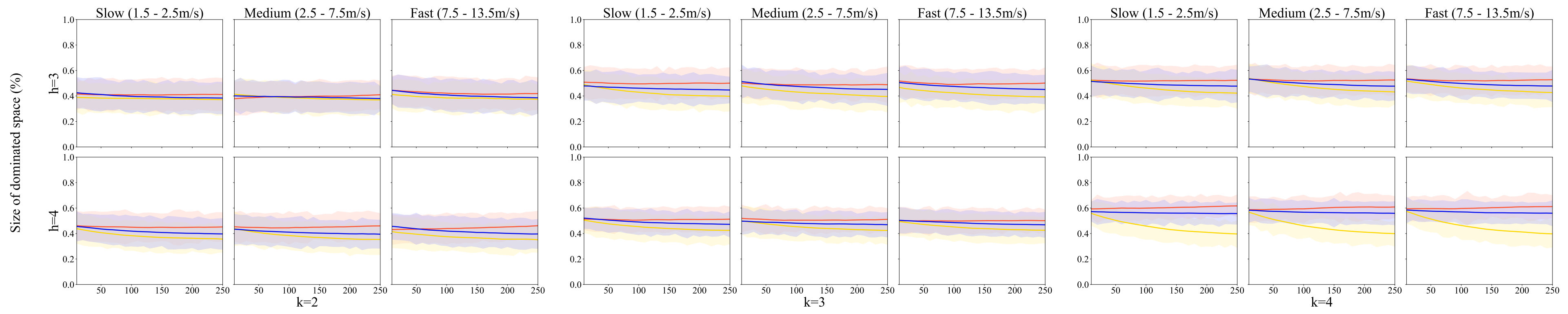
4. **GoCoMo**: presented in Section 5.4.

5. **SimDijkstra**: presented in Section 5.4.

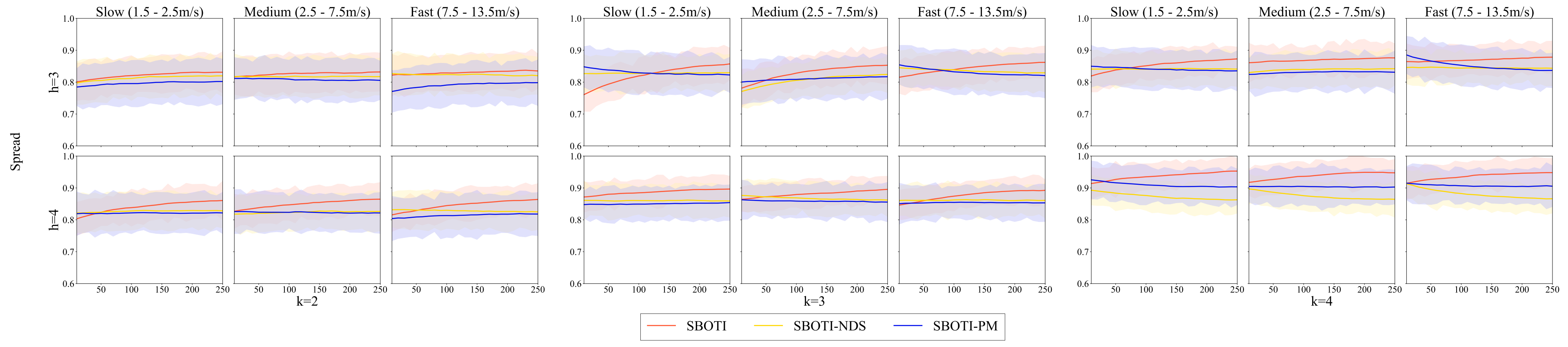
6. **Random**: presented in Section 5.4.

5.7.2 Evaluated Algorithms Settings

In addition to the parameters defined in Section 5.6.2, SBOTI-PM uses a pheromone smoothing coefficient δ (Section 3.4.4). Table 5.6 highlights two sets of values used in this evaluation. *Set 1* re-uses the values from Table 5.4. These values are used to evaluate SBOTI, SBOTI-NDS and SBOTI-PM. *Set 2* reduces the value associated with $\tau_{initial}$ and reduces the value of τ_{min} and τ_{max}, which are used in the *Reset Strategy* (Section 3.4.3). This set is used for evaluation of SBOTI and SBOTI-PM. The purpose of the second set is to check whether SBOTI-PM can be adjusted to identify more optimal solutions than SBOTI. By reducing the *min* and *max* values associated with the reset interval, the frequency of which knowledge about previous solutions is discarded is increased. This highlights the performance improvements of the pheromone smoothing technique evaluated in this study. Similar to study in Section 5.6, the number of mobile agents is set to the initial number of services in the service dependency graph.



(a) Size of Dominated Space.



(b) Spread.

Figure 5.7: The size of dominated space (higher is better) and the spread (lower is better), as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied.

5.7.3 Results

Size of Dominated Space

Figure 5.7a shows the trade-off between the size of the dominated space metric and the number of iterations (i.e., search time), for SBOTI, SBOTI-NDS and SBOTI-PM, where all the mechanisms are initialised using the *Set 1* values from Table 5.6. The figure shows this trade-off as the number and lengths of service compositions configurations available in the environment is varied. In all the evaluated cases, the size of the dominated space of SBOTI-NDS decreases as the iterations number control variable k (the number of service composition configurations in the solutions space) is increased. The size of dominated space of SBOTI-PM is higher than SBOTI-NDS in all the cases, but lower than SBOTI. As an example, when h is 4, the speed is Fast, and k is 2, after 10 iterations, the sizes of the dominated space for SBOTI, SBOT-NDS and SBOTI-PM are 43.15%, 41.49% and 45.66%, respectively. When k is 3, the results are 50.22%, 49.41%, and 50.38%, respectively. And when k is 4, the results are 59.4%, 57.4%, and 58.46%, respectively. SBOTI and SBOTI-PM trade-off exploration time (in number of iterations) for optimality, whereas SBOTI-NDS achieves a lower value for this metric. After 250 iterations, and when k is 2, the sizes of the dominated space are 46.13%, 35.16% and 39.66%, respectively. When k is 3, the results are 50.11%, 42.58%, and 46.84%, respectively. And when k is 4, the results are 61.16%, 39.77%, and 55.96%, respectively. Another observation with regards to these dimensions is that SBOTI-PM achieves super results compared to SBOTI-NDS, but inferior results compared to SBOTI. However, by increasing the length of the service composition configurations, the sizes of dominated space obtained by all three evaluated mechanisms increases. Another observation is that as the *speed* is increased, the sizes of dominated space obtained by SBOTI-NDS decreases significantly compared to the other two approaches. A statistical analysis is performed to measure whether the observed differences are statistically significant by following the procedure presented at the beginning of this section. If the observed differences are significant, the effect size is also measured (Section 5.2).

After applying the Kolmogorov-Smirnov and Shapiro-Wilk tests, this work concludes that the results do not follow a Gaussian distribution. A non-parametric analysis is performed to verify whether the differences observed between the means of the obtained results are statistically significant. The Wilcoxon-Mann-Whitney test by pairs of algorithms is performed. Table 5.7 shows the p-values

Table 5.7: Statistical analysis for the size of the dominated space and the spread indicator results. Statistically significant differences (Yes/No) and exact p-value are highlighted with bold. The effect size is calculated in the same cell if the results are statistically significant.

		Slow (1.5 - 2.5 m/s)			**Medium (2.5 - 7.5 m/s)**			**Fast (7.5 - 13.5 m/s)**		
		k=2	**k=3**	**k=4**	**k=2**	**k=3**	**k=4**	**k=2**	**k=3**	**k=4**
	Pair	**Size of Dominated Space**								
h=3	SBOTI vs. SBOTI-NDS	No (0.072)	**Yes (0.000) 0.7212**	**Yes (0.000) 0.7076**	**Yes (0.047) 0.5968**	**Yes (0.03) 0.6092**	**Yes (0.000) 0.7152**	No (0.135)	**Yes (0.000) 0.7716**	**Yes (0.000) 0.7732**
	SBOTI vs. SBOTI-PM	No (0.205)	No (0.05)	**Yes (0.005) 0.6476**	**Yes (0.002) 0.6644**	No (0.221)	No (0.073)	No (0.33)	**Yes (0.006) 0.6444**	**Yes (0.004) 0.6524**
	SBOTI-PM vs. SBOTI-NDS	No (0.303)	**Yes (0.002) 0.6644**	No (0.128)	No (0.806)	No (0.124)	**Yes (0.003) 0.6552**	No (0.266)	**Yes (0.006) 0.6448**	**Yes (0.026) 0.6124**
h=4	SBOTI vs. SBOTI-NDS	**Yes (0.000) 0.7576**	**Yes (0.000) 0.7712**	**Yes (0.000) 0.8972**	**Yes (0.000) 0.7152**	**Yes (0.000) 0.7544**	**Yes (0.000) 0.9316**	**Yes (0.000) 0.7856**	**Yes (0.000) 0.6876**	**Yes (0.000) 0.8852**
	SBOTI vs. SBOTI-PM	**Yes (0.011) 0.6328**	**Yes (0.008) 0.6388**	**Yes (0.000) 0.6844**	**Yes (0.000) 0.6896**	**Yes (0.000) 0.6812**	**Yes (0.000) 0.7208**	**Yes (0.000) 0.6876**	**Yes (0.002) 0.662**	**Yes (0.015) 0.626**
	SBOTI-PM vs. SBOTI-NDS	**Yes (0.023) 0.6152**	**Yes (0.001) 0.6712**	**Yes (0.000) 0.8176**	No (0.264)	**Yes (0.002) 0.6136**	**Yes (0.000) 0.8248**	**Yes (0.007) 0.64**	No (0.27)	**Yes (0.000) 0.8052**
	Pair	**Spread**								
h=3	SBOTI-NDS vs. SBOTI	No (0.918)	**Yes (0.032) 0.393**	**Yes (0.006) 0.355**	No (0.555)	No (0.071)	No (0.086)	No (0.771)	**Yes (0.023) 0.3848**	No (0.059)
	SBOTI-PM vs. SBOTI	No (0.491)	**Yes (0.000) 0.2968**	**Yes (0.000) 0.2688**	No (0.108)	No (0.054)	**Yes (0.008) 0.3608**	No (0.436)	No (0.185)	**Yes (0.028) 0.3896**
	SBOTI-PM vs. SBOTI-NDS	No (0.104)	No (0.152)	No (0.052)	**Yes (0.015) 0.375**	No (0.399)	No (0.098)	No (0.281)	No (0.826)	No (0.392)
h=4	SBOTI-NDS vs. SBOTI	No (0.313)	No (0.06)	**Yes (0.000) 0.2274**	No (0.283)	**Yes (0.006) 0.3544**	**Yes (0.000) 0.228**	No (0.424)	**Yes (0.011) 0.3682**	**Yes (0.000) 0.2988**
	SBOTI-PM vs. SBOTI	**Yes (0.033) 0.3938**	No (0.055)	**Yes (0.004) 0.3462**	No (0.063)	**Yes (0.013) 0.3712**	No (0.115)	No (0.0550	**Yes (0.037) 0.3968**	No (0.152)
	SBOTI-PM vs. SBOTI-NDS	No (0.055)	No (0.438)	No (0.972)	No (0.155)	No (0.537)	No (0.999)	No (0.306)	No (0.545)	No (0.944)

obtained from this statistical test. The cases in which the differences are statistically significant are highlighted with bold. These tests confirm the observations made about the variable *speed* and the size of the solution space (i.e., number of service composition configurations). The table shows that by increasing the *speed* of devices in the environment and the number of service composition configurations h, SBOTI-NDS achieves inferior results compared to SBOTI and SBOTI-PM. By increasing the length of the service composition configurations, the differences between the results obtained by each mechanism become significant. As the number of devices and their speed increases, SBOTI-PM offers better adaptation support as it has a better exploitation the service composition configurations that emerge from service providers mobility.

A second evaluation is performed within this study to verify whether the size of the dominated space obtained by SBOTI-PM can be improved by adjusting the parameters used to initialise the underlying optimization mechanism. For this evaluation, SBOTI and SBOTI-PM are initialised using *Set 2* values from Table 5.6. This evaluation is performed using only SBOTI and SBOTI-PM. The highest difference between these two mechanism can be observed when the *speed* variable is set to $Fast$. In the second evaluation, variable k is set to 2, variable *speed* is set to $Fast$, and variable h is varied from 3 to 6. Previous service composition proposals in mobile environments, showed that the success rate of service compositions is affected by the length of the service composition configurations [Karmouch and Nayak, 2012, Chen et al., 2018]. This evaluation is performed to check whether SBOTI-PM can outperform SBOTI, by using the pheromone smoothing mechanism to support adaptation to the mobile environment.

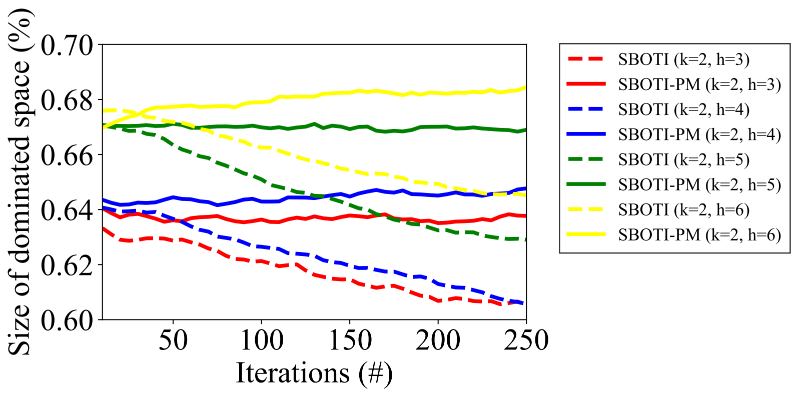


Figure 5.8: Trade-off between the Size of Dominated space metric (defined in Section 5.1.1) and the number of iterations for different number of service composition configurations.

Figure 5.8 shows how the size of the dominated space achieved by SBOTI and SBOTI-PM evolves as the number of iterations is increased. Both SBOTI and SBOTI-PM increase the size of their dominated space as the number of available paths increases. When h is set to 3, the size of the dominated space for SBOTI is 63.32% after 10 iterations and decreases to 60.54% after 250 iterations, whereas for SBOTI-PM maintains a level of 64%. When h is set to 4, SBOTI obtains 64.07% after 10 iterations and decreases to 60.56% after 250 iterations, whereas SBOTI-PM slowly increases from 64.35% to 64.76%. When h is set to 5, SBOTI achieves 67.05% after 10 iterations and 62.90% after 250 iterations, whereas SBOTI-PM achieves 66.96% after 10 iterations and 68.45% after 250 iterations. When h is set to 6, SBOTI achieves 67.59% and SBOTI-PM achieves 66.96%. SBOTI maintains the same decreasing slope and achieves 64.51% after 250 iterations, whereas SBOTI-PM achieves 68.45% after 250 iterations. As the length of the service composition configurations increases the smoothing mechanism used in SBOTI-PM can reduce the pheromone deposit on the optimal identified paths, and improve the exploration of the emerging service composition configurations. By reducing the pheromone deposited during each iteration, the exploration of new configurations as a result of mobility can be facilitated. Also, by reducing the pheromone associated with each *service agent* in the environment, in case of disconnections, SBOTI-PM can search for better solutions.

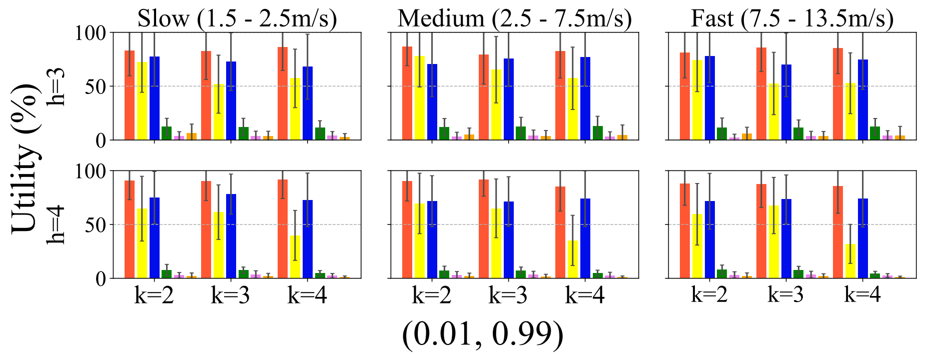
Spread

Figure 5.7b shows the trade-off between the spread of identified solutions and the number of iterations (i.e., search time), for SBOTI, SBOTI-NDS and SBOTI-PM. The figure shows this trade-off as the number and the lengths of service compositions configurations available in the environment is varied. A first observation is that the variables *speed* considerably influences the spread metric. For example, as variable *speed* is increased the spread values increases from 0.8 when h is 3 to 0.9 when h is 4, after 10 iterations. Another observation is that SBOTI-PM achieves similar spread value as SBOTI-NDS when the *speed* is slow and medium. A statistical analysis is performed to measure whether the observed differences in the obtained spread values are statistically significant by following the procedure presented at the beginning of this section. If the observed differences are significant, the effect size is also measured.

