TNC Transactions on Networks and Communications
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
VOLUME 3, ISSUE 1
ISSN: 2054 - 7420

Network-aware Composition for Internet of Thing Services

[1]Umar Shehu, [2]Ghazanfar Ali Safdar and [3]Gregory Epiphaniou
Department of Computer Science and Technology, University of Bedfordshire, UK
[1]umar.Shehu@beds.ac.uk; [2]Ghazanfar.Safdar@beds.ac.uk; [3]Epiphaniou@gmail.com

ABSTRACT

To enhance the adoption of Internet of Things (IoT) philosophy for the internet, research into IoT service composition has gathered momentum. In a distributed IoT environment, identifying IoT service among a set of similar service offerings that meets both functional and performance requirements of an IoT application has become important. However, the performance of a service cannot be guaranteed. Therefore service's QoS and network characteristics are required to aggregate IoT services. Most existing composition approaches only consider non-network related QoS properties at the application tier. However they do not consider the network parameters such as network latency at the application level in selection and composition of services. Therefore we propose two evolutionary algorithms for IoT service composition that consider not only QoS but also network latency at the IoT application layer. The algorithms are discussed and results of evaluation are presented. The results indicate that our algorithms are efficient in finding QoS optimal and low latency solutions.

Keywords: Internet of Things, Service Composition, QoS, Network latency, Composite service, Evolutionary Algorithm

1 Introduction

Internet of Things (IoT) envisions the internet as a set of interconnected objects. Objects refer to physical smart devices that utilize computing resources such as CPU, memory and network capabilities [1] in exposing their functionalities via the internet to the outside world. Recently, research studies [10][8][12][9] have attempted to map IoT as a service-oriented framework where smart device functionalities are exposed as services. This allows for the development of loosely coupled IoT applications [2] in which services can be discovered and selected according to user requirement. Given the immense potential of applying service-oriented concepts to IoT domain, several challenges have been identified. When single service is not sufficient in meeting user requirements, services will need to be combined into composite service that provide more complex capabilities that meet user needs. IoT service selection usually involves comparing service QoS scores. QoS represents the non-functional aspects of a service such as price, availability, reputation, response time, etc. It serves as a criteria that differentiates services offering similar capability. Once a service is selected based on functionality, composition can be carried out to find the right combination of services that yield composite service with optimal QoS. As IoT applications are becoming more distributed over the internet, network latency has become important in determining the performance of composite services. This is especially evident in IoT applications that require some form of data streaming. Data streaming is one of the most significant requirement of an IoT network [1] and is heavily dependent on network latency. Network latency, otherwise known as round trip time (RTT), is defined as the amount of time required for network packets to take a round

DOI: 10.14738/tnc.31.961
Publication Date: 19nd February 2015
URL: http://dx.doi.org/10.14738/tnc.31.961

trip from a source node to destination node [17]. For instance a data-intensive IoT application is presented in Figure 1. The purpose of the application is to provide video or audio feeds from a variety of smart devices like an internet-enabled surveillance camera or a Wifi-enabled push-to-talk (PTT) phone.

