Computation Offloading over Fog and Cloud using Multi-Dimensional Multiple Knapsack Problem

Junhua Wang*, Tingting Liu†, Kai Liu*, BaekGyu Kim‡, Jiang Xie§ and Zhu Han¶‖
*College of Computer Science, Chongqing University, Chongqing, China
†School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China
‡Toyota InfoTechnology Center, Mountain View, CA, USA
§Department of Electrical and Computer Engineering, The University of North Carolina at Charlotte, Charlotte, NC, USA
¶Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA
‖Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea

***Abstract*—Computation offloading over fog and cloud is critical to improve service quality and efficiency of future networks. Mobile vehicles have also been considered as potential fog nodes by sparing their computation capability to nearby users. In this paper, we propose a multi-layer computation offloading architecture, consisting of the user layer, mobile fog layer, fixed fog layer and cloud layer. Multiple wireless roadside units (RSUs) are deployed in the network to collect computation tasks from user layer, and offload the tasks to other layers. Each layer has distinct multi-dimensional characteristics, such as different transmission rates and computation capabilities. The computation tasks may consume different communication and computation resources when they are uploaded to different layers. However, the available resources of each layer are limited. Consider that each user will pay for the offloaded computation tasks according to their sizes, we aim to maximize the total profits of computation offloading from the infrastructure perspective. Specifically, the offloading problem is formulated as a generalized multi-dimensional multiple knapsack problem (MMKP), in which each layer is considered as a large knapsack and the computation tasks are treated as items. We propose a modified branch-and-bound algorithm to obtain the optimal solution, and a heuristic greedy method to obtain approximate performance with much lower computational overhead. A comprehensive simulation is conducted to compare the proposed two algorithms. Simulation results demonstrate that the proposed computation offloading architecture together with the task allocation algorithms can achieve the purpose of maximizing the total profits of offloaded tasks.**

I. Introduction

Recently, cloud computing has been widely adopted as an efficient solution to the extension of network computation capacity and application demands. However, it gradually becomes insufficient to support the phenomenal growth of big data generated by billions of intelligent devices. Fog computing, which was first introduced by Cisco in 2012 to address the challenges of IoT applications, has been considered as a promising solution to breakthrough the bandwidth bottleneck of wired/wireless access networks in cloud computing. Basically, fog nodes act as efficient spots in the middle layer between data sources and cloud by providing their computation, storage and networking resources. Since fog nodes are more accessible to end users, fog computing is expected to benefit from the advantages of geo-distribution, location awareness and fast response of fog nodes.

Currently, abundant researches have been developed on five types of fog nodes, namely, servers, networking devices, cloudlets, base stations and vehicles [1]. The fog servers can be geo-distributed at public places such as shopping centers and parks. As traditional network devices, gateway routers and switches can act as fog nodes and share their system resources with requesting users. In addition, cloudlets and base stations are expected to extend cloud services for seamless network communication. Furthermore, vehicles at network edge may also have attractive incentives to provide their computation facilities as fog nodes. With powerful onboard units (OBUs) and onboard computers installed on vehicles, they can download the computation tasks from roadside units (RSUs), and cooperatively handle with the requirements by forming a mobile fog layer.

Note that fog computing is not intended to replace cloud computing but to provide complimentary services [2]. Great efforts have been put on computation offloading between cloud and fog nodes. The work in [3] studied the workload offloading strategy to improve quality of experience (QoE). A fog node can forward part of its workload to other local fog nodes or to cloud with extra power consumption. Then, the tradeoff between QoE of users and power efficiency of fog nodes are analyzed before applying the optimal algorithm, which is based on alternative direction method of multipliers (ADMM). The work in [4] presented an architecture of vehicular fog computing (VFC) to augment the computation and storage power of fog computing architecture. Consider that parked vehicles nearby a shopping mall may have tremendous unexploited computing power, the maximum computation capacity of parked vehicles in a VFC zone is determined while predicting the need of computational resources.

In this paper, we consider a multi-layer network architecture, in which multiple RSUs are distributed to collect the computation tasks from user layer. The roadside cameras

* K. Liu is the corresponding author.
The authors would like to thank Dr. Chung-Wei Lin and Dr. Shinichi Shiraishi of Toyota InfoTechnology Center for their helpful suggestions.

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

and sensors can be the potential users. For example, the roadside cameras may collect traffic information without enough process capabilities to analyze them. They can upload their computation tasks to nearby RSUs, which will make the offloading decisions based on a certain criteria. There are three options for computation offloading. First, the computation tasks can be offloaded to the mobile fog layer. The mobile vehicles complete the computation tasks by utilizing the idle computation resources of onboard computers. Once the computation is finished, corresponding results will be returned to the nearby RSU directly or through multihop vehicle-to-vehicle (V2V) communications. Second, the computation tasks can be offloaded to the fixed fog or the cloudlet such as base stations and distributed computation servers in smart buildings. Third, the computation tasks can be offloaded to the remote cloud layer.