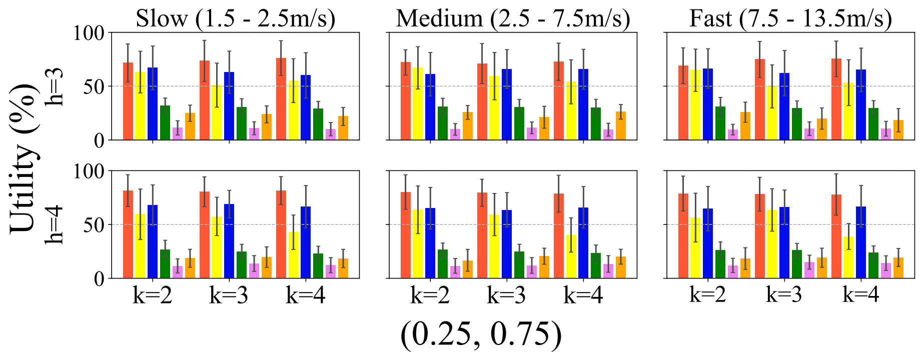
The Kolmogorov-Smirnov and Shapiro-Wilk tests indicate that the results do not follow a Gaussian distribution. A non-parametric analysis is performed to verify whether the differences observed between the means of the obtained results are statistically significant. The Wilcoxon-Mann-Whitney test by pairs of algorithms is performed. Table 5.7 shows the p-values obtained from this statistical test. The cases in which the differences are statistically significant are highlighted. These tests confirm the observations about the spread values obtained by SBOTI-PM as the *speed* and the size of the solution space is increased. The results show that SBOTI-PM trades-off the spread of solutions for optimality of solutions. For slow and medium speeds, SBOTI-PM achieves similar spread performances but more optimal compared to SBOTI-NDS.

Utility of Solutions

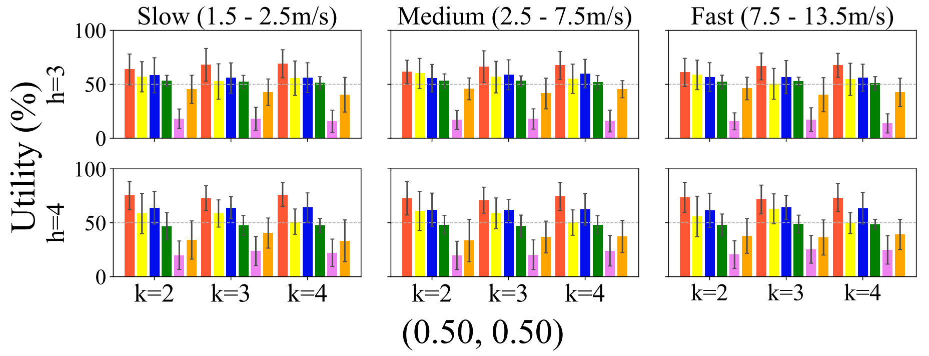
Figure 5.9 shows the results of the utility evaluation as the number of service composition configurations (i.e., variable k), the length of the service composition configurations (i.e., variable h) and the *speed* of devices in the environment is increased. The utility is calculated using the configurations produced by SBOTI, SBOTI-NDS and SBOTI-PM in their 250-th iteration, and the configurations produced by SimDijkstra, GoCoMo and Random. This metric was defined in Section 5.1.3. Two objectives are considered in this evaluation: minimising response time and maximising the throughput of the composition. The weights of these objectives as well as the values of variables k, h and *speed* are varied using



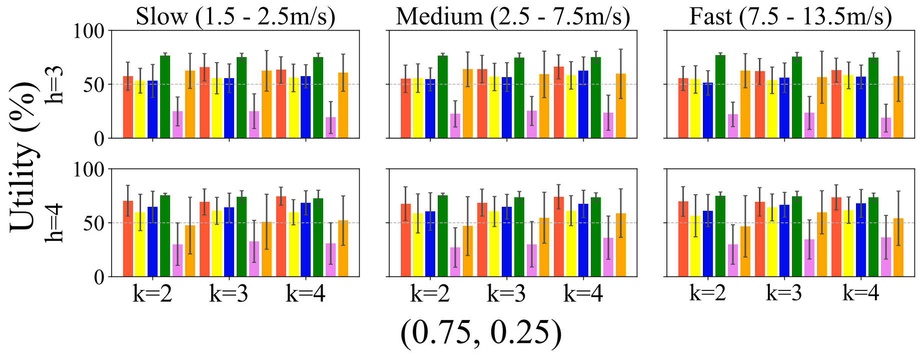
(a) Utility when weights (RT, TH) = (0.01, 0.99).



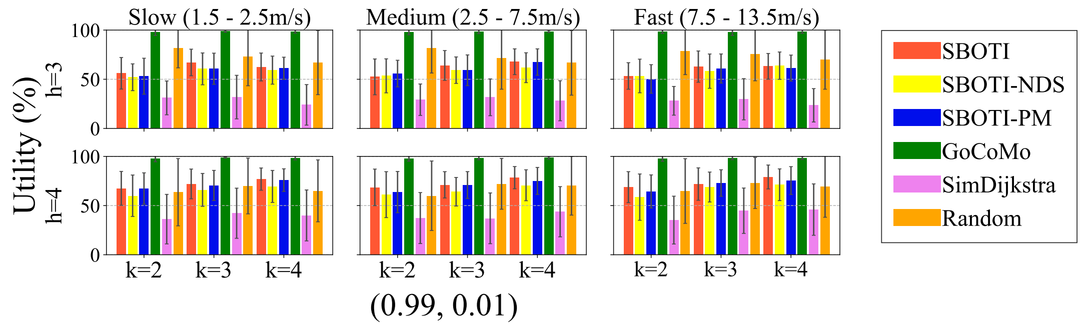
(b) Utility when weights (RT, TH) = (0.25, 0.75).



(c) Utility when weights (RT, TH) = (0.50, 0.50).



(d) Utility when weights (RT, TH) = (0.75, 0.25).



(e) Utility when weights (RT, TH) = (0.99, 0.01).

Figure 5.9: The utility (higher is better) of the evaluated algorithms as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The results are averaged over 50 executions.

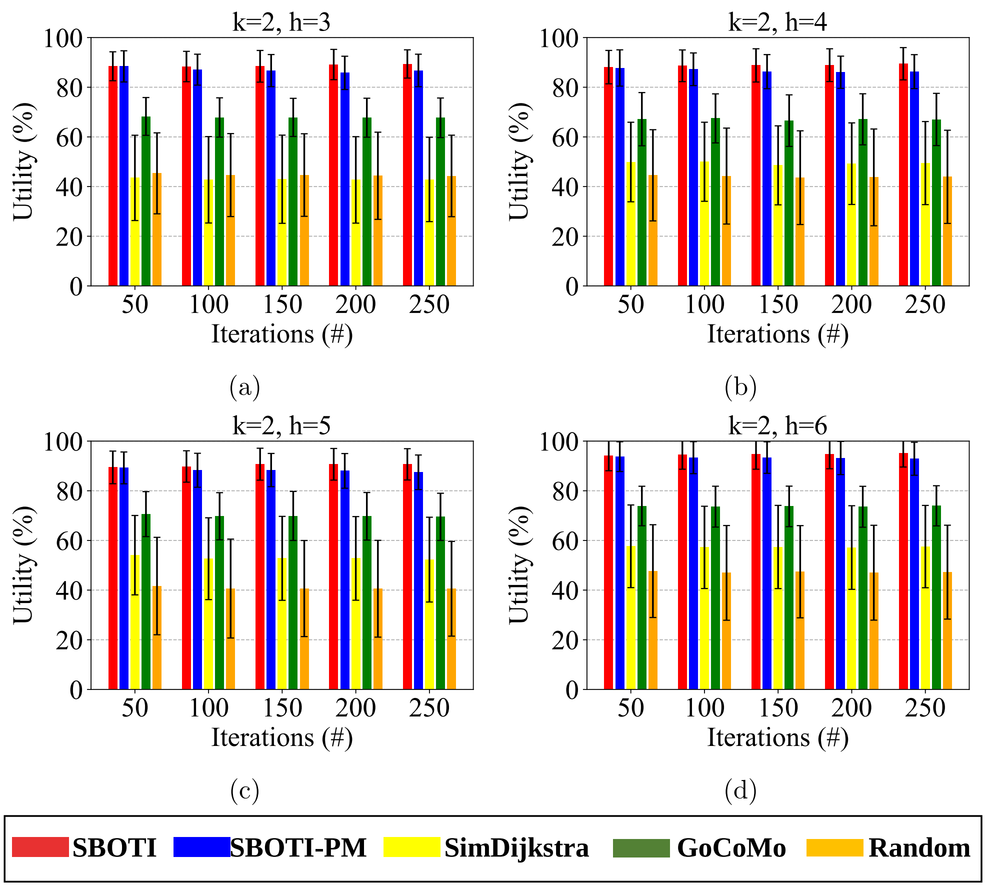


Figure 5.10: The utility of solutions produced by the evaluated algorithms (higher is better) after 250 iterations for different number of paths. The results are averaged over 50 executions.

the values in Table 5.3. Two trends can be observed for SBOTI, SBOTI-NDS and SBOTI-PM: (1) the utility of these variants confirm their optimality results from Figure 5.7a, and (2) the utility of these variants is higher than the utility of the SimDijkstra, GoCoMo and Random algorithms for a particular portion of the Pareto-front. The utility of SBOTI and SBOTI-PM is higher than SBOTI-NDS in all the cases. This is because these mechanisms focus on exploitation of the information learned from the environment, whereas SBOTI-NDS uses the diversify strategy to improve the exploration of the solution space. Compared to the evaluated baselines, the results show that the evaluated variants achieve a higher utility when the weights (RT, TH) are $\{(0.01, 0.99), (0.25, 0.75), (0.50, 0.50)\}$.

Figure 5.10 shows the results of the utility evaluation as the number of available paths (service composition configurations) is increased. In this evaluation, the results for when the length of the service composition configurations is increased from 3 to 6, while the value of variable k is set to 2 and the value of variable *speed* is set to $Fast$. The utility is calculated using the configurations produced by SBOTI and SBOTI-PM in the last 5 iterations, and the config-

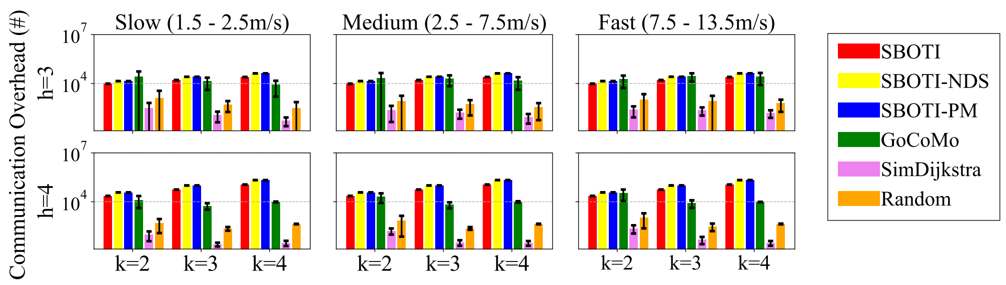


Figure 5.11: The overhead (lower is better) of evaluated algorithms as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The results are averaged over 50 executions.

urations produced by SimDijkstra, GoCoMo and Random. By increasing the number of iterations, two trends can be observed for SBOTI and SBOTI-PM: (1) the utility of both increases as the number of paths increases, and (2) the utility of SBOTI is higher than the utility of SBOTI-PM after 100 iterations. This difference decreases as the number of paths increases. When $h = 3$ (Figure 5.10a), SBOTI achieves an utility of 86.58% and SBOTI-PM achieves an utility of 80.38%. This decreases to 93.26% for SBOTI and 88.52% for SBOTI-PM, when $h = 6$ (Figure 5.10d). The SimDijkstra and Random algorithms have the lowest utility for almost all runs. GoCoMo performs comparably to SBOTI and SBOTI-PM. Figure 5.10b, Figure 5.10c and Figure 5.10d show that SBOTI has a higher utility than SBOTI-PM, which can be because of a bias towards a particular region of the Pareto-front.

Communication Overhead

Figure 5.11 shows the communication overhead (number of exchanged messages) after 250 iterations as the size of the solution space varies over the number of service composition configurations, the length of the configurations and the speed of devices. All the control variables influence the overhead. The *speed* introduces communication failures between devices, which require retransmissions. This increases the communication overhead, which can be observed in every presented scenario. The number and the length of the service composition configurations also affect the overhead as follows. 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 communication overhead introduced by these two algorithms is insignificant. The results show that the evaluated agent-based approaches trade-off communi-

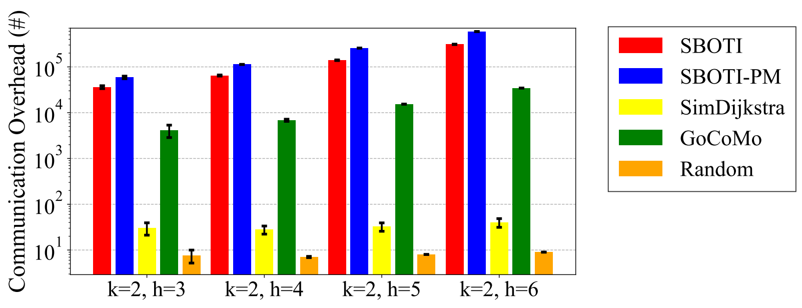


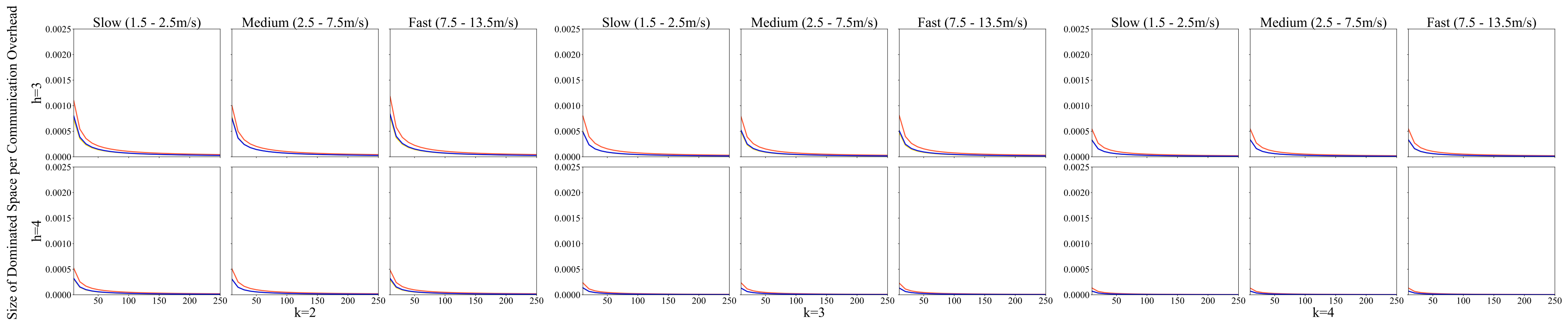
Figure 5.12: The overhead when various number of paths are available (lower is better). The results are averaged over 50 executions.

cation efficiency for optimality and diversity of solutions. SBOTI, SBOTI-NDS and SBOTI-PM introduce a higher overhead than the GoCoMo, SimDijkstra and Random approaches because of the large number of agents that exchanged messages to perform the optimisation process.

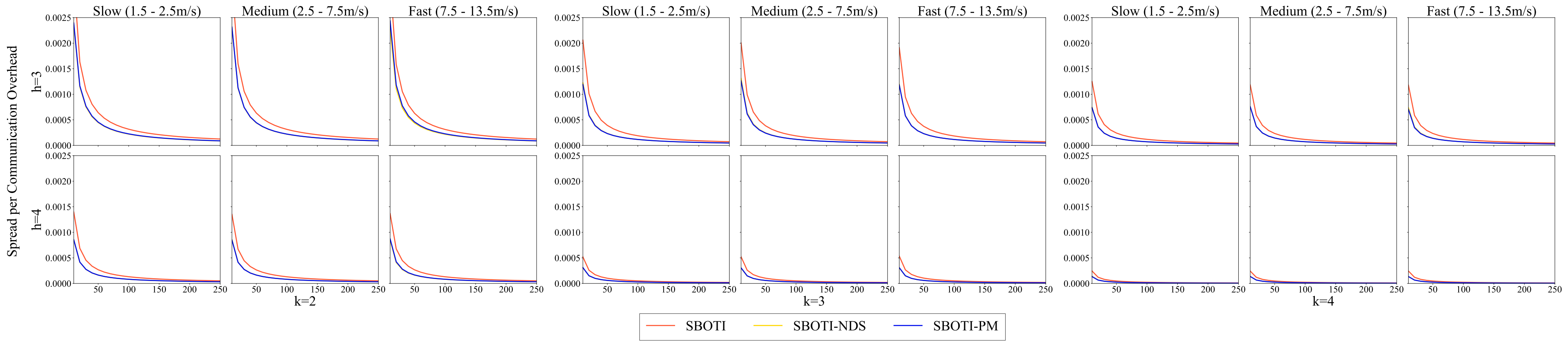
Figure 5.12 shows the overhead (number of exchanged messages) after 250 iterations as the length of service composition configurations in the service dependency graph varies. In GoCoMo, a global state mechanism periodically disseminates network state to all participating nodes, which explains the high overhead. This mechanism is not used in SimDijkstra and Random and the overhead introduced by these two algorithms is smaller. However, the utility of solutions of these algorithms is considerably lower than the utility produced by SBOTI and SBOTI-PM and GoCoMo. SBOTI-NDS and SBOTI-PM 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.

5.7.4 Discussion

The second study evaluated SBOTI, SBOTI-NDS (introduced in previous study) and SBOTI-PM, a proposed variant that uses a pheromone smoothing strategy to improve the optimality of identified solutions (Section 3.4.4). Two evaluations are performed in this study. In the first evaluation, SBOTI-PM is compared to SBOTI and SBOTI-NDS. The results show that SBOTI-PM can achieve superior solutions compared to SBOTI-NDS. SBOTI-PM improves the approximation of the set of QoS optimal service composition configurations that emerge as a results of providers mobility, while maintaining similar performances to SBOTI-NDS with regards to the spread metric. The decrease in utility of solutions when the



(a) Size of Dominated Space per Communication Overhead.



(b) Spread per Communication Overhead.

