Towards Network-Aware Composition of Big Data Services in the Cloud

Umar SHEHU
Department of Computer Science and Technology University of Bedfordshire Luton, UK

Ghazanfar SAFDAR
Department of Computer Science and Technology University of Bedfordshire Luton, UK

Gregory EPIPHANIOU
Department of Computer Science and Technology University of Bedfordshire Luton, UK

***Abstract*—Several Big data services have been developed on the cloud to meet increasingly complex needs of users. Most times a single Big data service may not be capable in satisfying user requests. As a result, it has become necessary to aggregate services from different Big data providers together in order to execute the user's request. This in turn has posed a great challenge; how to optimally compose services from a given set of Big data providers without affecting if not optimizing Quality of Service (QoS). With the advent of cloud-based Big data applications composed of services spread across different network environments, QoS of the network has become important in determining the true performance of composite services. However current studies fail to consider the impact of QoS of network on composite service selection. Therefore a novel network-aware genetic algorithm is proposed to perform composition of Big data services in the cloud. The algorithm adopts an extended QoS model which separates QoS of network from service QoS. It also uses a novel network coordinate system in finding composite services that have low network latency without compromising service QoS. Results of evaluation indicate that the proposed approach finds low latency and QoS-optimal compositions when compared with current approaches.**

Keywords—Big data; Service composition; QoS; Genetic Algorithm; Network latency; Cloud

I. INTRODUCTION

Service Oriented Computing (SOC) is a framework that allows for internet applications to be built by coupling web services together. In SOC, each web service represents a different functional aspect of a Service-oriented application (SOA) [32].

Web services are network-accessible objects that allow Big data vendors to build service-oriented Big data applications (SOA) which share business logic and application services with other vendors in order to meet growing consumer needs. A Big data service (BDS), also known as Big-data-as-a-service (BDaaS), is a data intensive web service that works on large scale unstructured or semi-structured datasets. They typically perform tasks such as data storage, processing, cleaning, extraction, modelling and virtualization on large datasets. BDaaS consist mainly of three layers namely; the infrastructure layer, platform layer and application layer. The infrastructure layer provisions the physical resources required to process large datasets. The platform layer houses the operating systems and virtual machines that run BDaaS applications. The application layer represents the models and software used to process Big data.

Every BDS is characterized by the ability to provide some task as identified by its functional and non-functional attributes [27]. The functional attributes define what the service is capable of doing e.g. Microsoft Azure BDS [30] provides cloud-based machine learning framework for analyzing large scale datasets. Non-functional attributes on the other hand determine how well a service can perform a given task e.g. how long it will take Microsoft Azure BDS to respond to a user request. Non-functional attributes are commonly referred to as QoS (Quality of Service). Examples of service QoS attributes include response time, cost, reputation, etc. They are often used as criteria in selecting services suitable for a user request especially in situation where there is more than one service with similar functionality. For instance, a Microsoft Azure BDS having response time equal to 10 seconds will be more suitable to a user requiring prompt service response than a similar service such as Amazon AWS BDS with a response time of 30 seconds. Thus service QoS is used to differentiate services that are similar in terms of their functionalities. In SOC, functionally similar services are usually categorized in the same service group and referred to as candidate services.

A. QoS-Aware Web Service Composition

Recently, the ability of services to aggregate their functionalities has gained much attention. This is as a result of increased complexity of user requests. Simple user requests may require only a single BDS to be completed. However, as user requests take more complex forms that are beyond the capabilities of a single BDS, aggregating service abilities is necessary to carry out the request. This process is known as service composition. It combines services in order to build a composite service [8, 9] that is viewed by the user as a single service. Within a composite service each constituent BDS takes care of a specific functional aspect of the user's request. For instance suppose a user issues a compound request like "Analyse e-books dataset" consisting of several sub requests at the task level such as Twitter feed analysis, Natural language processing and IOT device log analysis (as seen in Fig. 1). A single BDS is ill sufficient to satisfy the compound request, therefore services from different Big data vendors for each sub request will need to be discovered and aggregated together according to their QoS to complete the user request. QoS-aware service composition process is similar in principle to the behaviour of workflow management system [28] in which a workflow dictates how data should be processes. A workflow processes data by using different patterns that

transform data to an end result. Service composition uses similar patterns found in workflow systems. The patterns allow composition process to channel data flow from one BDS to another. Some major service composition patterns include sequence, parallel, exclusive choice and loop. Service composition process begins by breaking down a complex request into smaller functions or sub-requests organized according to one of several patterns. Depending on the pattern involved, service QoS are then orchestrated to determine QoS for a composite service. Usually there are several candidate services that exist within a service group that can execute a given functional aspect of the request.

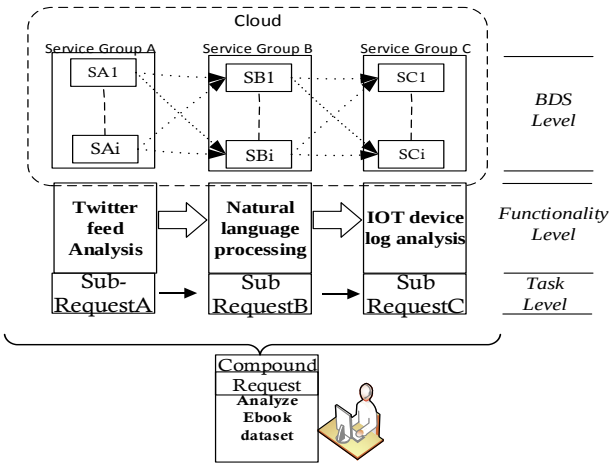


Fig. 1. Typical BDS composition scenario

Therefore choosing a service from each service group that maximizes the QoS of composite service while satisfying the user's constraints has become a research problem. The problem is also known as an NP-Hard optimization problem [18]. The problem has been solved using several techniques such as Linear Integer Programming [13] and Dynamic Programming [11]. Although techniques based on genetic algorithms [18] are usually used in finding near optimal compositions in polynomial time.

B. Service Composition in the Cloud

More and more BDS are increasingly being deployed on the cloud with the purpose of allowing Internet users from around the globe to access their functionalities for analysing large datasets. For instance organizations such as Amazon and Microsoft offer public cloud services using Amazon Web Services (AWS) [31] and Windows Azure [30] cloud platforms respectively. These services are deployed on cloud data centres via virtual machines (VM) where consumers can access them from literally any part of the world. VMs enable the processing resources such as CPU, storage and network resources needed to properly run cloud-based BDS. Traditionally, service providers deploy their VMs across several cloud data centres located in different geographical areas to host their BDS. Hence, each user will experience different network performances depending on the geographical location of the hosted service. Thus, when a user tries to invoke a composite service with candidate services spanning different cloud locations, the composition may not be able to deliver on the network performance needs of the user even if it has optimal service QoS. This is because the optimal service QoS only represents application level performance of a composite service but it does not account for its network performance. The impact of the network is usually quantified using a metric such as network latency [22]. The effect of network latency on application performance is noticeable in cloud environments where there is high degree of service distribution. Despite this, current studies do not separate QoS of network from service QoS. Hence, they may produce compositions that have sub-optimal performance when invoked by the user. An example is illustrated in Fig. 2. The example shows several BDS deployed on different clouds. Assuming each cloud consists of two or more BDS and is separated from other clouds by different round trip times (RTT). Also assuming a user request consists of a sequence pattern of the three tasks (t_1, t_2, and t_3) in Fig. 1, with each task having a set of candidate services and their respective QoS scores for cost (*P*), response time (*RT*) and execution time (*ET*). Current approaches will ordinarily pick the QoS optimal composite service (highlighted using bold boxes in Fig. 3) consisting of services ($|S_{A1}$-S_{B1}-$S_{C2}|$) with respect to cost, execution time and response time.

