Fog computing task scheduling optimization based on multi-objective simplified swarm optimization

Wei-Chang Yeh\textsuperscript{1}, Chyh-Ming Lai\textsuperscript{2}, Kuan-Cheng Tseng\textsuperscript{1}* \\
\textsuperscript{1}Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 300, Taiwan \\
\textsuperscript{2}Institute of Resources Management and Decision Science, Management College, National Defense University, Taipei 112, Taiwan \\

* Corresponding author: tseng17@gapp.nthu.edu.tw

Abstract. With the dramatic growth of data volume, the cloud computing structure has faced a severe difficulty, the latency. To deal with this problem, researchers have proposed the fog computing structure, which can successfully release the computation loads from one datacenter of cloud to multiple local fog devices. Hence, the tasks will be processed at the local fog device and avoid transmitting to datacenter which is not cost-effective, and the results can be transmitted to the users immediately. The main differences of task scheduling between cloud computing and fog computing are the processors specifications, such as processing rate, cost per unit time. The processing rate of processors in datacenter is fast but the cost per unit time is costly. However, the processing rate of fog devices is not as fast as datacenter but it is relatively cost-effective. Consequently, we are facing a multi-objective optimization problem. Hence, we adopted an elite MOSSO (Multi-Objective Simplified Swarm Optimization) with considering the characteristics of fog computing paradigm.

1. Introduction

1.1. Background

With the arrival of the era of Internet of Things (IoT), a huge amount of data will be generated from smart devices, e.g. mobile phones, automobiles, wearable devices...etc. It is estimated by IDC (International Data Corporation) that there will be 80 billion connected devices in 2025, helping generate 180 trillion gigabytes of new data that year. Cloud computing services, without a doubt, will suffer from this huge volume of data. Such enormous number of data is forcing the responding latency of cloud to be prolonged. Besides, many IoT applications is demanding either real-time or low latency \cite{1}. The traditional cloud computing is no longer be able to afford such a huge amount of data from plenty of IoT devices, and respond within an acceptable latency \cite{2}. Hence, fog computing, a new breed of computing structure, has born to decentralize the cloud computing structure and solve the latency problem. 

Fog computing was first well defined in by Flavio Bonomi, Rodolfo Milito, Jiang Zhu and Sateesh Addepalli in 2012. They defined the characteristics of fog computing paradigm and outlined the vision \cite{3} and it was defined as an intermediate computing power between the cloud and users \cite{2}, it helps the cloud to shoulder the burden of heavy computation loads from plenty of smart devices. In addition,
Fog computing is not replacing cloud computing but regarded as a complementary structure to cloud computing [4]. It extends the computational services offered by the cloud computing to the edge of the network [5]. Moreover, fog computing was defined as a scenario which usually consists of lots ubiquitous, heterogeneous, and decentralized devices communicate and potentially cooperate among them without the intervention of third parties [6]. Mung Chiang & Tao Zhang (2016) focused on the challenges and opportunities of fog, and they showed how fog computing solves the problems existing in IoT framework. Furthermore, they pointed out the End-to-End architectural tradeoffs problem which is a critical issue discussed in our study [7].

There are some features of fog computing which made it different from cloud computing, such as low latency and location awareness, wide-spread geographical distribution, mobility, very large number of nodes, predominant role of wireless access, strong presence of streaming, real time applications and heterogeneity [3]. Some of these features have changed some properties of task scheduling in fog computing. First and foremost, the low latency and location awareness facilitate the broker to assign the tasks to the nearby fog devices. Besides, a vast number of nodes made the problem large and complex. Last but not least, heterogeneity made the conflict between makespan and cost.

1.2. Motivation

To the best of our understanding, little did the past research in fog computing task scheduling problem have discussed about the relation between processing rate and cost per unit time of processor, which is the root cause of the tradeoff problem between makespan and cost. Hence, we present a simple example to demonstrate it in the rest of this section.

For instance, we feed a 12000MI (Million Instructions) task for each processor in 2 scenarios to show the relation. Scenario 1: if the growth of processing rate from processor 1 to processor 3 equal to the growth of cost per unit time, there exists no conflict as shown in Table 1.

Table 1. Scenario 1 configuration.

| Processors | 1   | 2   | 3   |
|------------|-----|-----|-----|
| Processing Rate | 1000 | 2000 | 4000 |
| Cost/s     | 2   | 4   | 8   |
| Makespan   | 12  | 6   | 3   |
| Total Cost | 24  | 24  | 24  |

In Table 1, we can reduce the makespan by selecting processor with higher processing rate and the cost will remain the same. We can obviously observe that the makespan and cost is not conflicted with each other.

