Cloudy Knapsack Problems: an Optimization Model for Distributed Cloud-assisted Systems

Harisankar Haridas, Sriram Kailasam, Janakiram Dharanipragada
Dept. of CSE, Indian Institute of Technology (IIT), Madras
harisankarh@gmail.com, {ksriram, djram}@cse.iitm.ac.in

Abstract

Cloud-assisted approaches for both peer-to-peer systems and mobile apps require optimized use of elastic cloud resources. Due to budget constraints, a subset of the tasks has to be selected for offloading considering context parameters like device battery level and task variability. This leads to the challenging problem of context-sensitive task scheduling on elastic resources with a limited global view, which is not addressed by existing works. We identify a new class of formal problems called cloudy knapsack problems to effectively model the same. Abstracting out the problem formally can spur future independent works related to different variants of the problem and corresponding bounds and optimal algorithms. We illustrate the global view related issues through simulations, identify some theoretical bounds for a variant of cloudy knapsack problems and discuss several open problems.

I. INTRODUCTION

Recently, several works [4], [10] have proposed cloud-assisted mobile apps to improve the response time as well as battery life of mobile devices. Several works have also explored cloud-assisted approaches to improve the properties of peer-to-peer systems (e.g., [8], [7]). Emerging augmented reality apps have a high task generation rate (e.g., of the order of frames per second) as well as high computation per task (e.g., object recognition from image frames). For popular apps with hundreds of millions of users, offloading of the possible tasks will be limited by cloud budget constraints. For co-operative peer-to-peer applications with a donation-based model, the budget constraints can be severe.

With limited budget, for each task generated, a decision has to be made on whether it has to be processed in the client (peers or mobile devices) itself or in the cloud. The decision on a task can depend on the context like current battery level of the device as well as the cost of processing it in cloud. Context and task characteristics across all the clients have to be considered in order to derive maximum value within the cloud budget. Maintaining a central scheduler for making the decisions will be infeasible for a popular application with high cumulative task generation rate. Hence this decision-making has to be performed either at the clients or at intermediate servers. However, this distributed decision making is a challenge because only a partial view of the global state will be available. This setting is characterized by distributed decision-making to handle large volume of low-latency requests, a certain duration for spending the budget, and resources that can be scaled dynamically within the available budget. To the best of our knowledge, existing works in literature address one or more of these characteristics, but not all. We identify a new class of formal problems called cloudy knapsack problems to model this optimization challenge. The problem setting is similar to online knapsack problems except that while deciding on inclusion of each item into the knapsack, only a limited knowledge of the current status will be available. Thus decisions has to be made with only a cloudy view of the current state of the knapsack and past items. Different variants of the problem can be defined based on the nature of the limited views corresponding to different global view synchronization mechanisms.

Abstracting out the optimization challenge as cloudy knapsack problems could spur future independent research works providing lower bounds and optimal algorithms for different variants. We discuss lower and upper bounds for a variant of the problem and identify several open problems. Online knapsack problems are actively studied in areas like keyword ad bidding [11]. Existing solutions for the online knapsack problems could be adapted to give solutions to their cloudy variants.

In Section II, we discuss the decision making challenge in detail. We study the distributed decision making problem, identify the formal problems and prove bounds for a variant in Section III. The related works are discussed in Section IV. We conclude and discuss future works and open problems in Section V.

II. DECISION MAKING CHALLENGES

We consider two cloud-assisted applications which will be used as examples in the remaining sections. First is an augmented reality app which includes a subsystem to detect characters present in incoming image stream. The detected characters can be used to provide value-added services like translation or suggestions. The character recognition task for each image can be performed either in the mobile device or on the cloud. Henceforth, we use mOCR to refer to this cloud-assisted application.

Second example application is a cloud-assisted peer-to-peer live streaming service. Specifically, we consider an application similar to the passive assistance-based streaming proposed in Clive [7] (Section IIIA in the paper). Each peer receives frames of the streaming media from other peers using a mesh-based overlay. A copy of the streamed content is also stored in a cloud-based storage like Amazon S3. If a peer does not obtain a frame (from other peers) within a certain threshold period before the frame’s playback time, it will request a copy of the frame from the cloud storage. Henceforth, we refer to this application as Clive-P.
Budget constraints on cloud usage: For a non-commercial peer-to-peer application with a donation-based model, the budget available for cloud usage can be severely limited. Hence a subset of the candidate tasks can only be offloaded to the cloud. Large user base and high processing required per task can also lead to budget constraints in even some commercial mobile apps. WhatsApp, a messaging service with modest processing requirements, requires a 11000 core cluster in its cloud backend. This is because of its large active user base consisting of hundreds of millions of users. A compute-intensive app like mOCR of similar popularity can require millions of cores if all its tasks are offloaded to the cloud. Hence the offloading decision making for such an app will need to consider the budget constraints also.