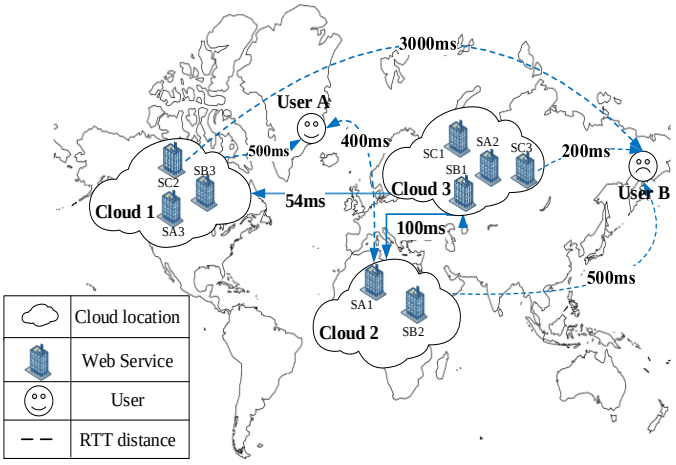


Fig. 2. BDS deployment locations

In doing so, users may experience different levels of performance for this optimal solution depending on the RTT between clouds of participating services.

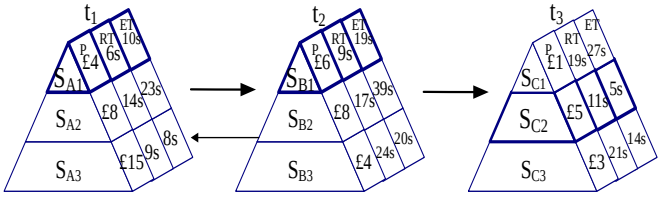


Fig. 3. Sequence workflow pattern with services and their QoS scores

BDS having shorter RTT will incur lower latency than those further away from each other. Therefore user A may experience low network latency for composite service $|S_{A1}$-S_{B1}-$S_{C2}|$ (i.e. end-to-end network latency for $|S_{A1}$-S_{B1}-$S_{C2}|$ is 400ms + 100ms + 54ms + 500ms = 1054ms), while user B experiences high network latency because of larger RTT (i.e.

500ms + 100ms + 54ms + 3000ms = 3654ms). Perhaps similar composite services like $|S_{A2}\text{-}S_{B1}\text{-}S_{C2}|$ (3087ms) or $|S_{A2}\text{-}S_{B1}\text{-}S_{C3}|$ (311ms) may be better suited for user B since they have lower network latency (as seen in Fig. 3.). This work differs from current approaches in that it separates QoS of network from service QoS. Integrating network latency property into the QoS model will allow us to find composition who's QoS in not only optimal at the application level, but also has near-optimal QoS of network from the user's perspective.

In this paper, a network aware approach to service composition which optimizes network latency and service QoS objectives such as cost, response time and execution time is proposed. It consists of a novel network model which first estimates network latency between BDS in the cloud. Estimation is necessary as traditional latency measurement methods which involve distribution of RTT pings to directly measure RTT between services are generally slow and computationally expensive [4, 6]. Information from the network model is fed to a novel network-aware composition technique based on genetic algorithm in order to find solutions that have optimal service QoS without compromising QoS of the network.

The paper is organized as follows: In Section II an analysis of recent research efforts is presented after which the proposed approach is described in Section III. Section IV presents a discussion of evaluation results. Finally, Section V concludes the paper.

II. Related Work

A. QoS-Based Service Composition

QoS-aware service composition problem has been modelled as an NP-Hard problem [24]. Several classes of approaches have been developed to address the problem. Earlier studies devised local optimization methods to finding optimal composite services. These methods employ search techniques to find services local to each subtask, then combines them into a composite service that will complete the user's request. Techniques developed include dynamic programming [11], learning depth first search [10] and simple additive weighing methods [12]. Another class of approaches widely used are linear integer programming techniques [13, 14]. These techniques use integer variables to search for optimal solutions without having to construct all possible combinations. Meta-heuristic (MH) approaches have been developed to tackle the NP-hard problem. These approaches are based on evolutionary concepts in nature. Some major MH approaches include Genetic [1, 2], particle swarm optimization methods [15] and artificial immune algorithms [29]. All these classes of approaches use similar QoS model which does not take QoS of network into consideration. In comparison, the network-aware genetic algorithm incorporates QoS of the network in the QoS model as it tackles the NP-Hard problem. Genetic algorithm has been chosen because it shows great promise in solving constrained multi-objective optimization. It is also capable of producing a set of solutions in which no solution is dominant to the others, thereby giving the user a wide range of near-optimal solutions to choose from.

B. Network-Aware Service Composition

Several studies have dealt with service composition while considering the impact of QoS of the network. Authors in [3] present a network-based service composition technique for component services in large scale overlay networks. Similarly, another study in [4] introduces network awareness in composing domain services in multi-domain networks. The authors try to optimize delay and available bandwidth. However, these studies do not consider service composition in the context of services in the cloud. A recent approach [2] develop a service composition technique that minimizes network latency of composite services in the Cloud. The authors use a network model based on Euclidean distance technique to estimate latency of composite services. Their work is similar with this study in that they consider network latency. The main difference is that they only consider QoS of network while this work considers QoS of network alongside service QoS objectives.

C. Network Coordinate System

Network coordinate systems (NCS) are used to estimate latency between nodes in a network [2]. Their purpose is to reduce the delay observed from sending physical round trip time (RTT) packets between nodes across the network path. They operate by predicting RTT measurements for a fraction of nodes on the network path using techniques such Euclidean distance estimation (EDE) [5, 16, 17] and matrix factorization (MF) [21]. EDE embed network distances between nodes as metric spaces where known network distances (RTT) are mapped into a two dimensional Euclidean space in order to predict unknown network distances. EDE is however susceptible to triangle inequality [21] which leads to inaccurate estimates. MF on the other hand estimate unmeasured network distances by factorizing distance matrix consisting of both known and unknown RTT values using mathematical concepts such as gradient descent [19]. MF does not use metric spaces and so is resistant to triangle inequality and produces more accurate than EDE. Current EDE and MF models adopt a centralized approach towards RTT estimation. The approach usually involves using a central server to collectively predict RTT values for all the nodes in the network path. This means that if one RTT value is inaccurately estimated, then the accuracy of other RTT values could be negatively affected. In this work, the problem is avoided by adopting a novel decentralized MF approach within the network model where each BDS takes charge of predicting RTT with its neighbouring services independently of other services on the cloud network.

