Received September 17, 2021, accepted September 29, 2021, date of publication October 4, 2021, date of current version October 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3117870

Server Placement and Task Allocation for Load Balancing in Edge-Computing Networks

PING-CHUN HUANG, TAI-LIN CHIN, (Member, IEEE), AND TZU-YI CHUANG
Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106335, Taiwan

Corresponding author: Tai-Lin Chin (tchin@mail.ntust.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST-109-2221-E-011-106.

ABSTRACT Offloading tasks to cloud servers has increasingly been used to provide terminal users with powerful computation capabilities for a variety of services. Recently, edge computing, which offloads tasks from user devices to nearby edge servers, has been exploited to avoid the long latency associated with cloud computing. However, edge server placement and task allocation strongly affect the offloading process and the quality of a user's experience. Therefore, appropriately deploying the edge servers within a network and evenly allocating the workload to the servers are vital. This paper thus considers both the workload of edge servers and the distances involved in offloading tasks to these servers. To improve the user experience, edge server locations are carefully selected and the workload for the servers are allocated in a balanced manner. This scenario is formulated as a mixed-integer linear programming problem, and a novel solution that searches for the best server placement using simulated annealing while integrating task allocation using the Lagrangian duality theory with the sub-gradient method is proposed. Numerical simulations verify that the proposed algorithm can achieve better results than conventional heuristics.

INDEX TERMS Cloud computing, edge computing, server placement, task allocation.

I. INTRODUCTION

With the increasing popularity of smartphones and Internet-of-Things (IoT) devices, many new applications that require significant computation and prompt responses have been developed. However, mobile devices and general embedded systems usually have limited computation capabilities. One solution to this problem is to offload individual tasks to cloud servers because, in general, cloud servers have massive computation resources [1]. Unfortunately, the servers are often located far from the users, resulting in high transmission latency. Recently, edge computing has been proposed to enhance the efficiency of the task offloading process [2], [3]. Edge computing deploys computational and storage resources to the edge of a network. Because the computational resources are relocated closer to the users, the communication latency can be significantly reduced. Nevertheless, given the rapid development of smartphone and IoT device applications, offloading demand can increase significantly and an unbalanced workload among edge servers may still result in longer computational latency when offloading tasks.

The associate editor coordinating the review of this manuscript and approving it for publication was Yougan Chen.

Some previous studies have considered load balancing by monitoring certain utilization metrics for load generated by server-side computation [4], [5], but rarely considered communication distance as the main factor impacting workload balancing. The transmission time is usually determined by the data size and data transmission rate [6], [7], with longer transmission distances also creating greater loads within the network. Traditional models for load balancing usually consider only the computational load on the servers and do not consider the transmission load due to distance.

In this paper, a task-offloading paradigm is considered. A number of edge servers are deployed with edge nodes within a network. Offloading requests from other edge nodes can be allocated to the edge servers in order to fulfill the demand for computational power. Edge server placement and task allocation are tailored by considering both the workload balance among the edge servers and the transmission distance for the offloaded tasks. Balancing the workload among the edge servers and reducing the transmission distance for offloaded tasks is equivalent to reducing the computational and communication latency, respectively. This is formulated as a mixed-integer linear programming problem. The Edge Server Placement (ESP) Algorithm is thus proposed as an efficient strategy to determine a better solution for edge server placement and task allocation. The ESP algorithm is based on simulated annealing and considers both edge server placement and task allocation. Simulated annealing is used to search the enormous solution space for edge server locations, while task allocation is resolved for the selected edge server locations using the Lagrangian duality theory [8] and the sub-gradient method. Simulations are conducted to evaluate the performance of the proposed scheme, with the results indicating that the ESP algorithm exhibits great flexibility in balancing the load among the edge servers and the transmission distance.

The main contributions of this paper are as follows:

- This paper proposes a load balancing model that considers both the computational load on the edge servers and the transmission distance for task offloading in edge-computing networks.
- An efficient and effective scheme is developed to provide better solution for both task allocation and server placement.
- The Lagrangian duality theory is integrated with the simulated annealing process to improve the search efficiency

The rest of this paper is organized as follows. Section II briefly reviews related work. Section III presents the problem formulation and solution in detail. Section IV describes the simulation experiments and evaluates the performance. Finally, Section V concludes the paper.

II. RELATED WORK

Task allocation and server placement problems have been intensively studied in recent decades. Both problems are often associated with optimization objectives and constraints that make them non-trivial and very challenging.

Some research has assumed that servers are already deployed and focus on allocating the clients to servers with fixed locations. For example, Ye *et al.* [9] studied user association policies in heterogeneous networks (HetNets) for load balancing and presented an efficient distribution algorithm that obtained a near-optimal solution. In [10], Athanasiou *et al.* modeled a client association problem that assigned a client to access points in a 60-GHz wireless access network. However, these studies did not consider the transmission distance between the clients and servers. Heuristic algorithms are often used for discovering the effective association between clients and servers.

The studies reported in [11], [12], and [13] employed genetic algorithms for server placement and task allocation, while Lim and Lee [11] aimed to find an effective strategy based on graph coloring to offload tasks to edge servers to balance the load. Tang and Pan [12] focused on the energy consumption in the communication network of a data center. They proposed a hybrid genetic algorithm that improved performance and efficiency by optimizing the placement of virtual machines. Xu *et al.* [13] presented a computational model based on vehicle-to-all communication (V2X) in edge computing. A genetic algorithm-based method was proposed as a balanced offloading strategy.

