Risk-Aware Application Placement in Mobile Edge Computing Systems: A Learning-based Optimization Approach

Hossein Badri*, Tayebeh Bahreini*, Daniel Grosu* and Kai Yang†
*Department of Computer Science, Wayne State University, Detroit, MI, USA
Email: {hossein.badri, tayebeh.bahreini, dgrosu}@wayne.edu
†Department of Industrial & Systems Engineering, Wayne State University, Detroit, MI, USA
Email: kai.yang@wayne.edu

***Abstract*—In this paper, we address the problem of application placement in MEC systems that takes into account the risk of exceeding the energy budget of the edge servers. We formulate the problem as a chance-constrained program, where the objective is to maximize the total quality of service in the system, while keeping the expected risk of exceeding the edge servers' energy budget within an acceptable threshold. We develop a learning-based method to solve the problem which requires a very small execution time for large size instances. We evaluate the performance of the proposed method by conducting an extensive experimental analysis.**

***Keywords*-Mobile edge computing; risk-aware application placement; stochastic optimization, machine learning.**

I. Introduction

Mobile Edge Computing (MEC) has been introduced with the aim of providing services to mobile users with a lower latency compared to the centralized data centers, which is based on the idea of bringing servers closer to the end users [1], [2]. Thus, a lower communication latency is expected to be achieved by locating the servers at the edge of the network. Due to the fact that MEC servers are more restricted in terms of computational capacity compared to the cloud servers, the resources must be utilized efficiently in order to take full advantage of these systems. One of the key decisions determining the efficiency of MEC systems is how to allocate applications to servers. Given the fact that MEC servers have a restricted energy budget, allocating too many tasks to those servers can result in the failure of services. On the other hand, an inefficient application placement employed by service providers can significantly impact the Quality of Service (QoS).

What makes the application placement in MEC systems more challenging is the existence of nondeterministic parameters such as the size of requests, the reliability of the network, and the movement of users. Due to the high volatility of the MEC systems, any optimal application placement might turn to a non-optimal or even infeasible placement in few seconds. Thus, to make an application placement reliable one needs to take the uncertainties of the network into account when placing applications on edge servers.

Several researchers have recently addressed the application placement problem in MEC from different perspectives and employed a wide range of methods. Due to the fact that the main goal of MEC systems is to reduce latency, many researchers have devoted their efforts to developing methods for improving the QoS. Ouyang et al. [3] proposed a service placement method which jointly optimizes users' perceived-latency and service migration costs. Maia et al. [4] proposed two algorithms for IoT service placement, where the objective is to minimize the potential violation of QoS requirements. Wang et al. [5] modeled the application placement as a graph problem and proposed an algorithm to find the optimal placement of a linear application graph. They also proposed online approximation algorithms for application placement. Sodhro et al. [6] proposed an algorithm for optimization of QoS in MEC-based medical video stream applications.

The necessity of improving the efficiency of energy consumption in MEC systems motivated researchers to take the energy consumption into account in the design of application placement algorithms [7]. Bahreini et al. [8] addressed the problem of energy-aware capacity provisioning and resource allocation in MEC systems. The objective in their proposed algorithm is to maximize the net profit of the service provider, where the profit is the difference between the aggregated users' payments and the total operating cost due to energy consumption. Badri et al. [9] considered the mobility of users in MEC as a non-deterministic parameter and proposed a multi-stage stochastic program for energy-aware application placement.Yang et al. [10] proposed an algorithm for resource management in MEC networks where the objective is to minimize the power utilization. The authors took latency and coverage into account in their proposed algorithm.

One of the key assumptions in the above-mentioned works is that the workload of mobile applications is known prior to allocating it to servers. However, this assumption might not be valid for many applications, that is, when a request is received by the MEC system, the workload is unknown. Compared to the cloud servers, MEC servers are expected to be more restricted in terms of capacity, that is, the

risk of exceeding the energy budget of servers is higher in MEC servers. Exceeding the energy budget may result in increased failure risk. One possible solution is to have a recovery scheme in place and offload the workload of an overloaded edge server to other servers with enough available capacity [11]. However, this approach does not manage the risk and might result in higher latency due to migrations. In order to manage the risk of exceeding the energy budget, one has to take the variance of workloads into account when allocating requests to MEC servers. Disregarding the stochasticity of applications' workloads might end up in either under-utilized servers or high failure risk. To the best of our knowledge, no application placement algorithm has been proposed that takes uncertainty of the loads into account and manages the risk of exceeding the energy budget of servers in MEC systems.