Figure 5.13: Figure 5.13a shows the size of dominated space per communication overhead and Figure 5.13b the spread per communication overhead, as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The horizontal axis represents the number of iterations.

adaptation procedure is used can be motivated by the choice of weights that are used to evaluate this metric. In the second evaluation, SBOTI-PM is compared to SBOTI. In this second evaluation, the configuration parameters are adjusted to verify whether SBOTI-PM can outperform SBOTI. The results show that the adaptation procedure can achieve a better approximation of the Pareto-optimal solutions than the no-adaptation variant. The results show that the proposed approach can pro-actively adapt to fluctuations in the QoS values, and can approximate the set of QoS-optimal of service composition configurations that may emerge in mobile environments. The results of this study addressed research question **Q2** in Section 1.3.4.

Figure 5.13a shows the size of dominate space per communication overhead ratio, and Figure 5.13b shows the spread per communication overhead ration as the length and number of service composition configurations increases. In all the cases SBOTI-PM and SBOTI-NDS have a lower value than SBOTI at the beginning of the optimisation process, but as the number of iterations increases the gap is reduced. SBOTI-PM and SBOTI-NDS achieve similar performances.

5.8 Study 3: Collaborative Agent Communities

Given the dynamic nature of the environment under consideration for this work, re-optimisation needs to be performed to closely track any changes in the available service composition configurations. SBOTI's optimisation process requires a number of hyper-parameters to be configured, such as the number of mobile agents required to explore the search space, the initial pheromone level τ associated with each service agent, the evaporation frequency ρ parameter that allows new paths to be explored. An important limitation of metaheuristic-based algorithms is that these hyper-parameters need to be tuned, because a universally optimal parameter values set does not exist [Talbi, 2009]. A tuning process allows for a larger flexibility and robustness, but it is difficult to perform since it may require problem specific information, or *a priori* knowledge about the environment [Hussein and Saadawi, 2003].

This study is focused on answering research question **Q3**. This section investigates to what extent will more search agents improve the optimality and diversity of identified service compositions. A collaborative approach to engage multiple communities of agents for provisioning QoS-optimal service compositions in mobile environments is evaluated in this section. New service compositions (with better QoS) can emerge from local decisions and interactions with agents from diverse communities. Several communities can independently tackle the optimisation problem in parallel, and maximise the explored/exploited search space. Each community searches and converges into different areas in the search space. When a dynamic change occurs, the multi-community approach has knowledge from a set of previously good solutions whereas a single community approach knows about only a single solution.

5.8.1 Evaluated Algorithms

The following algorithms are evaluated in this study:

1. **SBOTI**: the implementation of the QoS optimisation mechanism using a single community of agents as described in Section 3.4.2 (Fig. 3.6a).
2. **SBOTI-HOM**: an extension to SBOTI, which uses the homogeneous multi-community of agents as described in Section 3.4.5 and which optimises partitions of the service dependency graph (Fig. 3.6b).
3. **SBOTI-HET**: an extension to SBOTI, which uses the heterogeneous multi-community of agents as described in Section 3.4.5 and which optimises partitions of the service dependency graph (Fig. 3.6b).
4. **GoCoMo**: presented in Section 5.4.
5. **SimDijkstra**: presented in Section 5.4.
6. **Random**: presented in Section 5.4.

5.8.2 Evaluated Algorithms Settings

Single Agent Community Evaluation Settings

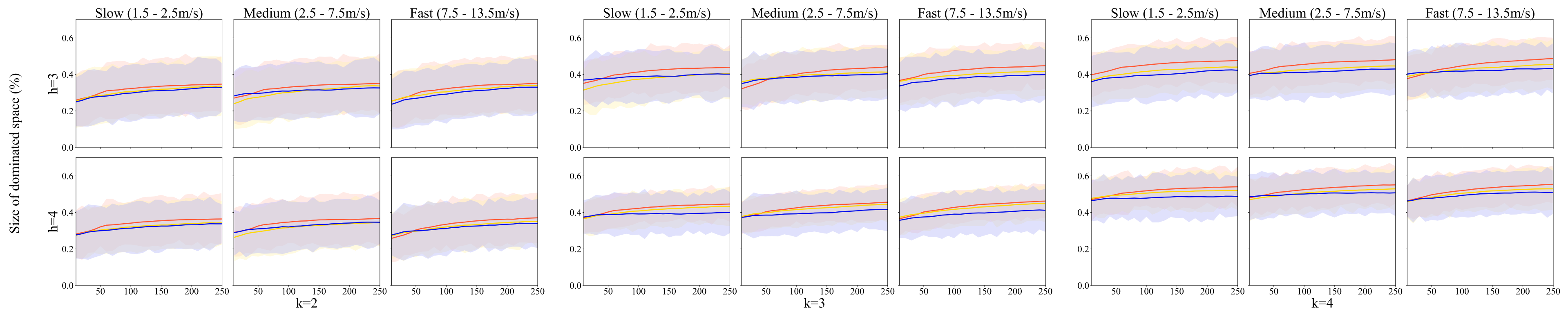
A number of parameters need to be set before running SBOTI. These are: the initial pheromone scalar $\tau_{initial}$ associated with each entry in each pheromone store (Definition 2), the parameter α which is used to control the influence of $\tau_{initial}$ (used in Equation 3.2 and Equation 3.6), parameter Q_1 which is the amount of pheromone added during the *Pheromone Deposit* procedure (Definition 7), and the pheromone evaporation coefficient ρ used for the *Pheromone Evaporation* procedure (Definition 6). A pheromone reset strategy is used to allow the previously explored solutions, but depleted by the *Pheromone Evaporation* procedure, to be re-explored. When a pheromone value is outside the interval $[\tau_{min}, \tau_{max}]$, it is reset to $\tau_{initial}$. This strategy is used because of the QoS values of each service in each solution are continuously changing. The settings for a single agent community case can be found in Table 5.8.

Multiple Agent Communities Evaluation Settings

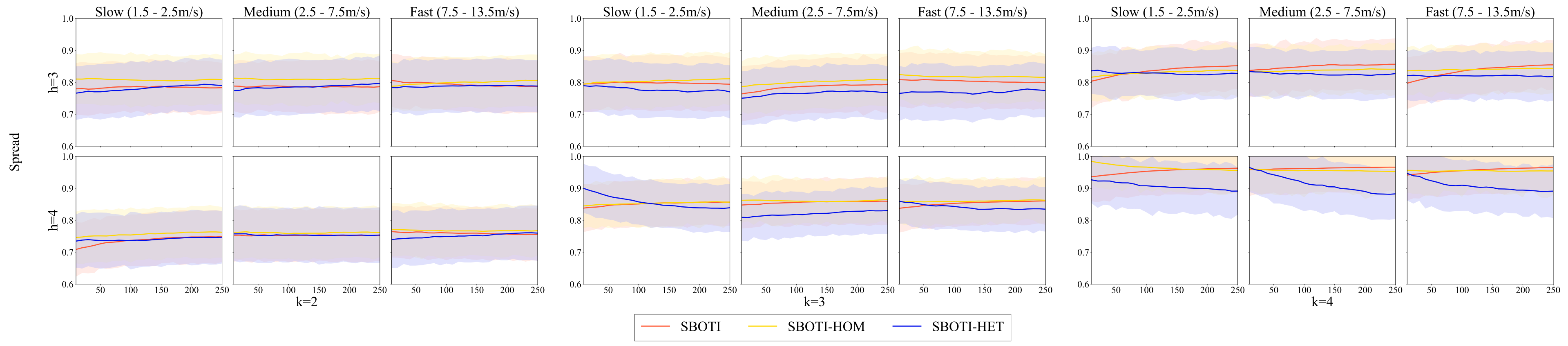
As introduced in Section 3.4.5, multiple agent communities may be used during the optimisation process instead of a single agent community. These communities may choose to search in the entire service dependency graph, like in the single agent community case, or in partitions of this graph (Figure 3.6b). In this evaluation, only the second case is considered, where one agent community has as a starting point for the optimisation process the first node in the service dependency graph, and one agent community is allocated to each node in the graph that has more than one outgoing node. As mentioned in Section 3.4.1, the available agent communities may be homogeneous or heterogeneous. The settings for each type of community can be found in Table 5.8. The agent communities are initialised as follows. In case of homogeneous agent communities, each community uses the values associated with the Homogeneous Communities column in this table. The period T of the *Evaporation Procedure* is set to the same value as in the case of single agent community.

In case of heterogeneous communities, each community selects a random value between a *Min* and a *Max* value, which are associated with the Heterogeneous Communities column in the same table. The period of the *Evaporation Procedure* is set to a random value in the interval $[T/2, T*3]$, where T is the period used in the single agent community case. These values were selected to introduce heterogeneity between the available communities. The choice of values of these parameters can affect the search 3.4.5. For instance, the evaporation period provides one means by which the trade-off between the exploration and exploitation can be controlled.

Only the mobile agents associated with non-dominated solutions are allowed to reinforce the traversed paths, and the number of mobile agents is set to the initial size of the service dependency graph.



(a) Size of Dominated Space.



(b) Spread.

Figure 5.14: The size of dominated space (higher is better) and the spread (lower is better), as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied.

Table 5.8: Parameter values used to initialise each type of agent community.

Parameter	**Single Agent Community**	**Homogeneous Agent Communities**	**Heterogeneous Agent Communities**	
	Value	**Value**	**Min**	**Max**
α	5.0	5.0	0.01	10.99
ρ	0.015	0.015	0.001	0.5
β	1.5	1.5	0.01	2.0
λ	0.95	0.95	0.01	0.99
Q_1	55.0	55.0	5.0	155.0
τ_{min}	10	10	10	15
τ_{max}	1000	1000	350	1500
$\tau_{initial}$	250	250	1.25 * τ_{min}	0.75 * τ_{max}

5.8.3 Results

Size of Dominated Space

Figure 5.14a shows the trade-off between the size of the dominated space metric and the number of iterations (i.e., search time), for SBOTI, SBOTI-HOM and SBOTI-HET. The figure shows this trade-off as the number and lengths of service compositions configurations available in the environment is varied. In all the evaluated cases, the size of the dominated space increases as the control variable k (the number of service composition configurations in the solutions space) is increased. As an example, when h is 4, the speed is Fast, and k is 2, after 10 iterations, the sizes of the dominated space for SBOTI, SBOT-HOM and SBOTI-HET are 24.82%, 26.89% and 27.14%, respectively. When k is 3, the results are 35.88%, 37.14%, and 35.19%, respectively. And when k is 4, the results are 45.66%, 45.57%, and 45.86%, respectively. After 250 iterations, the sizes of the dominated space increases for all the mechanisms. When k is 2, the sizes of the dominated space are 36.91%, 34.99% and 33.98%, respectively. When k is 3, the results are 46.09%, 44.81%, and 41.30%, respectively. And when k is 4, the results are 55.27%, 52.92%, and 50.89%, respectively. Another observation with regards to these dimensions is that SBOTI-HET achieves inferior results compared to SBOTI-HOM and SBOTI. However, by increasing the length of the service composition configurations, the sizes of dominated space obtained by all three evaluated mechanisms increases.

Another observation is that as the *speed* is increased, the difference between the obtained results by these mechanisms becomes smaller. A statistical analysis is performed to measure whether the observed differences are statistically significant by following the procedure presented at the beginning of this section. If the observed differences are significant, the effect size is also measured. After applying the Kolmogorov-Smirnov and Shapiro-Wilk tests, this work concludes

Table 5.9: Statistical analysis for the size of the dominated space and the spread indicator results. Statistically significant differences (Yes/No) and exact p-value are highlighted with bold. The effect size is calculated in the same cell if the results are statistically significant.

	Pair	**Slow (1.5-2.5 m/s)**			**Medium (2.5-7.5 m/s)**			**Fast (7.5 - 13.5 m/s)**		
		k=2	**k=3**	**k=4**	**k=2**	**k=3**	**k=4**	**k=2**	**k=3**	**k=4**
Size of Dominated Space										
h=3	SBOTI vs. SBOTI-HOM	No (0.248)	No (0.075)	**Yes (0.025) 0.5396**	No (0.251)	No (0.061)	No (0.199)	No (0.331)	No (0.184)	No (0.056)
	SBOTI vs. SBOTI-HET	**Yes (0.034) 0.6056**	**Yes (0.009) 0.6352**	**Yes (0.000) 0.6852**	No (0.28)	**Yes (0.024) 0.614**	**Yes (0.03) 0.6092**	No (0.294)	No (0.126)	**Yes (0.002) 0.6624**
	SBOTI-HOM vs. SBOTI-HET	No (0.162)	No (0.287)	No (0.052)	No (0.495)	No (0.37)	No (0.119)	No (0.447)	No (0.363)	No (0.108)
h=4	SBOTI vs. SBOTI-HOM	No (0.164)	**Yes (0.035) 0.6048**	**Yes (0.037) 06032**	No (0.273)	No (0.129)	No (0.379)	No (0.235)	No (0.298)	No (0.075)
	SBOTI vs. SBOTI-HET	**Yes (0.037) 0.6032**	**Yes (0.006) 0.6436**	**Yes (0.001) 0.674**	No (0.148)	**Yes (0.001) 0.6732**	**Yes (0.043) 0.5992**	No (0.087)	**Yes (0.005) 0.6464**	No (0.134)
	SBOTI-HOM vs. SBOTI-HET	No (0.247)	No (0.212)	No (0.111)	No (0.445)	**Yes (0.009) 0.6352**	No (0.084)	No (0.3)	**Yes (0.022) 0.616**	No (0.343)
Spread										
h=3	SBOTI vs. SBOTI-HOM	No (0.360)	No (0.592)	No (0.111)	No (0.674)	No (0.276)	No (0.444)	No (0.942)	No (0.928)	No (0.148)
	SBOTI vs. SBOTI-HET	No (0.534)	**Yes (0.041) 0.3996**	No (0.248)	No (0.625)	No (0.24)	No (0.465)	No (0.827)	No (0.171)	No (0.33)
	SBOTI-HOM vs. SBOTI-HET	No (0.637)	**Yes (0.01) 0.3664**	No (0.715)	No (0.49)	No (0.443)	No (0.561)	No (0.271)	**Yes (0.01) 0.6332**	No (0.702)
h=4	SBOTI vs. SBOTI-HOM	No (0.487)	No (0.515)	No (0.07)	No (0.667)	No (0.609)	No (0.162)	No (0.424)	No (0.358)	No (0.22)
	SBOTI vs. SBOTI-HET	No (0.54)	No (0.471)	**Yes (0.01) 0.3652**	No (0.652)	No (0.205)	**Yes (0.006) 0.3548**	No (0.424)	No (0.1)	**Yes (0.009) (0.3636)**
	SBOTI-HOMvs. SBOTI-HET	No (0.581)	No (0.548)	No (0.21)	No (0.556)	No (0.111)	No (0.094)	No (0.379)	No (0.138)	**Yes (0.041) (0.3664)**

that the results do not follow a Gaussian distribution. A non-parametric analysis is performed to verify whether the differences observed between the means of the obtained results are statistically significant. The Wilcoxon-Mann-Whitney test by pairs of algorithms is performed. Table 5.9 shows the p-values obtained from this statistical test. The cases in which the differences are statistically significant are highlighted. These tests confirm the observations made about the variable *speed*. The table shows that by increasing the *speed* of devices in the environment, the differences in sizes of dominated space become insignificant. For a slow speed SBOTI performs better than SBOTI-HOM and SBOTI-HET, but this difference becomes insignificant as the speed is increased. When the speed is increased the solution space is also changing faster, which continuously affects its size. When the solution space is small, using a single-agent community may converge to a more optimal set of solutions quicker than the multi-agent community approaches. Another highlight is that SBOTI-HOM outperforms SBOTI-HET when the length of the service composition configurations is increased (i.e., h). Although, the effect size of this performance is relatively small.

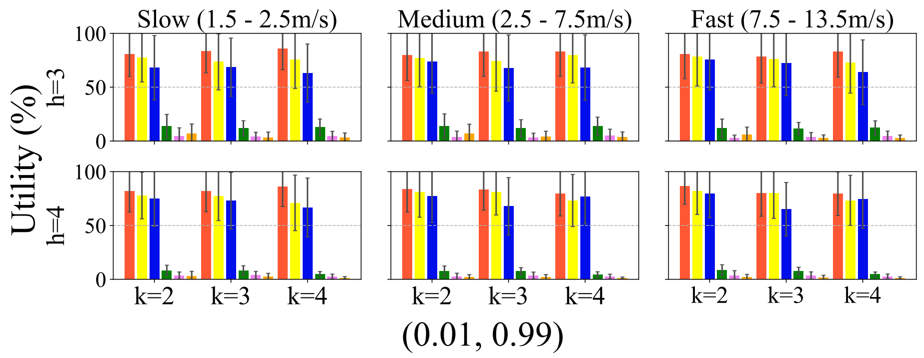
Spread

Figure 5.14b shows the trade-off between the spread of identified solutions and the number of iterations (i.e., search time), for SBOTI, SBOTI-HOM and SBOTI-HET. The figure shows this trade-off as the number and the lengths of service compositions configurations available in the environment is varied. A first observation is that variable h considerably influences the spread metric. As the length of the service composition configurations is increased, the interval of the spread values increases from 0.78 and 0.83 when h is 3, to 0.7 and 0.98 when h is 4, and k is varied between 2 and 4. Another observation is that the spread value obtained by SBOTI-HET improves as the *speed* of devices increases, whereas the SBOTI (single-agent community) and SBOTI-HOM (homogeneous multi-agent community) maintain the same spread level.