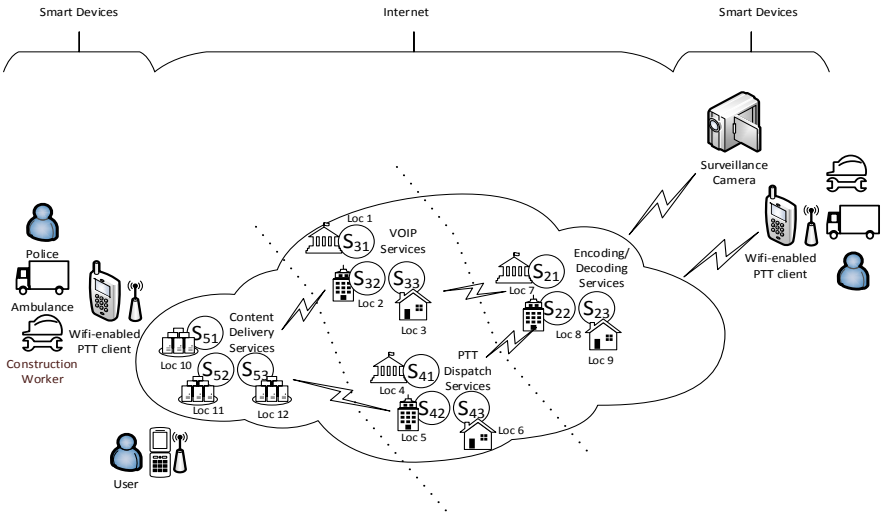

Figure 1: IoT services application

The application is composed of a variety of services deployed in different network locations on the internet and are participants of a composition process in order to provide much more advance features for the user. A typical scenario is when a police officer wants to view a video stream coming from a surveillance camera in a remote location on his mobile phone via a Wifi connection. As there exists many processes involved such as encoding/decoding, VoIP streaming and content delivery, such a scenario would involve the orchestration of different services that can perform each process. For example, alternative services (candidate services) like S31, S32 and S33 in Figure 1 can take care of VoIP streaming, while services S51, S52 and S53 representing different cloud-based content delivery networks (CDN) can handle content delivery to the police officer's smart phone. A simple composite services could be formed from integrating any of Encoding services with VoIP and CDN services. The problem now becomes how to integrate one service from each set of alternative services into a composite service in such a way that it satisfies the user's QoS and network requirement. By network requirement, we mean composition process should take into consideration the services that are closer to each other in terms of their network locations (as represented by inter-service RTT values). This will ensure optimum network performance of the application from the user's perspective.

In order to measure inter-service network latency, state of the art measurement tools could be deployed at the network layer. These tools function by measuring packet pings between all service nodes within a composition in order to obtain their network locations. The network locations are then fed to the application layer where the service composition process tries to obtain the end-to-end network latency for the composite service. Due to congestion and packet collisions, the tools are very slow in measuring RTT and therefore are not useful in situations where the IoT application requires data streaming or has strict time constraints such in Figure 1. As such, rather than measuring RTT, our proposed techniques estimate network latency QoS at the application level (i.e. during service composition process) so as to search for composite service that meets not only QoS

constraints, but also has near-optimal end-to-end network latency of its execution path. In other words, the optimal solution should have the best balance between optimal QoS score and network latency.

The problem of service composition has been described as NP-Hard Problem [18]. This is so because as the number of alternative services on the internet has increased, leading to a rise in number of possible composite services. This will also cause an exponential increase in the time it takes to find an optimal solution. To facilitate composition of IoT services which can be a time consuming process [15], research efforts have developed several algorithms capable of aggregating IoT services that contribute to optimum composite service and meets users' QoS constraints. [19] Tackle service composition in very large-scale IoT systems. They present an architecture that adapts composition process depending on the availability of constituent smart devices offering medical services such as ambulance, health insurance etc. The architecture considers the choreographic aspects of service composition. However it does not consider the QoS or network aspect of service composition. Another study [20] develop two probabilistic model for IoT service composition. The first model is based on finite state machine and deals with only the functional aspect of composition process while the second model is based on Markov Decision Process that handles service cost and reliability QoS properties. The study also falls short of considering Network-centric QoS. The most popular choice of algorithms for tackling QoS optimization of services are Evolutionary algorithms. Evolutionary algorithms are techniques that operate on the concepts of natural evolution [5]. They have shown great promise in tackling service composition problem because they add characteristics such as population diversity, reinforced learning, memory and adaptability to the composition process. They have also been shown to be more computationally efficient than other types of algorithms. Several evolutionary service composition techniques such as in [11] [13] [22] have been developed, although very few have been applied to the domain of IoT services. One such studies is in [21] which introduce a hybrid cooperative evolution algorithm that combines elements of Genetic and Particle Swarm Algorithms in searching for QoS-optimal composite services. The result is an algorithm that adapts to real-time data streaming from services running on smart devices. A similar study in [14] present a multi-objective approach to QoS-based service composition using a Genetic algorithm. Both approaches take service QoS into consideration, but once again ignore the network aspect of QoS.

