Coalition Formation towards Energy-Efficient Collaborative Mobile Computing

Liyao Xiang[1], Baochun Li[1], and Bo Li[2]

[1]Department of Electrical and Computer Engineering, University of Toronto
[2]Department of Computer Science, Hong Kong University of Science and Technology

***Abstract*—With mobile offloading, computation-intensive tasks can be offloaded from mobile devices to the cloud to conserve energy. In principle, the idea is to trade the relatively low communication energy expense for high computation power consumption. In this paper, we propose that computation-intensive tasks can be distributed among nearby mobile devices, and focus on the case that a group of mobile users may collaborate with one another with one common target job. In particular, a user can reduce its own energy consumption by delegating a portion of the job to nearby users in a coalition. We propose distributed collaboration strategies based on game theory, and formulate the problem as a non-transferable utility coalition formation game in which users join or split from coalitions depending on the local preference. The stability of the resulting partition is studied. We show through simulations that our proposed algorithm reduces up to 22% of the average energy costs compared to the non-cooperative case, and the running time scales well as the number of users grows.**

I. Introduction

It is increasingly common for users with mobile devices to depend on the assistance from the cloud for their computation-intensive tasks. Cyber foraging and cloudlets [1] have long been proposed as a way to liberate mobile devices from severe resource constraints. Existing works have largely focused on migrating computation to the cloud, but new features such as *Continuity* in Mac OS X and *WatchKit* in iOS have made it possible to offload tasks to nearby devices via Bluetooth Low Energy. For example, a user with an Apple Watch can easily offload an unfinished email from her watch to her iPhone.

Indeed, recent works (*e.g.*, [2]) in the literature have proposed that the computational resources of a collection of collaborative mobile devices can be pooled for more efficient energy consumption. This clearly points out a trend that more and more mobile applications are designed to be executed in a distributed fashion, in collaboration with nearby devices that the same group of users owns. In fact, new research areas such as social sensing and crowdsourcing require the ability to recruit more than one mobile device to jointly work as a group.

In previous works [2]–[7], mobile devices cooperate for the purpose of conserving energy or speeding up computation, under the assumption that the users are highly collaborative. However, mobile users are not altruistic in real life, rarely contributing freely to one another; typically, they tend to exchange computation results only with the collaborating users they choose, in order to reduce the workload on their own devices. Different from the previous work, we focus on how collaborating users are to be selected by each mobile user, where the goal is to minimize the energy cost. The intuitive optimal solution is to have everyone play the best strategy responses where each of them maximizes its own interest by acting according to the others' actions, but this scheme takes too many iterations before converging to an agreement. We thus consider an alternative way that multiple coalitions for collaboration are formed, and each user can choose its collaborators by leaving or joining a coalition.

In particular, we focus on the problem of how multiple coalitions are to be formed among a group of mobile users to reduce the average energy cost. We assume that as soon as a mobile user decides to form a coalition with other users, it enters a binding agreement with the other users within the coalition, and considers the benefit of the coalition as a whole. All users are cooperative and individually rational. With this assumption in place, we study the problem of how mobile users that are located in proximity to one another can form coalitions to complete a common computation-intensive job, with the objective of minimizing the energy consumption.

The key questions to ask are: given a job partitioned into several tasks, how do heterogenous mobile users form coalitions? Within each coalition, how do we distribute the tasks to each user? We assume that users are only concerned about the energy used for computation and network connections, in that the two constitute a major part of the energy consumption on a mobile device. The variability in energy characteristics of different devices and network connections should also be taken into consideration when allocating tasks. If we describe the computation capability and network connections of each mobile device by a resource graph, our problem can be formulated as an optimization problem over all partitions of the graph. However, seeking a centralized solution to minimize the overall energy cost is tricky and impractical.

To address this problem, our contributions in this paper are two-fold. *First,* we formulated the task distribution problem as a 0-1 integer quadratic programming problem to determine the assignment of tasks to devices, in order to minimize the overall energy costs. By assuming a central arbitrator exists, we are able to obtain an optimal partition of the resource graph, as well as an optimal task assignment within each partition. *Second,* we proposed a distributed coalition formation algorithm for multiple mobile users, who are self-organized into disjoint independent coalitions.

The remainder of this paper is organized as follows. The underlying infrastructure is introduced and the system model is described in Sec. III. Sec. IV formulates the problem as a centralized task distribution problem. Sec. V takes another perspective at the problem and reconsider it as a coalition game among users. A simple merge-and-split distributed algorithm is then proposed for forming coalitions. The performance of this algorithm is then evaluated against non-cooperative and centralized schemes.

II. Related Work

Most previous works in the area of application migration or code offloading, such as MAUI [8], CloneCloud [9], and ThinkAir [10], only consider offloading portions of a mobile application to the cloud. Even though the cloud has abundant computational power, the latencies and energy consumption may not be worth the effort of such offloading for applications that are sensitive to these factors. To mitigate the negative effects introduced by long latencies to the cloud over the Internet, it was proposed in [2], [4], [5] that the computing power in the proximity of a mobile device should be used to speed up execution or to save energy. This is especially useful when Internet access is expensive or even unavailable. Our work is focused on collaboration across mobile devices.