III. Proposed Approach

A. Problem Formulation

The problem can be described as follows:

Given a user request T that will require a set of tasks t_1 to t_n,

$$T = \{t_1, t_2, \ldots, t_n\},$$

Where *n* is the number of tasks to complete user request.

Each task is assigned a service group (*S*) which defines a set of candidate services (s_{ij}) capable of performing the given task (as seen in Fig. 4.),

$$S_i = \{s_{i1}, s_{i2}, \ldots, s_{ik_i}\},\ \forall i \in [1..n], \forall j \in [1..k]$$

Where k_i is the number of candidate services in the *i*-th service group.

For each task, only one candidate service within its service group can be bound to the task t_i to form a composite service *C*.

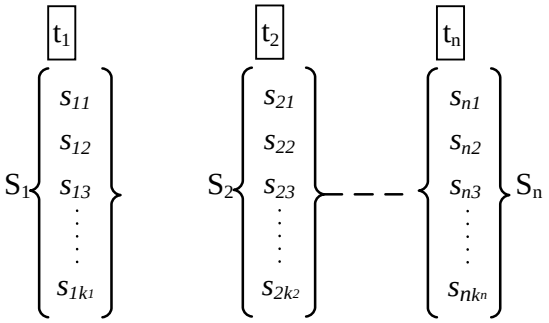


Fig. 4. Classification of candidate services into service groups and tasks

A composite service can be formed from the aggregation of candidate services;

$$C = \{s_{1j}, s_{2j}, \ldots, s_{nj}\},\ \forall j \in [1..k],\ \forall i \in [1..n]$$

Where s_{ij} is the BDS bound to its service group S_i.

Also, given a set of QoS objectives (cost, execution time, response time and network latency) that need to be optimized, the end-to-end QoS value of a composite service ($Q(C)$) is calculated by combining individual QoS values of its services (one per task) based on the following expressions.

In order to determine end-to-end cost of composite service, cost for each service ($P(s_{ij})$) are combined

$$Q_P(C) = \sum_{i=1}^{n} P(s_{ij}) \quad (1)$$

Similarly, both end-to-end response time ($RT(s_{ij})$) and execution time ($ET(s_{ij})$) are aggregated respectively

$$Q_{RT}(C) = \sum_{i=1}^{n} RT(s_{ij}) \quad (2),$$

$$Q_{ET}(C) = \sum_{i=1}^{n} ET(s_{ij}) \quad (3)$$

As for end-to-end network latency, RTT values are combined between each service in a given composite service

$$Q_{NL}(C) = \sum_{i=1}^{n} NL(s_{i,j}, s_{i+1,j}) \quad (4)$$

Where $NL(s_{i,j}, s_{i+1,j})$ represents the round trip time between each BDS in the cloud. Q_P, Q_{RT}, Q_{ET} and Q_{NL} represent end-to-end cost, response time, execution time and network latency of a composite service respectively.

Given weights w_P , w_{RT} , w_{ET} and w_{NL} which represent relative importance of QoS objectives from the user's perspective. Where,

$$\sum_{m=1}^{4} w_m = 1 \quad (5)$$

QoS objectives are normalized into fitness values using the expressions in Equations (6) and (7). Cost, response time and execution time are computed thus

$$F_m(C) = \sum_{\substack{i=1 \\ m \in \{P,RT,ET\}}}^{n} \left(\frac{Max_m(S_i) - Q_m(s_{ij})}{Max_m(S_i) - Min_m(S_i)} \right) \times w_m \quad (6)$$

$$\forall j \in [1..k]$$

Network latency fitness value for composite service (F_{NL}) is determined by an expression in Equation (6) which normalizes the end-to-end network latency QoS (Q_{NL}).

$$F_{NL}(C) = \frac{Q_{NL}(C)}{H} \times w_{NL} \quad (7)$$

Where H is a constant which normalizes value of $Q_{NL}(C)$ in the range of [0 1].

The research problem becomes a constrained multi-objective optimization problem where the aim is to find a set of composite services with near-optimal fitness values,

$$C_{best} = Min[F(C)]$$

Subject to:

- Selection constraint: Only one candidate service can be selected per service group.
- Minimum ($q_m^{\min}$) and maximum ($q_m^{\max}$) QoS constraints:

$$\forall Q_p \in \left[q_p^{\min} \quad q_p^{\max} \right], \forall Q_{RT} \in \left[q_{RT}^{\min} \quad q_{RT}^{\max} \right],$$
$$\forall Q_{ET} \in \left[q_{ET}^{\min} \quad q_{ET}^{\max} \right]$$

B. Network Model

In this study, a network model for estimating the RTT between BDS deployed on the cloud is adopted. The network model is composed of network coordinate system (NCS) based on Matrix factorization called LADMF (Learning-based Decentralized Matrix Factorization). Traditional MF techniques measure RTT ($d_{s_{ij}-s_{nk}}$) between a BDS and a subset of neighbors to build distance matrix *D*. These

measurements are then used to predict RTT values ($d^*_{s_{ij}-s_{nk}}$) for non-neighboring services as seen in Fig. 5. In mathematical terms, standard MF finds estimates of row matrix *X* and transposed column matrix *Y* that minimize the difference (ε) between measured RTT values in *D* and in estimated values in matrix D_{new},

$$\min[\varepsilon] \tag{8}$$

Where ε is the latency prediction error;

$$\varepsilon = (D - D_{new})^2 \tag{9}$$

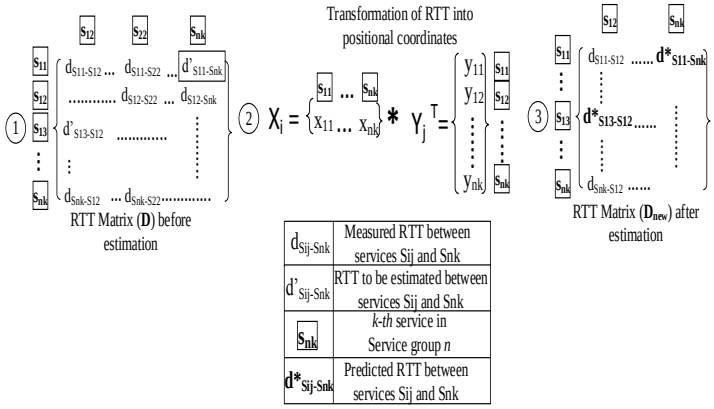


Fig. 5. Network distance estimation using matrix factorization

Also D_{new} is expressed as;

$$D_{new} = X * Y^T \tag{10}$$

Where X and Y are positional coordinates of all BDS on a given cloud network.

the standard MF technique is modified by adding learning automata concepts in order to further improve prediction accuracy of the estimation process. Instead of constructing a collective matrix (D_{new}) for all RTT estimates, LADMF decentralizes the process by allowing each BDS to estimate its own RTT values irrespective of other services. This is achieved by encoding each service as a learning automaton (LA) [5]. LA converts Equations (10) and (9) into Equations (11) and (12) respectively,

$$D_{ij_{new}} = X_i * Y_j^T \tag{11}$$

$$\varepsilon = \left[\left(D_{ij} - D_{ij_{new}} \right)^2 \right] \tag{12}$$

Where X_i is positional coordinate of *i*-th service, Y_j is positional coordinate of *j*-th neighbouring service, while D_{ij} .is the RTT between services *i* and *j*.