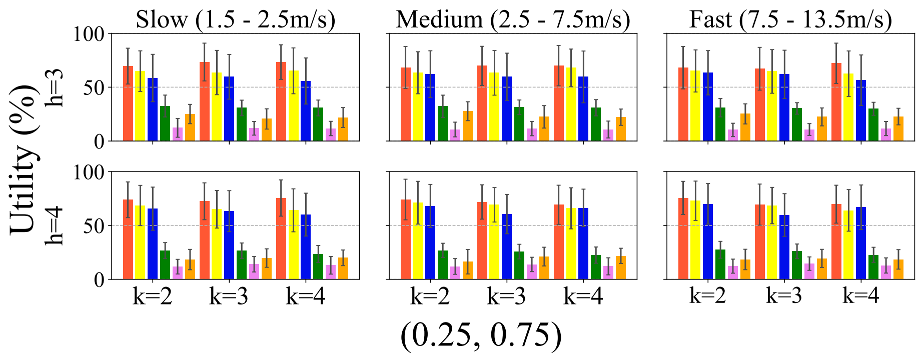
A statistical analysis is performed to measure whether the observed differences in the obtained spread values are statistically significant by following the procedure presented at the beginning of this section. If the observed differences are significant, the effect size is also measured. The Kolmogorov-Smirnov and Shapiro-Wilk tests indicate that the results do not follow a Gaussian distribution. A non-parametric analysis is performed to verify whether the differences observed between the means of the obtained results are statistically significant. The Wilcoxon-Mann-Whitney test by pairs of algorithms is performed. Table 5.9 shows the p-values obtained from this statistical test. The cases in which the differences are statistically significant are highlighted. These tests confirm the observations about the spread values obtained by SBOTI-HET as the *speed* and the size of the solution space is increased. The results show that SBOTI-HET trades-off optimality of solutions for a better distribution of solutions.

Utility of Solutions

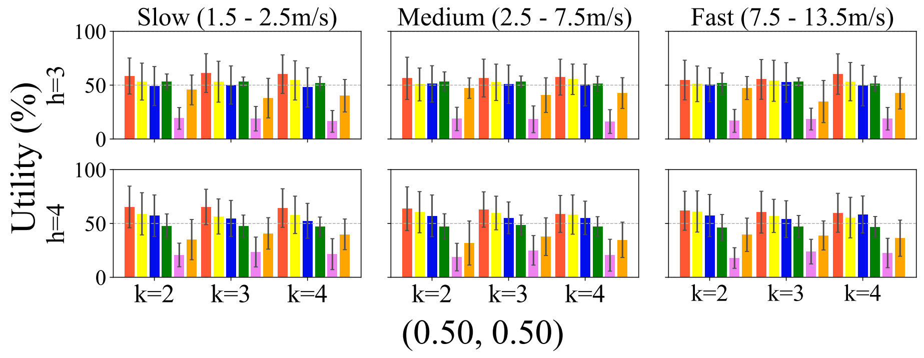
Figure 5.15 shows the results of the utility evaluation as the number of service composition configurations (i.e., variable k), the length of the service composition configurations (i.e., variable h) and the *speed* of devices in the environment is increased. The utility is calculated using the configurations produced by SBOTI, SBOTI-HOM and SBOTI-HET in their 250-th iteration, and the configurations produced by SimDijkstra, GoCoMo and Random. This metric was defined in Section 5.1.3. Two objectives are considered in this evaluation: minimising response time and maximising the throughput of the composition. The weights of these objectives as well as the values of variables k, h and *speed* are varied using the values in Table 5.3. By increasing the number of iterations, two trends can be observed for SBOTI, SBOTI-HOM and SBOTI-HET: (1) the utility of the variants increases as the number of paths increases, and (2) the utility of these variants is higher than the utility of the SimDijkstra, GoCoMo and Random algorithms. This is because of the efficient exploration approach used by the agent



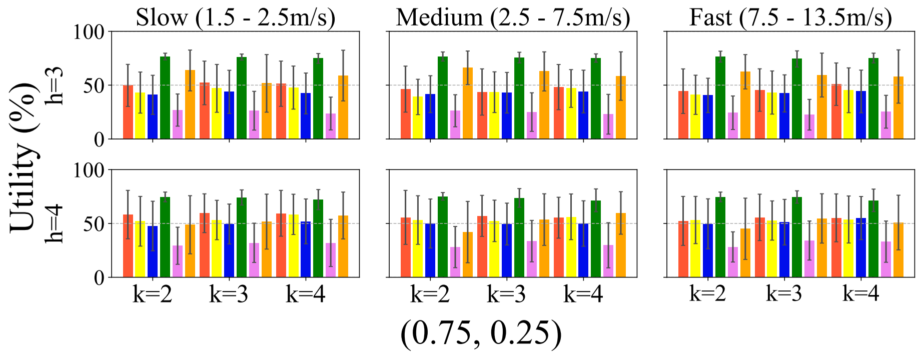
(a) Utility when weights (RT, TH) = (0.01, 0.99).



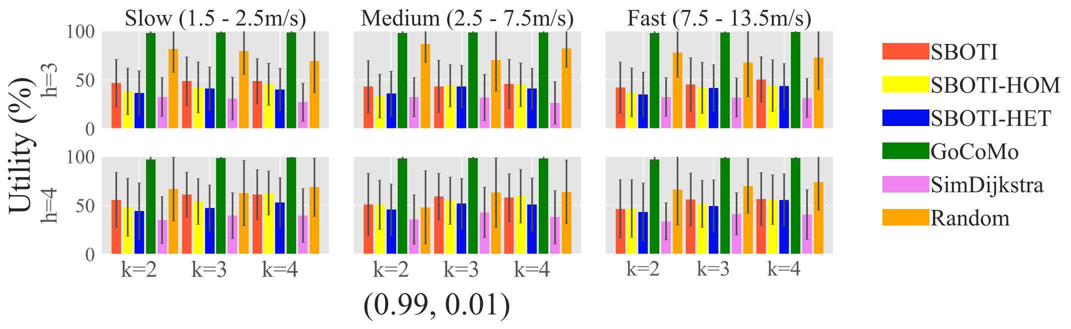
(b) Utility when weights (RT, TH) = (0.25, 0.75).



(c) Utility when weights (RT, TH) = (0.50, 0.50).



(d) Utility when weights (RT, TH) = (0.75, 0.25).



(e) Utility when weights (RT, TH) = (0.99, 0.01).

Figure 5.15: The utility (higher is better) of the evaluated algorithms as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The results are averaged over 50 executions.

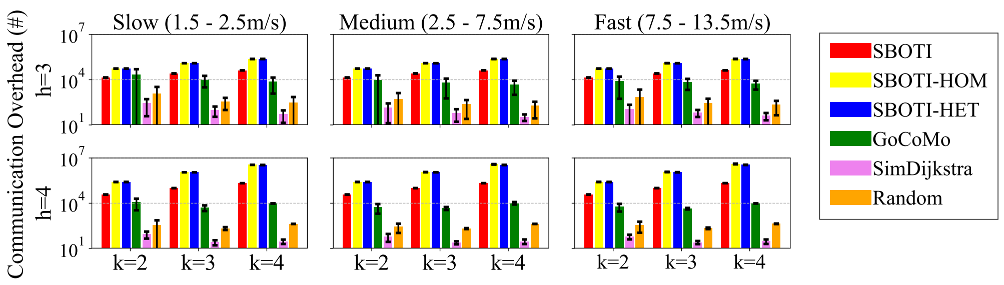


Figure 5.16: The overhead (lower is better) of the evaluated algorithms as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The results are averaged over 50 executions.

community-based variants, which is more efficient when compared to the other baselines, as they rely on continuous broadcasts of the environment state. The agent community-based approaches use a set of agents to continuously explore the solutions space, where each agent produces a service composition configuration. Also, the continuous sampling of the environment through each iteration, allows these approaches to avoid exploring previously explored solution space and explore new parts to identify better service compositions.

Communication Overhead

Figure 5.16 shows the communication overhead (number of exchanged messages) after 250 iterations as the size of the solution space varies over the number of service composition configurations, the length of the configurations and the speed of devices. All the control variables influence the overhead. The *speed* introduces communication failures between devices, which require retransmissions. This increases the communication overhead, which can be observed in every presented scenario. The number and the length of the service composition configurations also affect the overhead as follows. 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 communication overhead introduced by these two algorithms is insignificant. The results show that the evaluated agent-based approaches trade-off communication efficiency for optimality and diversity of solutions. SBOTI, SBOTI-HOM and SBOTI-HET introduce a higher overhead than the GoCoMo, SimDijkstra and Random approaches because of the large number of agents that exchanged messages to perform the optimisation process. Also, the results show that the multi-agent approaches introduce higher overhead than single-agent approaches.

This is because of the increased number of agent entities that exchange messages. The number of mobile agents was set to the number of nodes in the initial service dependency graph for both single agent and multi agent approaches.

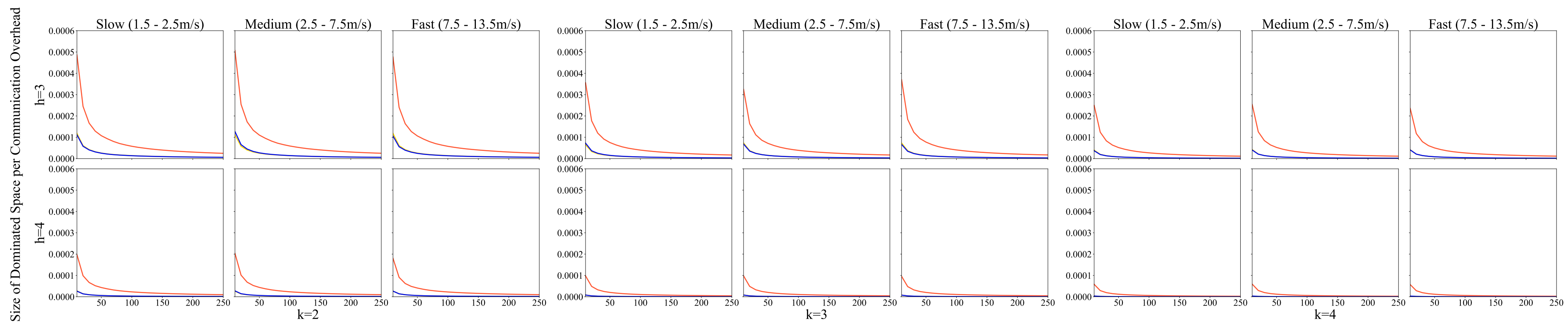
5.8.4 Discussion

The evaluation of these mechanisms is performed using an event-based simulator. The experimental results show the effectiveness of the multi-agent community approach in complex and highly dynamic, mobile environments. The results show that: (i) agent-based mechanisms that use stigmergic coordination may identify service composition configurations with better QoS in a mobile environment; (ii) using multiple heterogeneous agent communities can improve the diversity of identified solutions, which allows for a user to make better compromises between his objectives. The results of this study addressed research question **Q3** in Section 1.3.4. Tuning this parameter would likely reduce the overhead, though this needs to be verified.

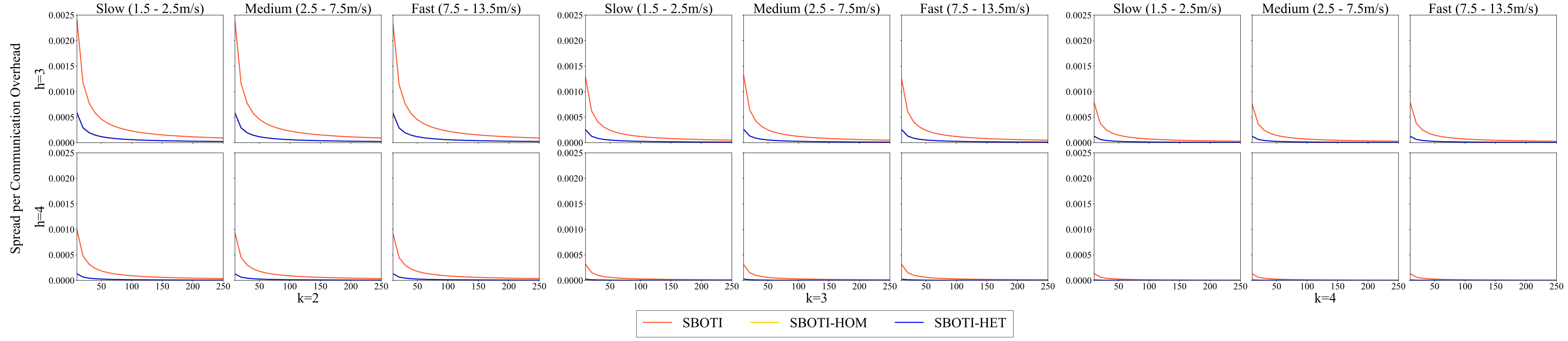
Figure 5.17a shows the size of dominate space per communication overhead ratio, and Figure 5.17b shows the spread per communication overhead ration as the length and number of service composition configurations increases. In all the cases SBOTI-HOM and SBOTI-HET have a lower value than SBOTI at the beginning, but as the number of iterations increases the gap between the two is reduced.

5.9 Threats to Validity

Construct threats can be introduced by the stochastic nature of the SBOTI's underlying swarm-based approach, which may introduce bias to the used metrics. To mitigate these threats, the experiments were repeated 50 times. Statistical significant tests were used to further validate the results. *Internal validity threats* may arise from the hyper-parameter settings used to initialise SBOTI. For instance, the mobile environment may remove the knowledge learned about the solutions in the environment if the pheromone evaporation frequency is too high and pheromone deposit quantity is very low. To mitigate these threats, a number of experiments with various interval settings were performed to identify a set of parameters which allow convergence on a set of approximately optimal solutions. *Threats to external validity* are linked to the selected benchmark setup are linked to the selected benchmark setup, the QoS data set used and the exper-



(a) Size of Dominated Space per Communication Overhead.



(b) Spread per Communication Overhead.

Figure 5.17: Figure 5.17a shows the size of dominated space per communication overhead and Figure 5.17b the spread per communication overhead, as the number of service composition configurations (k), the length of service composition configurations (h) and the speed of devices in the environment is varied. The horizontal axis represents the number of iterations.

imental setup environment. To mitigate these threats, SBOTI was evaluated in Simonstrator, which is a simulator for mobile environments. The WS-DREAM dataset, a popular data set used in many works, was used to initialise the QoS of each available service. The agent-based variants and the other baselines were evaluated on a cluster provided by TCHPC (see Section 5.3), where each node had similar resources and was running the same number of tasks.

5.10 Chapter Summary

This chapter presents three studies to assess the limitations of SBOTI, under various dynamic conditions, and to answer the research questions presented in Section 1.3.4. The evaluation is performed using an event-based simulator. The evaluation results are compared with the performance results of GoCoMo, SimDijkstra and a Random approach. In all the studies, the experimental results show the superior effectiveness of SBOTI in complex and highly dynamic, mobile environments, compared to existing baselines, but at a cost of increased communication overhead.

The first study evaluated SBOTI and SBOTI-NDS, a proposed variant that uses a solution diversity strategy to improve the diversity of identified solutions (Section 3.4.3). The results showed that the proposed diversity strategy, based on positive reinforcement of optimal solutions and negative reinforcement of non-optimal solutions, can improve the diversity of identified solutions. The results of this study addresses research question **Q1** in Section 1.3.4.

The second study evaluated SBOTI, SBOTI-NDS (introduced in previous study) and SBOTI-PM, a proposed variant (i.e.,) that uses a pheromone smoothing strategy to improve the optimality of identified solutions (Section 3.4.4). The results show that the proposed approach can pro-actively adapt to fluctuations in the QoS values, and can approximate the set of QoS-optimal of service composition configurations that may emerge in mobile environments. The results of this study addresses research question **Q2** in Section 1.3.4.

The third study focused on the limitation of always requiring an optimal hyper-parameter set to initialise SBOTI. This study explores whether by using an information exchange mechanism between multiple, potentially diverse, agent communities, the approximation of the real Pareto-front can be improved. Two variants SBOTI-HOM and SBOTI-HET are proposed. These mechanisms use the information provided through multi-agent communities to approximate the set of QoS-optimal service composition configurations. SBOTI-HOM exchanges information only with homogeneous agent communities, whereas SBOTI-HET exchanges information only with heterogeneous agent communities. The results show that: (i) agent-based mechanisms that use stigmergic coordination may identify service composition configurations with better QoS in a mobile environment; (ii) using multiple heterogeneous agent communities can improve the diversity of identified solutions, which allows for a user to make better compromises between his objectives. The results of this study addressed research question **Q3** in Section 1.3.4.

Each study has employed statistical tests to verify whether the observed differences in the performance indicators are statistically significant. The results have been presented in Table 5.5, Table 5.7, and Table 5.9, respectively. In case of single agent communities study (i.e., first and second studies), a general pattern can be observed. As the size of the search space increases, statistically significant differences can be observed in both performance indicators. In particular, in the first study, the use of the negative reinforcement has improved the size of the dominated space, which shows the optimality of the solutions. In the second study, the spread of solutions is improved by using the adaptation support mechanism. In case of multiple agent communities, the statistical tests has shown statistically significant differences in size of dominated spaces as the length of service composition configurations increases.

Chapter 6

Discussion and Conclusion

This thesis has investigated the limitations of existing QoS optimisation proposals for flexible service compositions in a mobile environment, and has focused on how to search for an optimal and a diverse set of service composition configurations in such an environment. This thesis has proposed SBOTI, a stigmergic-based optimisation mechanism for flexible, QoS-aware service composition in mobile environments. SBOTI can efficiently explore the environment and can identify a set of Pareto-optimal solutions, which allows users to make trade-offs between QoS objectives. SBOTI was evaluated under various dynamic conditions, and the results were compared with the performance of existing baseline algorithms.

This chapter is structured as follows: Section 6.1 summaries this thesis, Section 6.2 highlights the contributions to the body of knowledge, and Section 6.3 outlines future research questions and possible extensions to this work.