Recent works, such as Serendipity [3] and Langford *et al.* [6], focused on the collaboration among mobile devices. In contrast to Serendipity, our work does not make the assumption that there exists an “initiator” mobile device, which “borrows” idle CPU cycles from nearby devices for free. Instead, in our work, a device “trades” its computation power in exchange for the computation power of other devices that share the same target job with. In addition, our work is different from Langford *et al.* [6] in that the devices are not assumed to participate on a voluntary basis or based on external incentives. In reality, it is challenging to encourage a group of mobile devices to contribute on a voluntary basis, with or without incentive mechanisms.

In this work, we assume users cooperatively form multiple coalitions, and are individually rational. The coalitions are formed among users with certain agreements towards a common objective of reducing the energy costs. The users are individually rational in a sense that no user prefers being alone to its coalition if the coalition yields more benefit.

At first glance, being self-interested and choosing the best strategy response may be the natural choice of a user; however, it is highly impractical in reality. Instead, we adopt a merge-and-split stability concept to allow users to switch between different coalitions by either merging into one or splitting up into any groups, when at least one user sees strict cost reduction while not hurting other users’ benefits. The problem in our work is a non-transferable utility (NTU) game, since the energy expense cannot be transferred across different devices. The coalition formation approach we adopt can be considered as a special case of the generic solution [11].

III. System Model

In this paper, mobile users targeting the same job would potentially collaborate and share their computation and bandwidth resources. As a starting point, we first assume that there is an arbitrator in the cloud that profiles all the resource information beforehand, and responds to users’ requests of launching a job. The arbitrator decides the optimal way of organizing users into groups, and assigning tasks to each group. While the centralized scheme is impractical, it provides the optimal bound for the distributed schemes. In a distributed scheme, each mobile user initiates its job request by discovering neighbors who target at the same job. The energy profiles of each device and network channel are exchanged among them, based on which the estimated energy consumption on each device is computed. The group of users continues to either merge with new group members, or split up according to certain preference order until converging to a stable collaboration scheme. This local strategy has the advantage over the centralized scheme, in that it is resistant to network topology changes due to user mobility.

We assume that a typical job on the mobile device is a program which can be analyzed by static analysis tools, i.e., given inputs and the program code, the instruction cycles of each part, and the amount of data transferred between parts can be estimated. Dynamic analysis tools can also be adopted to gradually learn the behavior of the program. Eventually, we obtain the call graph of a job as such in Fig. 1, in which the partitions are determined by functionalities. Each block in the figure stands for an atomic task that can only be executed at one place as one piece; the arrow between the blocks represents the data flow or state transfer from one part to another. For example, the outputs of “Image Capturing” are the inputs of “Data Backup” and “Features Extraction”. If two sequential tasks are executed on different devices, the link between them is considered an external link. More specifically, we represent the job workflow with a directed graph $\mathbb{G}^t = (\mathbb{V}, \mathbb{E})$. Each node $i \in \mathbb{V}$ is a task associated with computation cycles c_i, and each link $e_{i,j} \in \mathbb{E}$ between nodes i and j is associated with $d_{i,j}$ which denotes the amount of data transferred from task i to task j.

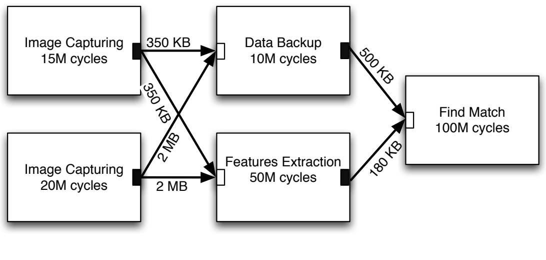


Fig. 1. The Call Graph of an Example Job

Due to the disparity in hardware and physical locations, different mobile devices have different energy profiles, and are connected through different network channels. We describe the group of collaborative mobile users using a directed resource graph $\mathbb{G}^r = (\mathbb{N}, \mathbb{L})$, of which each node $n \in \mathbb{N}$ stands for a device and a link $l_{n,m} \in \mathbb{L}$ represents the communication channel from user n to m. Here, we adopted the linear model in previous work [8] that the smartphone energy consumption is linear to the CPU utilization. The energy characteristics of each device are profiled beforehand, that a_n Joule is consumed to execute one unit CPU cycle on the device. The energy consumption on each network channel is also estimated with network profilers as $b_{n,m}$ Joules/KB representing the amount of energy consumed for a unit of data transferred via link $l_{n,m}$. Notice that, due to dynamic voltage and frequency scaling, as well as network condition changes, both a_n and $b_{n,m}$ are subject to changes in execution, thus energy profiling need to be done periodically.

From a holistic point of view, we are interested in partitioning the resource graph $\mathbb{G}^r$ into subgraphs of which users form coalitions, and then map the task graph $\mathbb{G}^t$ onto each subgraph. If the coalition consists of only one mobile user, the energy cost would entirely consist of the computation energy; if many users cooperate in that coalition, the computation energy consumption per device decreases at the cost of increasing communication energy. Between these two extremes, there should be a sweet spot which minimizes the average energy cost. The problem is further complicated by the fact that the decisions of coalition formation and task distribution are correlated with each other.

In the following two sections, we will respectively describe the centralized and distributed approaches to our problem.