Based on the above framework, we formulate the computation offloading problem as a generalized multi-dimensional multiple knapsack problem (MMKP). Each layer is considered as a knapsack with distinct two-dimensional features in terms of communication and computation resources. The computation tasks are treated as items which will consume different resources when they are allocated to different layers. The main contributions are outlined as follows:

- We present a multi-layer network architecture for computation offloading. We show the potential competitiveness of the mobile vehicles to share their computation resources by forming a mobile fog layer.
- We formulate the computation offloading problem as a generalized MMKP which aims to achieve the best profit of offloaded tasks. The problem itself is an integer programming problem.
- We modified the branch-and-bound algorithm to derive the optimal solution. By utilizing the Lagrangian relaxation, we decompose the original problem into multiple instances of two-dimensional knapsack problem. The upper bound at each branch node of original problem is obtained by computing the optimal solution of multiple subproblems. Further, the heuristic greedy algorithm is also developed to compare the results.
- Extensive simulations are conducted to demonstrate the effectiveness of the proposed offloading architecture and task allocation algorithm in maximizing the total profits of offloaded tasks. The simulation results also reflect the individual features of different layers in terms of communication and computation resources.

The reminder of this paper is organized as follows. Section II presents the system model. In Section III, we formulate a generalized MMKP. In Section IV, we propose an adapted branch-and-bound algorithm and a heuristic greedy algorithm. The simulation results are given in Section V. We conclude the paper in Section VI.

II. System Model

There are four layers in the proposed network architecture, including user layer, mobile fog layer, fixed fog layer and cloud layer, which are shown in Fig. 1. The last three layers are equipped with computation capability. These three layers differentiate from each other concerning the available communication and computation resources. A large number of RSUs are deployed in the concerned network to serve as access points, which support multiple communication interfaces such as 3G/4G and millimeter wave (mmwave) [5], etc. The RSUs will firstly collect users' computation tasks, and then offload tasks to different layers. In the following, we elaborate the distinct characteristics of each layer as well as the communication modes between RSUs and different layers.

1) *User Layer*: User layer includes intelligent devices, roadside cameras and sensors, etc. These terminal devices will produce numerous computation tasks which cannot be executed by local computation resources. Due to energy constraint or other concerns, they choose to offload their computation tasks to the nearby RSUs. We assume that multiple RSUs are connected with each other by wired backhauls.
2) *Mobile fog Layer*: Mobile fog layer is formed by a large number of mobile vehicles which drive in cities randomly (i.e. cars and taxis) or routinely (i.e. buses). They will receive computation tasks when passing through or stopping within the communication range of RSUs. Consider a certain area where several RSUs are connected to share computation resources, all mobile vehicles will contribute to the computation capability of mobile fog layer. We assume that vehicles are equipped with mmwave communication interfaces for task offloading. Therefore, mobile fog has a high transmission rate. Once the computation tasks are completed, if the RSU is still in their transmission range, they will return the results directly, or otherwise, via multi-hop V2V communication [6]. An example is shown in Fig. 1. Vehicle h received computation tasks from the first RSU, and computed it during driving. After completing computation, the vehicle (h') has arrived at another RSU. The computation results are transmitted to the nearby RSU and then returned to the original one.
3) *Fixed fog Layer*: Some smart buildings, parking lots and base stations provide computation resources to users by forming a fixed fog layer. Similar to the mobile fog layer, the computation capability of fixed fog layer is the total available computation resources of all fixed fog nodes located at the concerned area and connected to nearby RSUs. Considering that single fixed fog node has more powerful computation capability than a mobile vehicle, we assume the computation capability of the fixed fog layer is more powerful than that of the mobile fog layer. The fixed fog layer can support multiple communication technologies according to different types of fog nodes, such as WLAN, MAN, 3G/4G/LTE, etc. [7]. We assume RSUs offload computation tasks to the fixed fog layer through LTE. Thus, the available communication resources of the fixed fog layer is usually smaller than

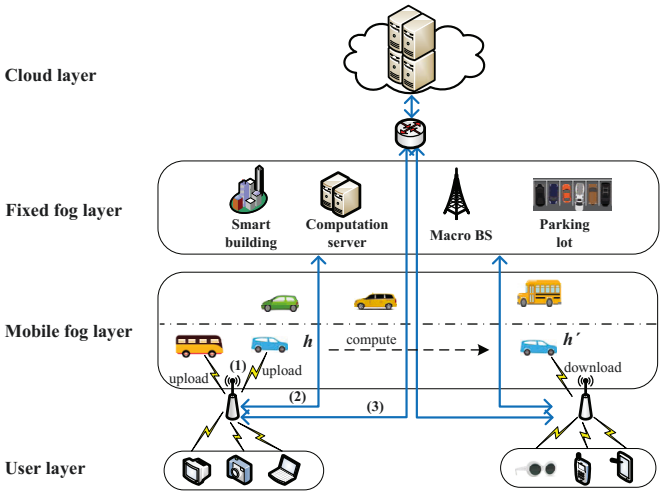

Fig. 1: System model

the mobile fog layer.