Conclusively, recent works demonstrate good capability in finding non network-centric QoS optimal solutions, however they fail to consider the impact of network-related QoS parameter such as network latency on service composition at the application level. In contrast, we propose two evolutionary approaches to network-aware IoT service composition. Our approaches utilize a QoS model that is extended with a network model which efficiently estimates the RTT between services running on IoT devices. The network model consists of a decentralized network coordinate system for fast estimation of end-to-end RTT. This will ensure that RTT estimation process does not negatively impact the overall computation time for our approach and for the IoT application. We also propose novel network-aware Genetic and Particle swarm algorithms for searching compositions with both optimal QoS and optimal network latency. We then compare our approaches against each other and against other state of the art techniques and present the results of our experiment. The remainder of the paper is organized as follows. Section 2 formulates the service composition problem. Section 3 presents our network model and proposed techniques. Section 4 discusses the result of evaluation of our approaches. Section 5 concludes this paper.

2 Problem Formulation

Service composition borrows its concepts from workflow management systems [3] where a functionality is implemented as a task, and process of task execution follows specific workflow pattern which could be one of the many patterns such as sequence, parallel, loop, etc. The goal of service composition is to find a set of interconnected services (one per task) that contribute to the optimal composite service and meets user's need. With this in mind, the service composition problem is described as follows:

Given a set of n interconnected tasks that are needed to satisfy a user requirement,

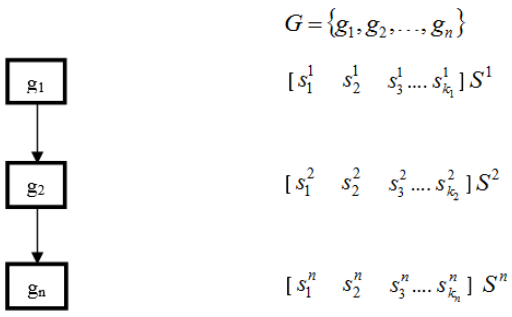

Figure 2: Arrangement of candidate services into tasks

Every task requires *k* number of similar services or candidate services that have the ability to complete the task,

$$S^i = \{s_1^i, s_2^i, \ldots, s_k^i\}, \ \forall i \in [1..n]$$

Where *i* identifies the class in which similar services are grouped according to their task g_i as seen in Figure 2. Only one candidate service s_j^i is selected and bound to its task. Once all tasks have been bound a composite service C is formed,

$$C = \{s_j^1, s_j^2, \ldots, s_j^n\}, \ \forall j \in [1..k]$$

Every service is assigned a number QoS values. In our work we consider four QoS objectives namely cost, response time, reputation and network latency. For a composite services, end-to-end QoS values are computed by aggregating the QoS values of constituent services depending of type of workflow. End-to-end cost, response time, reputation and network latency are calculated using (1) (2) (3) and (4) respectively;

$$Q_X(C) = \sum_{i=1}^{n} X(s_j^i) \tag{1}$$

$$Q_R(C) = \sum_{i=1}^{n} R(s_j^i) \tag{2}$$

$$Q_P(C) = \frac{\sum_{i=1}^{n} P(s_j^i)}{n} \tag{3}$$

$$Q_L(C) = \sum_{i=1}^{n} L_{s_j^{i+1}}^{s_j^i} \tag{4}$$

Where $L_{s_j^{i+1}}^{s_j^i}$ represents the round trip time (RTT) between candidate services S_j^i and S_j^{i+1}. Q_X, Q_R, Q_P and Q_L represent end-to-end QoS value for cost, response time, reputation and network latency. Also X, R, P and L represent service's QoS value for each QoS objective respectively. While

Normalization of cost, response time and reputation in the range [0 1] is achieved to obtain a fitness value (*F*) using (5). Where $Max_m(S^i)$ and $Min_m(S^i)$ represent maximum and minimum QoS values for service class *i.*

$$F_m(C) = \sum_{\substack{i=1 \\ m\in\langle X,R,P\rangle}}^{n} \left(\frac{Max_m(S^i) - Q_m(S_j^i)}{Max_m(S^i) - Min_m(S^i)} \right) \tag{5}$$

While fitness value for network latency is obtained using (6),

$$F_L(C) = \frac{Q_L(C)}{H} \tag{6}$$

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

The service composition problem becomes a multi-objective optimization problem where the aim is to search for composite services with optimal fitness values with respect to cost, response time, reputation and network latency, subject to the following constraints:

- One service should be selected for each task
- QoS boundary constraint:

$$\forall Q_X \in \left[q_X^{\min} \quad q_X^{\max} \right], \forall Q_R \in \left[q_R^{\min} \quad q_R^{\max} \right], \forall Q_P \in \left[q_P^{\min} \quad q_P^{\max} \right]$$