However, there is a conflict in Scenario 2. In scenario 2, the growth of processing rate is lower than to the growth of cost per unit time which means it is less cost-effective to process a task on higher processing rate of processor. In Table 2, if we process the task on processor 1, it will take 12 seconds and cost 24 cents. Nevertheless, if we process the task on processor 3, then the makespan can be reduce to 3 seconds but the cost will rise to 54 cents.

Table 2. Scenario 2 configuration.

| Processors | 1   | 2   | 3   |
|------------|-----|-----|-----|
From the previous example, we found the tradeoff only happens when the growth of cost overwhelmed the growth of the processing rate. That is, if a job is in an emergence, we ought to spend much more for shorter makespan. In reality, if a task is a time sensitive event, the scheduler will assign it to the cloud for higher processing rate or it will violate Service Level Agreement (SLA) and the service provider will be penalized. In fact, the scenario is exactly the situation of fog computing. A study compared the processing cost and transmission cost between fog and cloud computing paradigms against different number of terminal nodes [8], and it shows that fog computing costs significantly less than cloud computing. Furthermore, the impact on cost reduction becomes more obvious when the number of terminal nodes rises.

1.3. Purpose

1.3.1. More efficient methodology. Due to the outstanding performance characteristic of SSO (e.g., efficient convergence, simplicity of parameter setting and flexibility), most of the SSO related algorithms, e.g. MOSSO, NSSSO, performed significantly well in discrete problems. With the same idea, the task scheduling problem in fog computing is a discrete problem. Hence, we can reasonably expect that MOSSO could perform better than other multi-objective algorithms, e.g. MOPSO, NSGA-II...etc. Moreover, from the past experience, an elite version of MOSSO usually performs significantly well in multi-objective optimization problem.

1.3.2. Find the root cause of tradeoff in this problem. Little did the past research in fog computing task scheduling problem have discussed about the reason that cause such tradeoff between makespan and cost, especially in fog computing paradigm. As a result, we provide a simple example explaining the root cause of tradeoff problem between makespan and cost.

2. Literature review

2.1. Task scheduling problem in fog computing
Deng, Lu, Lai & Luan (2016) considered a tradeoff problem between the power consumption and computation latency, and the author decomposed the problem into three sub-problems [9], the first sub-problem is to find an optimal compromise between the computation latency and power consumption. The author utilized a convex optimization [10]. The second sub-problem is to find an optimal compromise between the power consumption and the computation delay in the cloud computing. A nonlinear integer programming is applied for this sub-problem [11]. The third sub-problem is to minimize communication delay in WAN subsystem, which is considered as an assignment problem. Hence, a Hungarian method is applied [12]. However, this study utilized a centralized approach to performs the optimization process to reduce computation delay and power consumption, which is less appropriate with a fog computing infrastructure, due the fact that central node will easily face a performance bottleneck when allocating workloads, and it will degrade the system performance.

Due to the performance bottleneck of centralized optimization approach, meta heuristic approach is considered. Bitam, Zeadally & Mellouk (2018) proposed a new bio-inspired optimization approach called Bees Life Algorithm (BLA) to address the job scheduling problem in fog computing.

| Processing Rate | 1000 | 2000 | 4000 |
|-----------------|------|------|------|
| Cost/s          | 2    | 6    | 18   |
| Makespan        | 12   | 6    | 3    |
| Total Cost      | 24   | 36   | 54   |
environment. They decompose jobs into tasks and allocate tasks to different fog devices considering the CPU execution time and Allocated memory size. During the foraging step, they adopted greed local search process in order to reach the optimal solution among different solutions. Though proving a great performance in solving such large scale problem, it did not consider some special characteristics of fog computing paradigm, e.g., the tradeoff problem whether send the tasks to the cloud or not, the communication cost arises from the distance between two distinct fog devices, penalty result from violation of SLA.

HAN Kui-kui, XIE Zai-peng & LV Xin (2018) proposed an improved genetic algorithm for a mixed cloud and fog computing infrastructure. In this study, they considered the cost as a performance evaluation metric which consist of operation cost of virtual machines and penalty result from violating SLA. In this way, the impact of makespan is simultaneously considered when considering the penalty. However, this study didn’t consider the tradeoff problem of whether sending the tasks to the cloud or not. That is, this conflict between the makespan and cost is not considered.