4) *Cloud Layer*: A cloud data center can act as a cloud layer, which can be physically located at remote areas far away from users. The cloud center usually has extremely powerful computation capability than the aforementioned two layers. However, due to the long distance and limited transmission power, the transmission rate of the cloud layer is much smaller than the fixed fog layer and the mobile fog layer. However, the wired communication link between the cloud center to the RSUs is more reliable than the other two layers.

We briefly summarize the characteristics of different offloading layers in Table I.

III. Problem Formulation

A. Notations

We consider the computation offloading problem in a limited city area. The mobile vehicles driving across this area, the fixed fog nodes located within this area and the remote cloud center form different network layers to provide their idle computation resources for the nearby users. Let $L = \{l_1, l_2, l_3\}$ represent the mobile fog layer, fixed fog layer and cloud layer, respectively. Denote the task set as $A = \{a_1, a_2, ..., a_{|A|}\}$. The set of profits is denoted as $P = \{p_1, p_2, ..., p_{|A|}\}$, where $p_j = p_0 \pi_j \, (1 \leq j \leq |A|)$, π_j represents the size of computation task a_j (i.e., *Giga cycles*), and p_0 denotes the price for one unit of computation resource (i.e., *$ per Giga cycles*). Note that we didn't consider the computation cost of different layers. Denote the maximum transmission rate (i.e., *bit per second*) of each layer as $R = \{R_1, R_2, R_3\}$ and denote the maximum computation capability (i.e., *the number of cycles per second*) of each layer as $V = \{V_1, V_2, V_3\}$. We use binary variables $x_{i,j} \, (1 \leq i \leq 3, 1 \leq j \leq |A|)$ to represent whether task a_j is processed by layer l_i. If a_j is processed by l_i, then $x_{i,j} = 1$; otherwise, it is 0. Denote the allocated transmission rate for task a_j by l_i as $r_{i,j}$. Denote the allocated computation capability for task a_j by l_i as $v_{i,j}$.

Without loss of generality, we assume that profit p_j, allocated transmission rate $r_{i,j}$ and computation capability $v_{i,j}$, as well as the maximum transmission capacity R_i and the maximum computation capability V_i are all positive integer numbers. Consider that in the wireless communication environment, the allocated resources cannot exceed the system capacity of each network layer, we have

$$r_{i,j} \leq R_i, v_{i,j} \leq V_i \, (i = 1, 2, 3, j = 1, ..., |A|) \, . \tag{1}$$

B. Problem Formulation

Given a task set A and the corresponding profit set P, as well as the intended transmission rate $r_{i,j}$ and the demanding computation capability $v_{i,j}$, we propose to maximize the total profits gained from offloading computation tasks under the constraints of transmission rate and computation capacity in each layer.

$$\begin{aligned}
\text{P1}: \quad & \max_{x_{i,j}} \sum_{i=1}^{|L|} \sum_{j=1}^{|A|} p_j \cdot x_{i,j}, \\
\text{s.t.} \quad & (\text{C1}): \sum_{j=1}^{|A|} r_{i,j} \cdot x_{i,j} \leq R_i, i = 1, ..., |L| \, , \\
& (\text{C2}): \sum_{j=1}^{|A|} v_{i,j} \cdot x_{i,j} \leq V_i, i = 1, ..., |L| \, , \\
& (\text{C3}): \sum_{i=1}^{|L|} x_{i,j} \leq 1, j = 1, ..., |A| \, , \\
& (\text{C4}): x_{i,j} \in \{0, 1\} \, , i = 1, ..., |L| \, , j = 1, ..., |A| \, ,
\end{aligned} \tag{2}$$

where $|L| = 3$, C1 is the constraint of communication resources of each network layer. It requires the sum of allocated transmission rates of all tasks uploaded to the ith layer no larger than the maximum transmission rate. C2 is the constraint of computation resources. It means that the total computation capabilities allocated for the tasks that are uploaded to the ith layer should be within its available computation resources. C3 guarantees that each task a_j can be allocated to at most one offloading layer. C4 is the constraint of binary variables, which represents the task's indivisibility. The above problem is a generalized multi-dimensional multiple knapsack problem (MMKP). Each offloading layer can be considered as a knapsack with distinct transmission rates and computation capabilities; and each task can be considered as an item with two weights (i.e. intended communication rate and demanding computation capability).

It has been proved that the standard 0-1 knapsack problem is a NP-hard problem [8]. Different from the traditional knapsack problem, the proposed problem P1 faces the challenges from the combination of the multiple dimensions and multiple knapsacks. Compared with the multi-dimensional knapsack problem (d-KP) and multiple knapsack problem (MKP), the weight of items (i.e. the allocated transmission rate and computation capability) is variable with the selected knapsack (i.e. offloading layer). For example, the task will be allocated with less communication resources but more computation resources by the cloud layer; while a higher transmission rate and lower computation capability from the mobile fog layer may be assigned.