Contributions. We propose a novel risk-based optimization approach to application placement in MEC that takes into account the risk of exceeding the energy budget of edge servers when making allocation decisions. Our objective is to maximize the QoS of the system which is defined with respect to the communication latency and is the sum of QoS of individual requests who receive services. We also take into account the risk of exceeding the energy budget of edge servers with the aim of controlling the failure risk. Because the resource requirements of mobile applications are stochastic parameters, and in order to control the risk of exceeding the energy budget of the edge servers, we formulate the application placement problem as a *chance-constrained stochastic program*. To solve the problem, we employ the Sample Average Approximation (SAA) method [12] and develop a fast machine learning-based optimization method to solve fairly large size instances. We develop a model using some basic features that can be used for application placement without having to solve any optimization problems online. We also propose a method where in addition to the basic features, some additional features from the solution of the linear programming (LP) relaxation of the problem are used to train the placement model. We evaluate the efficiency of the proposed approach by conducting an experimental analysis on instances with various problem settings based on real-world data.

Organization. The rest of this paper is organized as follows. Section II defines the problem and presents the risk-based optimization model. Section III is devoted to the description of our learning-based application placement method. Section IV describes the experimental setup and analyzes the results. Finally, Section V concludes the paper and presents possible directions for future research.

II. Risk-based Optimization Model

We consider a MEC system composed of edge servers and a cloud data center. The cloud servers situated in the data center are abstracted as a single cloud server of large capacity denoted by S_c. There are M^e servers at the edge level. The set of servers at the edge level is denoted by $\mathcal{M}^e = \{S_1, S_2, \ldots, S_{M^e}\}$, while the set of all servers including the cloud server is denoted by $\mathcal{M} = \{S_1, S_2, \ldots, S_{M^e}, S_c\}$. These servers provide computing resources to a set $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$ of N independent requests from mobile users. The MEC network model assumes that all cell towers are accessible to all users in the network, and that the cloud server is accessible through the cell towers. Our model is analogous to the network model considered in [3]. In our model, edge server S_j is co-located with a base station and is characterized by its *computational capacity*, C_j, and its *energy budget*, E_j expressed in Joules. The computational capacity C_j is the maximum number of unit-size containers that server S_j can host. The unit-size containers have a fixed resource configuration which is determined by the provider. The cloud server is located in a geographical area at a large distance from the users, and the cloud server is assumed to have enough capacity to handle all the requests from mobile users. We denote the distance from user i to server S_j by d_{ij}. The request U_i of user i is to offload and execute an application (a single container) on the edge or the cloud servers via 4G/5G/WiFi access networks. The request U_i is characterized by two parameters, (i) the amount of transferred data t_i, and (ii) the size of the requested container R_i, which is expressed in terms of the equivalent number of unit-size containers needed to execute the application. We assume that each created container serves only one request. We formulate the energy-aware application placement problem in MEC systems as a chance-constrained stochastic program. Chance-constrained stochastic programming was introduced by Charnes et al. [13] and has been employed extensively in a variety of applications.

The objective function is to *maximize the quality of service* of the system, which is defined as the sum of the QoS of individual requests who receive services. Here, we define the QoS based on two important factors that determine the latency, the distance between user and server, and the size of the transferred data from the mobile device to the system. Thus, the *quality of service* that user U_i receives from server S_j is defined as, $Q_{ij} = \frac{t_i}{d_{ij}}$, where, t_i is the size of data transferred from user i, and d_{ij} is the distance between user i and server S_j. In other words, the *quality of service* Q_{ij} that request U_i receives from server S_j is inversely proportional to the distance between the user and the server that provides the service, and directly proportional to the size of the transferred data. Here, we assume that the distance between users and servers is the Manhattan distance. Therefore, if user i is located at (a_u, b_u) and server S_j is located at (a_s, b_s), then the distance between the user and the sever is given by $|a_u - a_s| + |b_u - b_s|$. We assume that the amount of energy utilized by the request U_i when

it runs on server S_j is given by $\frac{\gamma \cdot R_i}{C_j}$, where γ is a constant coefficient. Thus, the energy utilized by the request U_i when it runs on server S_j is directly proportional to the fraction of S_j's capacity utilized by the request [14].

We formulate the energy-aware application placement as a chance-constrained integer program as follows.

$$\text{Maximize} \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{M}} \frac{t_i}{d_{ij}} \cdot x_{ij} \tag{1}$$

Subject to:

$$p\left\{ \sum_{i \in \mathcal{U}} \frac{\gamma \cdot R_i}{C_j} \cdot x_{ij} \leq E_j \right\} \geq (1-\alpha), \quad \forall j \in \mathcal{M}^e \tag{2}$$

$$\sum_{j \in \mathcal{M}} x_{ij} = 1, \quad \forall i \in \mathcal{U} \tag{3}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in \mathcal{U}, j \in \mathcal{M} \tag{4}$$

where x_{ij} is a binary variable, that is 1 if the request of user i is allocated to server S_j and, 0 otherwise. Constraint (2) ensures that the probability of loading an edge server beyond its energy budget is not greater than the risk factor α. Constraint (3) ensures that each request is satisfied and is not allocated to more than one server. Finally, Constraint (4) guarantees the integrality of the decision variables.

