KIPS Trans. Comp. and Comm. Sys.
Vol.10, No.11 pp.291~296
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2021.10.11.291

Energy-Efficient MEC Offloading Decision Algorithm in Industrial IoT Environments

Seolwon Koo† · YuJin Lim††

ABSTRACT

The development of the Internet of Things(IoT) requires large computational resources for tasks from numerous devices. Mobile Edge Computing(MEC) has attracted a lot of attention in the IoT environment because it provides computational resources geographically close to the devices. Task offloading to MEC servers is efficient for devices with limited battery life and computational capability. In this paper, we assumed an industrial IoT environment requiring high reliability. The complexity of optimization problem in industrial IoT environment with many devices and multiple MEC servers is very high. To solve this problem, the problem is divided into two. After selecting the MEC server considering the queue status of the MEC server, we propose an offloading decision algorithm that optimizes reliability and energy consumption using genetic algorithm. Through experiments, we analyze the performance of the proposed algorithm in terms of energy consumption and reliability.

Keywords : Mobile Edge Computing, Offloading, Genetic Algorithm, Industrial Internet of Things

산업용 IoT 환경에서 MEC 기반의 에너지 효율적인 오프로딩 결정 알고리즘

구 설 원† · 임 유 진††

요 약

사물인터넷의 발전으로 인하여 수많은 디바이스가 생겨나고, 큰 계산 자원을 요구하는 태스크들이 많이 발생된다. 이런 사물인터넷 환경에서 Mobile Edge Computing(MEC)는 지리적으로 사용자와 근접하여 서비스를 제공하기 때문에 많은 주목을 받고 있다. MEC 서버로의 태스크 오프로딩은 제한된 배터리 수명과 계산 능력을 갖고 있는 디바이스에게 효율적이다. 본 연구는 높은 신뢰도를 요구하는 산업용 IoT 환경을 가정하였다. 많은 디바이스와 여러 MEC 서버와 같은 환경으로 최적화에 있어서 복잡성이 발생한다. 이를 해결하기 위해 문제를 두 개로 나눠 해결한다. MEC 서버의 큐 상태를 고려하여 큐의 제한 길이를 충족하는 MEC 서버를 선택한 뒤, 유전 알고리즘을 사용하여 신뢰도를 고려하면서도 에너지 소모량을 최적화하는 오프로딩 결정 알고리즘을 제시한다. 본 연구는 실험을 통하여 에너지 소모량과 신뢰성 측면에서 제안 알고리즘의 성능이 효율적임을 분석하였다.

키워드 : 모바일 엣지 컴퓨팅, 오프로딩, 유전 알고리즘, 산업용 사물인터넷

1. 서 론

모바일 디바이스들은 사물인터넷(Internet of Things, IoT)의 발전으로 인해 다양한 서비스에서 이용되고 있다. 이러한 서비스 내에서 모바일 디바이스들은 많은 컴퓨팅 자원을 요구하는 태스크들을 생성하고 있다. 하지만 모바일 디바이스들의 제한된 배터리 수명과 계산 능력은 이런 태스크들을 지원하기에는 한계가 있다. 그래서 이 문제를 보완하기 위해서 클라우드 컴퓨팅 환경을 적용하였다. 이와 같은 모바일 디바이스들이 컴퓨팅 자원이 풍부한 클라우드 환경을 사용하기 위해서는 계산 오프로딩(computation offloading)이 요구된다. 계산 오프로딩은 모바일 디바이스에서 발생한 태스크를 클라우드 데이터 센터로 전송해 처리하는 것을 의미한다. 계산 오프로딩은 태스크를 어떻게 처리하느냐에 따라서 전체 오프로딩(full offloading)과 부분 오프로딩(partial offloading)으로 나누어진다. 전체 오프로딩이란 태스크 전체를 한 번에 클라우드 데이터 센터로 전송하여 처리하는 것을 의미한다. 부분 오프로딩은 태스크의 일부를 클라우드 데이터 센터로 전송하여 처리하고, 나머지 부분을 디바이스 자체 즉, 로컬에서 처리하는 것을 의미한다.