Game theory has also been widely adopted in task allocation for mobile-edge cloud computing [14], [15], [16], [17]. With distributed multiple users, every user is modeled as a game player, with each can independently determining their own offloading strategies. In [14], Chen presented a computation offloading model in which the tasks are assumed to be either processed locally or offloaded entirely to a single cloud server. The problem was formulated as a decentralized computation offloading game that promised to achieve a Nash equilibrium. Based on [14], Chen [15] further considered the problem of deciding whether to forward a user's tasks to a single remote cloud server with a single access point in each round of the game. In contrast, instead of considering a single access point, Ma *et al.* [16] took multiple access points into account. In addition, in [17], a multi-user offloading problem was formulated as a stochastic game, and a stochastic learning algorithm was proposed in order to reach a Nash equilibrium.

Other researchers have proposed the task-offloading schemes based on machine-learning techniques in edge computing. Li *et al.* [18] and Shuja *et al.* [19] surveyed solutions based on machine learning for caching in an edge network. These approaches trained the model to offload computationally-intense applications to a specified edge server. However, the offloading model could be complex, and a huge amount of data was required to train the model.

In term of server placement, previous studies have mostly concentrated on searching for candidate server positions as a cluster head to reduce the response time. Recently, studies have investigated the K-edge server placement (k-ESP) problem [20], [21]. k-ESP primarily focuses on minimizing the number of edge servers to cover the entire Internet while satisfying budget constraints. Zeng *et al.* [20] presented a greedy algorithm to determine the fewest number of servers required in wireless metropolitan area networks. The proposed method iteratively selected as many nodes as possible to maximize the edge servers' coverage. In [21], Yin *et al.* provided a dynamic, resource-provisioning framework to obtain feasible edge server locations according to the workload and users' proximity. However, they did not consider load balancing. Though the requirement for latency constraints was met, their proposal may cause overloading.

Li and Wang [22] proposed an energy-aware edge server placement model to reduce the energy consumption and computing resource utilization. A discrete particle swarm optimization algorithm was proposed for both server placement and task allocation. In [23], Xu *et al.* proposed a model to minimize a multi-objective problem for social media services within the Cognitive Internet of Vehicles. An integrated genetic algorithm was adopted to improve the quality of the services. Guo *et al.* [24] described a multi-objective optimization problem to minimize the communication delay with load balancing between devices. Although the studies aim to optimize multiple objectives, the weights for each term in the objective function are usually selected intuitively and, thus, can dramatically change the final results. In contrast, the model proposed in this paper seamlessly integrates the cost of computation and communication and provides a more meaningful control between the loads of computation and communication.

III. TASK ALLOCATION AND SERVER PLACEMENT

In this section, the system model for the server placement and task allocation problem is illustrated. An integer programming problem is then formulated based on the system model.

A. SYSTEM MODEL

Consider a network composed of nodes and links, as shown in Fig. 1. The network topology can be considered an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, N\}$ denotes the set of nodes that could be base stations or access points, and E is the set of links between the nodes. The nodes provide access to the network for mobile users. To facilitate task-offloading operations, a number of edge servers will be deployed on some of the nodes. Users' devices can attach to the nodes in their vicinity. A variety of computationally intensive and delay-sensitive tasks from user devices can be offloaded to the edge servers through the nodes to which the user is currently attached. For simplicity, each node allocates a non-dividable task entirely to a solitary node equipped with an edge server. The problem in which nodes can split tasks between different edge servers can be modeled in a similar way. In addition, every node can allocate its tasks to any node with an edge server in the topology.

B. PROBLEM FORMULATION

Assume that there are K edge servers deployed in the network to process the received tasks. In this system, each node i can be allocated to node j deployed with an edge server. The access delay between node i and j can be indicated by the number of hops as follows:

$$d_{ij} = (hop_{ij} + 1)^{\alpha}, \tag{1}$$

where hop_{ij} is the minimum number of hops between node i and node j, and α is the weight factor for distance. The distance is at least 1 because tasks from a mobile device are initially offloaded to the nearest base station i.

Let λ_i be the offloading task request rate originating from node i. The total workload is weighted by the distance. The largest workload among all edge servers can be expressed as

$$\eta = \max_{j \in V} \sum_{i \in V} d_{ij}\lambda_i y_{ij} \tag{2}$$

where y_{ij} is a binary variable that represents the allocation decision. In particular, for all $i, j \in V$,

$$y_{ij} = \begin{cases} 1, & \text{if the task originated from node } i \\ & \text{is allocated to edge server at node } j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

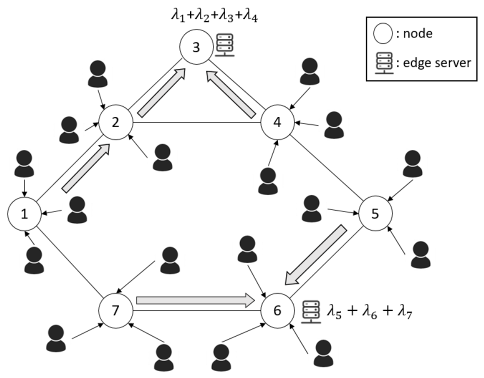

$\lambda_1+\lambda_2+\lambda_3+\lambda_4$
: node
: edge server
1
2
3
4
5
6
7
$\lambda_5+\lambda_6+\lambda_7$

FIGURE 1. Illustration of task allocation in edge computing.