Chance-constrained stochastic programs may be extremely hard to solve, due to the nonconvexity and feasibility checking issues. Several approaches have been proposed to solve chance-constrained programs efficiently [12], [15], [16]. Here, we employ the Sample Average Approximation (SAA) method which is a Monte Carlo simulation-based approach to solve chance-constrained programs [17]. In SAA, the actual distribution in the chance constraint is replaced by an empirical distribution of a random sample. In the following, we present the SAA formulation of the problem.

Let us define the unutilized energy budget of server S_j as,

$$G_j(x, \Theta^\xi) = E_j - \sum_{i \in \mathcal{U}} \frac{\gamma \cdot \tilde{R}_i^\xi}{C_j} \cdot x_{ij}, \qquad \forall j \in \mathcal{M}^e \tag{5}$$

where, $\tilde{R}_i^\xi$ is the realization of parameter R_i based on scenario ξ on the size of requests. Then, we can formulate the SAA problem as a mixed-integer program (MIP),

$$\text{Maximize} \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{M}} \frac{t_i}{d_{ij}} \cdot x_{ij} \tag{6}$$

Subject to:

$$G_j(x, \Theta^\xi) \geq W \cdot z_\xi, \quad \forall j \in \mathcal{M}^e, \xi \in \Theta \tag{7}$$

$$\sum_{\xi \in \Theta} z_\xi \leq \alpha \cdot \varphi \tag{8}$$

$$\sum_{j \in \mathcal{M}} x_{ij} = 1, \quad \forall i \in \mathcal{U} \tag{9}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in \mathcal{U}, j \in \mathcal{M} \tag{10}$$

$$z_\xi \in \{0,1\}, \quad \forall \xi \in \Theta \tag{11}$$

where the objective function in Equation (6) is the same as that of the original chance-constrained model in Equation (1). In Constraint (7), W is a very large negative integer, and z_ξ is a binary variable. If z_ξ is 1, the energy budget constraint can be violated under realization of scenario ξ, and if it is 0, otherwise. Also, Θ is the independent identically distributed (iid) sample of φ realizations of $\tilde{R}_i^\xi$. Constraint (8) guarantees that the number of violated capacity constraints is not greater than $\alpha \cdot \varphi$. Constraints (9) and (10) were described in the chance-constrained model. Constraint (11) guarantees the integrality of z_ξ.

III. Learning-based Application Placement

An efficient application placement algorithm for MEC systems has to be very fast to be useful in practice. Since the proposed SAA program is a MIP model, it might not be feasible to obtain the optimal solution in a reasonable amount of time, especially with a large number of users. In this research, we develop a fast machine learning-based algorithm to solve the SAA model.

In recent years, several researchers have attempted to leverage machine learning techniques to either solve combinatorial optimization problems or improve the performance of solvers on these problems. Alvarez et al. [18] developed a method for variable branching in branch-and-bound which is based on imitating the decisions taken by a strong branching strategy with an approximation. The approximation is obtained via a machine learning technique from a set of observed branching decisions taken by strong branching. Khalil et al. [19] also proposed a machine learning framework for variable branching in MIP. Kool et al., [20] proposed an algorithm based on reinforcement learning to solve combinatorial optimization problems. Nazari et al. [21] also developed a framework for solving the vehicle routing problem using reinforcement learning. Bertsimas and Stellato [22] focused on Mixed-integer Quadratic Optimization problems and transformed the optimization algorithm to a multi-class classification problem. They proposed a fast online optimization algorithm consisting of a feedforward neural network evaluation and a linear system solution. Bengio et al. [23] surveyed the recent attempts at leveraging machine learning to solve combinatorial optimization problems.