3 Network Model

In this work we adopt a network model that efficiently estimates network latency between service nodes in a network. The model consists of a network coordinate system [4] that computes round trip times (RTT) between service nodes. Ordinarily, RTT values are measured by sending network packets across the network and obtaining the time it takes them to reach their destination. Unfortunately this approach is not scalable and will cause computation overhead on the network. In comparison, our network coordinate system works by only measuring RTT from each service node to a small subset of neighbors. The measurements are then used to estimate un-measured RTT to other nodes. Here we adopt state of the art network coordinate system based on Matrix factorization [4]. Once RTT between all service nodes have been determined, the values are fed to our novel service composition approaches to establish network-awareness during composition process. The algorithm for network coordinate system is outlined in Figure 3.

1. Initialize environment parameters: *M, Z, maxIter*; Where *Z* is total number of service nodes and *maxIter* is maximum iteration no.
2. Measure RTT between each service node and a small number *M* of random neighbours;
3. For each node select *(Z-M)* neighbours with un-measured RTT;
4. Initialize dataset *D* with both measured and un-measured RTT.
5. For i = 1 to *maxIter*
6. U = rand (position_1); V = rand (position_2);
7. $X = U * V^T$;
8. $error = w(D - X)^2$;
9. If (*error* is not minimized)
10. Return;
11. End If;
12. End For;
12. Return

Figure 3: Outline of Network latency estimation algorithm

4 Evolutionary Algorithms for Network-Aware Service Composition

4.1 Evolutionary Particle Swarm Algorithm

Our first proposed approach is an Evolutionary Particle Swarm algorithm. Classic Particle Swarm optimization algorithm (PSO) [6] carries out optimization by encoding the problem using swarms of particles that iterate their velocity and position attributes until an optimal solution is found. However it is plagued with premature convergence, poor swarm diversity and lack of alternative optimal solutions. In order to avoid these problems, we adapt classic PSO with evolutionary concepts like multi-population and non-dominated sort ability in performing optimization. The resultant algorithm is called Evolutionary Particle Swarm or VPSO. It aims to search for a Pareto set of composite services that have optimal QoS.

Encoding

The algorithm encodes composite service as a particle array where each array element (e1, e2…en) represents a task that can be bound to any candidate service.

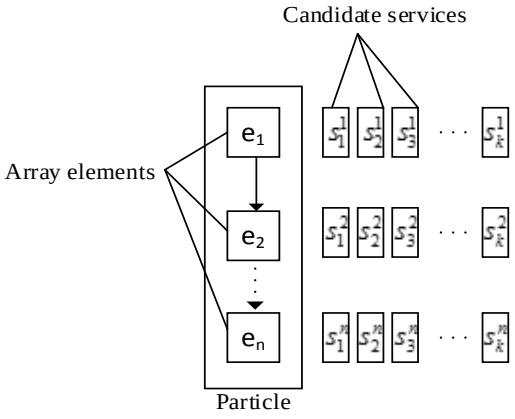

Figure 4: Encoding of a particle

Population initialization

It starts optimization process by initializing population of particles referred to as D population. This is achieved by arbitrarily selecting one candidate service for each task until all particles are initialized.

With the aid of RTT values earlier estimated by the network coordinate system, every particle is then assigned QoS values and a starting velocity and position.

Non-dominated Sorting and Multi-population creation

In the next step, the population is passed through a non-dominated sort process which involves sorting particles according to their fitness values and assigning fronts to each particle based on degree at which it dominates other individuals. For example, particle T_a dominates another particle T_b if its fitness value in all QoS objectives (cost, response time, reputation and network latency) are better than fitness values of T_b. Therefore T_a will be placed in a higher front than T_b. Once particles have been sorted, the top 25% of the population is placed in a second population called O population. Then crowding distance (CD) value is computed for each individual. This value determines the Euclidean distance between a particle and its neighbors. CD is an important value because it helps the algorithm to determine diversity or spread between individuals in the O population. The next stage involves creation of a third population known as N population consisting of individuals from D population with the best network latency value.