The effect is that each BDS will control their own path to RTT estimation without influencing estimation path of other services. Hence an inaccurate estimation of one service coordinate will not affect accuracy of other service coordinates.

In LADMF X_i and Y_j are encoded with additional LA parameters as seen in Fig. 6.

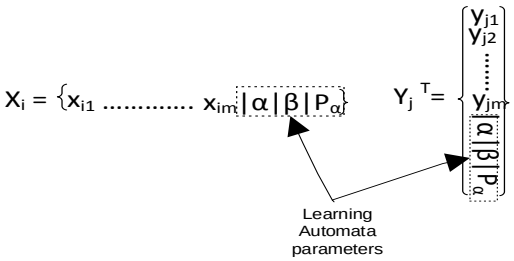


Fig. 6. Encoding of position vectors with LA parameters

Where

- α represents two alternative update strategies (α_1 and α_2) employed in updating position coordinates in X_i and Y_j:

$$\alpha \begin{cases} \alpha_1 \succ X_{i(new)} = D_{ij} Y_j (Y_j^T Y_j + (\Omega + J_1) I)^{-1}, \\ \quad Y_{j(new)} = D_{ij} X_i (X_i^T X_i + (\Omega + J_2) I)^{-1} \\ \alpha_2 \succ X_{i(new)} = D_{ij} Y_j (Y_j^T Y_j + (\Omega - J_1) I)^{-1}, \\ \quad Y_{j(new)} = D_{ij} X_i (X_i^T X_i + (\Omega - J_2) I)^{-1} \end{cases} \tag{13}$$

Also

- Ω - Regularization parameter that controls speed of update
- J_1 and J_2 are constants
- I - Identity matrix
- β represents feedback for every action in α. β = {$\beta_{\alpha 1}$, $\beta_{\alpha 2}$}
- P_α is action probability which is determined from feedback of estimation error.

If feedback for action α_1 is good ($\beta_{\alpha 1}$ = 0 i.e. ε is improved) then action probability $P_{\alpha 1}$ is rewarded while $P_{\alpha 2}$ is penalized,

$$\beta_{\alpha_1} = 0 \begin{cases} P_{\alpha_1(new)} = P_{\alpha_1} + c(1 - P_{\alpha_1}) & c = 0.5, \\ & e = 0.005 * c \\ P_{\alpha_2(new)} = P_{\alpha_2} - e(1 - P_{\alpha_2}) & \end{cases} \tag{14}$$

Else if feedback is bad ($\beta_{\alpha 1}$ = 1 i.e. ε is not improved) then reverse is the case,

$$\beta_{\alpha_2} = 1 \begin{cases} P_{\alpha_2(new)} = P_{\alpha_2} + c(1 - P_{\alpha_2}) & c = 0.5, \\ & e = 0.005 * c \\ P_{\alpha_1(new)} = P_{\alpha_1} - e(1 - P_{\alpha_1}) & \end{cases} \tag{15}$$

Actions are evaluated and assigned probabilities based on error feedback which in this case is the estimation error (

$\min[\varepsilon]$). The action with the highest probability is selected as the next action. The process is continued until the estimation error is minimized. LADMF algorithm is outlined in Algorithm 1. Afterwards, estimated RTT values are aggregated to determine end-to-end network latency for a composite service via Equation (4).

Algorithm 1 LADMF Algorithm

Input: *D, max_iter, L,*
Ouput: *Dnew*
1: [*X, Y*] = function *LADMF(D)*
2: { for(*i* =1: *maxIter*)
3: for(*j* =1: *max candidate service*)
4: $X \leftarrow$ rand(*x*)
5: $Y \leftarrow$ rand(*y*)
6: $\mathcal{E} \leftarrow w\,[D - (X * Y^T)]^2$
7: if ($\mathcal{E}$ is minimised)
8: $Dnew \leftarrow X * Y^T$
9: return
10: endif
11: endfor
12: endfor
13: }

C. Network-Aware Service Composition Algorithm

A novel network-aware service composition technique based on non-dominated sort genetic algorithm (NSGA) is presented. When applying genetic algorithm to service composition problem, each genome represents a possible composite service and is encoded in form of array of numbers or genes, each gene in turn represents a task and can be assigned to any one of its candidate services (as seen in Fig. 7). State of the art NSGA initiates optimization process by building an initial generation of genomes then sorts individuals according to their fitness value and crowding distance. The best individuals are placed in a mating pool where they are altered by crossover and mutation operators to generate children that will populate subsequent generations. The whole process is repeated until optimization is reached. The state of the art NSGA algorithm is enhanced in order to be able to solve research problem. The improved algorithm called INSGA is described step by step as follows:

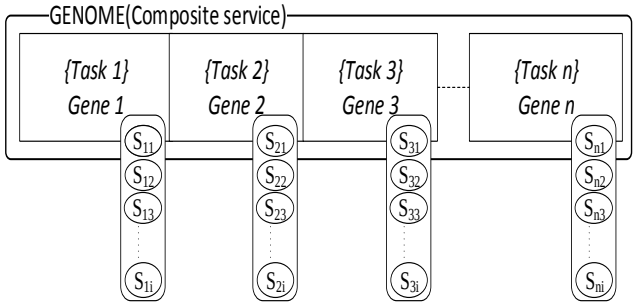


Fig. 7. Structure of composite service genome

Step.1. Initialization of Population. INSGA starts by randomly generating an initial population from the BDS that are part of the cloud. In order for this to be achieved, every service is first encoded as a two digit integer value. For example in Fig. 8, a BDS is encoded as "33" is the 3rd candidate service capable of executing task 3. In the next step only one candidate service is arbitrarily selected per task.

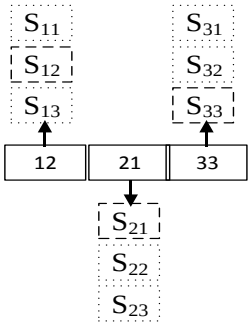


Fig. 8. Example of a composite service encoded as integer array

BDS QoS scores are then randomly initialized within their boundary constraints. With the aid of LADMF algorithm, the QoS scores are normalized and aggregated into values representative of composite service end-to-end cost, response time, execution time and network latency respectively.