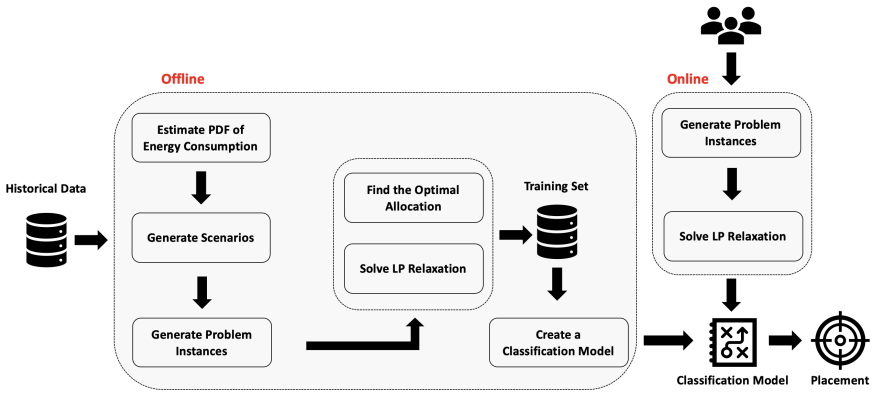


Figure 1: Learning-based Application Placement (LBAP) Framework.

Here, we leverage machine learning techniques to develop a *Learning-based Application Placement* (LBAP) algorithm which is a very fast algorithm to solve the SAA model for the energy-aware application placement in MEC systems. The framework of the LBAP algorithm is illustrated in Figure 1. It consists of two major components, offline and online. The role of the *offline component* is to create a classification model based on the historical data on the size of the transferred data as well as the resource requirements of applications. In this classification model, each request is considered as an observation, a server is considered as a class, and the true class of each observation (request) is the allocated server according to the optimal solution. Later in this section, we will describe the features, the target variables, and the structure of the training set. The output of the *offline component*, which is a trained classification model, is employed in the *online component* for the placement of requests.

The first step in the *offline component* is to estimate the Probability Distribution Function (PDF) of the resource utilization of applications. Then, the PDF is used to generate scenarios for the resource requirements of applications ($\tilde{R}_i^{\xi}$). Once the scenarios are generated using the PDF, a problem instance is created. Since in the classification model, the true class of each observation (request) is the allocated server according to the optimal solution, an exact solution method has to be employed in order to label observations. We also solve the Linear Programming (LP) relaxation model of the problem to extract some features from the solution of this model which will be used to improve the accuracy of the classification model. The LP relaxation model is created by replacing Constraint (10) and (11) with, $x_{ij} \geq 0,\ \forall i \in \mathcal{U}, j \in \mathcal{M}$, and $z_{\xi} \geq 0,\ \forall \xi \in \Theta$. Finally, features and the target values corresponding to the current problem instance are added to the training set. The process described above is repeated multiple times and for instances of different sizes. Once the training set is large enough, we use a multi-class classification method to create the classification model. Since this procedure is performed in an offline fashion, the complexity of the solution algorithm and the classification method is not a challenge. In our proposed approach, it is assumed that the number of servers is fixed, but every time, the MEC system might receive a different number of requests.

The *online component* of the LBAP uses the classification model for application placement in an online fashion. Every time the MEC system receives a batch of requests, it generates scenarios for the resource requirements of the received requests. We should note that the size of the transferred data is deterministically known at this stage. Then the LP relaxation model is solved and the value of all the features are obtained. The trained classification model is then used for application placement.

Training set. In the training set, each row (observation) corresponds to one request. Let us denote the observation corresponding to request U_i by tuple (F_i, y_i), where F_i is the set of features corresponding to user i, and y_i is the target variable which is the label of each request. The label of each request is the index of the allocated server and is obtained by solving the SAA model of the application placement problem and finding the optimal placement. Therefore, $y_i = \arg\max_j \bar{x}_{ij}$, where, $\bar{x}_{ij}$ is the optimal solution of the SAA problem. The set of features needs to be carefully created such that the structure of the instance and the main characteristics of each request are captured accurately. Since the LBAP model should be capable of solving instances of different sizes without having to be trained for all the possible sizes, all the features should be defined to be independent of the size of the instance.

The first subset of features is created using *statistics of the chance constraint*. We denote by

$$f_{ij\xi}^{(1)} = \frac{\gamma \cdot \tilde{R}_i^{\xi}/C_j}{E_j},\ \forall j \in \mathcal{M}^e, \xi \in \Theta, \tag{12}$$

the ratio of the energy consumed by request U_i on server S_j and the energy budget of S_j. The features in this subset are, $\min_{i\in\mathcal{U}} \left\{f_{ij\xi}^{(1)}\right\}$, $\max_{i\in\mathcal{U}} \left\{f_{ij\xi}^{(1)}\right\}$, $\sigma_{i\in\mathcal{U}} \left\{f_{ij\xi}^{(1)}\right\}$, $\mu_{i\in\mathcal{U}} \left\{f_{ij\xi}^{(1)}\right\}$, where σ_i and μ_i denote the standard deviation

and the mean value. The values of the features in this subset are identical across all the requests.