Note that the workload is weighted by the distance from the node to the edge server. Let x_i be another binary decision variable that indicates whether node i is deployed with an edge server and can be defined as follows:

$$x_i = \begin{cases} 1, & \text{if node } i \text{ is deployed with an edge server} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The goal of the problem is to minimize the largest workload for edge server from among the set of all edge servers. Specifically, the server placement and task allocation problem can be formally expressed as follows:

$$\begin{aligned} &\text{minimize } \eta && (5)\\ &\text{subject to } \sum_{i \in V} d_{ij}\lambda_i y_{ij} \le \eta, \quad \forall j \in V && (6)\\ &\sum_{j \in V} y_{ij} = 1, \quad \forall i \in V && (7)\\ &\sum_{i \in V} x_i = K && (8)\\ &y_{ij} \le x_j, \quad \forall i, j \in V && (9)\\ &y_{ij} \in \{0, 1\}, \quad \forall i, j \in V && (10)\\ &x_i \in \{0, 1\}, \quad \forall i \in V. && (11) \end{aligned}$$

Constraint (6) guarantees that the workload for all edge servers is less than or equal to η, which is the largest workload among the edge servers. Constraint (7) guarantees that tasks originating from a particular node will be allocated to a single edge server. The total number of edge servers is limited to K as in constraint (8). Constraint (9) ensures that nodes must be allocated to a node with an edge server. Instead of relying on exponentially complex global methods, this paper proposes an efficient approach that is feasible and better than simple heuristics.

The proposed solution can be separated into two phases: (1) the location of the edge servers is selected, and (2) the tasks from each node are allocated to the edge servers. To simplify the problem, the task allocation is resolved under the assumption that the edge server locations have already been selected. The edge server locations are then selected by integrating the task-allocation scheme.

C. TASK ALLOCATION PROBLEM

Assume that K-edge servers have already been deployed in the network. Let X be the set of nodes for which edge servers are deployed. Because the edge server locations have already been selected, the problem can be simplified as follows:

$$\text{minimize } \eta \tag{12}$$
$$\text{subject to } \sum_{i\in V} d_{ij}\lambda_i y_{ij} \le \eta, \quad \forall j \in X \tag{13}$$
$$\sum_{j\in X} y_{i,j} = 1, \quad \forall i \in V \tag{14}$$
$$y_{ij} \in \{0,1\}, \quad \forall i \in V, \quad \forall j \in X. \tag{15}$$

This is a combinatorial problem. In the worst case, the complexity of conventional algorithms in solving this problem would grow exponentially with the increase of the topology size.

The task allocation problem differs from the general assignment problem in that the objective is to reduce the largest load among the edge servers rather than the sum of the assignment costs. The Lagrangian duality theory applied for the association problem in [10] is referred to solve the problem. For completeness, the derivation is briefly described as follows. The Lagrangian duality theory aims to solve the original objective function by finding the solution for a dual function that is derived from the original function. The transformed dual function is usually easier to solve, and the obtained solution can place a bound on the primal function. First, denote $\boldsymbol{u} = (u_j)_{j\in X}$ as the vector of the Lagrange multipliers to dualize constraint (13) in problem (12). The partial Lagrangian can be formed as

$$L(\eta, \mathbf{y}, \boldsymbol{u}) = \eta(1 - \sum_{j\in X} u_j) + \sum_{i\in V}\sum_{j\in X} d_{ij}\lambda_i u_j y_{ij}. \tag{16}$$

To simplify the notation, let $\mathcal{Y}$ be the set of all possible solutions for the allocation vectors $\mathbf{y}$ according to the constraints in Eqs. (14) and (15). $\mathcal{Y}$ can further be denoted as a Cartesian product for the set of $\mathcal{Y}_i \subset \mathbb{R}$, $i \in V$

$$\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_n \tag{17}$$

where $\mathcal{Y}_i$ is given by

$$\mathcal{Y}_i = \{\mathbf{y}_i = (y_{ij})_{j\in X} | \sum_{j\in X} y_{ij} = 1, y_{ij} \in \{0,1\}\}. \tag{18}$$

Furthermore, the Lagrange dual function $g(\boldsymbol{u})$ can be obtained by minimizing the partial Lagrangian in (16) with the input including η, $\mathbf{y}$ and $\boldsymbol{u}$ as follows:

$$g(\boldsymbol{u}) = \inf_{\mathbf{y}\in\mathcal{Y}} L(\eta, \mathbf{y}, \boldsymbol{u}) \tag{19}$$
$$= \begin{cases} \inf_{\mathbf{y}\in\mathcal{Y}} \sum_{i\in V}\sum_{j\in X} d_{ij}\lambda_i u_j y_{ij}, & \sum_{j\in X} u_j = 1 \\ -\infty, & \text{otherwise} \end{cases} \tag{20}$$
$$= \begin{cases} \sum_{i\in V} \inf_{\mathbf{y}_i\in\mathcal{Y}_i} \sum_{j\in X} d_{ij}\lambda_i u_j y_{ij}, & \sum_{j\in X} u_j = 1 \\ -\infty, & \text{otherwise} \end{cases} \tag{21}$$
$$= \begin{cases} \sum_{i\in V} g_i(\boldsymbol{u}), & \sum_{j\in X} u_j = 1 \\ -\infty, & \text{otherwise.} \end{cases} \tag{22}$$