2.2. Multi-objective algorithm in task scheduling problem
As far as we know, studies of task scheduling in fog computing rarely applied multi-objective algorithm. However, it is of great importance to find an optimal compromise between makespan and cost in a fog computing paradigm. Fieldsend & Singh (2002) proposed Multi-Objective Algorithm (MOA) to solve two conflict metrics problem, and a non-dominated tree is applied to determine the global best for each particle [13]. Colleo, Pulido, Lechuga (2004) proposed multi-objective particle swarm optimization. Unlike other proposals extended PSO to solve multi-objective optimization problems, their algorithm uses an external repository of particles that is later used by other particles to guide their own flight [14]. Zhou, Qu, Zhao, Suganthan and Zhang proposed a Multi-Objective Evolutionary Algorithm (MOEA) to solve the task scheduling problem in grid computing [15]. Liu, Luo, Zhang, Zhang and Li proposed multi-objective genetic algorithm to solve the task scheduling problem in cloud computing [16]. Jena proposed Task Scheduling multi-objective nested Particle Swarm Optimization (TSPSO) for task scheduling with two performance evaluation metrics, power consumption and cost [17]. Fard, Prodan, Barrionuevo and Fahringer proposed Multi-Objective List Scheduling (MOLS) for workflow applications scheduling in heterogeneous systems such as Grids and Clouds [18]. Based on MOLS, Dogan and Ozguner proposed Bi-objective Dynamic Level Scheduling algorithm (BDLS) to maximize the reliability and minimize execution time [19]. In our study, we also adopt MOSSO to deal with the task scheduling problem in fog computing.

2.3. MOSSO
MOSSO is a multi-objective optimization method originated from SSO (Simplified Swarm Optimization) which was proposed by Yeh (2009) [20]. SSO is a novel population-based stochastic optimization method. It is one of the swarm optimization approach, and its simplicity and efficiency have captured many scholars’ attention. It has proved to solve discrete problems efficiently in many studies [21].

In simplified swarm optimization algorithm, we set three parameters, $C_g$, $C_p$ and $C_w$ where $C_g > C_p > C_w$. The update mechanism of SSO is defined by Eq. (1):

$$x_{ij}^t = \begin{cases} x_{ij}^{t-1} & \text{if } \rho \in [0, C_w) \\ p_{ij}^{t-1} & \text{if } \rho \in [C_w, C_p) \\ g_j & \text{if } \rho \in [C_p, C_g) \\ x & \text{if } \rho \in [C_g, 1] \end{cases}$$

(1)
Note that $x'_{ij}$ is the $j^{th}$ variable of $i^{th}$ solution at iteration $t$, $\rho$ is a uniform random number within [0, 1]. $p_{ij}^{t-1}$ is $j^{th}$ the variable of $p_{best}$ (best $i^{th}$ solution among $t-1$ iterations), $g_{j}$ is $j^{th}$ the variable of $g_{best}$ (i.e. best solution among $t-1$ iterations) and $x$ is a random variable between the lower bound and the upper bound of the feasible solution space.

For each update process, $\rho$ is generated first. If $\rho$ is located in $[0, C_{w})$, the value of variable will maintain the same as last generation. If $\rho$ is located in $[C_{w}, C_{p})$, the value of the variable will be generated from $p_{best}$. If $\rho$ is located in $[C_{p}, C_{g})$, the value of the variable will be generated from $g_{best}$. Otherwise, a random value $x$, will be generated and replace the current variable.

3. Problem statement

3.1. System model

A cloud system is composed of the cloud server and multiple fog devices. Each of fog devices is located at different areas. Tasks are assigned to different processors, either fog devices or the cloud. Each fog device receive request from the users and upload the data information to cloud computing infrastructure. After receiving the information, the cloud server runs the optimization procedure to determine an assignment. Any data which was assigned to be processed locally will be processed by the local fog device. Otherwise, it will be uploaded to the cloud server and sent back after completion.

In this paper, we assume each task can be processed only on one processor. That is to say, the virtual machine migration problem is not considered in this study. Furthermore, one virtual machine can only process one task at once. (Figure 1)

![System model](image)

Figure 1. System model.

The encoding policy is based on the assignment of each task. Each dimension represents the destination of a task. For example, assume the total number of tasks is 6, and there is a solution (3, 4, 1, 5, 2, 3). It means 1st task is assigned to processor 3 and 2nd task is assigned to processor 4…etc.