The second subset of features consists of *statistics of requests*. We normalize these features to make them independent of the size of instances. We denote by

$$f_{ij\xi}^{(2)} = \frac{\gamma \cdot \tilde{R}_i^{\xi}/C_j}{\sum_{i\in\mathcal{U}} \gamma \cdot \tilde{R}_i^{\xi}/C_j},\ \forall j \in \mathcal{M}^e, \xi \in \Theta, \tag{13}$$

the normalized energy consumed by request U_i on server S_j. The features in this subset are, $\min_{i\in\mathcal{U}} \left\{ f_{ij\xi}^{(2)} \right\}$, $\max_{i\in\mathcal{U}} \left\{ f_{ij\xi}^{(2)} \right\}$, $\sigma_{i\in\mathcal{U}} \left\{ f_{ij\xi}^{(2)} \right\}$, $\mu_{i\in\mathcal{U}} \left\{ f_{ij\xi}^{(2)} \right\}$, where σ_i and μ_i denote the standard deviation and the mean value of the feature. Similar to the first subset of features, the value of the features in this subset are identical across all the requests.

The third subset of features captures the main *statistics corresponding to each request*. We denote by

$$f_{ij\xi}^{(3)} = \frac{t_i/d_{ij}}{\gamma \cdot \tilde{R}_i^{\xi}/C_j},\ \forall j \in \mathcal{M}^e, \xi \in \Theta, \tag{14}$$

the ratio of the QoS of user i when the request is allocated to server S_j and the energy consumed by request U_i on S_j. In contrast to the first two sets of features, this subset consists of features that might have different values across requests. These features are normalized to make them independent of the size of instances. The features in this subset are, $\min_{\xi\in\Theta} \left\{ f_{ij\xi}^{(3)} \right\}$, $\max_{\xi\in\Theta} \left\{ f_{ij\xi}^{(3)} \right\}$, $\sigma_{\xi\in\Theta} \left\{ f_{ij\xi}^{(3)} \right\}$, $\mu_{\xi\in\Theta} \left\{ f_{ij\xi}^{(3)} \right\}$, where σ_i and μ_i denote the standard deviation and the mean value of the feature.

The fourth subset of features consists of the *normalized value of the transferred data size* that is generated for all possible combinations of (U_i, S_j). The features in this subset are,

$$f_{ij}^{(4)} = \frac{t_i/d_{ij}}{\sum_{k\in\mathcal{U}} t_k/d_{kj}},\ \forall i \in \mathcal{U}, j \in \mathcal{M}^e. \tag{15}$$

The last subset of features corresponds to the *value of the decision variables in the optimal solution of the LP relaxation model*,

$$f_{ij}^{(5)} = \dot{x}_{ij},\ \forall i \in \mathcal{U}, j \in \mathcal{M}, \tag{16}$$

where $\dot{x}_{ij}$ is the value of x_{ij} in the optimal solution of the LP relaxation model.

Machine learning algorithm. In LBAP, we employ multi-class classification to obtain allocation of requests to servers. The multi-class classification problem is to assign each observation into one of the classes [24]. In LBAP, each request is considered as an observation, and servers are the classes. Since each request can be allocated to any server, we take the allocation obtained in the optimal solution of the SAA model as the true classification. To create the LBAP function, we use the XGBoost package [25] which is based on a tree boosting algorithm and has been widely employed for classification problems in several applications.

IV. Experimental Analysis

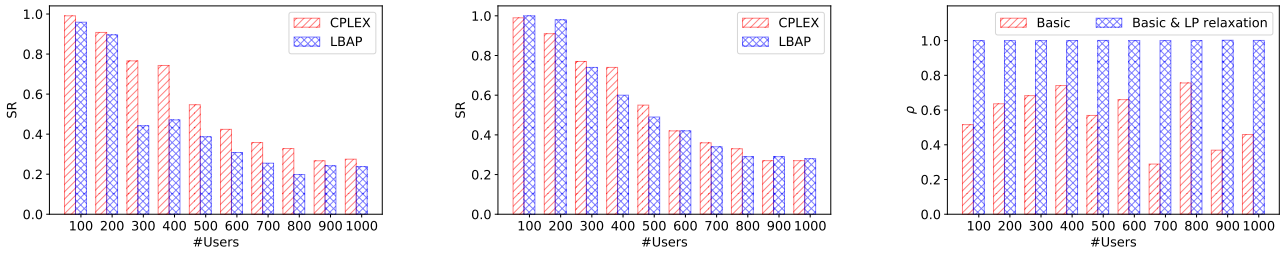
To evaluate the performance of the proposed application placement algorithm (LBAP), we perform an extensive experimental analysis using real-world data. We aim at evaluating the quality of solutions and the running times of the proposed method for problem instances of different sizes. We investigate the performance of the LBAP model when only basic features are considered in the classification model, and evaluate the impact of considering features from the solution of the LP relaxation model in addition to the basic features.