Updating Particle Velocity and Position

With the aid of first, second and third populations, new values for particle velocity and position are calculated with (7) and (8),

$$V_{j(new)}^{i} = wV_j^i + c_1 r_1 (N_j^i - D_j^i) + c_2 r_2 (O_j^i - D_j^i) \tag{7}$$

$$D_{j(new)}^{i} = D_j^i + V_{j(new)}^{i} \tag{8}$$

Where *V* is particle velocity; *w* is inertia weight, c_1 and c_2 represent constants; r_1 and r_2 are random numbers in range [0 1]; *N* is population of particles with best cost, response time and reputation; *O* denotes population of particles with best network latency; *D* represents initial population. Equations (7) and (8) force particle towards areas in search space where QoS objectives have good fitness values in terms of both QoS and Network latency as defined by their velocity. Where a particle's velocity is directly proportional to both the distance between the particle and particle with best network latency (i.e. $(N_j^i - D_j^i)$), and the distance between the particle and particle with best QoS score (i.e. $(O_j^i - D_j^i)$). Typically particles with lower velocities will move more slower than particles with higher velocities in the search space thereby keeping best particles (with low velocity) for participation in subsequent populations, while bad particles (with high velocity) are changed to new individuals. The effectiveness of equations (7) and (8) are demonstrated by result of experiment in the next section. VPSO is summarized in Figure 5.

```
1. Initialize PSO parameters: gen, pop_size, c1, c2
2. D = generate_population(gen, pop_size);
3.  O = non_dominated_sort(D);
4. For i = 1 to gen
5.          N = sort(O);
6.          For i = 1 to pop_size
7.                    V(new) = wV + c1r1(N – D)+c2r2(O – D);
8.                    D(new) = D + V(new);
9.                    D = compute_qos(D);
10.         End For
11.         O = non_dominated_sort(D);
12. End For
13. Return
```

Figure 5: Outline of VPSO algorithm

4.2 Network-aware Genetic Algorithm

In our second approach, we develop a novel Genetic algorithm based on non-dominated sort called N-Genetic algorithm or NGA. Non-dominated sort Genetic algorithms [16] are a class of Genetic algorithms that are capable of tackling problems of a multi-objective nature. They are also able to perform non-dominated sort operation in addition to standard operations such as crossover and mutation.

Encoding

NGA encodes composite service as a genome which consist of genes instead of elements as in the case of VPSO. A gene represents a candidate service bound to its task and can take an integer value.

Population Initialization

Similar to VPSO, NGA initiates optimization process by creating an initial population of individuals with initialized QoS values. Non-dominated sort operation is then performed on the initial population to find individuals with the best fitness values.

Crossover Operation

The individuals (parents) are subsequently placed in a mating pool and then subject to crossover operation. The operation involves intertwining sets of genes between any two parents. The type of operator used in our work is a single point crossover operator.

Mutation Operation

In order to integrate network awareness into NGA, we run k-means clustering algorithm [7] over our network model in order to effectively classify service nodes into separate clusters according to their RTT distance from other nodes. This way, services that are closer together in RTT are placed in the same cluster, while services that are further away are placed in different clusters. The clusters are then used by mutation operator to determine which genes within the same cluster that will be candidates for replacing the gene to be mutated. The operator will arbitrarily select only one gene among candidates available as seen in Figure 6. NGA algorithm is described in Figure 7.