6.1 Thesis Summary

Chapter 1 Introduction described the motivation behind this thesis and highlighted the challenges that arise when composing services deployed on mobile devices from a functional and a non-functional perspective. This thesis was especially concerned with the openness and dynamism of the mobile environment. In addition to the transient networks, the lack of composition infrastructure, unreliable communication, service composition needs to consider the service providers autonomy and mobility, and the limited resources available on the devices that are used to provide these services. The myriad of services available in service-sharing communities created by users within a limited geographic area, and through sensors deployed in the environment, introduces scalability concerns. To address these dynamic conditions, the existing proposals trade-off optimality for computational efficiency. The chapter hypothesised that by using a decentralised, iterative, stigmergic-based mechanism that can control the trade-off between the computational efficiency of the execution and optimality of identified solutions, in a mobile environment, service composition configurations with higher utility than the ones produced by the existing proposals can be identified.

Chapter 2 State of the Art analysed how the state of the art optimisation mechanisms for flexible, QoS-aware service composition meet the challenges of mobile environments. The chapter explored the common types of performing service composition, the existing proposals for multi-objective QoS optimisation for service composition, and how the existing approaches deal with the computational efficiency concerns in mobile environments. The analysis highlighted the gap for a novel QoS optimisation mechanism that can be used to control the optimality and communication overhead to find service composition configurations in mobile environments.

Chapter 3 Design returned to the challenges of QoS-aware service composition in mobile environments outlined in Chapter 1 and described the design objectives, and design decisions made in this thesis. The chapter outlined the design of SBOTI, a stigmergic-based QoS optimisation mechanism that uses an agent-based approach to solve the QoS optimisation problem for flexible service composition in mobile environment. Thereafter, the chapter outlined how SBOTI addressed the design decisions in detail. Each service available in the environment is associated with a service agent. A set of mobile agents is used to search the environment for QoS optimal and diverse service composition configurations. By using stigmergic-coordination of these agents, a set of solutions can be identified using a pheromone mechanism. SBOTI uses a non-dominated sorting technique to extract the Pareto-optimal solutions. SBOTI uses a diversity strategy where solutions belonging to this set are reinforced with a positive scalar, whereas solutions the other solutions are reinforced with a negative offset. An adaptation support mechanism is used to further improve the optimality and diversity of identified solutions. Given the dynamic nature of the environment, re-optimisation is required to closely track any changes available in the available service composition. SBOTI requires a number of hyper-parameters to be configured to efficiently explore such a dynamic space. To address this limitation, SBOTI uses a collective agent communities approach to exchange information about the identified solutions, to improve the optimality and diversity of solutions.

Chapter 4 Implementation introduced the SURF middleware, which provides the necessary components for service provisioning in dynamic environments. In addition to registration, discovery and composition, this middleware offers the necessary components for SLA-negotiation and QoS monitoring of available service composition configurations. Then, the chapter highlighted the implementation details of SBOTI, including the interaction between the main components of this mechanism. This chapter also described how SBOTI was implemented in Simonstrator, an event-based simulator that is used to simulate mobile devices, moving at various speeds and according to a specific trajectory in the environment.

Chapter 5 Evaluation evaluated how well SBOTI achieves its objective of finding QoS optimal service composition solutions compared to baseline approaches, subject to three research questions. SBOTI and four variants are evaluated in a (simulated) mobile environment to expose these mechanisms to various dynamic conditions such as different speeds of composition participants, and various dimensions of the (functional) service dependency graph which emerges based on the syntactic/semantic dependencies between service providers. The results show that SBOTI (and the variants) can identify solutions of higher utility than the evaluated baseline algorithms (Section 5.4). Section 6.2 of this chapter extends this discussion on how well SBOTI performed agains the baseline algorithms.

6.2 Discussion

SBOTI was evaluated under various dynamic conditions, and the results compared to the existing baselines. In all the studies, the experimental results show the superior effectiveness of SBOTI in complex and highly dynamic, mobile environments, compared to existing baselines, but at a cost of increased communication overhead. The contributions to the body of knowledge are outlined first. A discussion of the results and a description of the limitations of SBOTI conclude this section.

6.2.1 Thesis Contributions

This thesis has made three contributions to the body of knowledge. The first contribution is SBOTI, a stigmergic-based QoS optimisation mechanism that can control the trade-off between computational efficiency and optimality of solutions, using an iterative approach that continuously explores the search space for better solutions. To allow for a broad selection of service composition configurations, a diversity strategy is used where the service composition configurations in the Pareto-optimal set are reinforced using a positive scalar and the service composition configurations outside this set are reinforced using an negative scalar to reduce the probability of exploration. This strategy was implemented in SBOTI, and the final variant named SBOTI-NDS.

The second contribution is an adaptation mechanism, which uses a pheromone smoothing mechanism to limit the pheromone reinforcement on the previously explored (and optimal) service composition configurations. The purpose of this approach was to allow new service composition to emerge as a result of service providers mobility. This adaptation mechanism was implemented in SBOTI, and the final variant named SBOTI-PM.

The third contribution was a collaborative agent communities approach to assess how much can the optimality and diversity of identified service composition can be improved by increasing the number of search agents. In addition to increasing the number of agents, this work also evaluated the cases when the search agents have different properties. SBOTI uses an agent-community to search for service composition configurations. Multiple agent communities can be used to search for service composition configurations in various parts of a given service dependency graph. These communities may share similar properties (homogeneous) or may have different properties (heterogeneous). These agent communities may exachange information about the identified solutions to improve the exploration of the solution space. SBOTI-HOM and SBOTI-HET are the variants that were implemented as part of this contribution.

6.2.2 Quality of Solutions

Three studies have been performed in this thesis. The first study evaluated SBOTI and SBOTI-NDS, a proposed variant that uses a solution diversity strategy to improve the diversity of identified solutions (Section 3.4.3). The results showed that the proposed diversity strategy, based on positive reinforcement of optimal solutions and negative reinforcement of non-optimal solutions, can improve the diversity of identified solutions. The results of this study addresses research question **Q1** in Section 1.3.4. The quality of identified solutions identified using SBOTI is bounded by the limitations due to the design decisions. SBOTI is not suitable for time-critical applications that require strict QoS guarantees. While SBOTI uses an iterative mechanism to continuously explore the environment, the identified solutions do not evenly cover the entire Pareto-front. This is because of the pheromone reinforcement mechanism, which can favour *stagnation* on a particular set of identified solutions. The diversity strategy in SBOTI-NDS addresses this limitation. By reducing the pheromone on the previously non-optimal solutions, the exploration of the solution space can be improved at a cost of reduced exploitation. However, by increasing the exploration, does not necessarily mean that more optimal solutions can be identified. The service composition configurations that emerge as a results of the mobility of services providers and the fluctuations in the QoS values, exacerbates the requirement for controlling the trade-off between the exploitation and exploration.

The second study showed that SBOTI-PM can pro-actively adapt to fluctuations in the QoS values, and can approximate the set of QoS-optimal of service composition configurations that may emerge in mobile environments. The results of this study addresses research question **Q2** in Section 1.3.4. Two evaluations are performed as part of this study. In each study, SBOTI and SBOTI-PM were initialised with different hyper-parameters. The first evaluation showed that the optimality of identified solutions can be improved compared to the optimality achieved by SBOTI-NDS, while maintaining similar distribution over the Pareto-front (i.e., spread) as SBOTI-NDS. The results of the second evaluation show that SBOTI-PM can also outperform SBOTI. By reducing the pheromone associated with optimal solutions, the adaptation mechanism in SBOTI-PM allows further exploration of the solution space. This study also highlighted the limitation due to the design decisions of SBOTI. Carefully selecting the hyper-parameters used to initialise this mechanism affects the performance of the mechanism.

The third study focused on the limitation of always requiring an optimal hyper-parameter set to initialise SBOTI. This study explores whether by using an information exchange mechanism between multiple, potentially diverse, agent communities, the approximation of the real Pareto-front can be improved. The results of this study addresses research question **Q2** in Section 1.3.4. The results of the evaluation showed that by using multiple heterogeneous agent communities diversity of identified solutions can be improved, which allows for a user to make better compromises between his objectives.

6.2.3 Computational Efficiency

Each study performed in Chapter 5 has concluded that SBOTI (and the other evaluated variants: SBOTI-NDS, SBOTI-PM, SBOTI-HOM and SBOTI-HET) introduce a higher overhead than the existing proposals (see Section 5.6, Section 5.7 and Section 5.8). This is partly because of the mechanism and the rate (frequency) of updates used when evaluating the baseline proposals. While SBOTI (selectively) samples the environment, the existing proposals use broadcast-based protocols to inform other service providers about any state changes (e.g., GoCoMo [Chen et al., 2018]). By increasing the rate of broadcasts, the utility of identified solutions may increase. Even if by increasing this rate, and a service composition configuration with a higher utility may be identified, the existing proposals are still limited to single-objective optimisation, which produces a single solution.

6.3 Future Work

This thesis has shown that SBOTI, a QoS optimisation mechanism for flexible service composition in mobile environments, can be used for finding better service composition configurations in a mobile environment compared to existing proposals in an environment where services are deployed on mobile devices and made available through a Dynamic Composition Overlay Network (DCON). Further research work may focus on:

1. *Heterogeneous Environments*: Mobile environments use devices with varying resource capacities such as CPU, RAM, and storage. Also, these devices are generally equipped with more than one network interface such as BLE and NFC, with various maximum bandwidth available. Future work should assess SBOTI's performance in heterogeneous operating environments. The limited resources available on these devices may affect the resilience service compositions, which may favour exploitation rather than exploration.

2. *Trustworthy Service Composition*: This work relies on services that are offered by third-party service providers to resolve user's request and process

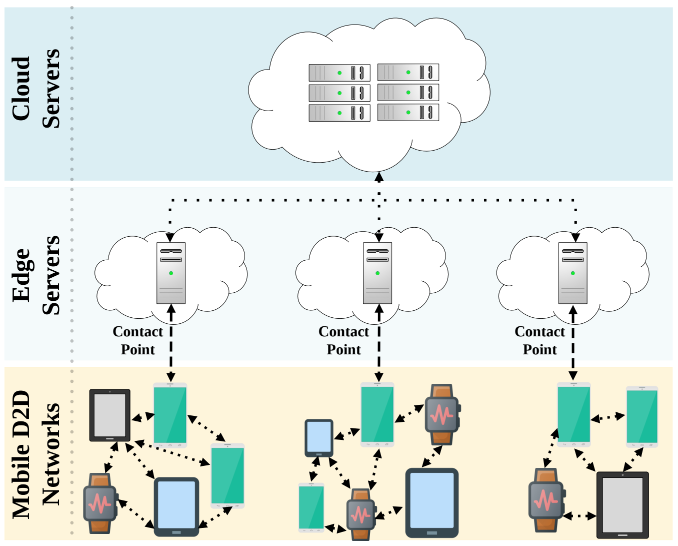


Figure 6.1: Device-to-Device (D2D) Networks of Mobile Devices using Edge Servers to access Cloud Servers.

their data. Fraudulent service providers may use attractive advertisements to deceive service consumers. Mechanisms to consider the trust of service providers has been considered in works such as Moustafa et al. [Moustafa et al., 2016]. Also, reputation based ratings or reviews can be used to represent service' trustworthiness levels represented by data from off-device authorities [Chang et al., 2006]. Future work should consider incorporating a trust-based scheme into SBOTI, and investigate the implications of such a scheme on SBOTI's performance in a mobile environment.

3. *More Realistic Mobility Models*: SBOTI's performance should be evaluated in more realistic mobility scenarios. All devices were set to the same speed during each test case, whereas in a real environment each device has its own speed. Future evaluations should consider scenarios with diverse characteristics such as the inter-contact time (ICT) distribution, node density and transmission ranges. Also, *a priori* information about the environment can be used to guide the search in SBOTI. In this context, future research questions may arise such as how to combine *a priori* environment knowledge to improve the optimality and diversity of identified solutions. How can the collaborative agent community approach be extended to improve the optimality of solutions?

4. *Cloud and Edge Computing*: Various IoT platforms have been previously proposed such as Mosden [Perera et al., 2014], Cheng et al. [Cheng et al.,

2016], FIoT [do Nascimento and de Lucena, 2017], Jin et al. [Jin et al., 2014], Le et al. [Le and Kwon, 2017] (a comprehensive list can be found in Razzaque et al. [Razzaque et al., 2016]). These platforms are either cloud-based. Future work should investigate the performance of the SURF middleware (Section 4.1) in a hybrid edge-clould computing model [Cicirelli et al., 2018, Olaniyan et al., 2018] (see Figure 6.1), and to investigate whether offloading some of the computational expensive tasks (such as data processing) into cloud may improve the resilience of service compositions in a mobile environment, and how this affects the service composition from a non-functional perspective.

Appendices

Appendix A

While Section 4.2 highlights how SBOTI is implemented in Simonstrator, the following appendix represents the configuration script used in Simonstrator to support its implementation.

This script contains a set of (XML) nodes. The *Default* node shows the values of the variables used in the configuration. These values are used to initialise the fields of rest of the nodes. In this work, the service providers rely on mobile devices to deliver their services, and each service provider is associated with a mobile device. The node *HostBuilder* illustrates the configuration of each service provider used during the evaluation. The sub-nodes of this node represent the elements that are part of the protocol stack of each service provider. The *Topology* node configures the topology and the speed of the mobile devices. The rest of the nodes in this script show how each layer in the protocol stack was configured.

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration xmlns:xi="http://www.w3.org/2001/XInclude">
    <Description>SBOTI</Description>
    <Default>
        <Variable name="seed" value="500" />
        <Variable name="startTime" value="0m" />
        <Variable name="finishTime" value="10080m" />
        <Variable name="actions" value="config/scenarios/sbotiTrans/dynamicBroadcast.dat" />
        <Variable name="WORLD_X" value="1000"/>
        <Variable name="WORLD_Y" value="1000"/>
        <Variable name="WIFI_RANGE" value="250" />
        <Variable name="WIFI_MODEL" value="80211" />
        <Variable name="WIFI_LOSS_EXPONENT" value="3.8" />
        <Variable name="SPEED_MIN" value="7.5" /><!-- Slow 1.5 and Medium 2.5-->
        <Variable name="SPEED_MAX" value="13.5" /><!-- Slow 2.5 and Medium 7.5-->
        <Variable name="PHY" value="WIFI" />
        <Variable name="ALGORITHM" value="GLOBAL_KNOWLEDGE" />
        <Variable name="ENABLE_FRAGMENTING" value="true" />
        <Variable name="upBandwidth" value="3150kbits" />
        <Variable name="downBandwidth" value="53665kbits" />
        <Variable name="enableChurn" value="true" />
        <Variable name="enableChurnController" value="false" />
        <Variable name="ENERGY_INITIAL" value="18648" />
        <Variable name="ENERGY_MODEL_ENABLED" value="true" />
        <Variable name="SERVICE_DENSITY" value = "1" />
        <Variable name="ITERATIONS_NUMBER" value = "250" />
        <Variable name="LOG" value="off" />
        <Variable name="ENABLE_DAO" value="false" />
    <Variable name="DAO_TABLE" value="test" />
    <Variable name="DAO_ENABLE_AT" value="0m" /> <!-- 30m -->
    <Variable name="DAO_STOP_AT" value="30m" /> <!-- 120m -->
</Default>

<SimulatorCore class="de.tud.kom.p2psim.impl.simengine.Simulator" static="getInstance" seed="$seed"
            finishAt="$finishTime" realTime="false" />
<Topology class="de.tud.kom.p2psim.impl.topology.TopologyFactory" worldX="$WORLD_X" worldY="$WORLD_Y">
    <View class="de.tud.kom.p2psim.impl.topology.views.RangedTopologyView" range="$WIFI_RANGE" phy="WIFI">
    <Latency class="de.tud.kom.p2psim.impl.topology.views.latency.DistanceBasedLatency" />
    <DropRate class="de.tud.kom.p2psim.impl.topology.views.droprate.StaticDropRate" dropRate="0" />
    <Placement class="de.tud.kom.p2psim.impl.topology.placement.RandomPositionDistribution" />
    <Movement class="de.tud.kom.p2psim.impl.topology.movement.GaussMarkovMovement"
            timeBetweenMoveOperations="15s" alpha="0.8"edgeThreshold="10" />
</Topology>
<LinkLayer class="de.tud.kom.p2psim.impl.linklayer.LinkLayerFactory" >
    <Mac class="de.tud.kom.p2psim.impl.linklayer.mac.configs.SimpleMac" phy="WIFI"
        trafficQueueSize="100" downBandwidth="53665kbits" upBandwidth="3150kbits" />
</LinkLayer>
<NetLayer class="de.tud.kom.p2psim.impl.network.routed.RoutedNetLayerFactory"
        enableFragmenting="$ENABLE_FRAGMENTING">
    <Routing class="de.tud.kom.p2psim.impl.network.routed.config.Routing" phy="$PHY" algorithm="$ALGORITHM"
            protocol="IPv4" />
</NetLayer>
<TransLayer class="de.tud.kom.p2psim.impl.transport.modular.ModularTransLayerFactory" />
<Overlay class="surfqosmos.overlay.sbotiTrans.SbotiTransNodeFactory" factoryType="SERVICE_PROVIDER"></Overlay>
<StartNode class="surfqosmos.overlay.sbotiTrans.SbotiTransNodeFactory" factoryType="START_NODE" ></StartNode>
<DestinationNode class="surfqosmos.overlay.sbotiTrans.SbotiTransNodeFactory"
                factoryType="DESTINATION_NODE" ></DestinationNode>
<AuxiliaryStartNode class="surfqosmos.overlay.sbotiTrans.SbotiTransNodeFactory"
                factoryType="AUXILIARY_START_NODE" ></AuxiliaryStartNode>
<AuxiliaryDestNode class="surfqosmos.overlay.sbotiTrans.SbotiTransNodeFactory"
                factoryType="AUXILIARY_DESTINATION_NODE" ></AuxiliaryDestNode>
<Controller class="surfqosmos.overlay.sbotiTrans.SbotiTransNodeFactory" factoryType="MAIN_CONTROLLER" >
</Controller>
<Monitor class="de.tud.kom.p2psim.impl.common.DefaultMonitor"
        static="getInstance" start="$startTime" stop="$finishTime" experimentDescription="CompositionExperiment">
    <Analyzer class="surfqosmos.overlay.generic.analyzer.CompositionLogger" />
</Monitor>
<Monitoring class="de.tudarmstadt.maki.simonstrator.overlay.monitoring.oracle.GlobalKnowledgeResolver$Factory" />
<SiS class="de.tudarmstadt.maki.simonstrator.service.sis.minimal.MinimalSiSComponent$Factory" />
<EnergyModel class="surfqosmos.overlay.generic.energy.EnergyModelFactory">
    <Battery class="de.tud.kom.p2psim.impl.energy.SimpleBattery" capacity="$ENERGY_INITIAL"
        initialEnergy="$ENERGY_INITIAL" />
    <Component class="de.tud.kom.p2psim.impl.energy.configs.TimeBased" phy="WIFI" />
</EnergyModel>