A. Experimental setup

For our analysis, we consider a MEC system with five edge servers and a cloud server which is located far from the users' area. We consider that the edge servers are co-located with the base stations. For the edge servers, we choose the Dell PowerEdge R740 Rack Server (Intel Xeon Gold 6240, 2.4GHz, 24 cores, 32GB RAM).

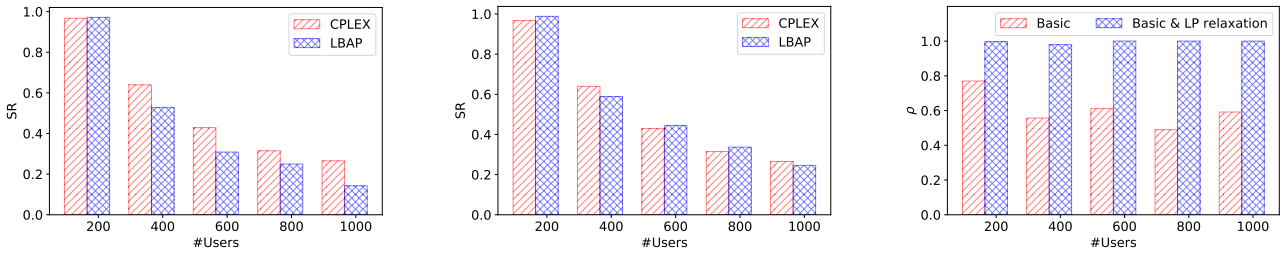
In our analysis, the number of requests ranges from 100 to 1000. In order to ensure high quality services, we define the unit-size container according to the computational capacity of a single core, and at most one container is allocated to a core. Therefore, the computational capacity, C_j, of a server is equivalent to the number of CPU cores. According to a survey on data centers energy consumption [26], the idle power of an edge server accounts for 60% of the full state power. Thus, in our setting, the energy budget of a server (in Joules) for a time slot is: $E_j = 0.4 \times 495\text{W} \times 120\text{s} = 23,760\,\text{J}$, where 495 W is the total power consumption of a Dell PowerEdge R740 Rack Server, and 120 seconds represents the length of the time slot considered in the experiments.

We assume that if a core is utilized, it requires full power. Therefore, the power of each core is obtained by dividing the total power of a server by the number of cores. Based on this assumption the power of each core of the PowerEdge R740 Rock server is 20.625 W. We also assume that when a container is allocated to multiple cores, all the cores are fully utilized. Therefore, the utilization rate of a server can be simply obtained by dividing the allocated cores by the total number of cores. We use the dataset on smartphones [27], to estimate the PDF of the size of the containers requested by users. In order to generate scenarios, we use the Categorical distribution, where each category corresponds to each possible size of requests. The probability of each category is obtained by dividing the number of observations of that category in the dataset by the total number of observations in the dataset. Throughout the experiments, the sample size is 50 scenarios.



(a) Service Ratio: Basic features (b) Service Ratio: Basic & LP features (c) QoS ratio

Figure 2: Service ratio & QoS ratio vs. number of users



(a) Service Ratio: Basic features (b) Service Ratio: Basic & LP features (c) QoS ratio

Figure 3: Service ratio & QoS ratio vs. number of users: unobserved size of instances

We compare the performance of LBAP with that of the optimal placement obtained by solving the problem with CPLEX [28] considering two main metrics. The first metric is the *Service Ratio* (SR) which is defined as the fraction of the total requests allocated to edge servers, that is, $SR = \frac{\sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{M}^e} x_{ij}}{N}$, where, x_{ij} is 1 if request U_i is allocated to the edge server S_j, and 0 otherwise. The second metric is the *QoS Ratio*, which is defined as, $\rho = \frac{Q^{LBAP}}{Q^{CPLEX}}$ where, Q^{CPLEX} is the optimal value of the QoS obtained by CPLEX, and Q^{LBAP} is the value of the QoS obtained by LBAP. For the experiments involving CPLEX, we use the CPLEX 12 solver provided by IBM ILOG CPLEX optimization studio for academics initiative [28]. Each experiment is performed five times and the analysis is conducted based on the average value of the metrics.

B. Experimental results

We perform our analysis using two different approaches. First, we investigate the performance of the proposed method by not considering the features from the solution of the LP relaxation model. We aim at investigating the accuracy of the LBAP model when only features from the SAA MIP model are included in the classification model. In this approach, we do not need to solve the LP relaxation model, and the placement is obtained without having to solve any optimization model in the online fashion. In the second part of our experimental analysis, we include the features from the solution of the LP relaxation model. Based on this approach, once the system receives the requests, the LP relaxation model is solved and the optimal value of the decision variables are used as features to generate the placement using the LBAP model.