In Eq. (16), $\eta(1 - \sum_{j\in X} u_j)$ should be zero or the value of this equation would be infinity. Therefore, $\sum_{j\in X} u_j = 1$ is needed to prevent the objective function from going to infinity. In particular, the constraints for task allocation in Eqs. (14) and (15) are implied in Eqs. (17) and (18). The optimal value of $g_i(\boldsymbol{u})$ in problem (22) can be obtained from

$$\min \sum_{j\in X} d_{ij}\lambda_i u_j y_{ij} \tag{23}$$
$$\text{s.t. } \mathbf{y}_i \in \mathcal{Y}_i. \tag{24}$$

Finally, the Lagrange dual problem can be formulated as

$$\max g(\boldsymbol{u}) = \sum_{i\in V} g_i(\boldsymbol{u}) \tag{25}$$
$$\text{s.t. } \sum_{j\in X} u_j = 1 \tag{26}$$
$$u_j \ge 0, j \in X. \tag{27}$$

The load-balancing problem in (12) is converted to the Lagrange dual problem which is now a convex problem.

According to the properties of the Lagrangian duality theory, if the primal problem is convex, the optimal value of the primal problem and the corresponding dual problem is the same. In contrast, if the primal problem is non-convex, there would be a duality gap between the optimal value of the two problems. Though a gap exists, a feasible solution approximating the optimal solution for the primal problem can still be obtained by solving the dual problem.

The objective $g(\boldsymbol{u})$ in problem (25) is a non-differentiable function. Therefore, instead of using gradient-based algorithms, a sub-gradient method [25] is used to solve this problem.

In problem (23), although it is combinatorial, the solution can be obtained trivially as follows:

$$y^*_{ij} = \begin{cases} 1, & j = \arg\min_{k\in X} d_{ik}\lambda_i u_k \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

The solution y^*_{ij} determines whether the task requests of node i are allocated to server j.

The sub-gradient method is used for the problem in (25). Let $\boldsymbol{s} = (s_j)_{j\in X}$ denote the sub-gradient of $-g$ at a feasible $\boldsymbol{u}$, where s_j can be obtained as follows:

$$s_j = -\sum_{i\in V} d_{ij}\lambda_i y^*_{ij} \tag{29}$$

where y^*_{ij} is the solution obtained from Eq. (28) for problem (23). The iterative projected sub-gradient method can be formulated as

$$\boldsymbol{u}^{(k+1)} = P((\boldsymbol{u})^{(k)} - \beta_k \boldsymbol{s}^{(k)}) \tag{30}$$

where k is the index of the iteration in the projected sub-gradient method, and P is the function of Euclidean projection that can project a value onto the unit simplex

$\Pi = (\boldsymbol{u} | \sum_{j \in V} u_j = 1, u_j \geq 0)$ [26]. β_k is the step size at the kth iteration. In this work, β_k is set as $\beta_k = \beta / k$, where β is a constant greater than 0. In the beginning, the value u_j is given randomly; $\boldsymbol{u}$ is optimized with the iteration of the projected sub-gradient method, and the dual problem (25) can be solved gradually.

```
Algorithm 1: Task Allocation Algorithm
Input:
K-edge servers' positions X
Output:
Task allocation result y*
1  Given $u_j$ randomly with $\sum_{j \in X} u_j = 1$
2  Set sub-gradient iteration number $k = 1$
3  Temporary allocation result set $\tilde{\boldsymbol{y}} = \emptyset$
4  for $k \leftarrow 1$ to $\varphi$ do
5      Step 1 Determine task allocation $y_{ij}$ by Eq. (28)
6      Step 2 Store the allocation result into $\tilde{\boldsymbol{y}}$
7      Step 3 Compute $s_j$ of each edge server $j$ by Eq. (29)
8      Step 4 Update each $u_j$ by projected sub-gradient
       method in Eq. (30) as $\boldsymbol{u}^{k+1}$
9  compute cost of primal problem (12) with each
     allocation result store in $\tilde{\boldsymbol{y}}$
10 $\boldsymbol{y}^*$ = the allocation with the lowest cost by problem (12)
     in $\tilde{\boldsymbol{y}}$
```

However, as mentioned previously, the primal problem (12) is non-convex, so a feasible solution for the primal problem cannot be obtained from the dual problem directly. In our problem, let φ be the number of iterations running for the projected sub-gradient method. The most feasible primal solution for task allocation is taken as the best dual solution from the φ iterations in Algorithm 1. The complete process for the allocation algorithm is presented in Algorithm 1.

D. THE PROPOSED SCHEME

In the next phase, the edge server locations are selected. The task allocation scheme presented above is integrated with the server placement. Selecting locations for the edge servers is important because different edge server locations may lead to different workloads, which are weighted with the distance between the servers and the allocated nodes. Given the increasing size of the toplogy, it has become very challenging to optimally deploy edge servers due to the exponential increase in the number of combinations in task allocation and server location selection. To deal with this large combinatorial optimization problem, simulated annealing is adopted in the search for the global optimal solution for edge server placement and task allocation. The proposed algorithm is listed in Algorithm 2.