<HostBuilder class="de.tud.kom.p2psim.impl.scenario.DefaultHostBuilder" experimentSize="$TOTAL_NODES_NUMBER">
    <!-- Group of service providers -->
    <Group groupID="G1" size="$NODES_NUMBER">
        <Topology /><LinkLayer />
        <NetLayer upBandwidth="$upBandwidth" downBandwidth="$downBandwidth" />
        <TransLayer /><EnergyModel /><Overlay /><Monitoring /><SiS />
        <Properties enableChurn="$enableChurn" maxMovementSpeed="$SPEED_MAX" minMovementSpeed="$SPEED_MIN" />
    </Group>
    <!-- This is the main start node -->
    <Host groupID="S" size="$NUMBER_OF_MAIN_START_NODES">
        <Topology /><LinkLayer />
        <NetLayer upBandwidth="$upBandwidth" downBandwidth="$downBandwidth" />
        <TransLayer /><EnergyModel /><StartNode />
        <Properties enableChurn="$enableChurnController" maxMovementSpeed="0" minMovementSpeed="0" />
    </Host>
    <!-- This is the main destination node -->
    <Host groupID="D" size="$NUMBER_OF_MAIN_DESTINATION_NODES">
        <Topology /><LinkLayer />
        <NetLayer upBandwidth="$upBandwidth" downBandwidth="$downBandwidth" />
        <TransLayer /><EnergyModel /><DestinationNode />
        <Properties enableChurn="$enableChurnController" maxMovementSpeed="0" minMovementSpeed="0" />
    </Host>
```

```
        <!-- This will be the auxiliary start nodes that will be used to start optimisation from multiple places -->
        <Group groupID="SA" size="$NUMBER_OF_AUXILIAR_START_NODES">
            <Topology /><LinkLayer />
            <NetLayer upBandwidth="$upBandwidth" downBandwidth="$downBandwidth" />
            <TransLayer /><EnergyModel /><AuxiliaryStartNode />
            <Properties enableChurn="$enableChurnController" maxMovementSpeed="0" minMovementSpeed="0" />
        </Group>
        <!-- This is the main destination node -->
        <Host groupID="C" size="$NUMBER_OF_MAIN_DESTINATION_NODES">
            <Topology /><LinkLayer />
            <NetLayer upBandwidth="$upBandwidth" downBandwidth="$downBandwidth" />
            <TransLayer /><Controller />
            <Properties enableChurn="$enableChurnController"
                        maxMovementSpeed="0"
                        minMovementSpeed="0" />
        </Host>
    </HostBuilder>
    <Scenario class="de.tud.kom.p2psim.impl.scenario.CSVScenarioFactory" actionsFile="$actions"
              componentClass="surfqosmos.overlay.sbotiTrans.SbotiTransNode"
              additionalClasses="surfqosmos.overlay.sbotiTrans.CentralController" />