We perform our analysis on instances of various sizes, where the size of an instance is specified by the number of users. In these experiments, the number of users ranges from 100 to 1000. Figure 2 compares the service and QoS ratios obtained by LBAP with the optimal ratios obtained by solving the MIP model using the CPLEX solver. In Figure 2a, we observe that the service ratio decreases as the number of users increases. The reason is that the capacity of the edge servers is fixed, and therefore as the number of users increases, a smaller fraction of them can be allocated to the edge servers. In most cases, the performance of the LBAP method based on the service ratio is within 0.1 distance from the optimal solution. Figure 2b shows the service ratio obtained by LBAP, where the LP relaxation features are used in addition to the basic features in the classification model. As expected, LBAP performs significantly better when the LP relaxation features are used in the classification model, and the service ratio obtained by LBAP is fairly close to that of CPLEX. In few instances, we observe that LBAP obtains a slightly higher service ratio compared to CPLEX. One possible reason for this is that a higher service ratio might not necessarily yield a higher QoS, which is the objective of the problem. Given this fact, CPLEX might obtain a solution that has a smaller fraction of users allocated to edge servers, but QoS of the system is maximized.

Figure 2c shows the performance of LBAP in terms of QoS. In this figure, we observe that considering only

the basic features, the QoS obtained by LBAP is within an acceptable distance from the optimal QoS obtained by CPLEX. In most instances, the QoS obtained by LBAP considering only the basic features is within 0.5 distance from the optimal QoS. We do not observe any significant decrease in ρ with the increase in the number of users. By including the LP relaxation features, the QoS obtained by LBAP gets very close to the optimal. We observe that in few cases the QoS ratio is slightly greater than 1, that is, the QoS obtained by LBAP is slightly greater than the optimal QoS obtained by using CPLEX. This indicates that the placement obtained by LBAP might have a higher risk of exceeding the energy budget than the placement obtained by CPLEX. This is not a significant issue, given that the risk is a soft constraint and could be violated under some scenarios. This also could be tackled through training the LBAP classification model with a higher value for the risk factor, α.

Generalization analysis. In the first set of experiments, the classification model was trained for all the sizes of instances in the test set. In the real world, it might not be practical to train the classification model using all possible sizes of instances. Therefore, we might need to train a model using only a subset of sizes. Here, we investigate the performance of LBAP when it is used for instances with sizes that were not included in the training set. For this purpose, we train the model using instances with 100, 300, 500, 700, and 900 requests and test it on instances with 200, 400, 600, 800, and 1000 requests. We should emphasize that none of the instances in the test set were included in the training set of the classification model. This setup provides an insight on the generalization power of LBAP.

Figure 3 shows the performance of LBAP for unobserved instances based on the service and QoS ratios. Figure 3a shows the service ratio versus the number of users when only basic features are used in the classification model. We observe that the service ratio of LBAP for unobserved instances is fairly close to that in the optimal solution. Figure 3b shows the service ratio when basic and LP relaxation features are used. We observe that LBAP performs very well when it is used for the sizes of instances that were not included in the training set. Similarly to the results in Figure 2b, we observe that in some instances the service ratio of LBAP is greater than that of the optimal solution obtained by CPLEX. The same justification applies here, the objective is to maximize QoS of the system, and a higher service ratio does not necessarily result in a higher QoS. Figure 3c shows the performance of LBAP in terms of the QoS ratio for unobserved instances. We observe that when only basic features are used in the classification model, the QoS obtained by LBAP is within 0.5 distance from the optimal QoS obtained by CPLEX. Based on these results, the size of instances does not have a significant impact on ρ.

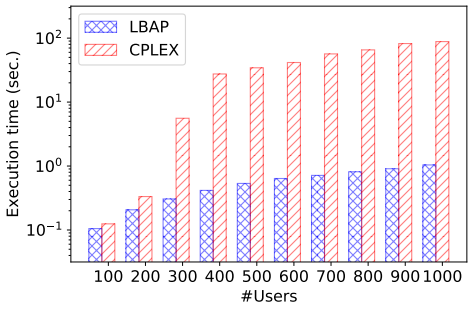


Figure 4: Execution time vs. number of users

We observe that with the complete set of features, i.e., basic and LP relaxation features, the performance of LBAP in terms of QoS is very close to the optimal solution.

Our experimental results on the generalization power of LBAP confirm that this method performs very well when applied to unobserved instances. These results indicate that LBAP can be trained for a subset of instances and be efficiently used for any unobserved instances. It is important to note that the selection of the subset of instances to train the classification model is of high importance, and can impact the performance of the method. For example, it might not be efficient to train the model using only small size of instances and use it for significantly larger sizes.