3.2. Notations

3.2.1. Indexes and Coefficients. The definition of indexes and coefficients are listed in Table 3.

| Index | Definition |
|-------|------------|
| i     | Index of certain task. $i = 1, \ldots, N$, where is the number of total tasks. |
| j     | Index of certain fog device. $j = 1, \ldots, M$, where is the number of |

Table 3. Indexes and Coefficients.
Coef
ficient
T
i
i
th
task of set T.
P
j
j
th
processor of set P.
L
i
instructions of
i
th
task.
r
j
processing rates of
j
th
fog device.

3.2.2. Functions. The definition of each function is presented in Table 4.

| Function | Definition |
|----------|------------|
| n(P
j
) | Number of tasks allocated on
j
th
processor. |
| ET(T
i
, P
j
) | The execution time of
i
th
task conducted on
j
th
processor. |
| ECU(P
j
) | The execution cost of
j
th
processor per unit time. |
| TCU | The transmission cost per unit distance |

3.3. Mathematical model
We formulate our problem in a mathematical model. An objective function is defined to evaluate the quality of a solution and our goal is to minimize two performance evaluation metrics, makespan and cost as shown in Table 5.

| Objective function | Constraint |
|---------------------|------------|
| Minimize Objective Function = Makespan (2) | Makespan < Deadline (4) |
| Minimize Objective Function = Cost (3) | |

3.3.1. Makespan.

\[
\text{Makespan} = \max_{j=1}^{M} \text{Time}_{P_j} = \max_{j=1}^{M} \sum_{i=1}^{n(P_j)} ET(T_i, P_j)
\]

We calculate the execution time of
i
th
task processed
j
th
processor, and sum up for the total execution time of execution time of
j
th
processor, Time
j
. Then determine the longest execution time as Makespan.

3.3.2. Cost. The cost evaluation consists of three parts, execution cost, machine idle cost and data transmission cost.

Execution Cost
The execution cost is calculated by multiplying the cost per unit time of the \(j\)th processor by the \(Time_{P_j}\) which was mentioned in makespan.

\[
\text{processors idle cost} = \sum_{j=1}^{M} ECU(P_j) \times (\text{Makespan} - Time_{P_j})
\] (7)

The machine idle cost is calculated by multiplying the cost of each processor by its idle time.

\[
\sum_{j=1}^{M} TCU \times \text{Distance}\left(F_i, P_j\right)
\] (8)

When a task is sent away from local area, transmission cost should be considered. Specifically, the cost linearly raised with the increase of distance between two computational devices.

3.3.3. Constraint. Considering reality situation, each of the job (i.e., a set of tasks) has a predetermined deadline requirement which was defined in the SLA. If the makespan exceeded the deadline, a penalty will be charged by the user. Hence, we define the deadline constraint in Eq. (9):

\[
\text{Deadline constraint} \quad \text{Makespan} < \text{Deadline},
\] (9)

4. Results and analysis

In this chapter, we compare the elite MOSSO with MOPSO in different scenarios. We set 3 scenarios with 12, 20 and 30 fog devices, respectively. In section 1, the parameters setting is presented. Section 2 shows the results and discussion in different scenarios.

4.1. Parameter setting

Table 6 shows the parameters of elite MOSSO and MOPSO. In the elite version of MOSSO, we delete \(C_p\) and keep \(C_g\). We regard the solutions in archive set are gBests, that is, we replace the variable by the randomly picked solution from archive set if the random number is lower that \(C_g\).

| Parameter | Value |
|-----------|-------|
| MOSSO     |       |
| \(C_g\)   | 0.6   |
| \(C_w\)   | 0.8   |
| MOPSO     |       |
| \(w\)     | 0.5   |
| \(C_1\)   | 1     |
| \(C_2\)   | 1     |
| \(r_1\)   | [-1,1]|
| \(r_2\)   | [-1,1]|

In Table 7, the algorithm parameters are provided. For each problem scale, we set different generations for a better convergence.

| #of Particles | #of Generations | #of Runs |
|---------------|-----------------|----------|
| Small         | 100             | 300      | 30       |
| Medium        | 100             | 600      | 30       |
4.2. Results and discussion

In this section, we set three problem scale to compare to evaluate the performance of each algorithm, we select the non-dominated solutions from all the results of the algorithms as an estimated Pareto front. (Figure 2)

4.2.1. Scenario1 - Small scale. In scenario 1, a small scale problem is considered. We have 1 cloud service and 5 fog devices process 20 tasks. The performance measurement of scenario1 is shown in Table 8.

| Measuremen | IGD | SP |
|------------|-----|----|
| MOSSO      | 3.7031 | 0.9777 |
| MOPSO      | 8.2519 | 0.9892 |

(Figure 2) Pareto front comparison in small scale.