IV. Task Distribution for Mobile Applications

There are a number of ways to partition the resource graph to form coalitions, as well as a number of different methods of assigning tasks within each coalition. Our objective is to find out the optimal way to partition the resource graph and map the job workflow graph onto it to minimize the average energy consumed on each device. More formally, let $\mathcal{B}$ be the set of all partitions of graph $\mathbb{G}^r$, and T be one user coalition of the partition $\mathcal{P} \in \mathcal{B}$. Our objective is:

$$\min_{\mathcal{P}\in\mathcal{B}} \sum_{T\in\mathcal{P}} \min C(T). \tag{1}$$

where $C(T)$ represents the sum of the energy expense on all mobile devices in the coalition T. Since the total number of mobile users is implicitly included in the resource graph, we minimize the overall energy consumption instead. Given one partition of the graph, we map the job onto the subgraph in a way shown by Fig. 2. Mobile devices n_1 to n_5 form two coalitions T_1, T_2 which execute the job that consists of three tasks i_1, i_2, i_3.

In addition to the notations in Sec. III, we define the following notations:

- $l_{n,m} = 1$ represents that device n and m are connected on $\mathbb{G}^r$, and 0 otherwise.
- $e_{i,j} = 1$ represents that the output of task i is the input of task j on $\mathbb{G}^t$, and $e_{i,j} = 0$ represents that the task i, j are not directly associated.
- $s_{i,n} = 1$ represents that task i is assigned to device n, and 0 otherwise.
- $r_{i,n} = 1$ represents that task i is able to be executed on device n, and 0 otherwise.

Partitions of the resource graph can be generated in different ways, for now, we just assume coalition T is given, and minimize the energy expense over that coalition as follows.

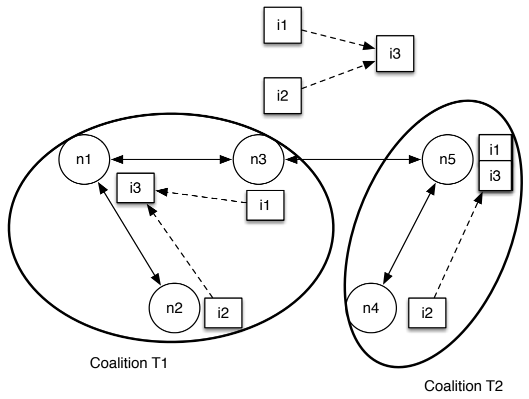


Fig. 2. An example of mapping jobs to a set of mobile devices

$$\min C(T) = \min \sum_{n\in T} \phi_n(T), \tag{2}$$

where $\phi_n(T)$ represents the energy consumption on device n, $\forall n \in T$:

$$\phi_n(T) = \begin{cases} \infty, \text{ if } |T| > |\mathbb{V}|, \\ E_n(T), \text{ otherwise.} \end{cases} \tag{3}$$

$$E_n(T) = a_n \sum_{i\in\mathbb{V}} s_{i,n} c_i + \sum_{i\in\mathbb{V}} \sum_{\substack{m \neq n,\\ m,n \in T\\ i\neq j,\\ i,j\in\mathbb{V}}} s_{i,n} s_{j,m} (e_{i,j} b_{n,m} d_{i,j} + e_{j,i} b_{m,n} d_{j,i}) \tag{4}$$

We consider it infeasible to divide an atomic task, thus setting a barrier when there are more than $|\mathbb{V}|$ users in a coalition. The first part of (4) is linear representing the computational energy costs, while the second part is quadratic w.r.t $s_{i,n}$ denoting the network energy expense on all external links of the device. The constraints are:

$$\sum_{n\in T} s_{i,n} = 1, \ \forall i \in \mathbb{V}, \tag{5}$$

$$\sum_{i\in\mathbb{V}} s_{i,n} \geq 1, \ \forall n \in T, \tag{6}$$

$$s_{i,n} s_{j,m} e_{i,j} \leq l_{n,m}, \forall i \neq j, \forall n \neq m, n, m \in T \tag{7}$$

$$s_{i,n} \leq r_{i,n}, \forall i, \forall n \in T, \tag{8}$$

$$s_{i,n} \in \{0,1\}, \forall n \in T, \forall i \in \mathbb{V}. \tag{9}$$

The given parameters in the above are a_n, c_i, $l_{n,m}$, $b_{n,m}$, $e_{i,j}$, $d_{i,j}$, and $r_{i,n}$. Given coalition T, the optimization problem in (2) is to find the optimal $s_{i,n}$ to minimize the energy consumption of a coalition. Constraint (5) means each task i is executed on one device only. And constraint (6) presents the requirement that each user in the coalition should be involved with at least one task. (7) shows that device n and m can collaborate with each other only when the external link between them exists. (8) denotes that executing tasks is constrained by the resource given on that device. The problem of (2) is quadratic and non-convex, and the constraints of (7) can be non-convex as well, thus there is no existing solver to solve this integer programming problem. However, since the solution space is not huge ($2^{|\mathbb{V}|*|T|} \leq 2^{|\mathbb{V}|^2}$), a heuristic approach can be used to find the optimal solution for the centralized approach.