Challenges: First, we discuss the feasibility of a cloud-based centralized scheduler to select tasks to be offloaded. For applications with hundreds of millions of users, the cumulative task generation rate can be very high. This issue is compounded in case of multimedia applications like mOCR where the task generation rate per device can be of the order of frames per second. This can lead to scalability bottlenecks for a centralized scheduler. Further, for many applications, most of the tasks (say 90%) can be processed at the client side (peer or mobile device) itself. In case of low latency tasks like OCR or video streaming, the additional latency of 100-150ms involved in contacting the remote scheduler for each task will increase the average latency significantly. Next, we discuss the challenges for a client-side scheduler to make the offloading decisions. The potential benefit from offloading a task can depend significantly on the context which can vary drastically. In the case of mOCR, the benefit from offloading tasks increases if the phone has low battery level. The cost involved in processing different images in the cloud can also vary drastically. In the case of Cloud-P, the benefit from downloading a video frame from cloud improves if many recent preceding frames were not rendered to the user. The overall task rate can also vary drastically at times during interest spikes. Because of the importance of context and the variability in the task rates, decision-making by individual client schedulers while maximizing overall value addition is challenging.

III. DISTRIBUTED DECISION MAKING WITH PARTIAL VIEW

Since a central scheduler with global view is not feasible, the offloading decision of each task has to be made in a distributed setting. Decisions could be made either at the clients or using a hierarchical scheme. In both cases, only an inexact view of the current global status will be available at decision-making time. The global status when deciding on a task will include knowledge of the amount of budget consumed till now, total value added till now and total number of items considered till now.

The degree of inexactness of the global view (while deciding on a task) will depend on the particular view synchronization mechanism used by the application as well as the scale.

Different mechanisms like periodic broadcast, polling or gossip combined with prediction techniques can be used for view synchronization ([2], [9]). Generally, obtaining a more accurate view will incur more overhead in terms of network and device resource usage. Different mechanisms can be abstracted based on the distribution of error in the global views they provide.

A. Formal decision making problem

System model: Each task $j$ generated can be associated with a weight ($w_j$) corresponding to the cloud resources consumed by it and a value ($\pi_j$) corresponding to the user experience improved if the task is offloaded. These weights and values can be dynamically assigned after the task is generated based on the task characteristics and client context. For example, in the case of mOCR, the value assigned to a task can be a function of the task’s estimated energy consumption and the current battery level. In case of peer-to-peer applications, tasks which remain unprocessed even after assistance from other peers (e.g., missing frames in Clive-P) will only be considered for offloading. Maximum budget ($B$) can be defined as a constant for a fixed time period (e.g., donation-based model) or relative to the total number of tasks (e.g., a commercial app with estimated revenue per task). In the latter case, budget can be defined in terms of $\beta = \frac{B}{N}$ where $\beta$ is application-specific and $N$ is the total number of tasks.

The above model assumes a task-level monetary price for cloud usage. This directly corresponds to the cases where the app provider uses a higher level service like Storage-as-a-Service (as in Clive-P) or Platform-as-a-Service (PaaS) from the cloud. To model a lower level service like Infrastructure-as-a-Service (IaaS), additional factors like delay in starting and stopping instances may have to be considered.

Cloudy knapsack problems: The goal of selecting items which maximize the overall value within the budget constraints can be expressed as:

$$\text{Maximize } \sum_{j=1}^{N} \pi_j x_j \text{ subject to } \sum_{j=1}^{N} w_j x_j \leq B, x_j \in \{0, 1\}$$

where $x_j$ is an indicator variable denoting whether task $j$ is offloaded to cloud. If a central scheduler was feasible for decision making, this optimization could be modelled as an online knapsack problem ([5], [11]). Each task $j$ with corresponding weight ($w_j$) and value ($\pi_j$) will correspond to items to be considered for filling in a knapsack with maximum capacity $B$. Decision of each task/item has to be made without knowledge of the future sequence of tasks/items. Deciding to process a task in the cloud will correspond to putting corresponding item in the knapsack.