Step 2. Ranking and Sorting. INSGA uses a non-dominated sorting technique that ranks individuals into different fronts according to the degree that they dominate other individuals in the population. A composite service C_i perfectly dominates another composite service C_j if all four fitness values of C_i are lower than the fitness values of C_j. Therefore C_i will be placed in a higher rank (front) than C_j. For each front, individuals are sorted in ascending order according to the magnitude of their fitness. This is used to establish the crowding distance (*CD*) which indicates the Euclidean distance between individual in the fitness value space. *CD* for a given composite service C_i is expressed as;

$$CD(C_i) = \frac{F(C_{i+1}) - F(C_{i-1})}{F_{\max} - F_{\min}} \tag{16}$$

Where

- $CD(C_i)$ is the crowding distance for the *i*-th individual.
- $F(C_{i+1})$ represents the fitness value of individual succeeding *i*-th individual.
- $F(C_{i-1})$ represents the fitness value of individual preceding *i*-th individual
- $F_{\max}$ and $F_{\min}$ represent the maximum and minimum fitness values in population

Step 3. Tournament Selection. A tournament selection of the best individuals that meet the user's satisfaction constraint is achieved to determine parents who will take part in crossover operation. The selection process ensures that only individuals with best fitness, rank and do not violate user constraint are selected for crossover operation.

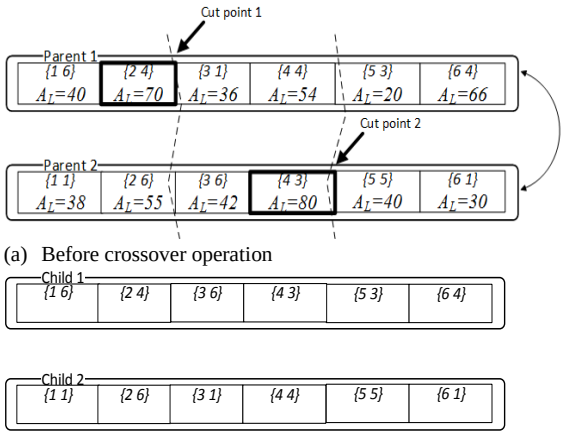
Step 4. Crossover Operation. Crossover operation combines any two parents into offspring (children) that are quite different from their parents and can have superior

properties of both parents. Traditional crossover operation picks arbitrary cut points where genes around cut points of one parent are replaced with genes of another parent to construct a set of children. INSGA employs a novel two-point crossover which cuts parents at two non-random cut points. The two cut points (one per parent) are chosen from points on each parent where average network latency is high. In order to determine which point on a parent constitutes poor average latency, every BDS assigned an average latency score (A_L) which is the arithmetic sum of RTT values over all outgoing paths divided by the number of outgoing paths from a given service,

$$A_L(s) = 1/|G| \sum_{\forall g \in G} Q_{NL}(g) \quad (17)$$

Where A_L *(s)* represents average latency score in milliseconds (ms) for service *s*, G is number of outgoing paths from *s*, and Q_{NL} *(g)* is RTT value for a given path.

Once average latency scores are known, the crossover operator selects a cut point from each parent where A_L is maximum. After the cut points are known then the genes around those points are interchanged between both parents. This ensures that genes having highest A_L are interchanged with genes having lower A_L . Fig. 9 depicts how crossover operation is performed.



(a) Before crossover operation

(b) After crossover operation

Fig. 9. INSGA's two point crossover operation when cut point 1 and cut point 2 are not the same

When cut points 1 and 2 are the same for both parents then the crossover operation translates to a single point crossover. The impact of the crossover operator is that children produced are low latency versions of their parents as demonstrated by the results.

Step 5. Mutation Operation. The function of mutation operation is to adjust a parent into new offspring that closely resemble its parent with the aim of further improving parent fitness values and discourage trapping into local optima. The standard mutation operator adjusts parents by using a uniform distribution index (DI) [23]. DI controls degree of similarity between parents and their children. The value for DI influences the diversity of offsprings in the population. A new mutation operation is presented. The operator uses a variable distribution index whose value depends on a parent's crowding distance and fitness value for network latency. Each parent is going to be mutated according to the value of its distribution index which is computed using the following expression:

$$mum_{par\,i} = \left\{ \frac{H}{F_{NL}(par_i) + (1 - CD(par_i))} \right\} \times CD(par_i) \quad (18)$$

Where

- mum_{par_i} is the distribution index for the parent.
- $F_{NL}(par_i)$ represents the parent's fitness value for network latency.
- $CD(par_i)$ indicates the parent's crowding distance.
- H is a constant.

The expression in Equation (18) will force a strong mutation for poor quality parents and a weak mutation for good quality parents. A large value for mum_{par_i} will indicate parent has good fitness and crowding distance therefore offspring's genes will closely resemble the parent (i.e. weak mutation), while a small value for mum_{par_i} indicates parent has poor fitness and crowding distance hence genes of offspring will differ greatly with the parent (i.e. strong mutation). This will ultimately improve the population diversity of new offspring and also increase the likelihood of finding the global solution. After mutation operation is performed, parents are replaced by newly formed off springs and the whole process is repeated until maximum number of generation is reached. INSGA algorithm is outlined in Algorithm 2 while the unique crossover and mutation operators are outlined in Algorithm 3 and 4 respectively.

Algorithm 2 INSGA Algorithm

Input: *D, g, n,* Ω *, h, max_iter, no_states, state, actions_prob, rp_env, w,* J_1*,* J_2*,*
Ouput: *pop*
1: Set environment parameters
2: *pop* ← Randomly generate population
3: P ← Randomly generate QoS values of solutions
4: *pop*[Q , f] ← Determine end-to-end QoS and fitness of solutions
5: *pop* ← Perform non-dominated sort (*pop*)
6: *pop* ← *LADMF* (*Input*)
7: **While** (*gen* ≠ *maxgen*)
8: {
9: *pop* ← tournament selection (*pop*)
10: *pop* ← Crossover (*pop*)
11: *pop* ← Perform non-dominated sort (*pop*)
12: *child_pop* ← Mutation (*pop*)
13: *combination_pop* ← *pop* + *child_pop*
14: *combination_pop* ←Perform non dominated sort (*combination_pop*)
15: *pop* ← replacement (*combination_pop*)
16: **endWhile**
17: }

Algorithm 3 INSGA Crossover operation

Input: *pop*
Ouput: *Child*
1: **For**(*i* = 1 to *popsize*)
2: {
3: Randomly pick *Parent1* and *Parent 2* from *pop*
4: Compute Average latency A_L of *Parent 1* and *Parent 2*
5: *index1* ← Find cut point of *Parent 1* with poorest latency
6: *index2* ← Find cut point of *Parent 2* with poorest latency
7: [*Child 1*, *Child 2*] ← Crossover genes for each parent around *index 1* and *index 2*
8: [*Child 1*, *Child 2*] ← Determine end-to-end QoS and fitness of children
9: *Child* ←Add *Child 1* and *Child 2* in the child population.
10: **endFor**
11: }