V. Coalition Formation of Mobile Users

As a first step, we seek a centralized solution that minimizes the energy consumption of all users. In the centralized approach, we assume that there is an arbitrator who makes decisions with respect to the coalition structures, based on the energy profiles of all participating users, and within each coalition assigns tasks to each user. However, it is shown in [12] that the problem in Eq. (1) is NP-hard, as the number of possible partitions in graph $\mathbb{G}^r$ grows exponentially with the number of users. Moreover, in the real world, an arbitrator hardly exists. Thus we proposed a distributed solution in this section to enable each coalition to make local decisions to transform itself depending on its preference.

A. A Coalition Game for Collaborative Mobile Devices

A coalition game is a game where groups of players may enforce cooperative behavior, hence the game is a competition between coalitions of players. A set of analytical tools have been developed [11] in the area of coalition formation, here we adopt a generic approach.

In the settings of coalition formation, the collaboration among mobile devices is modeled as a coalition game (N, v) where N is the set of players (or users), and v is the utility function or value of each coalition, which in our case corresponds to the energy cost $C(T)$ for users in coalition T. Notice that if the coalition size is larger than the number of atomic tasks of the job, i.e., $|T| > |\mathbb{V}|$, the job cannot be partitioned within a coalition without leaving any of them idle, thus the corresponding utility is defined as negative infinity. Generally, the utility $v(T)$ is defined as a decreasing function of the energy cost given by:

$$v(T) = -C(T) = -\sum_{n \in T} \phi_n(T) \tag{10}$$

By definition given in [13], a coalition game (N, v) is said to have *transferable* utility if the utility function $v(T)$ can be arbitrarily apportioned between the coalition's players. Obviously, the collaborative computing game (N, v) defined as above has *non-transferable* utility, since the energy cost per device cannot be arbitrarily apportioned once the tasks are assigned, thus the utility cannot be transfered between users. In the next section, we discuss how user groups can be formed within such a coalition game framework.

B. Coalition Formation: Proposed Algorithm

Before showing our proposed algorithm for coalition formation, we begin by formalizing the concept of comparison relation.

Definition 1: A *collection* is any family $T = \{T_1, ..., T_l\}$ of mutually disjoint coalitions. If additionally $\bigcup_{j=1}^{l} T_j = \mathbb{N}$, the collection T is called a *partition* of $\mathbb{N}$.

Definition 2: Assume A and B are partitions of the same set C, a *comparison relation* $\triangleright$ is defined as, $A \triangleright B$ means that the way A partitions C is preferable to the way B partitions C.
Notice that the comparison relation $\triangleright$ is defined on partitions of the same set of players. Partitions of different sets of players are incomparable w.r.t. $\triangleright$. As illustrated in [11], the comparison relations on partitions are induced in a canonical way from the corresponding comparison relations on multisets of reals as such:

$$A \triangleright B \iff v(A) \triangleright v(B). \tag{11}$$

The comparison relation $\triangleright$ on multisets of reals can be defined by different orders. The orders can be defined by the sum of the utility of all users in that coalition, or individual utility of each user in the coalition. Notice that individual utility here does not mean users make decisions based on its own interest, but rather the decision is agreed among all users w.r.t each user's interest in the coalition. More specifically, the individual value for users in collection C can be given by:

$$\psi(C) = \{\psi_n(C_i) | \forall n \in C_i, \forall C_i \in C\} \tag{12}$$

which can also be viewed as a sequence of individual utilities. It is defined in a way that the comparison relations are anonymous which means not distinguishing different players. The comparison relations should be based on the individual utility. In this paper, we adopt *Pareto order* as the comparison relation, so that a stable partition presents a Pareto optimal utility distribution for all users.

The transformation of coalitions through Pareto order can only happen when it at least strictly improves the utility of one user. As we prove later, its stable state is a state that no user can join or split from its original coalition to improve its own utility if the action hurts any other's interest. Given two partitions $T = \{T_1, ..., T_l\}$ and $T^{'} = \{T_1^{'}, ..., T_k^{'}\}$ of the same set of players $\mathbb{M}$, we let $\psi(T) = \phi(T)$, and the comparison relation between T and $T^{'}$ is given by:

$$T \triangleright T^{'} \iff \forall n, \phi_n(T) \leq \phi_n(T^{'}) \text{ and } \exists m, \phi_m(T) < \phi_m(T^{'}) \tag{13}$$

where T and $T^{'}$ is of the same length so they can be compared with Pareto order.

We now define two rules that allow us to transform coalitions w.r.t the comparison relation of (13). We define $T_1, ..., T_k, P \subseteq \mathbb{N}$ as disjoint user coalitions.

Merge: $\{T_1, ..., T_k\} \cup P \rightarrow \{\bigcup_{j=1}^{k} T_j\} \cup P$, where $\{\bigcup_{j=1}^{k} T_j\} \triangleright \{T_1, ..., T_k\}$.

Split: $\{\bigcup_{j=1}^{k} T_j\} \cup P \rightarrow \{T_1, ..., T_k\} \cup P$, where $\{T_1, ..., T_k\} \triangleright \{\bigcup_{j=1}^{k} T_j\}$.

Using the rules above, multiple coalitions can merge into a larger one if at least one user strictly reduces its energy usage. Likewise, one coalition can be split into smaller coalitions if splitting improves the utility of at least one user. Because the number of different partitions is finite, every iteration of the merge and split rules terminates. Hence we have

Theorem 1: Given the comparison relation defined in Definition 2, every iteration of merge and split rules terminates.