The online knapsack problems assume that an exact view of all past events will be known while deciding on an item. The solutions can use the up-to-date global view to improve their performance. Deeparnab et al. [11] proposes an almost optimal algorithm for a particular online knapsack problem setting. In the algorithm (ONLINE-KP-THRESHOLD), an item $j$ is included in the knapsack if $\frac{\pi_j}{w_j} \geq \Psi(z_j)$ where $\Psi$ is a function on the fraction of knapsack filled when item $j$ is considered ($z_j$). Such solutions are not possible when an up-to-date global view (here, that of $z_j$) is not available.

To address above issue, we identify a new class of problems called *cloudy knapsack problems*. The problem setting is similar to online knapsack problems where the goal is to choose items to be put in a knapsack such that the total value of the knapsack is maximized without exceeding its capacity. The decisions have to be made as soon as items arrive without knowledge of future items. However, when the decision on an item is made, an exact global view will not be available. The exact values of the fraction of budget consumed or the number of items considered till now will not be available. While deciding on item \( j \), only an inexact(cloudy) value of \( z_j \), \( Y(z_j) \) will be available, where \( Y(z_j) \) need not be equal to \( z_j \). Within the class of problems, different variants can be defined with different restrictions on the relationship between (cloudy) view(\( Y(z_j) \)) and exact view(\( z_j \)). A variant of the algorithm with a bounded error is defined below. Different variants can be used to model different cloud-assisted applications with different view synchronization mechanisms.

In a \( CLOUDY\ KP/r_z/* \) variant of the problem,
\[
\forall j: z_j - r_z \leq Y(z_j) \leq z_j + r_z, r_z \geq 0 \tag{1}
\]
where \( r_z \) is a parameter. Note that only the bounds(\( r_z \)) of the error is fixed and its distribution is assumed to be arbitrary(*). In the next section we theoretically analyze an existing online algorithm in \( CLOUDY\ KP/r_z/* \) setting and provide a theoretical upper bound for it in the setting. Further possible problem variants within the class are discussed in the future works section(Section V).

Different solutions to online knapsack problems are evaluated based on their competitive ratio with the corresponding optimal offline solution having full knowledge of the item sequence. For an online knapsack algorithm(solution) \( A \) with competitive ratio \( c \), the following condition should be true. For any item sequence \( \sigma \), if \( A(\sigma) \) denotes the total value obtained by \( A \) and \( OPT(\sigma) \) denote the total value obtained by the offline optimal algorithm, \( OPT(\sigma) \leq cA(\sigma) \). The performance of an algorithm(solution) for the cloudy knapsack problem can also be defined in the same manner(as its competitive ratio with the optimal offline algorithm).

**Effect of cloudy view:** Using simulation results, we illustrate the need for current global view while decision-making and the impact of a cloudy view. Marchetti et al.[5] proposes a fixed choice online knapsack algorithm for cases where the values and weights of items follow uniform distributions(0,1).

An item is inserted into knapsack if the item’s value to weight ratio is greater than \( r \), where \( r \) is calculated as
\[
r = \begin{cases} 
\frac{1}{2} - 3\beta, & 0 \leq \beta < \frac{1}{6} \\
\frac{1}{2}, & \frac{1}{6} \leq \beta < \frac{1}{2} 
\end{cases} \tag{2}
\]

\( \beta \) is defined as (total budget)/(total number of items(\( N \))) which is required to be less than \( \frac{1}{6} \). We simulated the algorithms with 1 million randomly generated items and a budget of 0.25 million. To evaluate the effect of high variations in the item characteristics(deviating from the expected distributions), 20% of the items had values and weights drawn from exponential distributions(with mean of 8) and those of remaining items were drawn from uniform distributions(0,1). We observed that the fixed choice algorithm performed poorly when high variability is present. To improve the performance, we developed an adaptive version of the algorithm where \( \beta \) used to calculate \( r \) was calculated adaptively based on current global view. Specifically, while deciding on each item, the \( \beta \) used in (2) was calculated as (remaining budget)/(remaining number of items) which is same as (initial budget - budget used)/(\( N \) - number of items processed). For the same input distribution mentioned, total value obtained by the adaptive algorithm was 48% better compared to the fixed choice algorithm. This shows the importance of global view in decision making.