Algorithm 4 INSGA Mutation operation

Input: *pop*
Ouput: *Child*
1: **For**(*i* = 1 to *popsize*)
2: {
3: Compute *Dist. Index* of *pop(i)* according to Equation (18)
4: *Child(i)* ← Mutate genes of *pop(i)* according to *DI*
5: *Child(i)* ← Determine end-to-end QoS and fitness of child
6: **endFor**
7: }

IV. EVALUATION

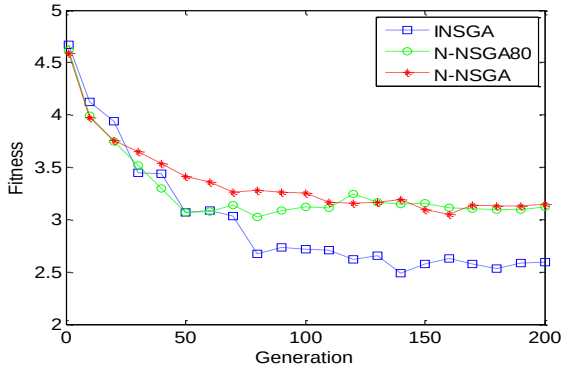
A. Setup

Experiments were run on a machine with Intel Core i7 CPU (3.8GHz) and with 8GB memory. All the algorithms and experiments are implemented in MATLAB 2013. A cloud network of BDS is simulated using planet lab meridian dataset [7] to provide RTT measurements between BDS. The dataset is chosen because it is expensive to implement a physically large cloud environment. The dataset contains symmetric round trip time (RTT) measurements between 1740 peer-to-peer nodes. Also, a test workflow is generated and will be used to evaluate INSGA algorithm. In the workflow, a set of thirteen tasks (t_1 to t_{13}) is defined. For each task, it is assume that each service group has equal number of candidate services for the sake of simplicity. The experiment is performed with 20 candidate services per task to simulate a large BDS cloud network.

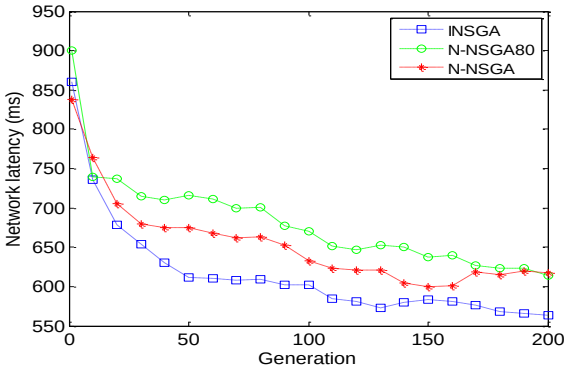
B. Results and Discussion

To demonstrate the efficiency of INSGA, its fitness latency and population diversity are compared against other traditional algorithms such as Particle swarm optimization (PSO) [26] and Genetic algorithms N-NSGA [25] and S-NSGA [24] in different environmental situations such as variations in number of tasks, candidate services and distribution index. Given the probabilistic nature of the test algorithms, each algorithm is run 50 times to obtain average values for fitness, latency and standard deviation which is often used to measure diversity of population.

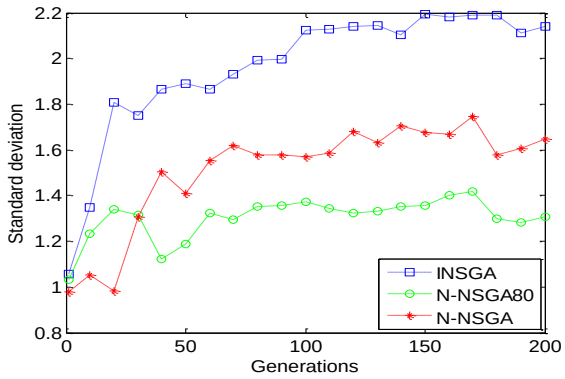
a) Impact of Distribution Index: In this experiment, an evaluation is done to determine the impact of distribution index on average fitness and population diversity of composite services. Here, the population size and maximum generation are set as 200 with network size of 260 services. In Fig. 10 (a) (b) and (c), it is observed that INSGA finds solutions with better fitness, latency diversity than N-NSGA and N-NSGA80. INSGA also avoids trapping in local optima while converging after 140 generations. This result shows that improvements in fitness, latency and population spread can be attributed to the proposed mutation and crossover operators.



(a) Graph indicating effect of distribution index on fitness



(b) Graph showing effect of distribution index on latency



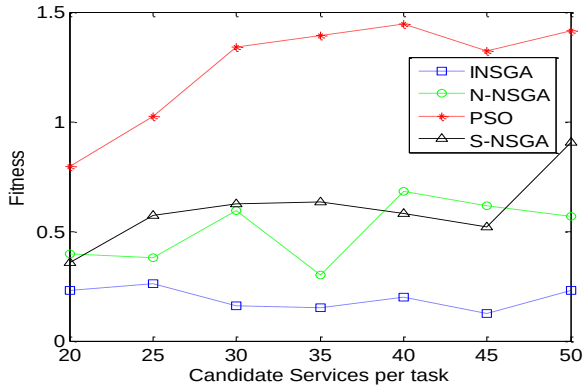
(c) Graph showing effect of distribution index on population diversity

Fig. 10. Plot of Distribution index against fitness, latency and diversity of population

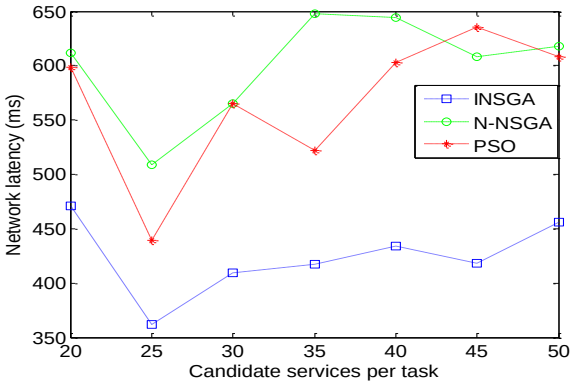
b) Size of Candidate Service per Task

In this experiment, the number of candidate services per task is increased from 20 to 50 and evaluate the impact on network latency, fitness, computation time and standard deviation of population. In Fig. 11(a) and (b), it is noticed that an increase in size of candidate services may ultimately lead to better quality solutions for all test algorithms with the exception of PSO whose quality worsens. It can also be seen

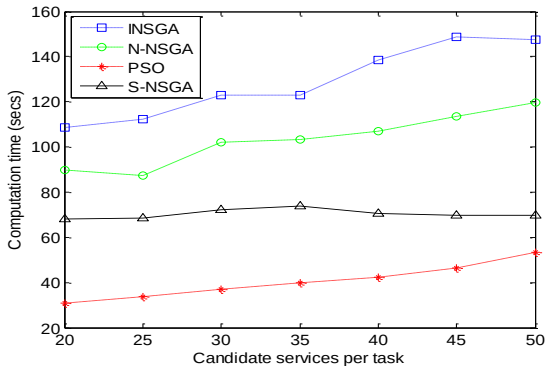
that INSGA finds solutions with the best fitness value and network latency when compared to the other algorithms. In Fig. 11(c) the computation times of all four algorithms are compared. It is observe that only INSGA has the highest computation time. This is as a result of the computational overhead generated by the network model. PSO has the lowest computation time which is about one third of INSGA's time. Also computation times for both N-NSGA, PSO and INSGA increase slightly with the number of candidate services except for S-NSGA whose computation time remains largely unchanged. Fig. 11(d) shows that increasing candidate service size doesn't influence the diversity of population for N-NSGA, PSO and S-NSGA. But in INSGA, standard deviation is improved slightly. Also, PSO shows the worst standard deviation while INSGA has the best standard deviation amongst the test algorithms.