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2021R1F1A1047113).
※ 이 논문은 2021년 한국정보처리학회 춘계학술발표대회에서 "IIoT 시나리오에서 신뢰성을 보장하는 에너지 효율적인 오프로딩"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 숙명여자대학교 IT공학과 석・박사통합과정
†† 종신회원 : 숙명여자대학교 IT공학과 교수
Manuscript Received : July 6, 2021
First Revision : August 2, 2021
Accepted : August 23, 2021
* Corresponding Author : YuJin Lim(yujin91@sookmyung.ac.kr)

하지만 클라우드 데이터 센터와 모바일 디바이스는 지리적으로 거리가 멀기 때문에, 이는 제한적인 지연 시간을 요구하는 서비스에 적합하지 않고, 경우에 따라 로컬에서 처리하는 것보다 더 많은 전력을 소비할 수 있다[1]. 이 문제를 보완하기 위해서 클라우드 데이터 센터를 모바일 디바이스에 가깝게 이동시켜서 이용하는 Mobile Edge Computing(MEC) 개념이 등장하였다. 이로 인해 지연 시간과 모바일 디바이스의 에너지 소모량을 줄일 수 있다.

유비쿼터스 컴퓨팅 특징을 갖고 있는 IoT는 많은 도메인에서 사용되고 있고, 그중 산업 환경에서 이용되는 Industrial Internet of Things(IIoT)는 기존에 이용되는 IoT에 비해서 신뢰성 측면에서 엄격한 요구 사항을 갖고 있다. 낮은 신뢰성으로 인해 발생하는 손실액이 다른 도메인과 비교하였을 때 훨씬 크기 때문에 신뢰성 측면에서 엄격하다. 이러한 강력한 요구 사항 외에 IIoT 디바이스는 IoT 디바이스와 마찬가지로 배터리나 계산 능력에 있어서 제한적이다[2].

본 연구는 앞서 언급한 산업 환경 도메인에서 모바일 디바이스의 오프로딩 결정 기법을 제안한다. IIoT 환경에서 모바일 디바이스의 제한적인 배터리 수명과 계산 자원 문제를 해결하기 위해서 신뢰성과 모바일 디바이스의 에너지 소모량을 동시에 최적화한다. 즉, IIoT 환경에서 모바일 디바이스의 에너지 소모량을 최소화하면서 신뢰성을 보장하는 오프로딩 결정 기법이다.

본 연구는 다음과 같이 구성된다. 2장에서는 관련 연구, 3장에서는 시스템 모델을 설명하였다. 그리고 4장은 제안 알고리즘을 다뤘으며, 5장에서는 제안한 알고리즘의 성능을 비교하였다. 마지막 6장에서는 결론과 추후 연구를 다룬다.

2. 관련 연구

디바이스의 다양한 서비스의 요구 사항을 고려하면서 MEC를 이용한 최적의 오프로딩 결정에 대해 많은 연구가 진행되고 있다. 기존 오프로딩에 대한 연구는 다음과 같다. 지연 시간을 최소화하기 위해서 매칭 이론을 이용한 오프로딩 기법 연구[3]와 태스크의 종속성을 보장하기 위해서 Directed Acyclic Graph(DAG)를 이용하며, 지연 시간 제약과 신뢰성을 만족시키면서도 에너지 소모량을 최적화하는 오프로딩 기법[4]이 있다. 또한 태스크들 내에서 게임 이론을 이용하여 지연 시간을 보장하면서 시스템 비용을 최적화하는 연구[5]가 있다.

또한 메타 휴리스틱 알고리즘을 이용한 다양한 오프로딩과 관련된 연구들이 있는데 대표적인 메타 휴리스틱 알고리즘 예로는 입자 군집 최적화(Particle Swarm Optimization, PSO)와 유전 알고리즘이 있다. 우선 PSO는 새와 같은 집단 행동을 하는 특징에서 영감을 얻고, 개인과 무리의 지식을 공유하면서 목적함수를 최적화하는 기법이다. 개별 잠재해인 입자들은 위치와 속도와 같은 속성을 갖고, 이들은 반복을 통해서 탐색 공간을 움직이면서 적합도 함숫값이 큰 쪽으로 위치로 이동한다[6]. 그리고 유전 알고리즘은 다윈의 자연 진화 법칙의 적자생존을 토대로 하여, 생물이 환경에 적응하면서 진화하는 과정으로 만든 최적화 기법이다. 이러한 과정은 반복적으로 모집단을 선택, 교차, 돌연변이, 적합성 평가를 이용하여 업데이트한다. 여기서 염색체는 잠재적인 해결책이며, 여러 반복을 통해 최종적으로 전역적인 해결책을 찾을 수 있다[7]. PSO는 수렴이 빠르지만, 국소 최적화에 쉽게 빠질 수 있다는 단점이 있으며, 유전 알고리즘은 유전자 조작으로 인해서 다양성을 제공하여 전역 최적화의 효율을 높여준다는 특징이 있다[8]. 이와 같은 장점을 갖고 있는 유전 알고리즘을 이용한 오프로딩 연구는 다음과 같이 진행되고 있다. 유전 알고리즘을 통해서 전체 지연 시간을 줄이고 최적의 오프로딩 비율과 채널 대역폭, MEC 서버의 할당된 자원을 결정하는 연구[9]가 있다. 그뿐만 아니라 제약 지연 시간 및 클라우드렛의 리소스 제약과 같은 여러 제약 조건에서 지연 시간과 에너지 소모량, 클라우드렛의 자원 효율성을 유전 알고리즘을 이용하여 최적화한 연구도 있다[10].