The adaptive algorithm uses “budget used”(\( b \)) and “number of items processed”(\( n \)) which are part of the exact global view while decision making. Next we simulated cases where only a stale global view is available to the algorithm. In different simulations, the observed values of both the view quantities(\( b \) and \( n \)) differed from their actual values by a percentage staleness which was uniformly drawn from (0,\( s_{max} \)). The results obtained for different values of \( s_{max} \) are shown in Figure 1. When staleness was between 0 and 3%, the total value obtained reduced by 5.6% compared to the case when an exact view was available.

**B. Preliminary results on a problem variant**

In this section we theoretically study the \( CLOUDY\ KP/r_z/* \) setting(defined in (1)) when the upper and lower bounds of the value-to-weight ratio are known(\( L \leq \frac{z_j}{w_j} \leq U \)) and the weight of each item is significantly smaller compared to size of the knapsack(\( \forall j : w_j \ll B \)). Note that here the item value and weight distributions can be arbitrary(within the \( U,V \) limits) unlike the previous setting where a uniform distribution was assumed. We theoretically analyze the ONLINE-KP-THRESHOLD algorithm of Deeparnab et al.[11] which is near optimal for the particular non-cloudy online setting in the \( CLOUDY\ KP/r_z/* \) setting. Based on the analysis, we derive loose upper and lower bounds for the performance of any algorithm in the cloudy setting. In the original algorithm we replace \( z_j \)(fraction of knapsack filled) with \( Y(z_j) \) as \( z_j \) is not available in the cloudy setting. The resulting algorithm is shown below.

**Algorithm ONLINE-KP-THRESHOLD-\( r_z \)**
\[ \Psi(z) \equiv \left( Ue/L \right)^z (L/e) \]

When the \( j^{th} \) item arrives, if item can be fitted in knapsack, put item \( j \) if \[ \frac{w_j}{z_j} \geq \Psi(Y(z_j)) \]

We prove an upper bound of the competitive ratio of the algorithm in the cloudy setting.

**Theorem 1.** For any input sequence \( \sigma \), if OPT(\( \sigma \)) is the maximum achievable value (obtained by an optimal offline algorithm) and \( A(\sigma) \) is the value obtained by ONLINE-KP-THRESHOLD-\( r_z \), then

\[ \frac{OPT(\sigma)}{A(\sigma)} \leq \left( \ln \left( \frac{U}{L} \right) \right) + 1 \left( \frac{UL}{L} \right)^{2r_z} \]

**Proof:** Let \( Z \) denote the fraction of the knapsack filled after execution of ONLINE-KP-THRESHOLD-\( r_z \) (henceforth denoted by algorithm \( A \)). Let \( S \) and \( S^* \) denote the set of items picked by the Algorithm \( A \) (selected by \( A \) and fitted in knapsack) and the optimal offline algorithm (OPT) respectively. Let \( W = w(S \cap S^*) \) and \( P = \pi(S \cap S^*) \) denote the total weight and profit of the common items picked by both algorithms.

\[ \begin{align*}
OPT(\sigma) &= \pi(S^*) \\
&= \pi(S \cap S^*) + \pi(S^* \setminus S) \\
&= P + \pi(S^* \setminus S) \\
\end{align*} \]

Since \( \Psi(z) \) is a monotonically increasing function, using eq(1), we get

\[ \Psi(Y(z_j)) \leq \Psi(z_j + r_z) \leq \Psi(Z + r_z) \quad \text{(4)} \]

For each item \( j \) not selected by algorithm \( A \), its \( \frac{w_j}{w_j} \) is less than \( \Psi(Y(z_j)) \). For items (\( j \)) selected by algorithm \( A \) but did not fit in the knapsack, \( (w_j + z_j)B > B, i.e., z_j > (1 - \frac{w_j}{w_j}) \). For these items, \( \Psi(z_j + r_z) > \Psi(1 - \frac{w_j}{w_j} + r_z) \). As \( w_j \ll B, r_z > \frac{w_j}{w_j} \).

Hence, \( \Psi(z_j + r_z) > \Psi(1) = U \geq \frac{w_j}{w_j} \). Combining both the classes of items not picked by algorithm \( A \), and considering eq (4), we get \( \forall j \in (S^* \setminus S) \)

\[ \frac{\pi_j}{w_j} < \Psi(Z + r_z) \quad \text{(5)} \]

From eq(3) and eq(5), we get

\[ \begin{align*}
OPT(\sigma) &\leq P + \Psi(Z + r_z)w(S^* \setminus S) \\
&\leq P + \Psi(Z + r_z)(B - W) \\
\end{align*} \]

as \( w(S^*) \leq B \).