Initially, the K-edge servers are deployed at the nodes with the highest request rates. There are several parts in the simulated annealing process. First, the configuration state of the system has to be defined. Server placement $\boldsymbol{S}^*$ and task allocation $\boldsymbol{y}^*$ are the configurations of the architecture. $\boldsymbol{S}^*$ is

```
Algorithm 2: Edge Server Placement Algorithm
Input:
A network topology $G = (V, E)$
Output:
Edge server location set $\boldsymbol{S}^*$
Task allocation result $\boldsymbol{y}^*$
1  $T = T_0$
2  $\eta^* = \inf, \eta = 0$
3  $\boldsymbol{S} \leftarrow$ the $K$ nodes with the largest request rates
4  while $T > T_f$ do
5      Obtain $\boldsymbol{y}$ by Algorithm 1 for the edge servers in $\boldsymbol{S}$
6      Compute the largest edge server workload $\eta$
7      if $\eta < \eta^*$ then
8          $\boldsymbol{S}^* \leftarrow \boldsymbol{S}, \boldsymbol{y}^* = \boldsymbol{y}, \eta^* = \eta$
9      else
10         $\Delta = \eta - \eta^*$
11         $P = \min(1, e^{(-\Delta f)/T})$
12         if with probability $P$ then
13             $\boldsymbol{S}^* \leftarrow \boldsymbol{S}, \boldsymbol{y}^* = \boldsymbol{y}, \eta^* = \eta$
14         else
15             $\boldsymbol{S} \leftarrow \boldsymbol{S}^*$
16     Update $\boldsymbol{S}$ by Algorithm 3
17     $T = T * \gamma$
18 return $\boldsymbol{S}^*, \boldsymbol{y}^*$
```

a set of nodes that are chosen to place edge servers, and $\boldsymbol{y}^*$ is the allocation results obtained from Algorithm 1 according to the specified edge server locations. Second, the acceptance probability for the generated configurations needs to be calculated. While the generation mechanism provides candidate configurations, the probability decides whether worse configurations are selected as the next state or not. In the proposed algorithm, if the new configuration exhibits a better objective value, it is accepted directly. If not, it may still be accepted based on this probability, which depends on Boltzmann's function, formulated as

$$P = \min(1, \exp^{-\Delta f / T}) \tag{31}$$

where Δf is the difference between the current and previous values of the objective function at the iteration, and T is the current temperature. The purpose for accepting worse configurations is to provide chance to jump out of local minima which may obtain better results.

Third, a generation mechanism for new configurations is required. To search efficiently in the large combinatorial problem, simulated annealing explores different configurations to obtain better results. At every iteration, the configurations are updated and compared, allowing the better configuration to be determined. In the proposed algorithm, at each iteration, the configuration is updated as follows. One of the edge servers including the group of nodes that are allocated to the edge server is selected. Then, in the cluster, the node which will has the least workload

Algorithm 3: Edge Server Updating Algorithm

```
Input:
Edge server location set $\boldsymbol{S}$
Task allocation result $\boldsymbol{y}$
Output:
New edge server location set $\boldsymbol{S}$
1 Randomly select an $s$ from $\boldsymbol{S}$
2 $\boldsymbol{S} \leftarrow \boldsymbol{S} / s$
3 $G_s = \emptyset$
4 for $i \leftarrow 1$ to $N$ do
5     if $y_{is} = 1$ then
6         $G_s \leftarrow G_s \cup i$
7 $s' = \arg\min_{i \in G_s} \sum_{j \in G_s} d_{ij}\,\lambda_j$
8 $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup s'$
9 return $\boldsymbol{S}$
```

if performing as the edge server is chosen as the new edge server. Algorithm 3 lists the sever updating process. Finally, the nodes in the network are then reallocated according to the new set of servers in the next iteration. The algorithm stops when the temperature is lower than the set threshold.

E. TIME COMPLEXITY

In the ESP algorithm, time is primarily spent on the allocation algorithm. At each iteration, server placement and task allocation are updated, with the complexity of updating the server in Algorithm 3 is about $O(N^2)$. In Algorithm 1, the calculation time is primarily consumed in Step 1, with the time complexity for determining the allocation being $O(NK)$. The total time complexity for Algorithm 1 is $O(\varphi(NK))$, where φ is the number of iterations for the sub-gradient method. The computation and comparison of the costs can be calculated in constant time. Let r be the number of moves executed in simulated annealing. The total time complexity is $O((N^2 + \varphi(NK)) * r)$ which is executed in polynomial time.

IV. SIMULATIONS

In this section, the proposed algorithm is verified and evaluated using topologies of various sizes. The experimental results are discussed to determine the effectiveness of the proposed algorithm in terms of the placement and allocation problem.

A. ENVIRONMENTAL SETUP

In the experiments, 100 different topologies are generated with different numbers of links ranging from $1.1 \times N$ to $2.0 \times N$, where N is the number of nodes. The offloading task request rate from the nodes are randomly selected, but the sum of the injected task request rates is controlled by the size of the topology, i.e., the number of nodes N. In the simulations, the total injected task rate is set to $20 \times N$. For the parameters used in the Algorithms, the total iteration number φ for sub-gradient is set to 100. The initial and final temperatures are 300 and 10, respectively. The cooling rate γ is set to 0.9. In addition, the number of hops between nodes is obtained using the Dijkstra algorithm. The value for every data point in the figures is the average over the 100 topologies.