Execution time analysis. We now investigate the execution time of LBAP. Since the execution time for extracting the features is negligible, we focus our analysis on the execution time of LBAP with LP relaxation features. For the sake of fairness, we use the CPLEX solver to solve the MIP model of the SAA method and the LP relaxation model in LBAP. Figure 4 compares the execution time required for solving the MIP model of the SAA method, labeled as CPLEX, with that of solving LBAP with LP relaxation features. We observe that the execution time for solving the MIP model increases at a much higher rate than that of the LBAP method. We observe that for an instance with 1000 users, the MIP model is solved in about 100 seconds, while the LBAP method is solved in about one second. These results imply that the execution time of LBAP with basic and LP relaxation features is within an acceptable amount of time and this method is suitable for deployment in real MEC systems.

V. Conclusion

We developed a risk-based optimization model for application placement in edge computing systems and proposed a Learning-based Application Placement (LBAP) method which is capable of solving large size problem instances within a second. We performed an extensive experimental analysis to investigate the performance of the proposed method compared to the optimal solution obtained by solving the MIP model using the CPLEX solver. Our results

showed that LBAP is very efficient for solving large size problem instances. As a future research, we plan to apply the proposed method to solve other resource management problems in MEC systems, where parameters are nondeterministic.

ACKNOWLEDGMENT

This research was supported in part by the US National Science Foundation under grant no. IIS-1724227.

REFERENCES

[1] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conference on Advances in Future Internet*, 2014, pp. 48 – 54.

[2] B. Liang, "Mobile edge computing," *Key Technologies for 5G Wireless Systems*, p. 76, 2017.

[3] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE Int. Conf. on Computer Communications*, April 2019, pp. 1468–1476.

[4] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "Optimized placement of scalable iot services in edge computing," in *IFIP/IEEE Symp. on Integrated Network and Service Management*, April 2019, pp. 189–197.

[5] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[6] A. H. Sodhro, Z. Luo, A. K. Sangaiah, and S. W. Baik, "Mobile edge computing based qos optimization in medical healthcare applications," *Int. J. Information Management*, vol. 45, pp. 308–318, 2019.

[7] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Comm. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.

[8] T. Bahreini, H. Badri, and D. Grosu, "Energy-aware capacity provisioning and resource allocation in edge computing systems," in *Int. Conf. on Edge Computing*. Springer, 2019, pp. 31–45.

[9] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 4, pp. 909–922, April 2020.

[10] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in uav-enabled mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4576–4589, 2019.

[11] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138 – 147, 2017.

[12] S. Ahmed and A. Shapiro, "Solving chance-constrained stochastic programs via sampling and integer programming," in *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*. INFORMS, 2008, pp. 261–269.

[13] A. Charnes, W. W. Cooper, and G. H. Symonds, "Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil," *Management Science*, vol. 4, no. 3, pp. 235–263, 1958.

[14] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007.

[15] A. Nemirovski and A. Shapiro, "Scenario approximations of chance constraints," in *Probabilistic and randomized methods for design under uncertainty*. Springer, 2006, pp. 3–47.

[16] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro, "Sample average approximation method for chance constrained programming: theory and applications," *Journal of Optimization Theory and Applications*, vol. 142, no. 2, pp. 399–416, 2009.

[17] J. Luedtke and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 674–699, 2008.

[18] A. M. Alvarez, Q. Louveaux, and L. Wehenkel, "A machine learning-based approximation of strong branching," *INFORMS J. Computing*, vol. 29, no. 1, pp. 185–195, 2017.

[19] E. B. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Thirtieth AAAI Conf. on Artificial Intell.*, 2016.

[20] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[21] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems*, 2018, pp. 9839–9849.

[22] D. Bertsimas and B. Stellato, "Online mixed-integer optimization in milliseconds," *arXiv preprint arXiv:1907.02206*, 2019.

[23] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *arXiv preprint arXiv:1811.06128*, 2018.

[24] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *J. of Machine Learning Res.*, vol. 5, no. Aug, pp. 975–1005, 2004.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.

[26] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[27] Y. Mirsky, A. Shabtai, L. Rokach, B. Shapira, and Y. Elovici, "Sherlock vs moriarty: A smartphone dataset for cybersecurity research," in *Proc. ACM Workshop on Artificial Intelligence and Security*, 2016, pp. 1–12.

[28] (2009) IBM ILOG CPLEX V12.1 user's manual. [Online]. Available: ftp://ftp.software.ibm.com/software/websphere/ilog/docs/