대부분 오프로딩에 대한 연구는 에너지나 지연시간을 최적화하는 연구를 위주로 진행되어 왔다. 본 연구는 IIoT 환경에서의 오프로딩을 진행하기 때문에, 신뢰성과 에너지 소모량을 동시에 최적화하도록 유전 알고리즘을 이용하여 오프로딩 기법을 제안한다.

3. 시스템 모델

본 연구는 N개의 디바이스와 M개의 MEC 서버 환경을 가정하였고, 무선 네트워크로 연결되어 있다고 가정하였다. 각 디바이스의 집합과 MEC 서버의 집합은 $U=\{u_1,u_2,\cdots,u_N\}$, $S=\{s_1,s_2,\cdots,s_M\}$으로 표기하였다. 그리고 전체 오프로딩을 가정하였으며, 오프로딩 여부는 하나의 시간 슬롯마다 결정하도록 하였다[11]. 디바이스 $n(n\in N)$에서 발생된 태스크의 집합은 $T_n=\{t_{n,1},t_{n,2},\cdots,t_{n,k}\}$으로 표기하며, k는 태스크가 발생된 시간 슬롯을 나타낸다. 각 태스크 별로 $t_{n,k}=\{w_{n,k},c_{n,k},L_{req}\}$의 속성을 갖는다. $w_{n,k}$는 태스크의 크기(KB)이며, $c_{n,k}$는 태스크의 실행을 완료하기 위해 요구되는 프로세서 사이클(Megacycles)의 개수이다. L_{req}는 태스크의 지연시간 제약 조건이다. 통신 모델은 Shannon-Hartley식을 이용하였다. $R_{n,m}$은 디바이스 n과 MEC 서버 $m(m\in M)$사이의 데이터 전송률이다.

$$R_{n,m}=B\bullet\log_2\left(1+\frac{p_n^{tran}\bullet g_{n,m}}{\sigma^2}\right) \tag{1}$$

$$g_{n,m}=d_{n,m}^{-\alpha} \tag{2}$$

B는 전송 채널 대역폭이고, p_n^{tran}는 디바이스 n의 전송 전력이다. σ^2는 잡음 전력이며, $g_{n,m}$는 채널 이득이다. $d_{n,m}$은 디바이스 n과 MEC 서버 m 사이의 거리이며, α는 경로 손실 인자이다. 본 연구에서는 태스크의 입력 데이터보다 실행이 완료된 태스크의 출력 데이터가 훨씬 작기 때문에 출력 데이터의 전송은 무시하였다. 태스크 $t_{n,k}$를 로컬에서 처리할 때의 디바이스 n의 에너지 소모량(E_n^l)은 다음과 같다.

$$E_n^l(k)=\kappa\bullet(f_n)^2\bullet c_{n,k} \tag{3}$$

κ는 디바이스에 의해서 결정되는 상수이며, f_n는 디바이스 n의 계산 능력이다. 태스크 $t_{n,k}$를 MEC 서버 m으로 오프로딩 했을 때의 디바이스 n의 소모되는 에너지는 다음과 같다.

$$E_{n,m}^{off}(k) = p_n^{tran} \bullet \left\{ \frac{w_{n,k}}{R_{n,m}} \right\} \tag{4}$$

MEC 서버 m의 blocking 개수($b_{m,k}$)는 신뢰성의 척도가 된다. $b_{m,k}$는 서버 m의 큐가 이전에 도착한 태스크들로 모두 찼기 때문에, 해당 MEC 서버에 할당되었지만 큐에 저장되지 못하여, 실행되지 못한 태스크들의 개수를 의미한다. 여기서 발생한 blocking된 태스크들은 실행되지 못하기 때문에, 본 연구에서는 이를 최소화하는 기법을 제안한다. 서버 m의 blocking 개수는 다음과 같다.

$$b_{m,k} = \begin{cases} 0, & if q_{m,k} \le q_{max} \\ q_{m,k} - q_{max}, & otherwise \end{cases} \tag{5}$$

$q_{\max}$는 MEC 서버 내의 큐의 최대 길이이다. $b_{m,k}$는 시간 k에 MEC 서버 m로 오프로딩 되었지만, MEC 서버의 큐 제한으로 인해 blocking된 태스크의 개수를 의미한다. $q_{m,k}$는 시간 k에 MEC 서버 m으로 오프로딩 된 태스크 개수이다.