</Configuration>
```

Bibliography

[dub, 2018] (2018). Dublin Bus. https://www.dublinbus.ie/RTPI/. [Online; accessed May 2, 2019].

[gsm, 2018] (2018). The Mobile Economy 2018.

[Al-Ani and Seitz, 2016] Al-Ani, A. and Seitz, J. (2016). QoS-Aware Routing for Video Streaming in Multi-Rate Ad hoc Networks. In *Wireless and Mobile Networking Conference (WMNC), 2016 9th IFIP*, pages 193–198. IEEE.

[Al-Oqily and Karmouch, 2011] Al-Oqily, I. and Karmouch, A. (2011). A Decentralized Self-Organizing Service Composition for Autonomic Entities. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 6(1):7.

[Al Ridhawi and Karmouch, 2015] Al Ridhawi, Y. and Karmouch, A. (2015). QoS-Based Composition of Service Specific Overlay Networks. *IEEE Transactions on Computers*, 64(3):832–846.

[Alba, 2005] Alba, E. (2005). *Parallel Metaheuristics: a New Class of Algorithms*, volume 47. John Wiley & Sons.

[Alférez and Pelechano, 2017] Alférez, G. H. and Pelechano, V. (2017). Achieving Autonomic Web Service Compositions with Models at Runtime. *Computers & Electrical Engineering*, 63:332–352.

[Aloi et al., 2017] Aloi, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., and Savaglio, C. (2017). Enabling IoT Interoperability through Opportunistic Smartphone-Based Mobile Gateways. *Journal of Network and Computer Applications*, 81:74–84.

[Alrifai and Risse, 2009] Alrifai, M. and Risse, T. (2009). Combining Global Optimization with Local Selection for Efficient QoS-Aware Service Composition. In *Proceedings of the 18th International Conference on World wide web*, pages 881–890. ACM.

[Alsaryrah et al., 2018] Alsaryrah, O., Mashal, I., and Chung, T.-Y. (2018). Bi-Objective Optimization for Energy Aware IoT Service Composition. *IEEE Access*.

[Ameller and Franch Gutiérrez, 2008] Ameller, D. and Franch Gutiérrez, J. (2008). Service Level Agreement Monitor (SALMon). In *ICCBSS 2008 proceedings: Seventh International Conference on Composition-Based Software Systems: 25-29 February*

2008, Madrid, Spain, pages 224–227. Institute of Electrical and Electronics Engineers (IEEE).

[Angus and Woodward, 2009] Angus, D. and Woodward, C. (2009). Multiple Objective Ant Colony Optimisation. *Swarm Intelligence*, 3(1):69–85.

[Ardagna and Pernici, 2007] Ardagna, D. and Pernici, B. (2007). Adaptive Service Composition in Flexible Processes. *IEEE Transactions on Software Engineering*, 33(6).

[Artaiam and Senivongse, 2008] Artaiam, N. and Senivongse, T. (2008). Enhancing Service-Side QoS Monitoring for Web Services. In *Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pages 765–770. IEEE.

[Barbon et al., 2006] Barbon, F., Traverso, P., Pistore, M., and Trainotti, M. (2006). Run-time Monitoring of Instances and Classes of Web Service Compositions. In *Web Services, 2006. ICWS'06. International Conference on*, pages 63–71. IEEE.

[Baresi et al., 2004] Baresi, L., Ghezzi, C., and Guinea, S. (2004). Smart Monitors for Composed Services. In *Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 193–202. ACM.

[Baresi and Guinea, 2005] Baresi, L. and Guinea, S. (2005). Towards Dynamic Monitoring of WS-BPEL Processes. In *International Conference on Service-Oriented Computing*, pages 269–282. Springer.

[Baryannis et al., 2008] Baryannis, G., Carro, M., Danylevych, O., Dustdar, S., Karastoyanova, D., Kritikos, K., Leinter, P., Rosenberg, F., and Wetzstein, B. (2008). Overview of the State of the Art in Composition and Coordination of Services. *S-CUBE Software Services and Systems Network Consortium*, 1.

[Bechikh et al., 2016] Bechikh, S., Datta, R., and Gupta, A. (2016). *Recent Advances in Evolutionary Multi-Objective Optimization*, volume 20. Springer.

[Bello and Zeadally, 2016] Bello, O. and Zeadally, S. (2016). Intelligent Device-to-Device Communication in the Internet of Things. *IEEE Systems Journal*, 10(3):1172–1182.

[Blackwell and Branke, 2006] Blackwell, T. and Branke, J. (2006). Multiswarms, Exclusion, and Anti-convergence in Dynamic Environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472.

[Blum and Li, 2008] Blum, C. and Li, X. (2008). Swarm Intelligence in Optimization. In *Swarm Intelligence*, pages 43–85. Springer.

[Botee and Bonabeau, 1998] Botee, H. M. and Bonabeau, E. (1998). Evolving Ant Colony Optimization. *Advances in Complex Systems*, 1(02n03):149–159.

[BoussaïD et al., 2013] BoussaïD, I., Lepagnot, J., and Siarry, P. (2013). A Survey on Optimization Metaheuristics. *Information Sciences*, 237:82–117.

[Cabrera et al., 2017a] Cabrera, C., Li, F., Nallur, V., Palade, A., Razzaque, M., White, G., and Clarke, S. (2017a). Implementing Heterogeneous, Autonomous, and Resilient Services in IoT: an Experience Report. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on*, pages 1–6. IEEE.

[Cabrera et al., 2017b] Cabrera, C., Palade, A., and Clarke, S. (2017b). An Evaluation of Service Discovery Protocols in the Internet of Things. In *Proceedings of the Symposium on Applied Computing*, pages 469–476. ACM.

[Cabrera et al., 2018a] Cabrera, C., Palade, A., White, G., and Clarke, S. (2018a). Services in IoT: A Service Planning Model based on Consumer Feedback. In *Proceedings of 2018 International Conference on Service-Oriented Computing (ICSOC 2018)*. Springer.

[Cabrera et al., 2018b] Cabrera, C., White, G., Palade, A., and Clarke, S. (2018b). The Right Service at the Right Place: a Service Model for Smart Cities. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE.

[Cabrera and Franch, 2012] Cabrera, O. and Franch, X. (2012). A Quality Model for Analysing Web Service Monitoring Tools. In *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, pages 1–12. IEEE.

[Cai et al., 2006] Cai, W., Jin, X., Zhang, Y., Chen, K., and Wang, R. (2006). ACO Based QoS Routing Algorithm for Wireless Sensor Networks. In *International Conference on Ubiquitous Intelligence and Computing*, pages 419–428. Springer.

[Calinescu et al., 2011] Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., and Tamburrelli, G. (2011). Dynamic QoS Management and Optimization in Service-Based Systems. *IEEE Transactions on Software Engineering*, 37(3):387–409.

[Callaway et al., 2010] Callaway, R. D., Devetsikiotis, M., Viniotis, Y., and Rodriguez, A. (2010). An Autonomic Service Delivery Platform for Service-Oriented Network Environments. *IEEE Transactions on Services Computing*, 3(2):104–115.

[Canfora et al., 2005a] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2005a). An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1069–1075. ACM.

[Canfora et al., 2005b] Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2005b). QoS-Aware Replanning of Composite Web Services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 121–129. IEEE.

[Cardellini et al., 2012] Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Presti, F. L., and Mirandola, R. (2012). Moses: A Framework for QoS Driven Runtime

Adaptation of Service-Oriented Systems. *IEEE Transactions on Software Engineering*, 38(5):1138–1159.

[Cardoso et al., 2004] Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. (2004). Quality of Service for Workflows and Web Service Processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308.

[Castelli et al., 2015] Castelli, G., Mamei, M., Rosi, A., and Zambonelli, F. (2015). Engineering Pervasive Service Ecosystems: The SAPERE Approach. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(1):1.

[Cervantes et al., 2017] Cervantes, F., Ramos, F., Gutiérrez, L. F., Occello, M., and Jamont, J.-P. (2017). A New Approach for the Composition of Adaptive Pervasive Systems. *IEEE Systems Journal*.

[Chakraborty et al., 2005] Chakraborty, D., Joshi, A., Finin, T., and Yesha, Y. (2005). Service Composition for Mobile Environments. *Mobile Networks and Applications*, 10(4):435–451.

[Chang et al., 2006] Chang, E., Hussain, F., and Dillon, T. (2006). *Trust and Reputation for Service-Oriented Environments: Technologies for Building Business Intelligence and Consumer Confidence*. John Wiley & Sons.

[Chattopadhyay and Banerjee, 2017] Chattopadhyay, S. and Banerjee, A. (2017). QoS Constrained Large Scale Web Service Composition using Abstraction Refinement. *IEEE Transactions on Services Computing*.

[Chen et al., 2015a] Chen, B., Peng, X., Yu, Y., and Zhao, W. (2015a). Requirements-Driven Self-Optimization of Composite Services using Feedback Control. *IEEE Transactions on Services Computing*, 8(1):107–120.

[Chen et al., 2018] Chen, N., Cardozo, N., and Clarke, S. (2018). Goal-Driven Service Composition in Mobile and Pervasive Computing. *IEEE Transactions on Services Computing*, 11(1):49–62.

[Chen and Bahsoon, 2017] Chen, T. and Bahsoon, R. (2017). Self-Adaptive Trade-off Decision Making for Autoscaling Cloud-Based Services. *IEEE Transactions on Services Computing*, 10(4).

[Chen et al., 2015b] Chen, Y., Huang, J., Lin, C., and Hu, J. (2015b). A Partial Selection Methodology for Efficient QoS-Aware Service Composition. *IEEE Transactions on Services Computing*, 8(3):384–397.

[Cheng et al., 2016] Cheng, B., Zhu, D., Zhao, S., and Chen, J. (2016). Situation-Aware IoT Service Coordination using the Event-Driven SOA Paradigm. *IEEE Transactions on Network and Service Management*, 13(2):349–361.

[Chiandussi et al., 2012] Chiandussi, G., Codegone, M., Ferrero, S., and Varesio, F. E. (2012). Comparison of Multi-objective Optimization Methodologies for Engineering Applications. *Computers & Mathematics with Applications*, 63(5):912–942.

[Chitty and Hernandez, 2004] Chitty, D. M. and Hernandez, M. L. (2004). A Hybrid Ant Colony Optimisation Technique for Dynamic Vehicle Routing. In *Genetic and Evolutionary Computation Conference*, pages 48–59. Springer.

[Cicirelli et al., 2018] Cicirelli, F., Guerrieri, A., Spezzano, G., Vinci, A., Briante, O., Iera, A., and Ruggeri, G. (2018). Edge Computing and Social Internet of Things for Large-Scale Smart Environments Development. *IEEE Internet of Things Journal*, 5(4):2557–2571.

[Cobo et al., 2010] Cobo, L., Quintero, A., and Pierre, S. (2010). Ant-Based Routing for Wireless Multimedia Sensor Networks using Multiple QoS Metrics. *Computer Networks*, 54(17):2991–3010.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT press.

[Cruz et al., 2011] Cruz, C., González, J. R., and Pelta, D. A. (2011). Optimization in Dynamic Environments: A Survey on Problems, Methods and Measures. *Soft Computing*, 15(7):1427–1448.

[Das et al., 2009] Das, S., Biswas, A., Dasgupta, S., and Abraham, A. (2009). Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications. In *Foundations of Computational Intelligence Volume 3*, pages 23–55. Springer.

[De Lemos et al., 2013] De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., et al. (2013). Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer.

[Deb, 2014] Deb, K. (2014). Multi-Objective Optimization. In *Search Methodologies*, pages 403–449. Springer.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Deng et al., 2017] Deng, S., Huang, L., Taheri, J., Yin, J., Zhou, M., and Zomaya, A. Y. (2017). Mobility-Aware Service Composition in Mobile Communities. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(3):555–568.

[Deng et al., 2014] Deng, S., Huang, L., Tan, W., and Wu, Z. (2014). Top-K Automatic Service Composition: A Parallel Method for Large-Scale Service Sets. *IEEE Transactions on Automation Science and Engineering*, 11(3):891–905.

[Deng et al., 2016] Deng, S., Huang, L., Wu, H., Tan, W., Taheri, J., Zomaya, A. Y., and Wu, Z. (2016). Toward Mobile Service Computing: Opportunities and Challenges. *IEEE Cloud Computing*, 3(4):32–41.

[Di Caro and Dorigo, 1998] Di Caro, G. and Dorigo, M. (1998). AntNet: Distributed

Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365.

[Di Caro et al., 2004] Di Caro, G., Ducatelle, F., and Gambardella, L. M. (2004). AntHocNet: An Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks. In *International Conference on Parallel Problem Solving from Nature*, pages 461–470. Springer.

[Di Caro et al., 2005] Di Caro, G., Ducatelle, F., and Gambardella, L. M. (2005). AntHocNet: an Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad hoc Networks. *European Transactions on Telecommunications*, 16(5):443–455.

[Di Caro et al., 2008] Di Caro, G. A., Ducatelle, F., and Gambardella, L. M. (2008). Ant Colony Optimization for Routing in Mobile Ad Hoc Networks in Urban Environments. *IDSIA*, pages 5–8.

[D'Mello et al., 2011] D'Mello, D. A., Ananthanarayana, V., and Salian, S. (2011). A Review of Dynamic Web Service Composition Techniques. In *International Conference on Computer Science and Information Technology*, pages 85–97. Springer.

[do Nascimento and de Lucena, 2017] do Nascimento, N. M. and de Lucena, C. J. P. (2017). Fiot: An Agent-Based Framework for Self-Adaptive and Self-Organizing Applications Based on the Internet of Things. *Information Sciences*, 378:161–176.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

[Dorigo and Stützle, 2001] Dorigo, M. and Stützle, T. (2001). An Experimental Study of the Simple Ant Colony Optimization Algorithm. In *2001 WSES International Conference on Evolutionary Computation (EC'01)*, pages 253–258.

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). Ant Colony Optimization.

[Dressler and Akan, 2010] Dressler, F. and Akan, O. B. (2010). A Survey on Bio-Inspired Networking. *Computer Networks*, 54(6):881–900.

[Duan et al., 2003] Duan, Z., Zhang, Z.-L., and Hou, Y. T. (2003). Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning. *IEEE/ACM Transactions on Networking (TON)*, 11(6):870–883.

[Durillo et al., 2006] Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B., and Alba, E. (2006). jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. *Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10.*

[Efstathiou et al., 2014] Efstathiou, D., McBurney, P., Zschaler, S., and Bourcier, J. (2014). Efficient Multi-Objective Optimisation of Service Compositions in Mobile Ad hoc Networks Using Lightweight Surrogate Models. *J. UCS*, 20(8):1089–1108.

[Engelbrecht, 2006] Engelbrecht, A. P. (2006). *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons.

[Erl, 2005] Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design.* Pearson Education India.

[Estévez-Ayres et al., 2009] Estévez-Ayres, I., Basanta-Val, P., García-Valls, M., Fisteus, J. A., and Almeida, L. (2009). QoS-Aware Real-Time Composition Algorithms for Service-Based Applications. *IEEE Transactions on Industrial Informatics*, 5(3):278–288.

[Faniyi and Bahsoon, 2016] Faniyi, F. and Bahsoon, R. (2016). A Systematic Review of Service Level Management in the Cloud. *ACM Computing Surveys (CSUR)*, 48(3):43.

[Farooq and Di Caro, 2008] Farooq, M. and Di Caro, G. A. (2008). Routing Protocols for Next-Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview. In *Swarm Intelligence*, pages 101–160. Springer.

[Feng et al., 2013] Feng, Y., Ngan, L. D., and Kanagasabai, R. (2013). Dynamic Service Composition with Service-Dependent QoS Attributes. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 10–17. IEEE.

[Ferber and Weiss, 1999] Ferber, J. and Weiss, G. (1999). *Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence*, volume 1. Addison-Wesley Reading.

[Freitas, 2004] Freitas, A. A. (2004). A Critical Review of Multi-Objective Optimization in Data Mining: a Position Paper. *ACM SIGKDD Explorations Newsletter*, 6(2):77–86.

[Fulcher, 2008] Fulcher, J. (2008). Computational Intelligence: an Introduction. In *Computational Intelligence: a Compendium*, pages 3–78. Springer.

[Gabrel et al., 2018] Gabrel, V., Manouvrier, M., Moreau, K., and Murat, C. (2018). QoS-Aware Automatic Syntactic Service Composition Problem: Complexity and Resolution. *Future Generation Computer Systems*, 80:311–321.

[Gai et al., 2018] Gai, K., Qiu, M., and Sun, X. (2018). A Survey on FinTech. *Journal of Network and Computer Applications*, 103:262–273.

[Gao et al., 2018] Gao, R., Ye, F., Luo, G., and Cong, J. (2018). Introduction of Indoor Map Construction. In *Smartphone-Based Indoor Map Construction*, pages 1–2. Springer.

[García-Martínez et al., 2007] García-Martínez, C., Cordón, O., and Herrera, F. (2007). A Taxonomy and an Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-Criteria TSP. *European Journal of Operational Research*, 180(1):116–148.

[Georgakopoulos and Papazoglou, 2008] Georgakopoulos, D. and Papazoglou, M. P. (2008). *Service-Oriented Computing.* The MIT Press.

[Georgantas, 2018] Georgantas, N. (2018). *Service Oriented Computing in Mobile Environments: Abstractions and Mechanisms for Interoperability and Composition.* PhD thesis, Sorbonne Université.

[Geyik et al., 2013] Geyik, S. C., Szymanski, B. K., and Zerfos, P. (2013). Robust Dynamic Service Composition in Sensor Networks. *Services Computing, IEEE Transactions on*, 6(4).

[Goldberg, 2006] Goldberg, D. E. (2006). *Genetic Algorithms.* Pearson Education India.

[Grissom and Kim, 2005] Grissom, R. J. and Kim, J. J. (2005). *Effect Sizes for Research: A Broad Practical Approach.* Lawrence Erlbaum Associates Publishers.

[Groba and Clarke, 2014] Groba, C. and Clarke, S. (2014). Opportunistic Service Composition in Dynamic Ad-hoc Environments. *IEEE Transactions on Services Computing*, 7(4).

[Gu et al., 2003] Gu, X., Nahrstedt, K., Chang, R. N., and Ward, C. (2003). QoS-Assured Service Composition in Managed Service Overlay Networks. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 194–201. IEEE.

[Gu et al., 2008] Gu, Z., Li, J., and Xu, B. (2008). Automatic Service Composition based on Enhanced Service Dependency Graph. In *2008 IEEE International Conference on Web Services*, pages 246–253. IEEE.

[Gubbi et al., 2013] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645–1660.

[Gui et al., 2016] Gui, T., Ma, C., Wang, F., and Wilkins, D. E. (2016). Survey on Swarm Intelligence based Routing Protocols for Wireless Sensor Networks: An Extensive Study. In *Industrial Technology (ICIT), 2016 IEEE International Conference on*, pages 1944–1949. IEEE.

[Hachem et al., 2014] Hachem, S., Pathak, A., and Issarny, V. (2014). Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution. In *IEEE GLOBECOM: Global Communications Conference (Accepted).*

[Halonen and Ojala, 2006] Halonen, T. and Ojala, T. (2006). Cross-Layer Design for Providing Service Oriented Architecture in a Mobile Ad Hoc Network. In *Proceedings of the 5th international Conference on Mobile and Ubiquitous Multimedia*, page 11. ACM.

[Hashmi et al., 2016] Hashmi, K., Malik, Z., Erradi, A., and Rezgui, A. (2016). QoS Dependency Modeling for Composite Systems. *IEEE Transactions on Services Computing.*

[Hayyolalam and Kazem, 2018] Hayyolalam, V. and Kazem, A. A. P. (2018). A Systematic Literature Review on QoS-Aware Service Composition and Selection in Cloud Environment. *Journal of Network and Computer Applications*.

[Hofmeyr and Forrest, 2000] Hofmeyr, S. A. and Forrest, S. (2000). Architecture for an Artificial Immune System. *Evolutionary Computation*, 8(4):443–473.

[Hossain et al., 2016] Hossain, M. S., Moniruzzaman, M., Muhammad, G., Ghoneim, A., and Alamri, A. (2016). Big Data-Driven Service Composition using Parallel Clustered Particle Swarm Optimization in Mobile Environment. *IEEE Transactions on Services Computing*, 9(5):806–817.

[Huebscher and McCann, 2008] Huebscher, M. C. and McCann, J. A. (2008). A Survey of Autonomic Computing-Degrees, Models, and Applications. *ACM Computing Surveys (CSUR)*, 40(3):7.

[Huhns and Singh, 2005] Huhns, M. N. and Singh, M. P. (2005). Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 9(1):75–81.

[Hull and Su, 2005] Hull, R. and Su, J. (2005). Tools for Composite Web Services: A Short Overview. *ACM SIGMOD Record*, 34(2):86–95.

[Hussein and Saadawi, 2003] Hussein, O. and Saadawi, T. (2003). Ant Routing Algorithm for Mobile Ad-hoc Networks (ARAMA). In *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*, pages 281–290. IEEE.

[Immonen and Pakkala, 2014] Immonen, A. and Pakkala, D. (2014). A Survey of Methods and Approaches for Reliable Dynamic Service Compositions. *Service Oriented Computing and Applications*, 8(2):129–158.

[Islam et al., 2015] Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., and Kwak, K.-S. (2015). The Internet of Things for Health Care: a Comprehensive Survey. *IEEE Access*, 3:678–708.

[Jaeger and Mühl, 2007] Jaeger, M. C. and Mühl, G. (2007). QoS-Based Selection of Services: The Implementation of a Genetic Algorithm. In *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, pages 1–12. VDE.

[Jaeger et al., 2004] Jaeger, M. C., Rojec-Goldmann, G., and Muhl, G. (2004). QoS Aggregation for Web Service Composition using Workflow Patterns. In *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International*, pages 149–159. IEEE.

[Jiang et al., 2007] Jiang, S., Xue, Y., and Schmidt, D. C. (2007). Minimum Disruption Service Composition and Recovery over Mobile Ad Hoc Networks. In *Mobile and Ubiquitous Systems: Networking & Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1–8. IEEE.

[Jiang et al., 2014] Jiang, W., Hu, S., and Liu, Z. (2014). Top K Query for QoS-Aware

Automatic Service Composition. *IEEE Transactions Services Computing*, 7(4):681–695.

[Jin et al., 2014] Jin, J., Gubbi, J., Marusic, S., and Palaniswami, M. (2014). An Information Framework for Creating a Smart City through Internet of Things. *IEEE Internet of Things Journal*, 1(2):112–121.

[Jurca et al., 2007] Jurca, R., Faltings, B., and Binder, W. (2007). Reliable QoS Monitoring Based on Client Feedback. In *Proceedings of the 16th International Conference on World Wide Web*, pages 1003–1012. ACM.

[Kalasapur et al., 2007] Kalasapur, S., Kumar, M., and Shirazi, B. A. (2007). Dynamic Service Composition in Pervasive Computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(7).

[Karaboga and Basturk, 2007] Karaboga, D. and Basturk, B. (2007). A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*, 39(3):459–471.

[Karaboga et al., 2014] Karaboga, D., Gorkemli, B., Ozturk, C., and Karaboga, N. (2014). A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications. *Artificial Intelligence Review*, 42(1):21–57.

[Karmouch and Nayak, 2012] Karmouch, E. and Nayak, A. (2012). A Distributed Constraint Satisfaction Problem Approach to Virtual Device Composition. *IEEE Transactions on Parallel and Distributed Systems*, 23(11):1997–2009.

[Khosrowshahi-Asl et al., 2011] Khosrowshahi-Asl, E., Noorhosseini, M., and Pirouz, A. S. (2011). A Dynamic Ant Colony Based Routing Algorithm for Mobile Ad-hoc Networks. *Journal of Information Science and Engineering*, 27(5):1581–1596.

[Klein et al., 2014] Klein, A., Ishikawa, F., and Honiden, S. (2014). SanGA: A Self-Adaptive Network-Aware Approach to Service Composition. *IEEE Transactions on Services Computing*, 7(3):452–464.

[Ko et al., 2008] Ko, J. M., Kim, C. O., and Kwon, I.-H. (2008). Quality-of-Service Oriented Web Service Composition Algorithm and Planning Architecture. *Journal of Systems and Software*, 81(11):2079–2090.

[Krishna et al., 2012] Krishna, P. V., Saritha, V., Vedha, G., Bhiwal, A., and Chawla, A. S. (2012). Quality-of-Service-Enabled Ant Colony-Based Multipath Routing for Mobile Ad hoc Networks. *IET Communications*, 6(1):76–83.