However, when different sequences of merge and split rules are applied to the initial partition, the outcome may be different. In the following section, we study the conditions under which it is guaranteed that arbitrary sequences of these two rules yield the same outcome. But before that, we illustrates our algorithm based on these two simple rules in Algorithm 1.

Algorithm 1 Collaborative Computing Game through Merge and Split

```
Input: Initial partition $T = \{T_1, ..., T_l\} = \mathbb{N}$
Output: Final partition $T^{final}$
repeat
  $T$ = Merge($T$);
  $T$ = Split($T$);
until merge and split terminates.
$T^{final} = T$.
```

VI. Stable Coalitions

In this section, we study the stability of the proposed coalition formation game (N, v).

A. Stability Analysis

Since in the collaborative mobile computing game, we assume that all users are cooperative and individually rational rather than purely motivated by its own benefit, the stability defined is similar but different from core or Nash stability. We will prove that our stability indicates *contractually individual stability* which refers to a state that no player can benefit from moving its coalition to another without making others worse off. Formally, we consider a partition T of the grand coalition N is stable if for every collection $C \subset N$ that,

$$C[T] \triangleright C, \tag{14}$$

where

$$C[T] = \{T_1 \cap \bigcup\{C\}, ..., T_k \cap \bigcup\{C\}\} \backslash \{\emptyset\}.$$

$C[T]$ actually refers to the collection C partitioned in T's way. The intuition of the stability defined in Eqn. (14) is that in partition T, no group of players is interested in leaving T and form any other collection. It is also called $\mathbb{D}_c$-stable in [11]. We can prove that

Theorem 2: The stability defined in Eqn. (14) indicates contractually individual stability.

Proof: To prove the theorem, we first give the definition of contractually individual stability. It refers to a partition in which no player can benefit from moving from its coalition to another existing coalition, while making any member in either its old or its new coalition worse off. In the collaborative mobile computing scenario, it is formally defined as: $\nexists m \in T_k$, and $T_i \in T \cup \{\emptyset\}$, such that

$$\phi_m(T_i \cup \{m\}) < \phi_m(T_k), \tag{15}$$

$$\phi_n(T_i \cup \{m\}) \leq \phi_n(T_i), \forall n \in T_i, \tag{16}$$

and

$$\phi_s(T_k \backslash \{m\}) \leq \phi_s(T_k), \forall s \in T_k \backslash \{m\} \tag{17}$$

are satisfied simultaneously.

Assume T is a stable partition as defined in Eqn. (14), and let $C = \{T_k \backslash \{m\}, T_i \cup \{m\}\}$. $C[T]$ is the collection C partitioned by means of T thus $C[T] = \{T_k, T_i\}$. By rewriting the comparison relation with Pareto order, we get:

$$\phi_n(\{T_k, T_i\}) \leq \phi_n(\{T_k \backslash \{m\}, T_i \cup \{m\}\}), \forall n, \tag{18}$$

$$\phi_x(\{T_k, T_i\}) < \phi_x(\{T_k \backslash \{m\}, T_i \cup \{m\}\}), \exists x. \tag{19}$$

We prove that if Eqn. (18) and (19) are satisfied, Eqn.(15) –(17) cannot hold at the same time for any $m \in T_k$. We first assume Eqn. (15)–(17) are true simultaneously. Rewriting Eqn. (18) as two inequalities, we have:

$$\phi_s(T_k) \leq \phi_s(T_k \backslash \{m\}), \forall s \in T_k \backslash \{m\}, \tag{20}$$

and

$$\phi_n(T_i) \leq \phi_n(T_i \cup \{m\}), \forall n \in T_i. \tag{21}$$

Hence Eqn. (16), (17) are true only if

$$\phi_s(T_k) = \phi_s(T_k \backslash \{m\}), \forall s \in T_k \backslash \{m\}, \tag{22}$$

and

$$\phi_n(T_i) = \phi_n(T_i \cup \{m\}), \forall n \in T_i. \tag{23}$$

Considering Eqn. (19), Eqn. (22)(23) hold only if $x = m$, i.e.,

$$\phi_m(T_k) < \phi_m(T_i \cup \{m\}). \tag{24}$$

However, the last step violates Eqn. (15). Thus Eqn. (15) –(17) cannot hold true at the same time for any $m \in T_k$. We hereby prove the defined stability indicates contractually individual stability. ■

B. Revisiting Stability

Ideally, we wish that all users are partitioned in a way that no players in T could leave that partition and form any new coalition. However, the stability concept defined in the last section is too strict to satisfy all general cases. Hence, we relax the definition a bit by redefining stability. We first give the definition of a *defection function*: a function $\mathbb{D}$ that assigns to each partition a family of collections in the grand coalition. And we redefine stability as in Eqn. (14) for all collections $C \in \mathbb{D}(T)$. By defection function we assume users can only leave the original partition by forming new coalitions in $\mathbb{D}$. And the redefined stability refers to a state that if we only allow users transforming coalitions following the defection function $\mathbb{D}$, no group of players can transform coalitions without sacrificing the others' benefit.

In this paper, we consider the case where all coalitions are formed by rules of merge and split. A partition T is defined as *merge-and-split stable* if no group of players is able to leave T by means of merges or splits without violating comparison relations. More formally, we give two conditions to determine if a partition T is merge-and-split stable. A partition $T = \{T_1, ..., T_l\}$ of N is merge-and-split stable if and only if:

1) $T_i \triangleright \{C_1, ..., C_k\}$ where $\bigcup_{j=1}^{k} C_j = T_i, \ \forall T_i \in T$.