본 연구의 목적은 신뢰도를 최대화하면서 모바일 디바이스의 에너지를 최소화하는 오프로딩 결정 기법이다. 증가하는 디바이스와 시간에 따라 상황이 변하는 MEC 서버들을 한 번에 고려하여 최적의 해를 구하려면 매우 복합하다. 그래서 본 연구에서는 유전 알고리즘을 기반으로 하는 알고리즘을 제안한다.

4. 제안 알고리즘

본 연구는 디바이스들이 MEC 서버들의 위치 및 큐의 길이와 같은 정보를 기반으로 오프로딩 대상 MEC 서버를 선택한 다음 유전 알고리즘을 이용하여 오프로딩 여부를 최종 결정하도록 하였다. 하나의 시간 슬롯 내에서 제안 알고리즘의 순서도는 Fig. 1과 같으며, 디바이스들의 에너지 소모량과 신뢰성의 최적화를 목적으로 하는 MEC 서버로의 오프로딩 결정을 위한 알고리즘은 Table 1과 같다.

Fig. 1에서 따라, 먼저 태스트는 디바이스 내에서 발생하게 된다. MEC 서버들은 자신의 큐의 길이와 위치를 태스크들에게 알린다. 이 정보를 토대로 모든 태스크들은 MEC 서버를 선택한다. 그리고 선택된 MEC 서버로 태스크의 데이터 사이즈와 위치, 프로세서 사이클 개수와 같은 정보를 보낸다. 각 MEC 서버는 해당되는 태스크들로 유전 알고리즘을 실행하여 최적의 오프로딩 결정값을 얻을 수 있다. 이렇게 얻은 최적의 오프로딩 결정값을 디바이스로 전송하고, 그 값에 따라 디바이스는 오프로딩을 한다.

유전 알고리즘은 유전자(gene)이 0과 1로만 이루어진 이진 유전 알고리즘을 이용하였다. 하나의 염색체(chromosome)은 발생된 모든 태스크들의 오프로딩 여부의 배열을 나타낸다. 각

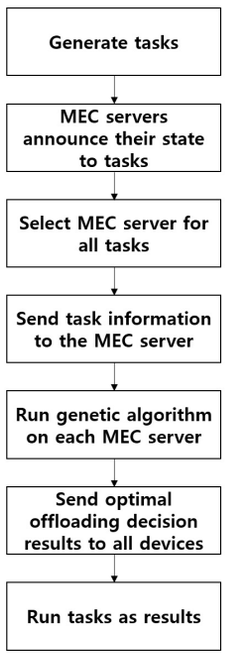

Fig. 1. Proposed Algorithm Flowchart in One Time Slot

Table 1. Offloading Decision Strategy using Genetic Algorithm

Algorithm 1. Offloading decision strategy using Genetic Algorithm

Input: initialization of random population P_0,
time iteration T_{iter},
probability of crossover C_p,
probability of mutation M_p,
maximum iteration I_{max}
Output: optimal offloading decision of tasks O
1 : i = 0
2 : for t = 1: T_{iter}:
3 : while i 〈 I_{max}:
4 : i = i+1
5 : calculate the fitness value of each individual
6 : select the individuals according to their *fitness*
7 : perform crossover with C_p
8 : perform mutation with M_p
9 : population = selected individuals after crossover and mutation
10 : end
11 : end

연산자는 다음과 같이 설정하였다. 선택(selection)은 룰렛 휠 선택을 이용하였고, 교배(crossover)는 single point를 사용하였다. 돌연변이(mutation)은 reverse를 이용하였다. 유전 알고리즘에서 사용된 적합도 함수는 다음과 같다.

$$fitness = 1/(\alpha \bullet E^{nor} + (1-\alpha) \bullet b^{nor}) \tag{6}$$

α는 0과 1사이의 가중치 값이고, b^{nor}는 MEC 서버에서 blocking된 태스크의 개수를 정규화한 값이며, E^{nor}는 하나의 시간 슬롯에서 실행된 태스크들의 에너지 소모량을 정규화한 값이다. 해당 식은 α를 조정하여 더 집중적으로 정규화 값을 최적화할 수 있다. 예를 들어 α가 0.8일 땐 에너지를 정규화하는 값을 더 집중적으로 최적화하는 것을 의미한다. 또

한 유전 알고리즘은 적합도 함수의 최대화하는 방향으로 구성되어있다. 그래서 에너지 소모량과 blocking 개수를 최소화하는 것이 목적이기 때문에, 정규화된 값에 가중치를 곱한 식을 Equation 6과 같이 설정해주어 제안 알고리즘이 적합도 함수가 최소화 될 수 있도록 하였다.