TABLE I: Characteristics of different offloading layers

Properties / Layers	Transmission Rate	Computation Capability	Connection to RSU
Mobile fog layer	High	slightly weak	Near, dynamic
Fixed fog layer	Medium	Medium	Medium distance
Cloud layer	Lower	Powerful	Remote, reliable

IV. Proposed algorithm

A. Upper bound at branch node

To obtain the optimal solution, we modify the branch-and-bound algorithm which is intended for MKP [9]. A good upper bound is critical to prune unfeasible search space and achieve fast convergency. As shown in P2, by adding constraint C3 to the objective function, we get the Lagrangian relaxation of original problem P1.

$$\begin{aligned}\text{P2}: \quad & z\left(L\left(P1,\lambda\right)\right) = \\ & \max_{x_{i,j}} \sum_{i=1}^{|L|}\sum_{j=1}^{|A|} p_j \cdot x_{i,j} - \sum_{j=1}^{|A|}\lambda_j\left(\sum_{i=1}^{|L|} x_{i,j} - 1\right), \\ \text{s.t.} \quad & \sum_{j=1}^{|A|} r_{i,j}\cdot x_{i,j} \le R_i, i = 1,...,|L|, \\ & \sum_{j=1}^{|A|} v_{i,j}\cdot x_{i,j} \le V_i, i = 1,...,|L|, \\ & x_{i,j} \in \{0,1\}, i = 1,...,|L|, j = 1,...,|A|. \end{aligned} \tag{3}$$

Then, we set Lagrangian multipliers $\lambda_1,...,\lambda_{|A|} = 0$ and decompose P1 into $|L|$ subproblems. Each subproblem is actually a two-dimensional knapsack problems (2-KP).

In the following, we describe how to compute the upper bound at branch nodes by using the optimal solution of $|L|$ subproblems. In the branch-and-bound algorithm, to determine whether a branch $x_{i,j} = 1$ can lead to a potential feasible solution, we compute the upper bound associated with branch node l_i. Denote as G the set of items that have been put into different knapsacks. The upper bound at l_i is computed by

$$B\left(i\right) = \sum\nolimits_{l_h \in L, h \ne i} z_h^*\left(O\right) + z_i^*\left(A - G\right) + p\left(G\right), \tag{4}$$

where $O = A - G - \{a_j\}$, $z_h^*\left(O\right)\left(O \subseteq A\right)$ represents the optimal solution of knapsack l_h when only considering the tasks in subset O. $p\left(G\right)$ is the total profits of computation tasks in set G. The first part in (4) computes the sum of optimal solution of the other subproblems by considering the remaining tasks without a_j; while the second part computes the optimal solution of knapsack l_i by considering all remaining tasks.

We can observe that, by decomposing P1 into multiple independent subproblems, the constraint that the computation tasks consume different communication and computation resources on different layers is eliminated when computing the upper bound of the branch nodes. For each subproblem of 2-KP, we still use the branch-and-bound algorithm to get the optimal solution. For the ith subproblem, the Lagrangian relaxation is obtained by adding the constraint of transmission rate to the objective function.

$$\begin{aligned}\text{P3}: \quad & z\left(L\left(P^i,\gamma_i\right)\right) = \\ & \max_{x_{i,j}} \sum_{j=1}^{|A|} p_j \cdot x_{i,j} - \gamma_i \cdot \left(\sum_{j=1}^{|A|} r_{i,j}\cdot x_{i,j} - R_i\right), \\ \text{s.t.} \quad & \sum_{j=1}^{|A|} v_{i,j}\cdot x_{i,j} \le V_i, \\ & x_{i,j} \in \{0,1\}, j = 1,...,|A|. \end{aligned} \tag{5}$$

Then the upper bound associated with each branch node is determined by solving the Lagrangian dual problem, where we select a nonnegative multiplier γ_i such that $z\left(L\left(P^i,\gamma_i\right)\right)$ is minimized.

$$z\left(LD\left(P^i\right)\right) = \min_{\gamma_i} z\left(L\left(P^i,\gamma_i\right)\right). \tag{6}$$

The subgradient optimization technique is used to yield the least upper bound. For each 2-KP, since the following procedure to derive the upper bound and obtain the optimal solution follows the standard branch-and-bound algorithm, we ignore detailed description due to page limitation.