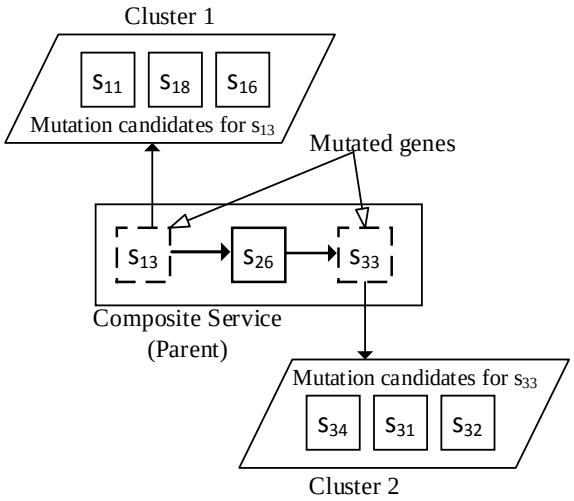

Figure 6: Mutation Operation of NGA

1. Initialize GA paramete: : *gen, pop_size.*
2. *pop* = generate_population(*gen, pop_size*);
3. *pop* = non_dominated_sort(*pop*);
4. For *i* = 1 to *gen*
5. *parent_pop* = tournament_selection*(pop);*
6. *parent_pop* = single_crossover_operation*(parent_pop);*
7. *parent_ pop* = non_dominated_sort(*parent_pop*);
8. *child_pop* = mutation_operation(*parent_pop*);
9. *combination_pop* = *pop* + *child_pop;*
10. *combination_pop* = non_dominated_sort(*combination_pop*);
11. *pop* = replacement (*combination_pop*);
12. End For
13. Return

Figure 7: Outline of NGA algorithm

5 Experiments and Analysis

In order the test the proposed algorithms VPSO and NGA, we present in Figure 8 a set of sequence workflows from our IoT application scenario in Figure 1 for sake of simplicity. It is expected that the results obtained will be similar irrespective of the sequence workflow used.

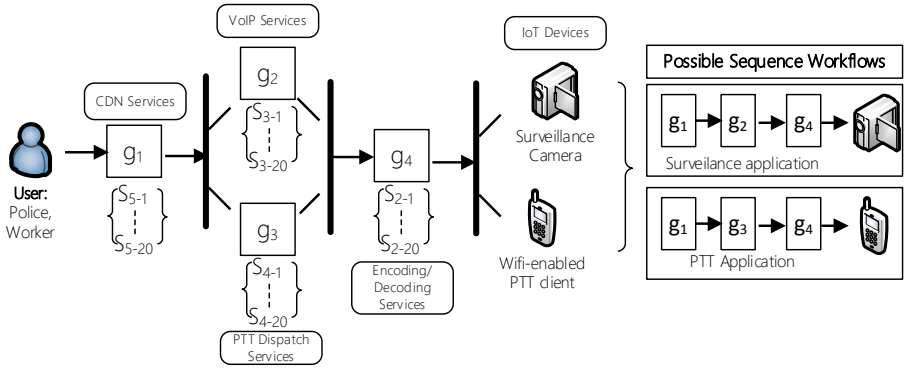

Figure 8: Experimental Service Composition Scenario

The experiment was executed in MATLAB 2013 running on Intel Core i7 (3.6GHz) CPU with 8GB RAM memory. Part of our experiment will aim to investigate how our algorithms cope with a large service

environment. This is achieved by expanding sequence workflow tasks up to 40 and candidate services per task up to 20.

In order to cater for RTT measurements between subset of nodes, we make use of meridian RTT dataset [23] which consist of a set of asymmetric RTT measurements between 1740 peer-to-peer nodes.

After the experiment is set up we compare our algorithms against state of the art approaches based on Genetic Algorithm (SGA) and Particle swarm Algorithm (SPSO). The results are presented in the following sub-sections.

5.1 Fitness

We run all algorithms over 200 generations and observe the fitness value during each generation. Figure 9 shows that both VPSO and NGA outperform SGA and SPSO in finding solutions with better fitness, with NGA demonstrating the ability to find the best solutions and SGA finding the worst solutions. In terms of convergence, VPSO and SPSO converge much earlier than NGA and SGA, This hampered their ability to find solutions as good as NGA.

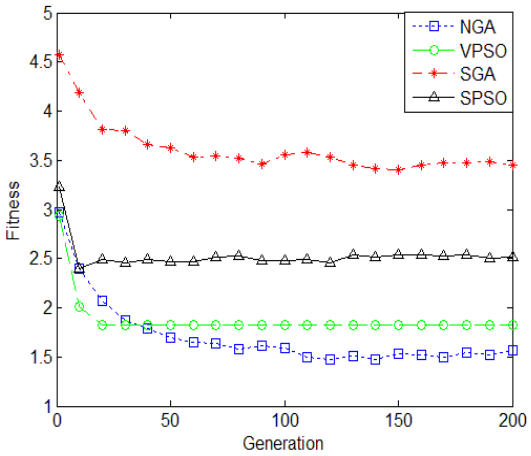

Figure 9: Fitness versus Generations

5.2 Network latency

Here, we compare the network latency of composite services for our algorithms. From Figure 10 it is discovered that VPSO is the best in finding low latency solutions followed by SPSO and NGA, leaving SGA with the worst set of solutions among the lot. VPSO's ability is attributed to uniquely identify and combine population with best network latency (N-population) with other populations. On the other hand, NGA is capable of searching for comparably low latency solutions thanks to its unique mutation operator which utilizes k-means clustering to find low latency solutions without compromising population diversity. The trend observed from SGA, which is a representation of how current evolutionary techniques behave with respect to network latency, shows that without network-awareness in service composition process optimal solutions may have good fitness for cost, response time and reputation but suffer from high network latency which can affect their performance.