[Kritikos et al., 2013] Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., Brandic, I., Kertész, A., Parkin, M., and Carro, M. (2013). A Survey on Service Quality Description. *ACM Computing Surveys (CSUR)*, 46(1):1.

[Kulkarni et al., 2011] Kulkarni, R. V., Forster, A., and Venayagamoorthy, G. K. (2011). Computational Intelligence in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys & Tutorials*, 13(1):68–96.

[Küster et al., 2005] Küster, U., Stern, M., and König-Ries, B. (2005). A Classification of Issues and Approaches in Automatic Service Composition. In *Intl Workshop WESC*, volume 5, pages 25–35.

[Le and Kwon, 2017] Le, M. and Kwon, Y.-W. (2017). Utilizing Nearby Computing Resources for Resource-Limited Mobile Devices. In *Proceedings of the Symposium on Applied Computing*, pages 572–575. ACM.

[Lemos et al., 2016] Lemos, A. L., Daniel, F., and Benatallah, B. (2016). Web Service Composition: a Survey of Techniques and Tools. *ACM Computing Surveys (CSUR)*, 48(3):33.

[Leung et al., 2010] Leung, C., Wong, T., Mak, K.-L., and Fung, R. Y. (2010). Integrated Process Planning and Scheduling by an Agent-Based Ant Colony Optimization. *Computers & Industrial Engineering*, 59(1):166–180.

[Li and Yan-Xiang, 2010] Li, W. and Yan-Xiang, H. (2010). A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 218–224. Springer.

[Li and Shen, 2012] Li, Z. and Shen, H. (2012). Game-Theoretic Analysis of Cooperation Incentive Strategies in Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 11(8):1287–1303.

[Liang et al., 2018] Liang, H., Du, Y., Jiang, T., and Li, F. (2018). A Comprehensive Multi-Objective Approach of Service Selection for Service Processes with Twofold Restrictions. *Future Generation Computer Systems*.

[Liang et al., 2009] Liang, Q., Wu, X., and Lau, H. C. (2009). Optimizing Service Systems Based on Application-Level QoS. *IEEE Transactions on Services Computing*, 2(2):108–121.

[Liu et al., 2017] Liu, C., Cao, J., and Wang, J. (2017). A Reliable and Efficient Distributed Service Composition Approach in Pervasive Environments. *IEEE Transactions on Mobile Computing*, 16(5):1231–1245.

[Liu et al., 2004] Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pages 66–73. ACM.

[Liu et al., 2005] Liu, Z., Kwiatkowska, M., and Constantinou, C. (2005). A Biologically Inspired QoS Routing Algorithm for Mobile Ad Hoc Networks. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 1, pages 426–431. IEEE.

[Lopez-Ibanez and Stutzle, 2012] Lopez-Ibanez, M. and Stutzle, T. (2012). The Automatic Design of Multiobjective Ant Colony Optimization Algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875.

[Ludwig et al., 2003] Ludwig, H., Keller, A., Dan, A., King, R. P., and Franck, R. (2003). Web Service Level Agreement (WSLA) Language Specification. *IBM Corporation*, pages 815–824.

[Lv et al., 2015] Lv, C., Jiang, W., Hu, S., Wang, J., Lu, G., and Liu, Z. (2015). Efficient Dynamic Evolution of Service Composition. *IEEE Transactions on Services Computing*.

[Ma et al., 2013] Ma, H., Bastani, F., Yen, I.-L., and Mei, H. (2013). QoS-driven Service Composition with Reconfigurable Services. *IEEE Transactions on Services Computing*, 6(1):20–34.

[Ma et al., 2015] Ma, H., Wang, A., and Zhang, M. (2015). A Hybrid Approach using Genetic Programming and Greedy Search for QoS-aware Web Service Composition. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVIII*, pages 180–205. Springer.

[Mabrouk et al., 2009] Mabrouk, N. B., Beauche, S., Kuznetsova, E., Georgantas, N., and Issarny, V. (2009). QoS-Aware Service Composition in Dynamic Service Oriented Environments. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 123–142. Springer.

[Mahbub and Spanoudakis, 2004] Mahbub, K. and Spanoudakis, G. (2004). A Framework for Requirements Monitoring of Service Based Systems. In *Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 84–93. ACM.

[Martin et al., 2004] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). OWL-S: Semantic Markup for Web Services. *W3C member submission*, 22(4).

[Mascitti et al., 2018] Mascitti, D., Conti, M., Passarella, A., Ricci, L., and Das, S. K. (2018). Service Provisioning in Mobile Environments through Opportunistic Computing. *IEEE Transactions on Mobile Computing*.

[Mehboob et al., 2016] Mehboob, U., Qadir, J., Ali, S., and Vasilakos, A. (2016). Genetic Algorithms in Wireless Networking: Techniques, Applications, and Issues. *Soft Computing*, 20(6):2467–2501.

[Mishra and Harit, 2010] Mishra, K. and Harit, S. (2010). A Fast Algorithm for Finding the Non Dominated Set in Multi Objective Optimization. *Int. Journal of Computer Applications*, 1(25):35–39.

[Model, 2011] Model, B. P. (2011). Notation (BPMN) version 2.0. *OMG Specification, Object Management Group*, pages 22–31.

[Mostafa and Zhang, 2015] Mostafa, A. and Zhang, M. (2015). Multi-Objective Service Composition in Uncertain Environments. *IEEE Transactions on Services Computing*.

[Moustafa et al., 2016] Moustafa, A., Zhang, M., and Bai, Q. (2016). Trustworthy Stigmergic Service Composition and Adaptation in Decentralized Environments. *IEEE Transactions on Services Computing*, 9(2):317–329.

[Nielsen, 2014] Nielsen (2014). Is Sharing the New Buying? *Nielsen Company, New York, May*, 28.

[Oasis, 2007] Oasis, B. (2007). Web Services Business Process Execution Language.

[Olaniyan et al., 2018] Olaniyan, R., Fadahunsi, O., Maheswaran, M., and Zhani, M. F. (2018). Opportunistic Edge Computing: Concepts, Opportunities and Research Challenges. *arXiv preprint arXiv:1806.04311*.

[Omicini, 2013] Omicini, A. (2013). Nature-Inspired Coordination for Complex Distributed Systems. In *Intelligent Distributed Computing VI*, pages 1–6. Springer.

[Opricovic and Tzeng, 2004] Opricovic, S. and Tzeng, G.-H. (2004). Compromise Solution by MCDM Methods: A Comparative Analysis of VIKOR and TOPSIS. *European Journal of Operational Research*, 156(2):445–455.

[O'Sullivan et al., 2002] O'Sullivan, J., Edmond, D., and Ter Hofstede, A. (2002). What's in a Service? *Distributed and Parallel Databases*, 12(2-3):117–133.

[Owyang et al., 2014] Owyang, J., Samuel, A., and Grenville, A. (2014). *Sharing is the New Buying: How to Win in the Collaborative Economy.* Vision Critical/Crowd Companies.

[Page, 2010] Page, S. E. (2010). *Diversity and Complexity.* Princeton University Press.

[Palade et al., 2018a] Palade, A., Cabrera, C., Li, F., White, G., Razzaque, M., and Clarke, S. (2018a). Middleware for Internet of Things: an Evaluation in a Small-Scale IoT Environment. *Journal of Reliable Intelligent Environments*, pages 1–21.

[Palade et al., 2018b] Palade, A., Cabrera, C., White, G., and Clarke, S. (2018b). Stigmergic Service Composition and Adaptation in Mobile Environments. In *In Proceedings of 2018 International Conference on Service-Oriented Computing (ICSOC 2018).* Springer.

[Palade et al., 2017] Palade, A., Cabrera, C., White, G., Razzaque, M., and Clarke, S. (2017). Middleware for Internet of Things: A quantitative evaluation in small scale. In *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on*, pages 1–6. IEEE.

[Palade and Clarke, 2018] Palade, A. and Clarke, S. (2018). Stigmergy-Based QoS Optimisation for Flexible Service Composition in Mobile Communities. In *IEEE World Congress on Services (IEEE SERVICES).* IEEE.

[Papazoglou et al., 2008] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2008). Service-Oriented Computing: A Research Roadmap. *International Journal of Cooperative Information Systems*, 17(02):223–255.

[Park and Shin, 2008] Park, E. and Shin, H. (2008). Reconfigurable Service Composition and Categorization for Power-Aware Mobile Computing. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1553–1564.

[Parunak, 1997] Parunak, H. V. D. (1997). “Go to the Ant”: Engineering Principles from Natural Multi-Agent Systems. *Annals of Operations Research*, 75:69–101.

[Patikirikorala et al., 2012] Patikirikorala, T., Colman, A., Han, J., and Wang, L. (2012). A Systematic Survey on the Design of Self-Adaptive Software Systems using Control Engineering Approaches. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 33–42. IEEE Press.

[Patnaik et al., 2017] Patnaik, S., Yang, X.-S., and Nakamatsu, K. (2017). *Nature-Inspired Computing and Optimization*, volume 10. Springer.

[Peng et al., 2018] Peng, X., Gu, J., Tan, T. H., Sun, J., Yu, Y., Nuseibeh, B., and Zhao, W. (2018). CrowdService: Optimizing Mobile Crowdsourcing and Service Composition. *ACM Transactions on Internet Technology (TOIT)*, 18(2):19.

[Perera et al., 2013] Perera, C., Jayaraman, P., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2013). Dynamic Configuration of Sensors Using Mobile Sensor Hub in Internet of Things Paradigm. In *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on*, pages 473–478. IEEE.

[Perera et al., 2014] Perera, C., Jayaraman, P. P., Zaslavsky, A., Georgakopoulos, D., and Christen, P. (2014). Mosden: An Internet of Things Middleware for Resource Constrained Mobile Devices. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 1053–1062. IEEE.

[Pop et al., 2010] Pop, C. B., Chifu, V. R., Salomie, I., Dinsoreanu, M., David, T., and Acretoaie, V. (2010). Ant-Inspired Technique for Automatic Web Service Composition and Selection. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on*, pages 449–455. IEEE.

[Qiqing et al., 2009] Qiqing, F., Xiaoming, P., Qinghua, L., and Yahui, H. (2009). A Global QoS Optimizing Web Services Selection Algorithm Based on MOACO for Dynamic Web Service Composition. In *Information Technology and Applications, 2009. IFITA'09. International Forum on*, volume 1, pages 37–42. IEEE.

[Rao and Su, 2004] Rao, J. and Su, X. (2004). A Survey of Automated Web Service Composition Methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer.

[Razzaque et al., 2016] Razzaque, M., Milojevic-Jevric, M., Palade, A., and Clarke, S. (2016). Middleware for Internet of Things: A Survey. *Internet of Things Journal, IEEE*, 3(1).

[Richerzhagen et al., 2015] Richerzhagen, B., Stingl, D., Rückert, J., and Steinmetz, R. (2015). Simonstrator: Simulation and Prototyping Platform for Distributed Mobile Applications. In *Proc. 8th International Conference on Simulation Tools and Techniques (SIMUTOOLS)*, pages 99–108. ACM.

[Rodriguez-Mier et al., 2017] Rodriguez-Mier, P., Mucientes, M., and Lama, M. (2017). Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition. *IEEE Transactions on Services Computing*, 10(4).

[Sadiq et al., 2015] Sadiq, U., Kumar, M., Passarella, A., and Conti, M. (2015). Service Composition in Opportunistic Networks: A Load and Mobility Aware Solution. *IEEE Transactions on Computers*, 64(8):2308–2322.

[Saleem et al., 2011] Saleem, M., Di Caro, G. A., and Farooq, M. (2011). Swarm Intelligence Based Routing Protocol for Wireless Sensor Networks: Survey and Future Directions. *Information Sciences*, 181(20):4597–4624.

[Satoh, 2010] Satoh, I. (2010). Mobile Agents. In *Handbook of Ambient Intelligence and Smart Environments*, pages 771–791. Springer.

[Shanshan et al., 2012] Shanshan, Z., Lei, W., Lin, M., and Zepeng, W. (2012). An Improved Ant Colony Optimization Algorithm for QoS-Aware Dynamic Web Service Composition. In *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, pages 1998–2001. IEEE.

[Shen et al., 2011] Shen, J., Beydoun, G., Yuan, S., and Low, G. (2011). Comparison of Bio-Inspired Algorithms for Peer Selection in Services Composition. In *IEEE International Conference on Services Computing*. IEEE.

[Sim and Sun, 2003] Sim, K. M. and Sun, W. H. (2003). Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(5):560–572.

[Sindhya, 2011] Sindhya, K. (2011). Hybrid Evolutionary Multi-Objective Optimization with Enhanced Convergence and Diversity. *Jyväskylä Studies in Computing*, (131).

[Singh et al., 2005] Singh, M. P., Huhns, M. N., and Huhns, M. N. (2005). *Service-Oriented Computing: Semantics, Processes, Agents.* John Wiley & Sons.

[Srinivasan and Treadwell, 2005] Srinivasan, L. and Treadwell, J. (2005). An Overview of Service-Oriented Architecture, Web Services and Grid Computing. *HP Software Global Business Unit*, 2.

[Stavropoulos et al., 2013] Stavropoulos, T. G., Vrakas, D., and Vlahavas, I. (2013). A Survey of Service Composition in Ambient Intelligence Environments. *Artificial Intelligence Review*, pages 1–24.

[Strunk, 2010] Strunk, A. (2010). QoS-Aware Service Composition: A Survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74. IEEE.

[Stutzle and Dorigo, 2002] Stutzle, T. and Dorigo, M. (2002). A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(4):358–365.

[Stützle and Hoos, 2000] Stützle, T. and Hoos, H. H. (2000). MAX–MIN Ant System. *Future Generation Computer Systems*, 16(8):889–914.

[Sydow and Cheney, 2018] Sydow, L. and Cheney, S. (2018). 2017 Retrospective: A Monumental Year for the App Economy.

[Talbi, 2009] Talbi, E.-G. (2009). *Metaheuristics: from Design to Implementation*, volume 74. John Wiley & Sons.

[Talbi et al., 2012] Talbi, E.-G., Basseur, M., Nebro, A. J., and Alba, E. (2012). Multi-Objective Optimization using Metaheuristics: Non-standard Algorithms. *International Transactions in Operational Research*, 19(1-2):283–305.

[Teixeira et al., 2011] Teixeira, T., Hachem, S., Issarny, V., and Georgantas, N. (2011). Service Oriented Middleware for the Internet of Things: a Perspective. In *European Conference on a Service-Based Internet*, pages 220–229. Springer.

[Temglit et al., 2017] Temglit, N., Chibani, A., Djouani, K., and Nacer, M. A. (2017). A Distributed Agent-Based Approach for Optimal QoS Selection in Web of Object Choreography. *IEEE Systems Journal*.

[Theraulaz and Bonabeau, 1999] Theraulaz, G. and Bonabeau, E. (1999). A Brief History of Stigmergy. *Artificial Life*, 5(2).

[Trummer et al., 2014] Trummer, I., Faltings, B., and Binder, W. (2014). Multi-Objective Quality-Driven Service Selection-a Fully Polynomial Time Approximation Scheme. *IEEE Transactions on Software Engineering*, 40(2):167–191.

[Urbieta et al., 2008] Urbieta, A., Barrutieta, G., Parra, J., and Uribarren, A. (2008). A Survey of Dynamic Service Composition Approaches for Ambient Systems. In *Proceedings of the 2008 Ambi-Sys workshop on Software Organisation and MonITоring of ambient systems*, page 1. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Vakili and Navimipour, 2017] Vakili, A. and Navimipour, N. J. (2017). Comprehensive and Systematic Review of the Service Composition Mechanisms in the Cloud Environments. *Journal of Network and Computer Applications*, 81:24–36.

[van Der Aalst et al., 2003] van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., and Barros, A. P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51.

[Veldhuizen and Lamont, 2000] Veldhuizen, D. A. V. and Lamont, G. B. (2000). Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147.

[Wada et al., 2012] Wada, H., Suzuki, J., Yamano, Y., and Oba, K. (2012). E^3: A Multiobjective Optimization Framework for SLA-Aware Service Composition. *IEEE Transactions on Services Computing*, 5(3):358–372.

[Wagner et al., 2012] Wagner, F., Klein, A., Klöpper, B., Ishikawa, F., and Honiden, S. (2012). Multi-Objective Service Composition with Time-and Input-Dependent QoS. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 234–241. IEEE.

[Wang et al., 2014] Wang, D., Huang, H., and Xie, C. (2014). A Novel Adaptive Web Service Selection Algorithm Based on Ant Colony Optimization for Dynamic Web Service Composition. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 391–399. Springer.

[Wang, 2011] Wang, J. (2011). Exploiting Mobility Prediction for Dependable Service Composition in Wireless Mobile Ad Hoc Networks. *IEEE Transactions on Services Computing*, 4(1):44–55.

[Wang and Shen, 2017] Wang, L. and Shen, J. (2017). A Systematic Review of Bio-Inspired Service Concretization. *IEEE Transactions on Services Computing*, 10(4):493–505.

[Wang et al., 2015] Wang, L., Shen, J., and Luo, J. (2015). Facilitating an Ant Colony Algorithm for Multi-Objective Data-Intensive Service Provision. *Journal of Computer and System Sciences*, 81(4):734–746.

[Wang and Du, 2016] Wang, P. and Du, X. (2016). QoS-Aware Service Selection Using An Incentive Mechanism. *IEEE Transactions on Services Computing*.

[Wang et al., 2017] Wang, Y., Chen, R., Cho, J.-H., Swami, A., and Chan, K. S. (2017). Trust-Based Service Composition and Binding with Multiple Objective Optimization in Service-Oriented Mobile Ad Hoc Networks. *IEEE Transactions on Services Computing*, 10(4):660–672.

[Wang et al., 2011] Wang, Z.-J., Liu, Z.-Z., Zhou, X.-F., and Lou, Y.-S. (2011). An Approach for Composite Web Service Selection Based on DGQoS. *The International Journal of Advanced Manufacturing Technology*, 56(9-12):1167–1179.

[Weerawarana et al., 2005] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and more.* Prentice Hall PTR.

[Wetzstein et al., 2009] Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Dustdar, S., and Leymann, F. (2009). Monitoring and Analyzing Influential Factors of Business Process Performance. In *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*, pages 141–150. IEEE.

[Weyns et al., 2013] Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., and Göschka, K. M. (2013). On Patterns for Decentralized Control in Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems II*, pages 76–107. Springer.

[White et al., 2018a] White, G., Cabrera, C., Palade, A., and Clarke, S. (2018a). Augmented Reality in IoT. In *Service-Oriented Computing: 16th International Conference, ICSOC*, pages 12–15.

[White et al., 2017a] White, G., Palade, A., Cabrera, C., and Clarke, S. (2017a). Quantitative Evaluation of QoS Prediction in IoT. In *Dependable Systems and Networks Workshop (DSN-W), 2017 47th Annual IEEE/IFIP International Conference on*, pages 61–66. IEEE.

[White et al., 2018b] White, G., Palade, A., Cabrera, C., and Clarke, S. (2018b). IoTPredict: Collaborative QoS Prediction in IoT. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE.

[White et al., 2019] White, G., Palade, A., Cabrera, C., and Clarke, S. (2019). Autoencoders for QoS Prediction at the Edge.

[White et al., 2017b] White, G., Palade, A., and Clarke, S. (2017b). Qos Prediction for Reliable Service Composition in IoT. In *International Conference on Service-Oriented Computing*, pages 149–160. Springer.

[White et al., 2018c] White, G., Palade, A., and Clarke, S. (2018c). Forecasting QoS Attributes using LSTM Networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Wiesemann et al., 2008] Wiesemann, W., Hochreiter, R., and Kuhn, D. (2008). A Stochastic Programming Approach for QoS-Aware Service Composition. In *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*, pages 226–233. IEEE.

[Wu et al., 2016] Wu, Q., Ishikawa, F., Zhu, Q., and Shin, D.-H. (2016). QoS-Aware Multigranularity Service Composition: Modeling and Optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(11):1565–1577.

[Wu and Zhu, 2013] Wu, Q. and Zhu, Q. (2013). Transactional and QoS-aware Dynamic Service Composition Based on Ant Colony Optimization. *Future Generation Computer Systems*, 29(5):1112–1119.

[Xia et al., 2008] Xia, Y.-m., Chen, J.-l., and Meng, X.-w. (2008). On the Dynamic Ant Colony Algorithm Optimization Based on Multi-Pheromones. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 630–635. IEEE.

[Xiao and Boutaba, 2005] Xiao, J. and Boutaba, R. (2005). QoS-Aware Service Composition and Adaptation in Autonomic Communication. *IEEE Journal on Selected Areas in Communications*, 23(12):2344–2360.

[Yan et al., 2007] Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., and Zhang, J. (2007). Autonomous Service Level Agreement Negotiation for Service Composition Provision. *Future Generation Computer Systems*, 23(6):748–759.

[Yan et al., 2012] Yan, Y., Chen, M., and Yang, Y. (2012). Anytime QoS Optimization over the PlanGraph for Web Service Composition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1968–1975. ACM.

[Yang et al., 2013] Yang, S., Li, M., Liu, X., and Zheng, J. (2013). A Grid-Based Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, 17(5):721–736.

[Yang et al., 2010] Yang, Z., Shang, C., Liu, Q., and Zhao, C. (2010). A Dynamic Web Services Composition Algorithm based on the Combination of Ant Colony Algorithm and Genetic Algorithm. *Journal of Computational Information Systems*, 6(8):2617–2622.

[Ye et al., 2016] Ye, D., He, Q., Wang, Y., and Yang, Y. (2016). An Agent-based Integrated Self-evolving Service Composition Approach in Networked Environments. *IEEE Transactions on Services Computing*.

[Yin et al., 2014] Yin, H., Zhang, C., Zhang, B., Guo, Y., and Liu, T. (2014). A Hybrid Multiobjective Discrete Particle Swarm Optimization Algorithm for a SLA-Aware Service Composition Problem. *Mathematical Problems in Engineering*, 2014.

[Yu et al., 2015] Yu, Q., Chen, L., and Li, B. (2015). Ant Colony Optimization Applied to Web Service Compositions in Cloud Computing. *Computers & Electrical Engineering*, 41:18–27.

[Yu et al., 2007] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6.

[Yunwu, 2009] Yunwu, W. (2009). Application of Chaos Ant Colony Algorithm in Web Service Composition Based on QoS. In *Information Technology and Applications, 2009. IFITA'09. International Forum on*, volume 2, pages 225–227. IEEE.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327.

[Zhang et al., 2017] Zhang, H., Wang, X., Memarmoshrefi, P., and Hogrefe, D. (2017). A Survey of Ant Colony Optimization Based Routing Protocols for Mobile Ad Hoc Networks. *IEEE Access*, 5:24139–24161.

[Zhang et al., 2007] Zhang, L.-J., Cai, H., and Zhang, J. (2007). *Services Computing*. Springer.

[Zhang et al., 2010] Zhang, W., Chang, C. K., Feng, T., and Jiang, H.-y. (2010). QoS-Based Dynamic Web Service Composition with Ant Colony Optimization. In *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*, pages 493–502. IEEE.

[Zhao et al., 2014] Zhao, D., Li, X.-Y., and Ma, H. (2014). How to Crowdsource Tasks Truthfully without Sacrificing Utility: Online Incentive Mechanisms with Budget Constraint. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1213–1221. IEEE.

[Zheng et al., 2016] Zheng, H., Yang, J., and Zhao, W. (2016). Probabilistic QoS Aggregations for Service Composition. *ACM Transactions on the Web (TWEB)*, 10(2):12.

[Zheng et al., 2007] Zheng, X., Luo, J., and Song, A. (2007). Ant Colony System Based Algorithm for QoS-Aware Web Service Selection. In *GSEM*, pages 39–50.

[Zheng et al., 2014] Zheng, Z., Zhang, Y., and Lyu, M. R. (2014). Investigating QoS of Real-World Web Services. *IEEE Transactions on Services Computing*, 7(1):32–39.

[Zitzler and Thiele, 1999] Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: a Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

[Zou et al., 2014] Zou, G., Lu, Q., Chen, Y., Huang, R., Xu, Y., and Xiang, Y. (2014). QoS-Aware Dynamic Composition of Web Services using Numerical Temporal Planning. *IEEE Transactions on Services Computing*, 7(1):18–31.