B. Initial upper and lower bound

Except the upper bound associated with each branch node, we compute the initial upper bound and lower bound for the original problem P1. We use the surrogate relaxation to get the initial upper bound by merging the set of constraints into one constraint.

$$\begin{aligned}\text{P4}: \quad & z\left(S\left(P1,\mu,\sigma\right)\right) = \max_{x_{i,j}} \sum_{i=1}^{|L|}\sum_{j=1}^{|A|} p_j \cdot x_{i,j}, \\ \text{s.t.} \quad & \sum_{i=1}^{|L|}\mu_i \sum_{j=1}^{|A|} r_{i,j}\cdot x_{i,j} \le \sum_{i=1}^{|L|}\mu_i R_i, \\ & \sum_{i=1}^{|L|}\sigma_i \sum_{j=1}^{|A|} v_{i,j}\cdot x_{i,j} \le \sum_{i=1}^{|L|}\sigma_i V_i, \\ & \sum_{i=1}^{|L|} x_{i,j} \le 1, j = 1,...,|A|, \\ & x_{i,j} \in \{0,1\}, i = 1,...,|L|, j = 1,...,|A|. \end{aligned} \tag{7}$$

For any instance of MKP, the optimal choices of multipliers $\mu_1,...,\mu_m$ in $z\left(S\left(P,\mu\right)\right)$ is set to k, where m is the number of knapsacks, and k is a positive constant [8]. Therefore, we set $\mu_i = \sigma_i = 1$ for $i = 1,...,|L|$. Let binary variable $x'_j = \sum_{i=1}^{|L|} x_{i,j}$ indicate whether task a_j has been selected in any knapsack $l_i (i = 1,...,|L|)$. By considering only one constraint each time, we get two relaxed subproblems P5-1 and P5-2.

$$\begin{aligned}\text{P5}-1: \quad & \max_{x'_j} \sum_{j=1}^{|A|} p_j \cdot x'_j, \\ \text{s.t.} \quad & \sum_{j=1}^{|A|} r'_j \cdot x'_j \le R, \\ & x'_j \in \{0,1\}, j = 1,...,|A|. \end{aligned} \tag{8}$$

$$\begin{aligned} \mathrm{P5-2}: \quad & \max_{x'_j} \sum_{j=1}^{|A|} p_j \cdot x'_j, \\ \text{s.t.} \quad & \sum_{j=1}^{|A|} v'_j \cdot x'_j \le V, \\ & x'_j \in \{0,1\}, j = 1, ..., |A|, \end{aligned} \tag{9}$$

where $r'_j = \min\{r_{i,j} | i = 1, ..., |L|\}$, $R = \sum_{i=1}^{|L|} R_i$, $v'_j = \min\{v_{i,j} | i = 1, ..., |L|\}$, and $V = \sum_{i=1}^{|L|} V_i$. Denote the optimal solution of (8) and (9) as zr^* and zv^*. We get the upper bound as

$$U_1 = \min\{zr^*, zv^*\}. \tag{10}$$

Another way to get the upper bound is based on the Lagrangian relaxation. As described above, we remove the constraint C3 to enable each subproblem of 2-KP to be solved independently. Each knapsack is filled by selecting from all computation tasks. Then, we can compute the total profits of $|L|$ knapsacks and get the upper bound U_2. Therefore, the upper bound of MMKP can be derived by

$$U = \min\{U_1, U_2\}. \tag{11}$$

We compute the initial lower bound using a heuristic greedy method. Define the efficiency of task a_j as [8]:

$$e_j = \frac{p_j}{\sum_{i=1}^{|L|}\left(r_{i,j} \cdot \left(\sum_{j=1}^{|A|} r_{i,j} - R_i\right) + v_{i,j} \cdot \left(\sum_{j=1}^{|A|} v_{i,j} - V_i\right)\right)}. \tag{12}$$

We roughly estimate the overall approximate capacity of knapsack l_i as $C_i = R_i \cdot V_i$. Firstly, we fill the knapsack with the minimum approximate capacity by selecting tasks from set $|A|$. The branch-and-bound algorithm for 2-KP is adopted to get the optimal solution for the first knapsack. Then, the knapsack with the second minimum capacity is filled by optimally choosing from the remaining tasks. Finally, the remaining tasks are filled in the last knapsack. Actually, on the basis of initial lower bound, another heuristic method is to exchange tasks from one knapsack to another knapsack if this change can lead to increasing profit of the computation tasks which will be put in.

C. Exact algorithm

As described above, by setting the Lagrangian multipliers $\lambda_j = 0, j = 1, ..., |A|$, the original MMKP can be decomposed into $|L|$ instances of 2-KP. First, by selecting a task which appears in k' ($2 \le k' \le |L|$) solutions of independent 2-KP, k' nodes are generated with the first $k'-1$ branches to assign the task to each knapsack, and the last branch to exclude the first $k'-1$ knapsacks. For example, if a_j is assigned to knapsacks k_1, k_2 and k_3. Three branches are generated, i.e., $a_{1,j} = 1$, $a_{2,j} = 1$ and $a_{1,j} = a_{2,j} = 0$. For the last branch, if the upper bound corresponding to the exclusion of the three knapsacks is lower than current optimal solution, then the third branch is represented by $a_{3,j} = 1$. The branch-and-bound algorithm for MMKP is shown as Table II.

TABLE II: The branch-and-bound algorithm.

Algorithm 1 The branch-and-bound algorithm.