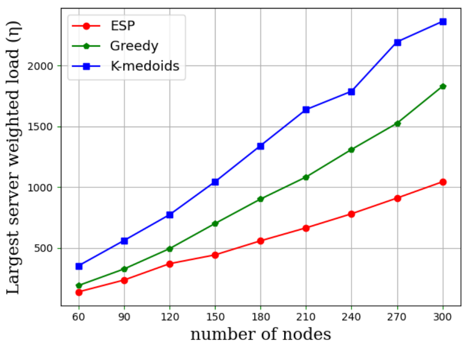


FIGURE 2. The impact of the number of nodes.

The performance of the proposed scheme is compared to classic K-medoids clustering [27] and a greedy algorithm. The greedy algorithm deploys the edge server one by one. In each iteration, it places an edge server at the node with the highest request rate among the remaining nodes that have not yet been allocated to an edge server. The nodes in its vicinity are allocated to that edge server starting with the closest node until the sum of the allocated task request rate reaches the average, i.e., $\sum_{i \in V} \lambda_i / K$. This process repeats until K edge servers are deployed and all of the nodes are allocated.

B. SIMULATION RESULTS

Fig. 2 presents the results for the largest server workload against the number of nodes in the topology. Twenty edge servers are deployed in the topology. Intuitively, the average objective value (η) increases with an increase in the number of nodes. The greedy approach is better than K-medoids because its task allocation strategy focuses on balancing the weighted load, which considers the task request rates and the distance from the nodes to the edge servers. In contrast, K-medoids only considers the distance between the nodes and the edge servers. The larger the environment, the stronger the impact of this load balancing mechanism. The proposed ESP algorithm always exhibits a lower value than the greedy and K-medoids algorithms and has the lowest increase rate, indicating that its performance will improve for larger topologies.

In Fig. 3, N is fixed at 200 and K is varied from 10 to 30 to observe the impact of the number of servers on the average objective value. With an increase in the number of edge servers, each node has a higher chance of finding a closer edge server, thus lowering the total workload. The results reveal that the optimization mechanism used by the proposed method is much better than the greedy and K-medoids algorithms. Fig. 4 investigates the impact of the number of links in the topology. N is fixed at 200 and K is fixed at 20. The proposed ESP produces a better performance than the other approaches. With a rise in the number of

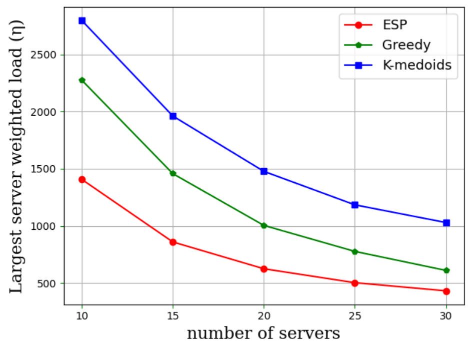


FIGURE 3. The impact of the number of servers.

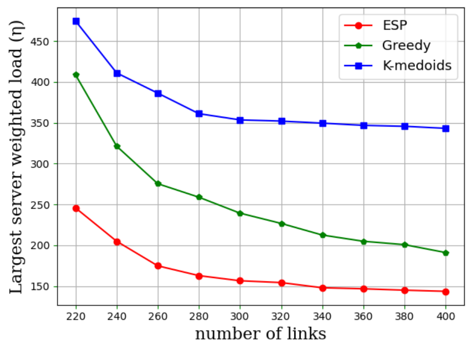


FIGURE 4. The impact of the number of links.

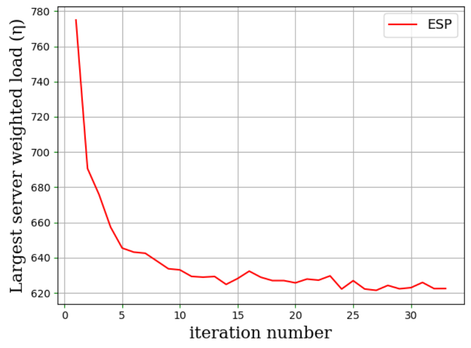


FIGURE 5. The process of the edge placement algorithm.

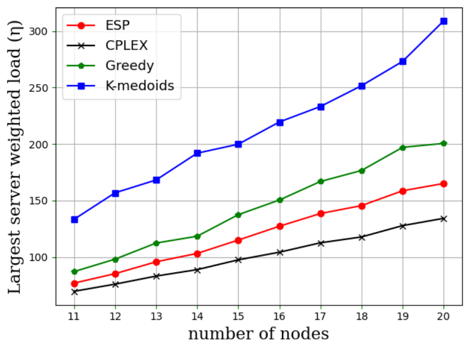


FIGURE 6. Comparison to the results from CPLEX.