We can infer from Table 8 that MOSSO has a better convergence and stability than MOPSO in this problem. We attribute the result to the great performance of SSO in discrete problems. However, when it comes to the standard deviation of SP, MOPSO is better than MOSSO.

4.2.2. Scenario2 - Medium scale. In scenario 2, We have 1 cloud service and 11 fog devices process 30 tasks. The performance measurement of scenario2 is shown in Table 9.

| Measuremen | IGD | SP |
|------------|-----|----|
| MOSSO      | 34.992 | 0.9926 |
| MOPSO      | 38.633 | 0.9954 |

(Figure 2) Pareto front comparison in medium scale.
4.2.3. Scenario3 - Large scale. In scenario 3, We have 1 cloud service and 19 fog devices process 50 tasks. The performance measurement of senario3 is shown in Table 10.

Table 10. Performance measurement of senario3.

| Large | IGD | SP  |
|-------|-----|-----|
| Measuremen | π | σ   | π   | σ   |
| MOSSO | 40.119 | 23.910 | 0.9926 | 0.0066 |
| MOPSO | 69.021 | 29.437 | 0.9968 | 0.0015 |

In large scale problem, still, MOSSO converges faster than MOPSO. From the two previous scenarios, we could infer that MOSSO has a better global search ability, especially in discrete problems. Moreover, the advantage is strengthened with the growth of problem dimensions. (Figure 4)
5. Conclusion
Due to the distributed computing structure, fog computing are able to release the workload from the
datacenter. Every device could be seen as a fog device in this structure, and tasks could be either
processed via the surrounding fog devices or datacenter. Hence, the makespan and cost could be
reduced by different assignment.

To meet the fog computing paradigm, we take the transmission cost for consideration. It is because
that tasks could be sent away from current fog device to another fog device or datacenter for a lower
cost or makespan, so the transmission cost should be considered in this case. Another included cost
evaluator is processors idle cost, we not only included it for the realistic purpose but evaluate the
balance of the assignment. It is known that if all tasks are processed on one processor, the processor
will be heated and lead the efficiency to be lower and the power consumption to be higher. Hence, we
evaluate the processors idle cost to strengthen the balance of assignment.

In this paper, we adopted the elite version of MOSSO and the results shows that MOSSO has a faster
convergence ability in fog computing task scheduling problem. In the future, batch processing
strategies or dimension reduction methods can proposed or applied in fog computing paradigm to
accelerate the simulation time.

6. References
[1] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, M. Nemirovsky, 2014 *CAMAD* (2014)
[2] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, A. V. Vasilakos, CSUR, 50(3), 1–43
(2017)
[3] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, MCC, 12 (2012)
[4] C. Matt, BISE. 60 (4), 351–355 (2018)
[5] S. Bitam, S. Zeadally, A. Mellouk, EISs, 12(4), 373–397 (2017)
[6] L. M. Vaquero, L. Rodero-Merino, ACM, 44(5), 27–32(2014)
[7] M. Chiang, T. Zhang, IEEE IoT-J, 3(6), 854–864 (2016)
[8] S. Sarkar, S. Misra, IET Networks, 5(2), 23–29 (2016)
[9] R. Deng, R. Lu, C. Lai, T. H. Luan, H. Liang, IEEE IoT-J, 1(1) (2016)
[10] J. He, P. Cheng, L. Shi, J. Chen, Y. Sun, IEEE Trans. Autom. Control, 59(3), 660–675 (2014).
[11] D. Li, X. Sun, Springer Science & Business Media (2006)
[12] H. W. Kuhn, *The Hungarian Method for the Assignment Problem. 50 Years of Integer
Programming 1958-2008*, 29–47. (2009)
[13] J. E. Fieldsend, R. M. Everson, IEEE CEC (2005)
[14] C. A. C. Coello, G. T. Pulido, M. S. Lechuga, IEEE Trans. Evol. Comput, 8(3), 256–279
(2004)
[15] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, Q. SWARM EVOL COMPUT, 1(1),
32–49 (2011)
[16] Bagchi Tapan P., *Job scheduling model for cloud computing based on multi-objective genetic
algorithm*. 10(1): p. 134(2013)
[17] R. K. Jena, Procedia Comput Sci, 57, 1219–1227 (2015)
[18] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, T. Fahringer, *CCGrid* (2012).
[19] A. Dogan, COMPUT J., 48(3), 300–314 (2005).
[20] W.-C. Yeh, Inf. Sci, 197, 65–76(2012)
[21] W.-C. Yeh, KNOWL-BASED SYST, 64, 1–12 (2014)