5. 실험 및 결과 분석

제안한 알고리즘 성능 평가를 위해서 실험을 진행하였고, 비주얼-트랙 응용 서비스를 가정하였다. 비주얼-트랙 응용 서비스란 컴퓨터 비전에서 많이 사용된다. 이는 2D나 3D에서 배경과 우리가 찾고자 하는 객체를 구분 지으면서 추출한다. 이렇게 추출한 객체를 추적하는데, IIoT 환경에서 발생되는 일들을 모니터링하고 추적하도록 본 응용 서비스를 이용하였다. IIoT 환경은 산업 환경이므로 생산품에 대한 추적과 모니터링들이 필수적이다. 실험을 위해 이용한 파라미터의 설정값은 Table 2와 같다.

제안한 알고리즘의 성능 평가를 위해 유전 알고리즘과 같은 메타 휴리스틱 기법인 입자 군집 최적화(Particle Swarm Optimization, PSO)를 활용하여 6가지의 기법을 비교하면서 실험을 진행하였다. 우선 첫 번째 기법은 거리를 기준으로 MEC 서버를 선택한 후 입자 군집 최적화를 이용하여 오프로딩을 결정하는 PSO(D) 기법이고, 두 번째 기법은 거리를 기준으로 MEC 서버를 선택한 후 유전 알고리즘을 이용한 proposed(D) 기법이다. 세 번째 기법은 MEC 서버의 큐 길이를 고려하여 MEC 서버를 선택한 후, 입자 군집 최적화를 이용한 PSO(Q) 기법이다. 네 번째 기법은 본 연구에서 제안하는 알고리즘인 proposed(Q)이다. 나머지 비교 기법으로는 모든 태스크를 로컬에서 처리하는 AL과 모든 태스크를 오프로딩 하는 AO이다.

Fig. 2와 Fig. 3의 비교를 통해서 태스크 발생 분포에 따른 성능 비교를 할 수 있다. Fig. 2는 태스크 발생 분포에 따른 MEC 서버 당 평균 blocking 개수이다. 태스크 발생 분포는 균등 분포와 푸아송 분포로 나누어 평균 디바이스 개수가 150개 일 때를 가정하여 진행하였다. 균등 분포에서는 제안하는 알고리즘인 proposed(Q)가 PSO(D)에 비해 약 80.1%, proposed(D)와는 약 44.4%, PSO(Q)와 비교하였을 때 약 33.3% 성능이 좋았다. 그리고 푸아송 분포에서는 제안하는 알고리즘이 PSO(D)에 비해 약 64.8%, proposed(D)와는 약 47.6%, PSO(Q)에 비해 약 39.9% 성능이 좋았다. 결과적으로 MEC 서버의 큐 길이를 고려했기 때문에 PSO(Q)와 proposed(Q) 기법들이 거리를 기반으로 한 다른 기법들에 비해 평균 blocking 수가 월등히 적었다. 국소 최적화에 빠질 수 있는 PSO의 단점으로 인해 proposed(Q)가 PSO(Q)에 비해 성능이 훨씬 좋았다. Fig. 3은 태스크 발생 분포에 따른 전체 디바이스 에너지 소모량이다. 균등 분포일 때 제안 알고리즘이 PSO(D)에 비해 약 4.4%, PSO(Q)에 비해 9.4% 좋았다. 또한 푸아송 분포에 있어서 제안 알고리즘은 PSO(D)와 비교하여 약 2.4% 좋은 성능을 보였고, proposed(D)와 비슷한 성능을 보였다. 앞서 PSO의 단점인 국소 최적화로 인한 문제로 인해서 디바이스의 에너지 소모량과 blocking 개수를 동시에 전역적으로 최적화를 하지 못하였다는 것을 알 수 있다. 결과적으로 제안 알고리즘은 에너지 소모량 측면에서는 비슷한 성능을 보였지만,