Input: Task set A, profit set P, required transmission rate $r_{i,j}$ and computation capability $v_{i,j}$ from a_j to l_i; transmission rate R_i and computation capacity V_i
Output: Allocation matrix $[x]$
Steps:

1: *Step 1: Initialize*
2: Compute the lower bound of MMKP; set $[x^*] = x$ and $P^* = LB$.
3: **for** $i = 1, ..., |L|$: **do**
4: solve 2-KP associated with each knapsack l_i using branch-and-bound algorithm; denote the solution vector and solution value in $[x]$ and $Z_{e,i}$, respectively.
5: **end for**
6: Set $F = A$, set $d_e = \left\{h \middle| x_{h,j} = 1\right\}$, and $e = 1$. Compute upper bound U of MMKP.
7: *Step 2: Branch*
8: **for** increasing $a_j \in F$ such that $\sum_{i \in |L|} x_{i,j} > 1$ **do**
9: Set $d_e = \left\{h \middle| x_{h,j} = 1\right\}$.
10: **end for**
11: if no such task exists, a feasible solution is found, and go to step 5.
12: *Step 3: Bound*
13: **for** $h \in d_e$ **do**
14: Solve the single knapsack problem associated with knapsack h using the tasks excluding the ones that have been assigned and the current one. Denote the solution value by z. Set $\overline{Z}_h = z + \sum_{j \in |A|, x_{h,j}=1} p_j$.
15: **end for**
16: **for** $h \in d_e$ **do**
17: Compute the upper bound of each node as $B_{e,h} = \sum_{i \in (A - d_e + \{h\})} Z_{e-1,i} + \sum_{i \in (d_e - \{h\})} \overline{Z}_{e,h}$. Order the nodes in decreasing order of $B_{e,h}$ values.
18: **end for**
19: *Step 4: Go on*
20: For nodes with $B_{e,h} \le P^*$, go to step 6. If the condition $x_{h,j} = 1$ can be assigned to the branch, set $F = F - \{a_j\}$, $R_h = R_h - r_{h,j}$, $V_h = V_h - v_{h,j}$.
21: **for** $i \in (d_e - \{h\})$ **do**
22: Set $Z_{e,h} = \overline{Z}_{e,h}$, $e = e + 1$.
23: Set the ith row of $[x]$ equal to the last solution vector determined at level e for knapsack i and go to Step 2.
24: **end for**
25: *Step 5: Update*
26: If $B_{e,h} \le P^*$, go to Step 6. If $P^* = U$, stop. Otherwise, set $P^* = B_{e,h}$, $[x^*] = [x]$.
27: *Step 6: Backtrack*
28: Go to the last layer $e = e - 1$. If $e = 0$, stop.
29: **if** $a_j \in (A - F)$ **then**
30: Set $F = F \cup \{a_j\}$, $R_h = R_h + r_{h,j}$, $V_h = V_h + v_{h,j}$.
31: **end if**
32: **for** $i \in (d_e - \{h\})$ **do**
33: Set $Z_{e,i} = Z_{e-1,i}$, and set the ith row of $[x]$ equal to the last solution vector determined at level $e - 1$ for knapsack i.
34: **end for**
35: Find the next node following h in d_e and go to Step 4.

In the first step, the lower bound is computed using the heuristic method as described above. The upper bound of MMKP is computed based on (11). Variable e is the current tier of branch tree. The set d_e includes all knapsacks that have repeatedly added the same task. In step 3, the right part of $Z_{e,h}$ adds the profits of tasks that have been assigned to node l_h. The upper bound $B_{e,h}$ at node l_h is computed according to (4). In step 4, for the nodes whose upper bound is lower than the current best profit, we prune the branch and execute backtrack in step 6. The condition that task a_j can be assigned to l_h includes two cases: one is that a_j is not

the last one in set d_e; the other one is that the upper bound computed by removing a_j from all knapsacks is smaller than the current optimal solution. For above example, the second condition remove the possibility of $a_{3,j} = 0$ from condition $a_{1,j} = a_{2,j} = 0$, and thus lead to $a_{3,j} = 1$. If task a_j can be imposed on the branch, then the corresponding knapsack will reduce the capacity. The allocation solution of other knapsacks in d_e will be set as the solution without a_j. Then the tier of branch tree increases, and the algorithm will go to step 2 to repeat the process of depth search. In step 5, it checks the upper bound and update the optimal value is as the current optimal solution. In step 6, the algorithm goes back to tier $e-1$ and recover the solutions at tier e by removing the task from current knapsack and trying to put it in the next knapsack in set d_e. Once there is no task to be found in multiple knapsacks, a feasible solution is found. Then, the algorithm will continue going backtrack until reach the upper bound or traverse all feasible solution space.

D. A scheduling example

We give an example with 6 tasks, the demanding communication rates, computation capacities for three offloading layers are r= {10 6 8 13 4 9; 4 5 8 10 12 3; 12 8 10 3 9 11}, and v= {3 6 4 5 8 10; 8 3 4 9 4 5; 5 12 8 10 8 9}, respectively. The profits of the tasks are p= {6 1 2 8 10 3}. The communication and computation capacities of different layers are $R = \{12\ 10\ 8\}$ and $V = \{8\ 11\ 15\}$, respectively. By running the branch-and-bound algorithm, we get the exact solution as follows: task a_5 is offloaded to k_1; task a_1 and a_2 are offloaded to k_2; and task a_4 is offloaded to k_3. The optimal solution value is 25. By using the heuristic method, the solution value is 19. It assigns task a_5 to k_1, task a_4 to k_2, and task a_2 to k_3. We can find that the exact algorithm assigns tasks to a layer when it has smaller requirements for this layer.