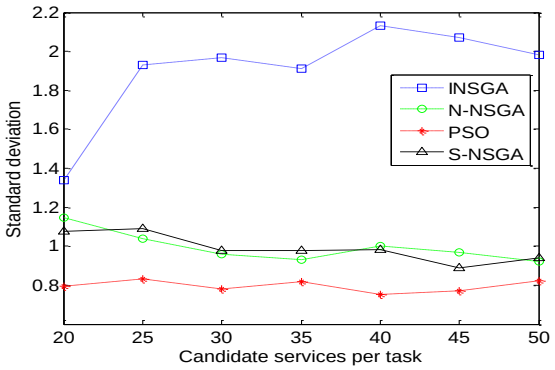
(a) Graph showing impact of number of candidate services on fitness



(b) Effect of number of candidate services on network latency



(c) Effect of number of candidate services per task on computation time

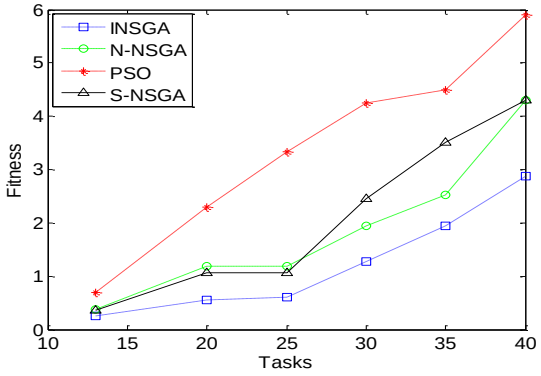


(d) Effect of number of candidate services per task on population diversity

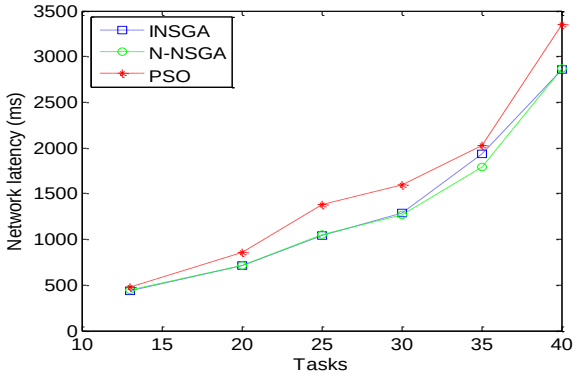
Fig. 11. Plot of candidate service size against fitness, network latency, computation time and standard deviation

c) Size of Tasks

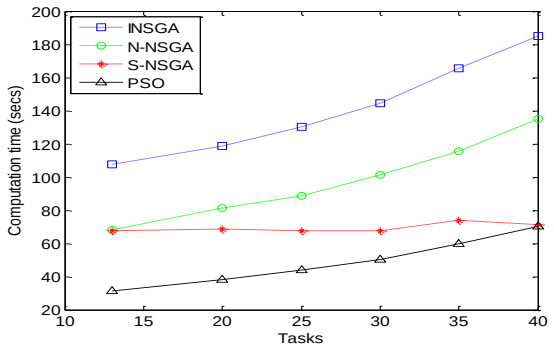
In this experiment the number of tasks are varied from 13 to 40 then the impact of fitness, network latency, computation time and standard deviation on the algorithms are determined. In Fig. 12 (a) and (b), it is observed that quality of fitness and network latency degrades with size of tasks for all test algorithms. INSGA is seen to produce the best quality solutions in terms of fitness and latency (tied with N-NSGA) while PSO produces worst quality of solutions. In Fig. 12 (c) a pattern similar to Fig. 11 (c) is observed, the only difference noticed is that computation time peaks at higher values when compared to graph in Fig. 12 (c). Lastly Fig. 12 (d) shows that population diversity increases linearly with size of tasks.



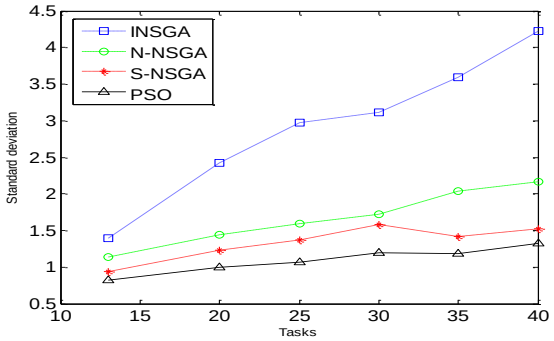
(a) Graph showing impact of number of tasks on fitness



(b) Graph showing impact of number of tasks on network latency



(c) Effect of number of tasks on computation time



(d) Effect of number of candidate services per task on standard deviation

Fig. 12. Plot of size of task against fitness, network latency, computation time and standard deviation

V. CONCLUSION

In this paper a novel approach to network-aware and QoS based service composition in the cloud is presented. Contrary to current works, this study separates QoS of network from service QoS. It consists of a network model which is composed of a novel network coordinate system called LADMF. LADMF uses matrix factorization to estimate the network latency (Round trip time) between BDS on the cloud. LADMF uses learning automata to encode service positional coordinates with additional learning parameters. This way the estimation process becomes decentralized where every service governs its own path to latency estimation. The latency information is then passed to a novel service composition algorithm based on non-dominated sort genetic algorithm called INSGA.

The aim of INSGA is to multi-objectively optimize cost, response time execution time and network latency QoS. INSGA uses a custom crossover and mutation operator. The crossover operator non-randomly picks two cut points where average latency is maximum while the mutation operator varies distribution index as a function of crowding distance and network latency. When compared with other state of the art service composition algorithms, results show that INSGA finds better quality solutions in terms of fitness, network latency and global search ability as indicated by its standard deviation.

REFERENCES

[1] Lifeng, Ai; "QoS-aware Web Service Composition Using Genetic Algorithms," PhD Thesis Queensland University of Technology, USA on, vol., no., pp.30, 2011.

[2] Adrian, K.; Fuyuki I.; Shinichi Honiden;,"Towards network-aware service composition in the cloud," In *Proceedings of the 21st international conference on World Wide Web* (WWW '12). ACM, New York, NY, USA, on, vol., no., pp.959-968, 2012.