Table 2. Setting Values for Parameters

Parameters	Values
B	10MHz
σ^2	10^{-9}W
κ	10^{-27}
p_n^{tran}	0.5W
$w_{n,k}$	[150, 200]KB
$c_{n,k}$[11]	[285, 380]Megacycles
q_{max}	5
Frequency of the Device(f_n)	1GHz
Frequency of the MEC server(f_{mec})	10GHz
Latency requirement(L_{req})	0.5sec
k	100ms
Total time iteration(T_{iter})	4
Max Iteration(I_{max})	100
α	0.5
C_p	0.9
M_p	0.2
size of population	100
the number of MEC servers	9
Coverage[12]	150m

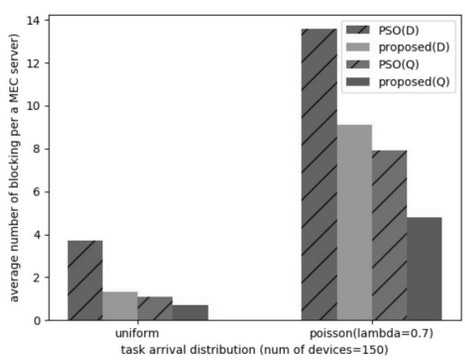

Fig. 2. Average Number of Blocking Per a MEC Server Depending on Task Arrival Distribution

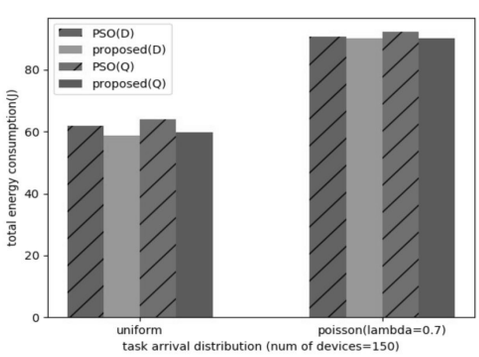

Fig. 3. Total Energy Consumption Depending on Task Arrival Distribution

blocking 수 측면에서 높은 성능을 보여 신뢰성 측면을 중시하는 IIoT 환경에 더 적합하다고 말할 수 있다.

Fig. 4, Fig. 5와 Fig. 6의 비교를 통해서 디바이스 개수 변화에 따라 성능 비교를 수행하였다. 태스크 발생 분포는 푸아송 분포를 이용하였으며 발생률은 0.7로 설정하였다. Fig. 4는 디바이스 개수 변화에 따른 MEC 서버 당 평균 blocking 개수이다. 디바이스 개수가 150개일 때, 제안 알고리즘은 AO와 비교하여 약 89.2%, PSO(D)와는 약 64.8% 정도 성능 향상을 보였다. 그리고 proposed(D)는 약 47.6%, PSO(Q)는 약 39.9%로 성능 차이를 보였다. 디바이스 개수가 250개일 때는 제안 알고리즘은 AO와 비교하여 약 31.6%, PSO(D)의 경우 약 17.8%로 성능이 향상됐다. proposed(D)와는 약 13.7%, PSO(Q)와 비교하였을 때는 약 7.9% 정도 향상된 성능을 보였다. 마지막으로 디바이스 개수가 350개일 때, 제안 알고리즘과 비교하였을 때 AO의 경우 약 14.4%, PSO(D)의 경우 약 3.4%로 좋은 성능을 보였다. 디바이스 수가 일정 이상 증가하게 되면 시스템 전체의 부하가 증가되어 각 기법 간 성능 차이가 감소되게 된다. 그러나 그렇지 않은 경우 (디바이스 개수 = 150)에는 제안 기법이 높은 성능을 보임을 알 수 있다. PSO(Q) 기법과 proposed(Q)가 다른 두 기법에 비해 성능이 좋은 이유는 MEC 서버를 선택하는데 있어서 큐의 상태를 먼저 고려하기 때문에, blocking되는 개수가 거리만으로 MEC 서버를 선택하는 PSO(D)와 proposed(D)에 비해 blocking 개수가 월등히 적다. 또한 국소 최적화에 빠질 수 있는 단점을 가진 PSO를 이용하였기 때문에 PSO(Q)는 proposed(Q)에 비해 성능이 좋지 않다. Fig. 5는 디바이스 개수 변화에 따른 전체 디바이스 에너지 소모량이다. 디바이스 수가 150개일 때 제안 알고리즘은 AL 기법에 비해 약 38.2% 나은 성능을 보였다. 디바이스 수가 250개일 때 제안 알고리즘은 AL에 비해 약 60.8% 좋은 성능을 보였다. PSO(Q), proposed(D)와 PSO(D)는 비슷한 성능을 보였다. 디바이스가 350개일 때, 제안 알고리즘은 AL에 비해 약 74.5% 성능이 좋았다. 또한 PSO(Q) 기법과 비교하였을 때 약 14.2% 성능이 좋았다. AL를 제외하고는 전반적으로 성능이 비슷할뿐더러, 디바이스가 250일 때는 PSO(D)의 에너지 소모량이 제일 낮다. 하지만 에너지 소모량과 blocking 개수는 상반적인 관계이다. 태스크의 수행이 로컬에서 이뤄지는 개수가 늘어나면 에너지 소모량은 높아지고 blocking의 개수는 줄어들고, 오프로딩되는 태스크의 개수가 증가하면 에너지 소모량은 낮아지고 blocking 개수는 늘어난다. 그렇기 때문에 단순히 에너지 소모량이 조금 좋아졌다고 해당 기법의 성능이 좋다고 할 수 없다. Fig. 4의 디바이스 250개의 PSO(D) blocking 개수는 AO 기법을 제외하고 제일 많기 때문이다. 결과적으로 디바이스의 에너지 소모량과 blocking 개수 사이에서의 둘 다 동시에 줄일 수 있는 최선의 오프로딩 결정을 해야 한다. 그렇기 때문에 비슷한 디바이스의 에너지 소모량을 갖고 있는 기법들 중 제일 blocking 개수가 적은 proposed(Q)가 성능이 제일 좋다.