2) $P \rhd \bigcup_{T_i \in P} T_i, \ \forall P \subseteq T$.

The first condition means that given the stable partition T, none of the coalitions can be split up following the defined comparison relation. And the second condition refers to that any of the existing coalitions cannot be merged into one w.r.t. comparison relation.

C. The Unique Outcome

We now investigate the conditions under which every iteration of the merge and split rules yields the same outcome regardless of their order. According to the results of [11], if $\rhd$ is a comparison relation, and T is a $\mathbb{D}_c$-stable partition, then T is the outcome of every iteration of the merge and split rules. Thus, we have

Theorem 3: If a partition T satisfying Eqn. (14) exists for the grand coalition N, every iteration of the merge and split rules yields the same unique outcome T.

The proof of the theorem above is omitted here due to space constraints. Interested readers can refer to [11] for a detailed proof. Here we illustrate the conditions to find if a partition is $\mathbb{D}_c$-stable. Given a partition $T = \{T_1, ..., T_l\}$ of N, T is $\mathbb{D}_c$-stable if and only if the following are satisfied:

1) for each pair of disjoint coalitions A and B such that $A \cup B \subseteq T_i, \forall T_i \in T$,
$$A \cup B \rhd \{A, B\}.$$
2) for each coalition $C \subseteq N$ such that $C \nsubseteq T_i, \forall T_i \in T$
$$C[T] \rhd \{C\},$$
where
$$C[T] = \{T_1 \cap \bigcup\{C\}, ..., T_l \cap \bigcup\{C\}\} \backslash \{\emptyset\}.$$

Compared to merge-and-split stability, the $\mathbb{D}_c$-stable is stricter in that it requires two conditions. First, any two disjoint subsets of a stable coalition must be joined w.r.t comparison relation; second, any collection partitioned in the stable partition's way is preferred over other methods.

VII. Performance Evaluation

A. Simulation Setup

For a typical job, we can apply program partition and analysis tools such as MAUI profilers [8] to estimate the CPU cycles per task and the amount of data transferred between them. All these information is fed as input to the task distribution optimization problem to determine which task should be assigned to which device to minimize the overall energy consumption. In our simulations, we use the call graph produced by the program profiler as the task graph. As a starting point, we run simulation against a typical call graph structure as shown in Fig. 2. The job is partitioned into three tasks, and the results of the sequentially former two tasks are merged as input to the third task. The CPU cycles of each task are chosen randomly between $20-100$ M cycles and the data transferred from 10 to 1000 KB on each link.

For setting the parameters in the mobile device resource graph, we adopted data from the energy profiles in the previous literature. As pointed out in [14], the energy for transmitting a fixed amount of data is related to the bandwidth and connection condition at that time. Typically, as verified in [8], downloading 50 KB data over WiFi consumes approximately 20 mJ/KB while downloading over GSM or 3G network costs around $50 - 200$ mJ/KB. However, transmitting data in bad connectivity conditions consumes even more since much energy is wasted on repeated attempts for transmission. For the typical CPU cycles energy consumption, we use the device profiling results of [8], which shows a simple linear model is sufficient in describing the relation between CPU cycles and corresponding energy consumption. Without loss of generality, we assume the amount of energy consumption in data transmission is uniformly distributed on $[20, 200]$ mJ/KB, and the computation energy cost on each device is uniformly drawn from $[40, 60]$ mJ/M cycles.

In simulations, we compare the performance of our merge-and-split algorithm against the non-cooperative, centralized approaches for different tasks on varied resource graphs. The centralized algorithm traverses through all partitions of the given resource graph to obtain the optimal partition, *i.e.*, the partition that minimizes the total energy cost. In the non-cooperative setting, each user finishes the job individually without collaborating with others. The metric is the average energy consumed per user since we concern about how much energy saved compared to the non-cooperative case and how far it is from the optimal bound. The average running time of each algorithm is also evaluated to show the efficiency of our algorithm. Besides that, the components of the energy cost are examined to see how the computation and communication energy proportion vary respectively against different settings of the resource graph. In the end, the average size of the resulting coalition is compared with the optimal case.

B. Performance Evaluation

First, we compare the performance of the merge-and-split algorithms with the centralized one, and the non-cooperative case in terms of average energy cost per user for different user numbers and connecting ratios among the users. The connecting ratio reflects the degree to which users are connected. Simulation results are given in Table I and Fig. 3(a), 3(b) respectively.

Table I shows the average energy cost per user over all different settings of the resource graph. The merge and split algorithm reduces the cost by a significant 21.79% compared with the non-cooperative case, and leaves only 1.59% gap to the optimal centralized approach.