Now, \( A(\sigma) = P + \pi(S \setminus S^*) \). Using (6), we get

\[ \begin{align*}
\frac{OPT(\sigma)}{A(\sigma)} &\leq \frac{P + \Psi(Z + r_z)(B - W)}{P + \pi(S \setminus S^*)} \\
\end{align*} \]

For each item \( j \) in \( S \) (picked by \( A \)),

\[ \frac{\pi_j}{w_j} \geq \Psi(Y(z_j)) \quad \text{(8)} \]

Since \( \forall j, L \leq \frac{w_j}{w_j} \), when \( \Psi(Y(z_j)) \leq L \), algorithm \( A \) will attempt to put item \( j \) into the knapsack. In this case, from eq(1) and because \( \Psi(z) \) is a monotonically increasing function, we get \( \Psi(z_j - r_z) \leq L \). Substituting the value for \( \Psi(z) \), we get

\[ \left( \frac{Ue}{L} \right)^{z_j - r_z} \cdot L \leq L \]

\[ z_j \leq \Psi(z_j) = z_j + \frac{1}{\ln \left( \frac{U}{L} \right)} \quad \text{(9)} \]

Hence, algorithm \( A \) will attempt to put all items till the knapsack is filled upto \( z \). If \( z > Z \), then all the items can be put into the knapsack without exceeding its capacity. Thus, algorithm \( A \) will perform as good as the optimal algorithm and the claim in the theorem will be trivially satisfied. We now consider cases where \( z \leq L \).

By partitioning the input sequence into those items which appear before \( z \) fraction is filled and afterwards, we can express \( P \) as

\[ P = \sum_{j \in S \cap S^* \land j \leq z} \pi_j + \sum_{j \in S \cap S^* \land j > z} \pi_j \quad \text{(10)} \]

Since \( \forall j : L \leq \frac{\pi_j}{w_j}, \pi_j \geq \Psi(Y(z_j)) \).

For these \( j \) in \( S \), from eq(8), \( \pi_j \geq \Psi(Y(z_j))w_j \).

Using these inequalities in eq(10), we get

\[ P \geq \sum_{j \in S \cap S^* \land j \leq z} \Psi(Y(z_j))w_j + \sum_{j \in S \cap S^* \land j > z} \Psi(Y(z_j))w_j \quad \text{(11)} \]

Similarly,

\[ \pi(S \setminus S^*) \geq \sum_{j \in S \setminus S^* \land j \leq z} \Psi(Y(z_j))w_j + \sum_{j \in S \setminus S^* \land j > z} \Psi(Y(z_j))w_j \quad \text{(12)} \]

As \( P \geq 1 \), eq(7) can be modified as

\[ \frac{OPT(\sigma)}{A(\sigma)} \leq \frac{P + \Psi(Z + r_z)(B - W)}{P + \pi(S \setminus S^*)} \quad \text{(13)} \]

Considering eq(12), eq(13) can be modified as

\[ \frac{OPT(\sigma)}{A(\sigma)} \leq \frac{P + \Psi(Z + r_z)(B - W)}{P + \pi(S \setminus S^*)} \quad \text{(14)} \]

From definition of \( \Psi(z) \), we get \( \Psi(z - r_z) = L \). As \( z \leq L \), we get \( z - r_z \leq Z + r_z \) and \( \Psi(z - r_z) \leq \Psi(Z + r_z) \). Hence,

\[ L \leq \Psi(Z + r_z) \quad \text{(15)} \]

From equations (15) and (4), (11) can be modified as

\[ P \leq \Psi(Z + r_z) \sum_{j \in S \setminus S^*} w_j = \Psi(Z + r_z)W \quad \text{(16)} \]

Using (16) in numerator of (14), we get

\[ \frac{OPT(\sigma)}{A(\sigma)} \leq \frac{\Psi(Z + r_z)B}{P + \pi(S \setminus S^*)} \quad \text{(17)} \]

From (11) and (12),

\[ \frac{P + \pi(S \setminus S^*)}{B} = \frac{1}{B} \left( \sum_{j \in S \setminus S^* \land j \leq z} \Psi(Y(z_j))w_j + \sum_{j \in S \setminus S^* \land j > z} \Psi(Y(z_j))w_j \right) \geq \frac{1}{B} \left( \sum_{j \in S \setminus S^* \land j \leq z} \Psi(Y(z_j))w_j + \sum_{j \in S \setminus S^* \land j > z} \Psi(Y(z_j))w_j \right) \quad \text{(18)} \]
where $\Delta z_j \equiv z_{j+1} - z_j = w_j / B$. As $\forall j : w_j \ll B$, $\Delta z_j \approx 0$.