Fig. 6은 디바이스 수에 따른 MEC 별 blocking 수에 대한 표준편차이다. 앞서 살펴본 실험 결과를 기반으로 상대적으로 좋은 성능을 보였던 MEC 서버의 큐 길이를 기준으로 MEC 서버를 선택한 기법들의 blocking 수에 대한 표준 편차를 구하였다. 디바이스가 150개 일 때, 제안 알고리즘인 proposed(Q)의 표준편차는 PSO(Q)에 비해 약 58.3% 정도로 좋은 성능을 보였고, 디바이스 250개일 때는 약 6.8%, 350개일 때는 약 2.5% 좋은 성능을 보였다. 결과적으로 디바이스 수가 점점 늘어날 때는 시스템 전체에 대한 부하가 증가하므로 표준편차의 차이는 줄어들었지만, 그럼에도 제안하는 알고리즘이 PSO 기법에 비하여 작은 표준편차를 보였으며, 이는 태스크 오프로딩이 특정 MEC 서버에 집중되지 않고 시스템 내의 MEC 서버들

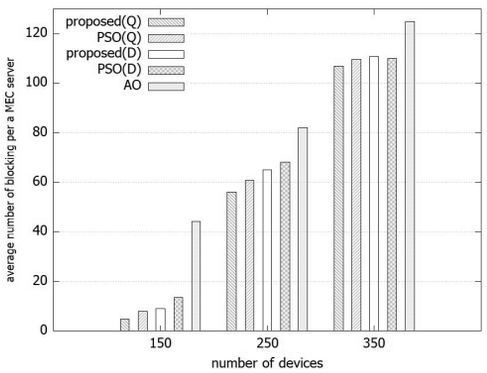

Fig. 4. Average Number of Blocking Per a MEC Server with Different Numbers of Devices

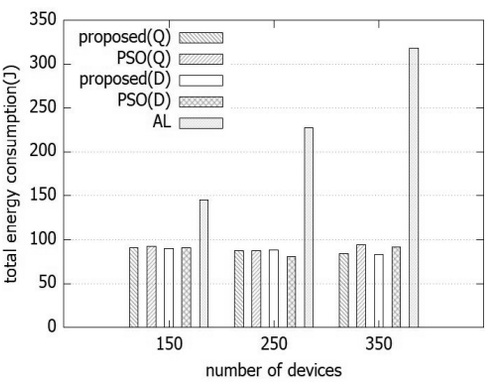

Fig. 5. Total Energy Consumption with Different Numbers of Devices

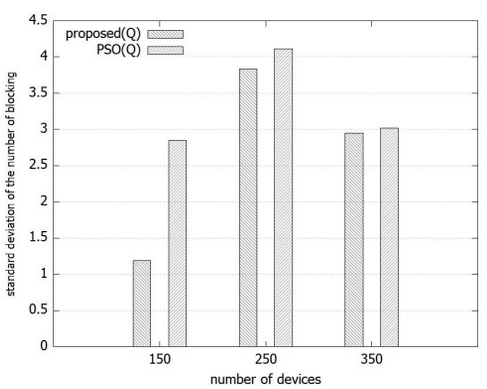

Fig. 6. Standard Deviation of the Number of Blocking with Different Numbers of Devices

에게 고르게 오프로딩 되었음을 보인다. PSO는 돌연변이와 교배와 같은 연산을 이용하여 모집단의 다양성을 높인 유전 알고리즘과 달리 몇 번의 반복만으로 개별 입자에 대한 이동하는 방향이 정해져 모집단의 다양성이 떨어져 국소 최적화를 이룰 확률이 높다. 이러한 특징 때문에 PSO는 상대적으로 유전 알고리즘에 비해 매번 모든 상황에서 최적의 가까운 해결책을 내기에 무리가 있다. 이러한 PSO의 특징은 유전 알고리즘에 비해 표준편차도 높게 나타난 이유이기도 하다.