TABLE I
Average Energy Cost per User over All Cases

	Non-coop	Centralized	Merge & Split
Energy Cost(J)	8.49	6.86	6.97

Fig. 3(a) and 3(b) show respectively the average energy cost per user. We average the results over the randomly generated resource graph given the number of users and their connecting ratio. Low connecting ratio means most of the

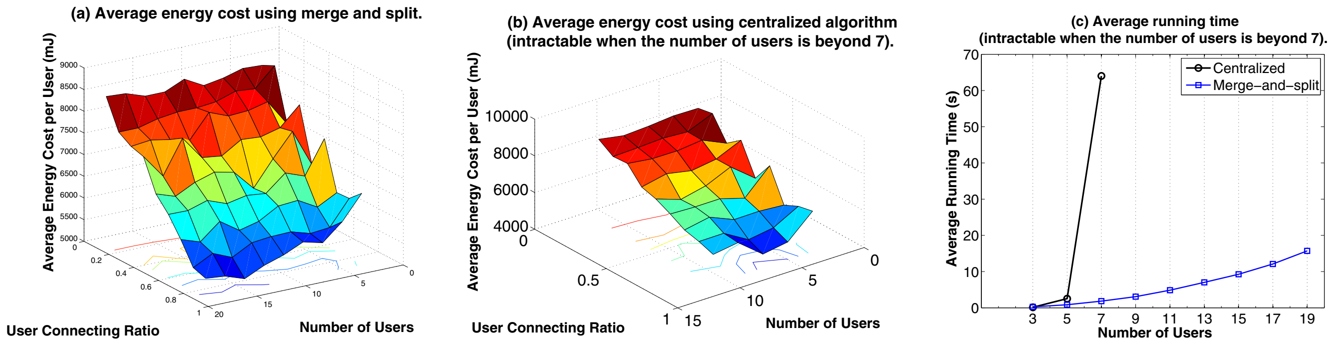


Fig. 3. Simulation results: Average energy cost per user and average running time.

nodes on the resource graph are disconnected nodes, while high connecting ratio indicates the resource graph is close to a fully-connected graph. The general trend of Fig. 3(a) shows that as the connecting ratio gets higher, or the number of users gets higher, the average energy cost per user decreases. If we take a closer look, we found when the connecting ratio is low, the variance of the average energy cost remains low w.r.t. the number of users; however, as the connecting ratio increases, the average energy cost decreases more dramatically as the number of users increases. This is because coalitions are easier to form when user connecting ratio is high. From the mathematical point of view, the reason is that higher number of users in proximity usually allow the user to have more options when choosing collaborators, *i.e.*, the constraints (7) are relatively more loose, thus leading to a decreased average energy cost per user.

However, with the increase of the connecting ratio, the gap between the merge-and-split algorithm and the centralized algorithm also gets bigger. The gap exists for two reasons. First, a stable state does not minimize the overall energy cost as it does not allow the user to switch between coalitions even if it improves the overall benefits, when the act hurts other user's benefit. Second, even if the stable state exists, merge-and-split algorithm may terminate before reaching that state. In that case, some partition sets of the resource graph has not been traversed upon termination.

In each set of experiments, although the centralized algorithm gives the optimal bound, it becomes hardly tractable as the total number of users increases beyond 7. Fig. 3(c) shows the average running time over all cases w.r.t. different number of users for each algorithm. As we can see from the figure, the average running time of the centralized method increases exponentially with the number of users while merge and split algorithm performs significantly better in scaling as the number of users increases.

Fig. 4 shows the average coalition size when the number of users and the connecting ratio vary. We observe that, with merge and split algorithm, the coalition size becomes larger with the increase of the total user number, and it also increases with higher connecting ratio. We also observe that as the connecting ratio gets higher, the centralized approach gives larger coalition size than the merge and split scheme, and the gap between them increases with the connecting ratio. That is to say, in the optimal case, users tend to form larger coalitions to reduce the overall energy cost; but using merge and split by Pareto order, users are more restricted by others' interests leading to the result that larger coalition is harder to form. The coalition effect is more obvious when connecting ratio gets higher.

At last, we illustrate the simulation results for the average proportion of computation and communication cost in different cases. Fig. 5(a) shows the proportion varying with different conditions of users. Overall, as the number of users grows, or user connecting ratio increases, the proportion of computation energy decreases and the network connection energy shows the complimentary trend. This trend corresponds to the growth of coalition size since the larger the coalition is, the more communication energy cost is incurred.

Fig. 5(b) shows the energy cost of a face recognition application for the non-cooperative, centralized, and merge-and-split methods respectively. The first three columns show the results for the application with the call graph from [8], and the rest are the results when we reduce the volume of transferred data to 1/50, 1/100, 1/1000 of the original amount. We found while the application may be fit to run in a mobile cloud environment, it does not really save energy by running on a group of collaborative mobile users, essentially because the non-trivial amount of data required for transferring states between devices. When we reduce the transferred data, we observe substantial reduction in the overall energy cost with our algorithm. The results show our merge and split algorithm is typically more effective on jobs with less data transferred across different tasks.