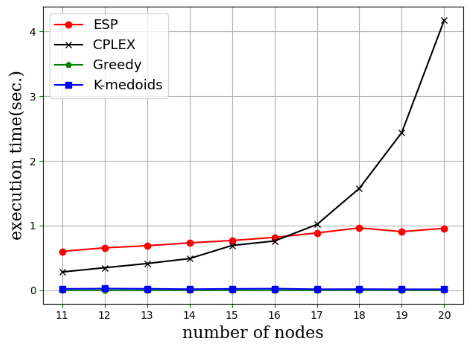


FIGURE 7. The execution time.

links, the average objective value steadily decreases. This is because the nodes can find shorter paths to servers when there are more links. However, once a certain number of links has been reached, the decline in η gradually slows down because a load balance status can be maintained. In Fig. 5, the convergence behavior of the proposed ESP algorithm is presented with N fixed at 200 and K fixed at 20. During the simulated annealing process, the largest workload among edge servers is reduced along the execution. The average objective value becomes stable after about 30 iterations.

In addition, the algorithms are also compared with the approximate solution obtained using CPLEX, an IBM tool for optimization problems. CPLEX is used as a comparison because its solution is considered to be close to the optimal. The community edition of the optimizer is employed in the experiments, so the solver can only handle small topologies. The number of edge servers is fixed at $K = 5$, and the number of nodes N ranges from 11 to 20. Fig. 6 presents the average objective value for each scheme. The results of the proposed ESP algorithm are the closest to those from CPLEX, which implies that the results of ESP are much closer to the optimal than the other schemes.

The computational efficiency of each algorithm is also compared. Fig. 7 shows the average execution time according to the number of nodes. K is fixed at 5 and n ranges from 11 to 20. Although the ESP algorithm has a higher computation time than the greedy and K-medoids algorithms, it is still within an acceptable range because the average objective value it obtains is much better. For CPLEX, the average execution time grows exponentially with an increase in the number of nodes. The execution time of the proposed algorithm also grows as the nodes increase, but this increase is much slower than the CPLEX. Predictably, CPLEX is much more computationally expensive for large-scale topologies.

Experiments are also conducted to evaluate the impact of the distance factor α by the ESP algorithm. In Fig. 8, by adjusting the distance factor α, the average number of hops is evaluated for different numbers of nodes. K is fixed at 20. By increasing α, the average number of hops between the nodes and edge servers decreases gradually. Because when the weight of distance is higher, nodes tend to offload to closer edge servers to reduce the transmission load, which

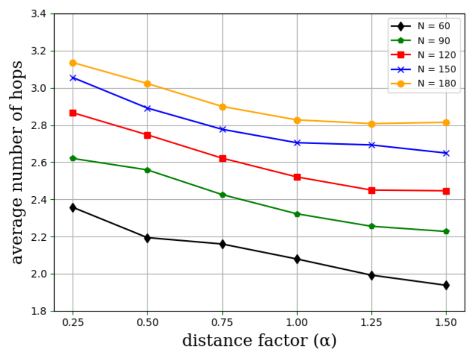


FIGURE 8. The impact of distance factor α on node distance.

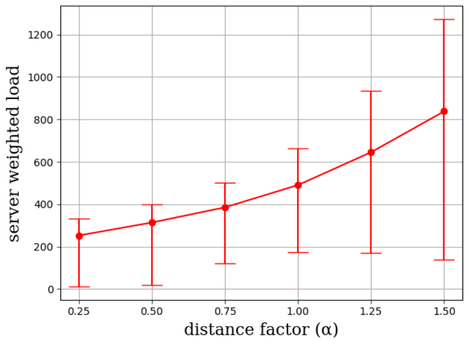
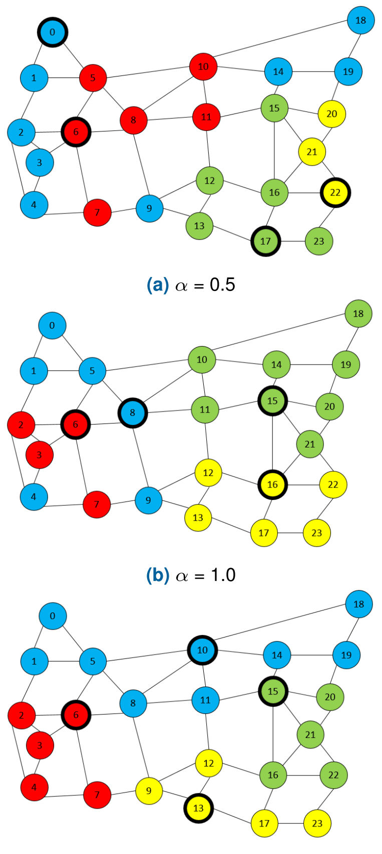


FIGURE 9. The impact of distance factor α on server workload.

results in a lower average number of hops. In addition, when n increases, the topology becomes larger. The number of hops also increases since the nodes are farther away from the edge servers.

In Fig. 9, the difference in the weighted load between servers is evaluated with different distance factor α. N is fixed at 200 and K is fixed at 20. In the figure, the upper bound of the vertical bars represents the largest server weighted load, while the lower bound represents the smallest, and the data point is the average. With an increase in α, the difference in the weighted load becomes larger. When α is large, the weight of the distance becomes higher and offloading tasks to more distant edge servers is prevented to reduce the load. However, this prevents nodes from being allocated to servers that would produce a better load balance and leads to a larger difference between the loads of edge servers. α can be adjusted based on specific scenarios; for example, if transmission costs are expensive, α could be set higher.