Eq(18) becomes

$$\frac{P_1 + P_2}{B} \geq \int_0^\bar{z} L dz + \int_\bar{z}^Z \Psi(z_j - r_z) dz = L \left( r_z + \left( \frac{U_e}{T} \right)^2 \frac{z_j}{\frac{U_e}{T} + r_z} \right)$$

Substituting (19) in (17) and simplifying, we get

$$\frac{OPT(\sigma)}{A(\sigma)} \leq \frac{(ln \left( \frac{U_e}{T} \right) + 1) \left( \frac{U_e}{T} \right)^2 r_z}{1 + F(Z)}$$

where $F(Z) = \frac{L r_z (ln \left( \frac{U_e}{T} \right))^z}{\Psi(Z)}$. Since $F(Z) \geq 0$,

$$\frac{OPT(\sigma)}{A(\sigma)} \leq \left( ln \left( \frac{U}{L} \right) + 1 \right) \left( \frac{U_e}{T} \right)^2 r_z$$

Thus algorithm ONLINE-KP-THRESHOLD-$r_z$ can achieve a competitive ratio of $(ln \left( \frac{U_e}{T} \right) + 1) \left( \frac{U_e}{T} \right)^2 r_z$ in the CLOUDY KP/r_z setting under the specific assumptions. This is an upper bound for the competitive ratio of any optimal algorithm for the setting. The competitive ratio obtained is $(\frac{U_e}{T})^{2r_z - z_j}$ times the ratio for the algorithm[11] in a non-cloudy setting $(ln \left( \frac{U}{L} \right) + 1)$. The lower bound for competitive ratio of any algorithm in a non-cloudy setting[11] is $ln \left( \frac{U}{L} \right)$. Since any solution to the cloudy version of the problem cannot give a lower bound, the currently known range of the competitive ratio of an optimal algorithm(cloudy setting) becomes $\left[ ln \left( \frac{U_e}{T} \right), (ln \left( \frac{U}{L} \right) + 1) \left( \frac{U_e}{T} \right)^2 r_z \right]$. Finding the exact lower bound and an optimal algorithm which achieves the same is an open problem.

IV. RELATED WORKS

Works in mobile cloud computing consider several context parameters like device’s energy consumption, local execution time, cost incurred for offloading, etc. to decide which portions of the app should be offloaded [4], [10]. However, many of these works are restricted to single mobile users and therefore may give suboptimal aggregate performance as the number of clients becomes very large and the budget is constrained. Recent works [4], [10] incorporate budget constraints, but rely on centralized scheduling and hence cannot scale to large number of users as in our setting. Some works [1] in cloud bursting consider certain duration for spending the budget as well as an elastic set of resources. But they also do not address huge volume of user requests and rely on centralized scheduling.

In contrast, existing works in P2P with cloud assistance deal with large number of clients and focus on improving the aggregate system performance [8], [7], [6], [3]. However, most of them do not constrain the cloud budget explicitly. CloudCast[6] constrains the cloud budget to be used for content dissemination application. However, the application does not have high task variability and does not target devices with context concerns like battery level. Hence, context and task variability related issues are not addressed.

The online knapsack problem has been studied by [5], [11] under different assumptions on input distribution, knowledge about the number of items, range of weights, and values. Unlike our setting, these works do not restrict the view of the knapsack during decision making.

V. CONCLUSIONS AND SCOPE FOR FUTURE WORKS

Cloudy knapsack problems is an effective way to model the distributed optimization required for large scale cloud-assisted applications with budget and context constraints. We illustrated the effect of cloudiness in global view through simulations and theoretical results. Insight into the relationship between the cloudiness in global view and the value addition can also lead to design of novel peer-to-peer view synchronization mechanisms suitable for cloud-assisted systems. Further variants can be defined with different distributions for the error like Normal distribution. Lower bounds and optimal algorithms for these variants are open problems. Techniques to handle selfish peers which try to maximize their private gain without considering the overall value addition in this setting is also an open area of research.

REFERENCES