결과적으로 에너지 측면에서 다른 기법들과 비교하였을 때 비슷한 성능을 보이면서도 blocking 평균값과 표준편차 측면에서 제안 기법이 더 나은 성능을 보였다. 본 연구는 IIoT 환경을 가정하였고, 이 환경은 신뢰성을 중시하기 때문에 신뢰성과 관련된 blocking 수 측면에서 제안 기법이 효율적이라고 할 수 있다.

6. 결론 및 향후 연구

본 연구에서는 IIoT 환경에서 유전 알고리즘을 사용하여 디바이스에서 발생한 태스크들의 오프로딩 여부를 결정하는 기법을 제안하였다. 이 알고리즘은 다른 비교 기법들에 비해서 MEC 서버 당 평균 blocking 개수를 줄여 신뢰성을 높이면서도 에너지 소모량에 있어서 좋은 성능을 보였다. 향후 연구에서는 실제 환경에서 보다 동적인 MEC 서버와 디바이스를 고려한 강화 학습 기반 오프로딩 결정 기법을 연구할 예정이다.

References

[1] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys & Tutorials*, Vol.19, No.3, pp.1657-1681, 2017.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *IEEE Transactions on Industrial Informatics*, Vol.14, No.11, pp.4724-4734, 2018.

[3] F. Chiti, R. Fantacci, and B. Picano, "A Matching Theory Framework for Tasks Offloading in Fog Computing for IoT Systems," *IEEE Internet of Things Journal*, Vol.5, No.6, pp. 5089-5096, 2018.

[4] H. Liu, L. Cao, T. Pei, Q. Deng, and J. Zhu, "A Fast Algorithm for Energy-Saving Offloading with Reliability and Latency Requirements in Multi-Access Edge Computing," *IEEE Access*, Vol.8, pp.151-161, 2020.

[5] Qiuping Li, Junhui Zhao, Yi Gong, and Qingmiao Zhang, "Energy-Efficient Computation Offloading and Resource Allocation in Fog Computing for Internet of Everything," *China Communications*, Vol.16, No.3, pp.32-41, 2019.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol.4, pp.1942-1948, 1995.

[7] L. Davis, "Handbook of genetic algorithms," CumInCAD, 1991.

[8] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-Optimized Partial Computation Offloading in Mobile-Edge Computing With Genetic Simulated-Annealing-Based Particle Swarm Optimization," *IEEE Internet of Things Journal*, Vol.8, No.5, pp.3774-3785, Mar. 2021.

[9] Z. Li and Q. Zhu, "Genetic Algorithm-based Optimization of Offloading and Resource Allocation in Mobile-Edge Computing," *Information*, Vol.11, No.2, pp.2-11, 2020.

[10] K. Peng, B. Zhao, S. Xue, and Q. Huang, "Energy- and Resource-Aware Computation Offloading for Complex Tasks in Edge Environment," *Complexity*, Vol. 2020, 2020.

[11] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for Computation Offloading in Mobile Edge Computing," *IEEE Transactions on Communications*, Vol.66, No.12, pp.6353-6367, 2018.

[12] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks," *IEEE Conference on Computer Communications (INFOCOM)*, pp.10-18, Apr. 2019.

구 설 원

https://orcid.org/0000-0000-1267-4470
e-mail : tjrgkr501@sookmyung.ac.kr
2019년 숙명여자대학교 IT공학과(학사)
2019년 ~ 현 재 숙명여자대학교 IT공학과 석・박사통합과정
관심분야 : 지능형 시스템, IIoT, Edge Computing

임 유 진

https://orcid.org/0000-0002-3076-8040
e-mail : yujin91@sookmyung.ac.kr
2000년 숙명여자대학교 전산학과(박사)
2013년 Tohoku University, Dept. of Information Sciences(박사)
2004년 ~ 2015년 수원대학교 정보미디어학과 부교수
2016년 ~ 현 재 숙명여자대학교 IT공학과 교수
관심분야 : 지능형 시스템, IoT, Edge Computing