In Fig. 10, an example of the effect of the distance factor α is illustrated using the USNET topology [28], which contains 24 nodes and four edge servers are deployed. The ESP algorithm is used to solve the server placement and task allocation problem. For lower values of α, task allocation in each cluster is more widely dispersed. With an increase in α, each cluster becomes more centralized because the distance becomes more important, so nodes tend to offload their tasks to a closer edge server.



(a) α = 0.5

(b) α = 1.0

(c) α = 2.0

FIGURE 10. The impact of the distance factor α.

V. CONCLUSION

Edge server placement strongly affects the efficiency of task allocation. In this paper, a server placement and task allocation method for an edge-computing network is studied. We formulate the scenario as a mixed-integer linear programming problem and propose a simulated annealing-based edge server placement and task allocation algorithm. To evaluate the performance of the proposed algorithm, different topology sizes are considered, including real-world networks. The impact of the distance factor is also examined in detail. The results show that, by adopting the proposed ESP algorithm, the costs of the largest server, i.e., the objective value in this paper, can be effectively reduced while the runtime remains manageable.

In reality, the ability of each server to handle the workload may differ. Furthermore, mobile users may offload different types of task, and those tasks could be processed by different edge servers. Thus, a more advanced model with different types of task and different computing capacities for the edge servers should be considered in the future.

REFERENCES

[1] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. 5th Int. Joint Conf. (INC, IMS IDC)*, 2009, pp. 44–51.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[3] S. K. U. Zaman, A. I. Jehangiri, T. Maqsood, Z. Ahmad, A. I. Umar, J. Shuja, E. Alanazi, and W. Alasmary, "Mobility-aware computational offloading in mobile edge networks: A survey," *Cluster Comput.*, Apr. 2021.
[4] S. Wan, X. Li, Y. Xue, W. Lin, and X. Xu, "Efficient computation offloading for internet of vehicles in edge computing-assisted 5G networks," *J. Supercomput.*, vol. 76, pp. 2518–2547, Apr. 2020.
[5] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, and W. Dou, "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *J. Netw. Comput. Appl.*, vol. 133, pp. 75–85, May 2019.
[6] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *J. King Saud Univ. Comput. Inf. Sci.*, to be published.
[7] G. Li, Y. Yao, J. Wu, X. Liu, X. Sheng, and Q. Lin, "A new load balancing strategy by task allocation in edge computing based on intermediary nodes," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–10, Dec. 2020.
[8] A. Auslender and M. Teboulle, "Lagrangian duality and related multiplier methods for variational inequality problems," *SIAM J. Optim.*, vol. 10, no. 4, pp. 1097–1115, Jan. 2000.
[9] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2706–2716, Jun. 2013.
[10] G. Athanasiou, P. C. Weeraddana, C. Fischione, and L. Tassiulas, "Optimizing client association for load balancing and fairness in millimeter-wave wireless networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 836–850, Jun. 2015.
[11] J. Lim and D. Lee, "A load balancing algorithm for mobile devices in edge cloud computing environments," *Electronics*, vol. 9, no. 4, p. 686, Apr. 2020.
[12] M. Tang and S. Pan, "A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers," *Neural Process. Lett.*, vol. 41, no. 2, pp. 211–221, 2015.
[13] X. Xu, Y. Xue, X. Li, L. Qi, and S. Wan, "A computation offloading method for edge computing with vehicle-to-everything," *IEEE Access*, vol. 7, pp. 131068–131077, 2019.
[14] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[15] M.-H. Chen, M. Dong, and B. Liang, "Multi-user mobile cloud offloading game with computing access point," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (Cloudnet)*, Oct. 2016, pp. 64–69.
[16] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. 18th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst.*, Nov. 2015, pp. 271–278.
[17] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.
[18] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.
[19] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 181, May 2021, Art. no. 103005.
[20] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, no. 1, p. 32, 2019.
[21] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li, "Edge provisioning with flexible server placement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1031–1045, Apr. 2017.
[22] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 66–73.
[23] X. Xu, B. Shen, X. Yin, M. R. Khosravi, H. Wu, L. Qi, and S. Wan, "Edge server quantification and placement for offloading social media services in industrial cognitive IoV," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2910–2918, Apr. 2021.
[24] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," *Softw., Pract. Exper.*, vol. 50, no. 5, pp. 489–502, May 2020.
[25] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[26] W. Wang and M. Á. Carreira-Perpiñán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," 2013, *arXiv:1309.1541*. [Online]. Available: http://arxiv.org/abs/1309.1541
[27] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, 2009.
[28] N. L. Van Adrichem, B. J. Van Asten, and F. A. Kuipers, "Fast recovery in software-defined networks," in *Proc. 3rd Eur. Workshop Softw. Defined Netw.*, Sep. 2014, pp. 61–66.

PING-CHUN HUANG received the B.S. degree from the Department of Medical Informatics, Chung Shan Medical University, Taiwan, in 2019, and the M.S. degree from the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, in 2021. His current research interests include edge computing and industrial networks.

TAI-LIN CHIN (Member, IEEE) received the B.S. degree in computer science and information engineering from the National Chiao-Tung University, Taiwan, in 1995, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Wisconsin–Madison, in 2004 and 2006, respectively. Since 2007, he has been with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, where he is currently an Associate Professor. His research interests include wireless networking, mobile computing, cloud computing, and sensor networks.

TZU-YI CHUANG is currently working toward the M.S. degree in the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology. His current research interests include edge computing and industrial networks.