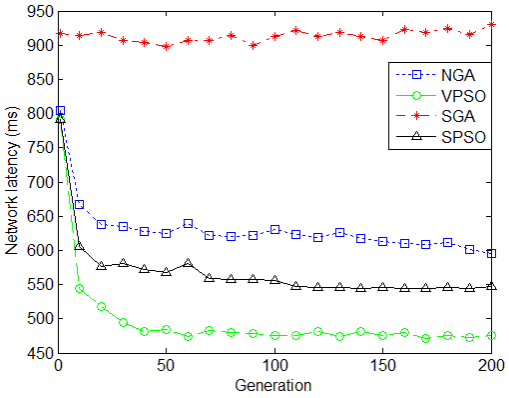

Figure 10: Network latency versus Generations

5.3 Standard deviation

In this experiment, we compare population diversities for the algorithms. Typically the better the population diversity as indicated by standard deviation, the less likely an algorithm will trap into local optimum. Figure 11 indicates that, as expected, VPSO and SPSO demonstrate the poorest diversity hence the reason they converge much earlier than the other two algorithms. NGA shows relatively better diversity while SGA shows the best diversity. This result also indicates that NGA's improved diversity when compared to VPSO is consequence of its novel mutation operator.

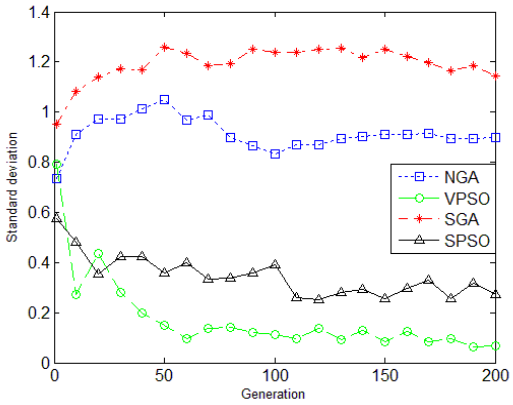

Figure 11: Standard Deviation versus Generations

5.4 Computation time

Table 1 shows that VPSO takes one third of NGA and SGA's computation times to achieve slightly similar results in terms of fitness and network latency. While it takes about half of SPSO's computation time to find significantly better solutions than SPSO. Therefore VPSO is the most efficient amongst the four algorithms, although at the cost of its population diversity. Table 2 indicates that VPSO has the worst fitness of the four algorithms despite obtaining the best network latency while SPSO has the best fitness with SGA showing the worst latency.

Table 1: Computation times (in seconds) of the four algorithms

SGA.	VPSO	NGA	SPSO
110.96s	**30.055s**	109.24s	66.54s

Table 2: Comparison of Algorithms' best results

ALGORITHM	BEST FITNESS	BEST NETWORK LATENCY	BEST STANDARD DEVIATION
NGA	**0.1793**	557.68ms	**1.2949**
VPSO	1.5585	**470ms**	0.9163
SGA	1.0310	574.22ms	1.2895
SPSO	0.6015	531.54ms	0.8326

6 Conclusion

In this paper we propose two evolutionary algorithms that perform network-aware IoT service composition. The aim of the algorithms is to search for composite services with optimum cost, response time, reputation and network latency QoS. The first algorithm is an Evolutionary Particle swarm Algorithm known as VPSO. The algorithm employs evolutionary techniques such as non-domination sort and multi-populations in its operation. The second approach is an N-Genetic Algorithm or NGA. NGA uses a k-means clustering algorithm to classify IoT services into clusters based on their RTT distance to other services and then tries to mutate individuals with other individuals in same cluster. From the results of experimentation, we observe that NGA outperforms in terms of quality of fitness, while VPSO outperforms in terms of computation speed. While both algorithms find low latency solutions, they fall short of population diversity when compared with state of the art Particle swarm and Genetic algorithms, this slight compromise is necessary in order to improve both fitness and network latency. Our results also demonstrate that VPSO is most efficient approach when compared to NGA.