VIII. Conclusions

In this work, we study the coalition formation among a group of collaborative mobile users. To minimize the energy consumption in collaboration, we formulate the task assignment problem as a 0-1 integer programming problem and apply heuristic method to solve it. However, in real world it is not easy to assign tasks with a global view in a centralized way, nor does it tractable to compute. Thus we devise a distributed merge-and-split algorithm which allows collaborative and individually rational users to form coalitions. Through stability analysis, we also reveal the conditions under which the scheme yields a stable partition. Finally, in the

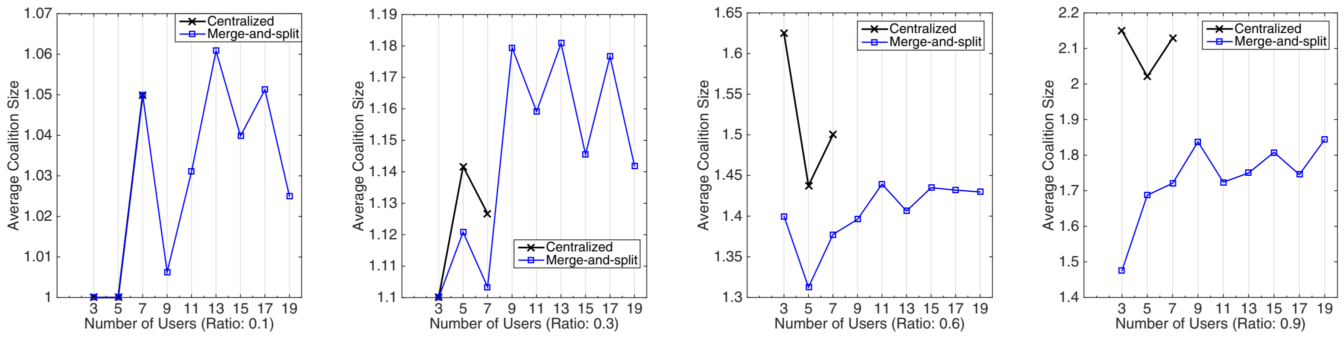
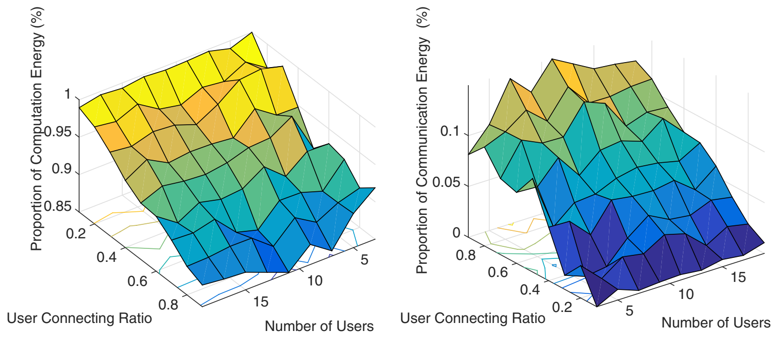
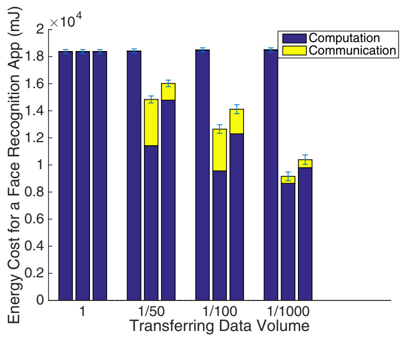


Fig. 4. Simulation results: Average coalition size.



(a) Average proportion of computation and communication cost.



(b) Energy cost for a face recognition app.

Fig. 5. Simulation results: Energy cost components for different cases.

performance evaluation, the simulation results show that our algorithm obtains near-optimal average energy cost, and is highly efficient compared to the centralized strategy.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE*, vol. 8, pp. 14–23, 2009.

[2] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. Ammar, "Towards Resource Sharing in Mobile Device Clouds: Power Balancing across Mobile Devices," in *Proc. 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing*, 2013.

[3] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling Remote Computing among Intermittently Connected Mobile Devices," in *Proc. 13th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2012.

[4] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: mClouds-Computing on Clouds of Mobile Devices," in *Proc. 3rd ACM Workshop on Mobile Cloud Computing and Services*, 2012.

[5] M. Guirguis, R. Ogden, Z. Song, S. Thapa, and Q. Gu, "Can You Help Me Run These Code Segments on Your Mobile Device?" in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, 2011.

[6] T. Langford, Q. Gu, A. Rivera-Longoria, and M. Guirguis, "Collaborative Computing On-Demand: Harnessing Mobile Devices in Executing On-the-Fly Jobs," in *Proc. 10th Int'l Conf. on Mobile Ad-Hoc and Sensor Systems (MASS)*, 2013.

[7] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan, and S. V. Krishnamurthy, "Computing While Charging: Building a Distributed Computing Infrastructure Using Smartphones," in *Proc. 8th Int'l Conf. on Emerging Networking Experiments and Technologies (CoNEXT)*, 2012.

[8] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making Smartphones Last Longer with Code Offload," in *Proc. 8th Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys)*, 2010.

[9] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution between Mobile Device and Cloud," in *Proc. 6th Conference on Computer Systems (EuroSys)*, 2011.

[10] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading," in *Proc. IEEE INFOCOM*, 2012.

[11] K. R. Apt and A. Witzel, "A Generic Approach to Coalition Formation," *International Game Theory Review*, vol. 11, pp. 347–367, 2009.

[12] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition Structure Generation with Worst Case Guarantees," *Artificial Intelligence*, vol. 111, pp. 209–238, 1999.

[13] R. B. Myerson, "Game Theory: Analysis of Conflict," *Harvard University Press*, 1991.

[14] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, and B. Li, "eTime: Energy-Efficient Transmission between Cloud and Mobile Devices," in *Proc. IEEE INFOCOM*, 2013.