V. Simulation Results

We build the simulation model and implement the exact algorithm and the heuristic method with Matlab. The experiments are conducted in the regular laptop with 4GHz CPU frequency. The communication rate from RSU to cloud is set to 15 Mbps [3], and 1.5 Gbps to mobile fog [5], and 80 Mbps to fixed fog. The computation capability of fog nodes is 20×10^8 cycles/s [10]. We consider an area with 10 vehicles and 2 fixed fog nodes. Then, the maximum computation capacity of mobile fog layer is 200×10^8 cycles/s. Consider that the fixed fog node such as base station has much more computation resources than mobile vehicles, we assume the available computation resources of a fixed fog node is 10 times than that of a mobile vehicle. Thus, the maximum computation capability of fixed fog layer is 400×10^8 cycles/s. The computation capacity of cloud is unlimited, but it usually can assign the speed of 100×10^8 cycles/s [11] for each user or regular task. Assume that $|A|$ is not larger than 40. Then the maximum computation capacity can be set as 4000×10^8. For each task, the intended transmission rates for different

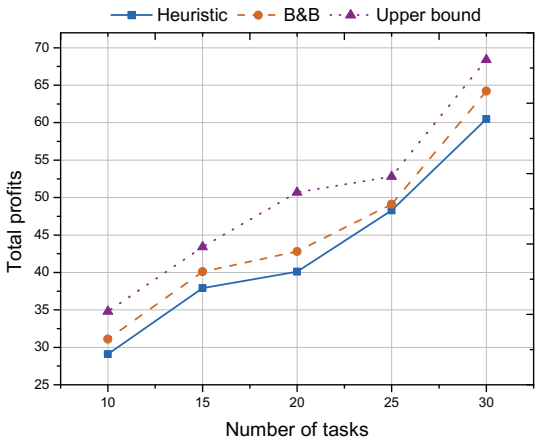

Fig. 2: The total profits of completed tasks

layers are uniformly generated from the range $[1, 50]$ Mbps, $[1, 20]$ Mbps and $[1, 10]$ Mbps, respectively. The demanding computation capabilities for different layers are uniformly distributed in the range of $[1, 15] \times 10^8$ cycles/s, $[1, 20] \times 10^8$ cycles/s and $[1, 200] \times 10^8$ cycles/s, respectively. The unit price p_0 of computation resource is set to 0.1 [12]. The size of computation task is uniformly distributed in $[1, 50]$ Gcycles. We run each sample for 50 times to compute the average value of all samples.

As shown in Fig. 2, with the number of tasks increasing, more computation tasks can be allocated to upper offloading layers. Although not all the tasks can be offloaded, the chances of tasks to be uploaded increase accordingly. Then the total profits of offloaded tasks will increase. The purple dotter line represents the upper bound of the total profits. The results of branch-and-bound algorithm and heuristic method are shown in dashed line and solid line, respectively. As expected, the branch-and-bound algorithm gains the optimal solution, which is a little lower than the upper bound, and higher than the heuristic method. The heuristic method can produce a quite good performance in all scenarios.

Furthermore, we compare the running time of branch-and-bound and heuristic method. As shown in Fig. 3, with the increasing number of computation tasks, the running time of branch-and-bound algorithm increases rapidly. As described above, we modified the branch-and-bound algorithm which was proposed for MKP to the proposed MMKP. Although, the algorithm complexity cannot be described explicitly since the actual number of nodes in branch tree cannot be bounded, the running time complexity in worse-case is extremely high (i.e. exponential growth). Instead, the heuristic algorithm has no obvious changes along with the increasing number of tasks, and the values of running time keep lower than 10 s in all settings.

In default setting, there is a potential relationship between the allocated transmission rate and computation capability. Assume that the average size of input data of these tasks is CM, and the average size of computation tasks is CP, then $(CM/r_{i,\max} + CP/v_{i,\max}) \in [ED - \varepsilon, ED + \varepsilon]$ holds for all layers $i = 1, 2, 3$. $r_{i,\max}$ and $v_{i,\max}$ are the maximum