REFERENCES

[1] Martinez-Julia, P.; Torroglosa Garcia, E.; Ortiz Murillo, J.; Skarmeta, AF., "Evaluating Video Streaming in Network Architectures for the Internet of Things," *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on* , vol., no., pp.411,415, 3-5 July 2013

[2] Guinard, D.; Trifa, V.; Karnouskos, S.; Spiess, P.; Savio, D., "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services," *Services Computing, IEEE Transactions on* , vol.3, no.3, pp.223,235, July-Sept. 2010

[3] 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

[4] 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

[5] Natallie Kokash; “An Introduction to Heuristic Algorithms,” *Department of Informatics and Telecommunications,* vol., no., pp.1-8, 2006.

[6] Miranda, V.; Fonseca, N., "EPSO-evolutionary particle swarm optimization, a new algorithm with applications in power systems," *Transmission and Distribution Conference and Exhibition 2002: Asia Pacific. IEEE/PES* , vol.2, no., pp.745,750 vol.2, 6-10 Oct. 2002

[7] Master, Chen Peng; Professor, Xu Guiqiong, "A brief study on clustering methods: Based on the k-means algorithm," *E -Business and E -Government (ICEE), 2011 International Conference on* , vol., no., pp.1,5, 6-8 May 2011

[8] Fenye Bao; Ing-Ray Chen, "Trust management for the internet of things and its application to service composition," *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a* , vol., no., pp.1,6, 25-28 June 2012

[9] Spiess, P.; Karnouskos, S.; Guinard, D.; Savio, D.; Baecker, O.; Souza, L.; Trifa, V., "SOA-Based Integration of the Internet of Things in Enterprise Services," *Web Services, 2009. ICWS 2009. IEEE International Conference on* , vol., no., pp.968,975, 6-10 July 2009

[10] Li L; Xinrui, L.; Xinyu, L.;”Cloud-Based Service Composition Architecture for Internet of Things,” *Communications in Computer and Information Science,* Springer, vol.312,no.,pp.559-564, 2012.

[11] 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.

[12] Edgardo Avilés-López, J Antonio García-Macías;”Mashing up the Internet of Things: a framework for Smart Environments,” *EURASIP Journal on Wireless Communications and Networking*, vol.2012, no.1, pp.1, 2012.

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

[14] Qian, L.; Runliang, D.; Fuzan, C.; Guofang, N.; “A QoS-oriented Web Service Composition Approach based on multi-population genetic algorithm for Internet of Things,” I*n International Journal of Computational Intelligence Systems*, vol.7, no.2, pp.26-34, 2014.

[15] Farhan, H.; “QoS Based Dynamic Web Service Composition,” In *International Journal of Computer Science and Information (IJCSIS),* vol., no., pp., 2010.

[16] Yijie, S.; Gongzhang, S.; “Improved NSGA-II Multi-objective Genetic Algorithm Based on Hybridization-encouraged Mechanism,” In *Chinese Journal of Aeronautics,* vol., no.21, pp.540-549, 2008.

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

[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] Kashif, D.; Amirhosein, T.; Romain, R.; Frank, E.; "Adaptable Service Composition For Very-large-scale Internet of Things Systems," *in Proceedings of the 8th Middleware Doctoral Symposium*, vol., no., pp.1-6, 2011.

[20] Lixing Li; Zhi Jin; Ge Li; Liwei Zheng; Qiang Wei, "Modeling and Analyzing the Reliability and Cost of Service Composition in the IoT: A Probabilistic Approach," *Web Services (ICWS), 2012 IEEE 19th International Conference on* , vol., no., pp.584,591, 24-29 June 2012.

[21] Liu, J; Yuxi, C; Xu, C; Jianli, D;, "A Cooperative Evolution for QoS-driven IoT Service Composition," *Automatika Journal for Control, Measurement, Electronics, Computing and Communications*, vol.55, no.4, pp.438-447, 2014.

[22] 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.

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