[3] Jingwen Jin; Jin Liang; Jingyi Jin; Nahrstedt, K., "Large-Scale QoS-Aware Service-Oriented Networking with a Clustering-Based Approach,"*Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, vol., no., pp.522, 528, 13-16 Aug. 2007.

[4] Jin Xiao; Boutaba, R., "QoS-aware service composition in large scale multi-domain networks," *Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on*, vol., no., pp.397, 410, 15-19 May 2005.

[5] Frank Dabek; Russ Cox, frrans Kaashoek;"Vivaldi: A Decentralized Network Coordinate System," *ACM SIGCOMM '04 NY USA on*, vol., no., pp.15-26, 2004.

[6] Damien Saucez; "Securing Network Coordinate Systems," *in Master thesis: Université catholique de Louvain on*, vol.,no., pp.1-110, June 2007.

[7] Wong, B.; Slivkins, A.; Sirer, E.; "Meridian: A lightweight network location service without virtual coordinates," *In: Proc. the ACM SIGCOMM.*, vol., no., pp., 2005).

[8] Casati, F.; Georgakopoulos, D.;,Proceedings of the international workshop on Technologies for E-Services Roma, Italy on, vol., no., pp., September 2001.

[9] Tsur, S.; Abiteboul, S.; Agrawal, S.; Dayal, U., Klein, J; Weikum, G.;,"Are web Services the Next Revolution in e-commerce?," *Proceedings of the International Conference on very large databases on*, vol., no., pp. 614-617, September 2001.

[10] Bonet Blai;," Learning Depth-First Search: A Unified Approach to Heuristic Search in Deterministic and Non-Deterministic Settings, and its application to MDPs," *In Proceedings of ICAPS™, pp.3-23 2006.*

[11] Yan Gao; Jun Na; Bin Zhang; Lei Yang; Qiang Gong;, "Optimal Web Services Selection Using Dynamic Programming," *Computers and Communications, 2006. ISCC '06. Proceedings. 11th IEEE Symposium on*, vol., no., pp. 365- 370, 26-29 June 2006.

[12] HaiTao Song; Yanming Sun; Yingyu Yin; Shixiong Zheng;, "Dynamic Weaving of Security Aspects in Service Composition," *Service-Oriented System Engineering, 2006. SOSE '06. Second IEEE International Workshop*, vol., no., pp.189-196, Oct. 2006.

[13] Yoo, J.J.-W.; Kumara, S.; Dongwon Lee; Seog-Chan Oh; , "A Web Service Composition Framework Using Integer Programming with Non-functional Objectives and Constraints," *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on* , vol., no., pp.347-350, 21-24 July 2008.

[14] Zhifeng Gu; Bin Xu; Juanzi Li; "Inheritance-Aware Document-Driven Service Composition," E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services, 2007. CEC/EEE 2007. The 9th IEEE International Conference on, vol., no., pp.513-516, 23-26 July 2007.

[15] Chen Ming; Wang Zhen wu;,"An Approach for Web Service Compositon Based on QoS and Discrete Particle Swarm Optimization," Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2007, Eight ACIS International Conference on, vol.2, no., pp. 37-41, August 2007.

[16] Adrian Klein; Fuyuki Ishikawa; Shinichi Honiden;,"Towards network-aware service composition in the cloud," In *Proceedings of the 21st international conference on World Wide Web* (WWW '12) on, vol., no., pp.959-968, 2012.

[17] Ng, T.S.E; Zhang, H.A.;'"A Network Positioning system for the Internet," *in Proc USENIX ATL, on*, vol., pp., 2004

[18] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. An Approach for QoS-aware Service composition based on Genetic Algorithms. In GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, pages 1069–1075, New York, NY, USA, 2005. ACM

[19] Yongjun Liao; Wei Du; Geurts, P.; Leduc, G., "DMFSGD: A Decentralized Matrix Factorization Algorithm for Network Distance

Prediction," *Networking, IEEE/ACM Transactions on* , vol.21, no.5, pp.1511,1524, Oct. 2013

[20] Rony Kay; "Pragmatic Network Latency Engineering Fundamental Facts and Analysis," *cPacket Networks on* vol., no., pp.1-13, 2009

[21] Yongjun Liao; Geurts, P.; Leduc, G., "Network Distance Prediction Based on Decentralized Matrix Factorization," *Lecture Notes in Computer Science Springer* , vol.6091, no., pp.15-26, 2010

[22] Chen, S., Joshi, K., Hiltunen, M., Schlichting, R., W. Sanders; "Link gradients: Predicting the impact of network latency," on multi-tier applications. *In IEEE INFOCOM*, vol., no., pp., 2009.

[23] Mohammad Hamdan; "The Distribution Index of Polynomial Mutation for Evolutionary Multiobjective Optimisation Algorithms: An Experimental Study," *Yarmouk University*,vol.,no.,pp.1-5,2012.

[24] L. Li; P. Yang; L. Ou; Z. Zhang; P.Cheng;"Genetic Algorithm-Based Multi-objective Optimisation from QoS-Aware Web Services Composition," *In Springer Knowledge Science, Engineering and Management* vol.,6291,no.,pp.549-554, 2010.

[25] S. Umar; S. Ghazanfar; E. Gregory;."Network Aware Composition for Internet of Thing Services," *In Transactions on Networks and Communications*, vol.3, no.1, pp., 2015.

[26] H. Rezaie; N. NematBaksh; F.Mardukhi;"A Multi-Objective Particle Swarm Optimization for Web Service Composition" *In Springer Journal for Communications in Computer and Information Computer Science*, vol.88,no.,pp.112-122, 2010.

[27] M. Chen; T. Tan; J. Sun; Y. Liu; J. Pang; X. Li;"Verification of functional and Non-functional Requirements of Web Service Composition," *Singapore University of Technology and Design*;vol., no., pp. 1-15, 2014.

[28] Jaeger, M.C.; Rojec-Goldmann, G.; Muhl, G., "QoS aggregation for Web service composition using workflow patterns," *Enterprise Distributed Object Computing Conference, 2004. EDOC 2004. Proceedings. Eighth IEEE International* , vol., no., pp.149,159, 20-24 Sept. 2004.

[29] X. Zhao; P. Huang; T. Liu; X. Li; "A Hybrid Clonal Selection Algorithm For Quality of service-aware Web Service Selection Problem." *International Journal Innovative Computing and Information Control IJICIC*, vol.8, no.12, pp. 8527-8544, 2012.

[30] Jennings, Roger. *Cloud Computing with the Windows Azure Platform*. John Wiley & Sons, vol., no., pp., 2010.

[31] H. Keqiang, A. Fisher, L. Wang, A. Gember, A. Akella, T.Ristenpart; "Next stop, the cloud: Understanding Modern Web Service Deployment in EC2 and Azure." *Proceedings of the 2013 conference on Internet measurement conference*. ACM, vol., no., pp.177-190, 2013.

[32] T. Magedanz, N. Blum, and S. Dutkowski; "Evolution of SOA concepts in telecommunications," *IEEE Computer Magazine*, vol. 40, no. 11, pp. 46–50, 2007