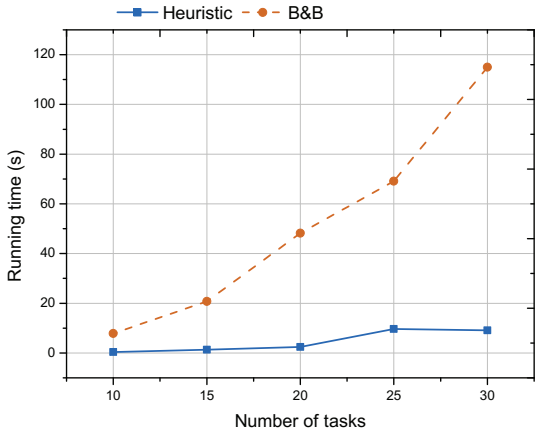

Fig. 3: The running time of two algorithms

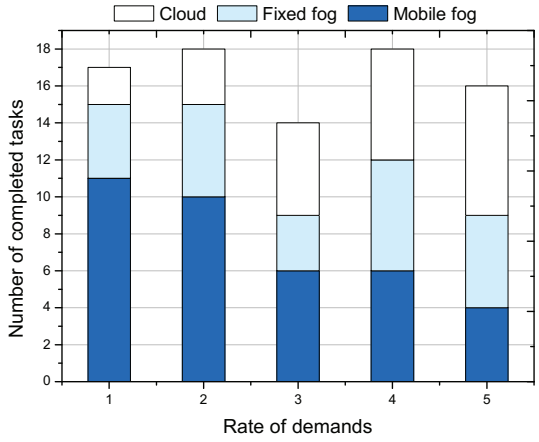

Fig. 4: The percentage of tasks completed at each layer

transmission rate and maximum computation capability demanded by the tasks. $ED \pm \varepsilon$ is the range of average expected delay. In above settings, we set the ratio of CP to CM to be 1. By varying the ratio as $1/3$, $1/2$, $1/1$, $2/1$ and $3/1$, we get five scenarios as shown in Fig. 4. From scenarios 1 to 5, the ratio of computation demands becomes higher, and the tasks consume more computation resources than communication resources, and the number of tasks completed by cloud (i.e. shown as blank columnar) increases. On the contrary, from scenarios 5 to 1, the number of tasks completed by mobile fog (i.e. shown as deep blue) increases because that mobile fog can satisfy the increasing demands of communication resources.

VI. Conclusion

In this paper, we have presented a multi-layer network architecture for computation offloading. The mobile vehicles are utilized and modeled as a powerful mobile fog layer to provide computation resources for users. We have described the unique characteristics of different offloading layers in terms of transmission rates and computation capabilities. On this basis, the task allocation is formulated as a generalized MMKP, which is an integer programming problem. The differences between MMKP and the d-KP or the MKP are described. By modifying the branch-and-bound algorithm, we decompose MMKP into multiple independent 2-KP. Then the constraint that the computation tasks consume different communication and computation resources when uploaded to different layers is eliminated. We adopt the Lagrangian relaxation and surrogate relaxation to obtain the upper bound of MMKP. The adapted branch-and-bound algorithm as well as the heuristic greedy method are realized and compared. Through a comprehensive performance evaluation, we have demonstrated that the proposed offloading architecture and the task allocation algorithms are able to achieve the maximization of total profits of computation offloading.

VII. Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (No. 61572088, 61702258), the China Postdoctoral Science Foundation (No. 2016M591852), Postdoctoral research funding program of Jiangsu Province (No. 1601257C), US MURI, NSF CNS-1717454, CNS-1731424, CNS-1702850, CNS-1646607, and the DGIST R&D Program of the Ministry of Science and ICT (18-EE-01), the Frontier Interdisciplinary Research Fund for the Central Universities (No. 2018CDQYJSJ0034).

References

[1] R. Mahmud, R. Kotagiri, and R. Buyya, *Fog Computing: A Taxonomy, Survey and Future Directions*. Springer Singapore, 2018, pp. 103–130.

[2] M. H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *IEEE Conference on Computer Communications (INFOCOM)*, Atlanta, GA, May 2017.

[3] Y. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation," in *IEEE Conference on Computer Communications (INFOCOM)*, Atlanta, GA, May 2017.

[4] M. Sookhak, F. R. Yu, Y. He, H. Talebian, N. S. Safa, N. Zhao, M. K. Khan, and N. Kumar, "Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 55–64, Sept. 2017.

[5] T. Baykas, C. S. Sum, Z. Lan, J. Wang, M. A. Rahman, H. Harada, and S. Kato, "Ieee 802.15.3c: the first ieee wireless standard for data rates over 1 gb/s," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 114–121, Jul. 2011.

[6] K. Liu, J. K. Ng, V. C. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: Vanet as a software defined network," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1759–1773, Jun. 2016.

[7] H. Zhang, Q. Zhang, and X. Du, "Toward vehicle-assisted cloud computing for smartphones," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5610–5618, Dec. 2015.

[8] D. Pisinger and P. Toth, *Knapsack Problems*. Springer US, 1999, pp. 299–428.

[9] S. Martello and P. Toth, "Solution of the zero-one multiple knapsack problem," *European Journal of Operational Research*, vol. 4, no. 4, pp. 276 – 283, 1980.

[10] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, Dec. 2017.

[11] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[12] H. Shah-Mansouri, V. W. S. Wong, and R. Schober, "Joint optimal pricing and task scheduling in mobile cloud computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5218–5232, Aug 2017.