Volunteered Mobile Sourcing with Multi-objective Ant Colony Optimization

Katchaguy Areekijseree
Computer Engineering Department
King Mongkut's University of Technology Thonburi
126 Pracha-Utid, Bangkok 10140, Thailand
katchaguy.are@gmail.com

Tiranee Achalakul
Computer Engineering Department
King Mongkut's University of Technology Thonburi
126 Pracha-Utid, Bangkok 10140, Thailand
tiranee@cpe.kmutt.ac.th

***Abstract*—Volunteered computing has been one of the popular distributed computing concepts recently. The basic idea is to allow computer owners to donate the computing power and storage to scientific applications. In this research, we are interested in the utilization of volunteered mobile devices. The implementation of such a concept is complicated since it is hard to accurately estimate the execution time of workflow tasks on numerous mobile devices. To efficiently schedule application workflows can thus be a real challenge. In this paper, we proposed a practical way to construct a workflow with estimated overhead and execution time, as well as a scheduling algorithm for a highly distributed computing platform. The main idea is to effectively optimize task scheduling onto the currently available mobile devices with two objectives of maximizing both cost and execution time saved. Therefore, the cost will be covered by the volunteers. We adapt the Multi-objective Ant Colony Optimization (MOACO) algorithm in our framework. We perform an experiment with different sizes of scientific workflows under different numbers of volunteered devices. The results show a good potential in using mobile sources to minimize the energy consumption at the data center while keeping the execution time within a reasonable deadline.**

Keywords—Volunteer Computing, Mobile Computing, Ant Colony Optimization, Multi-objective Optimization, Scheduling algorithm

I. INTRODUCTION

Mobile devices usually refer to handheld or pocket-size computing devices such as smart phones and tablets. At present, such devices have become a part of our daily life. People use mobile devices for several social networking, messaging, documentation, browsing, and photo editing. The statistics from the International Telecommunication Union has revealed that the number of active mobile-broadband subscriptions in the world has increased from 268 million in 2007 to 2,096 million in 2013. The computation ability of the mobile technology is now comparable to those of regular desktop computers. Thus, it is now possible to utilize mobile devices for actual computation, such as image processing, 3D rendering, 3D game, and mathematical modeling. Moreover, mobile users rarely turn off the devices leaving it in idle state for several hours each day. In order to capitalize on the CPU cycles and put them to good use, we are inspired to design a scheduling algorithm for mobile sourcing of scientific applications. The mobile users can contribute the computing power to be public-resources for Sciences.

A scientific workflow is large consisting of multiple compute nodes, where nodes represent a series of atomic tasks (small unit of computation or data manipulation). Edge between nodes of a workflow represents dependency between those atomic tasks. In general, a large amount of execution time and resources are required in order to serve the need of the applications. In order to execute the task, a cluster of computers or cloud computing are often required. However, the availability of these technologies is limited in some countries and the costs are sometimes unaffordable for many non-profit associations.

In this paper, we discuss the utilization of the computing performance contributed from mobile users under the concept of volunteer computing. Volunteer computing refers to a distributed computing platform constructing from resource donation of the users around the world. The more resources being donated, the faster the system will be. The basic idea of our mobile sourcing platform is to allow the atomic tasks of the scientific problems to be packed and distributed to mobile devices and use mobile power to solve the problem.

However, scheduling appropriate tasks to the mobile devices is a challenge due to the availability of volunteers, hardware limitation, and the wireless network connection. To tackle this challenge, this research proposes the design of a scheduling algorithm for the mobile sourcing platform. We adopt the Ant Colony Optimization (ACO) algorithm to be used with multiple objectives optimization. The first objective is to maximize the energy saving from the local execution (save cost at the datacenter). Second, the algorithm should also optimize the total computing time by utilizing mobile sources appropriately. Weighted sum method is applied to the Multi-Objective Ant Colony Optimization or MOACO.

The organization of this paper is as follows: Section 2 and 3 present literature survey and background on ACO. Section 4 describes the proposed technique. Section 5 discusses an experimental methodology and result discussion. The last section is a conclusion of this research.

978-1-4799-5822-1/14/$31.00 ©2014 IEEE

II. LITERATURE SURVEY

As the use of mobile devices grow exponentially, recent literature has been focusing on an integration of the mobile devices to the cluster, grid, and cloud computing. This section summarizes the previous works and their challenges.

Antonios et al. [1] initiated the concept of mobile grid computing platform. They discussed the efficient adoption of mobile devices in the grid computing environment. Then, Thomas [2] introduced the method to solve the scientific problems using the mobile grid computing. The system integrated the mobile wireless consumer devices into the grid environment. The system consisted of a proxy component which was called "interlocutor". The component was responsible for scattering the tasks to execute on the "minions", that were the mobile devices, and retrieving the results back to the grid.

Marinelli [3] developed Hyrax, a platform derived from Hadoop which supported cloud computing on Android smartphones. Hyrax allowed the users to utilize data and execute the tasks on heterogeneous networks of smartphones and servers. Unfortunately, Hadoop is too compute-intensive for mobile devices. Thus, the results showed that the overhead costs of running MapReduce were too high for Android phones due to the limitation of memory.

Later on, Gonzalo [4] extended the idea of Hyrax. In this work, they presented the guidelines for a framework to create a virtual mobile cloud computing providers. The goal of the framework was to utilize the mobile devices to perform some tasks. The framework was capable of detecting the mobile devices that were in a stable mode and create a virtual cloud computing on the fly among the users. The implementation of the framework was developed based on Hadoop and the communication between devices was P2P based on the Extensible Messaging and Presence Protocol (XMPP). However, the result of the experiment showed that the execution time of the platform was slower than the execution on the mobile devices alone. The problem was caused by the preparation and waiting time, which was approximately 44% of the execution time. Moreover, the result showed a poor performance due to the fact that Hadoop created a new JVM per each map processing. The framework also faced with a memory problem when it handled multiple files.

With such limitations posted in previous literature, form a grid or a cloud of mobile devices seems impractical with the current technology. Thus, in order to utilize the computing power from mobile sources, the method must be carefully designed. In our research we aim at utilizing mobile sourcing for data-intensive scientific applications, where computation can be divided into a large amount of independent tasks. These tasks can then be distributed over a large network of mobile source volunteered by its owners. This type of sourcing has been accomplished in the previous literature for certain types of scientific applications, such as SETI@home [5], Folding@home [6], and Cloud@home[7]. All three projects have a common concept. They obtain the computing power from millions of computer owners worldwide. The client software runs either as a screen saver or while the users work, meaning that the use of processor time will not be wasted. The client software gets a work unit from the volunteered source, analyzes the results, and sent them back. This computing model allows Sciences to benefit from free computing power donated by the world population.

In our proposed work, we adopted the idea of volunteered computing to Hyrax's concept of mobile cloud computing. In order to reduce the effect of preparation and wait time as well as the data transfer overhead, we emphasize the scheduling process. Mobile sources will only be utilized if the cost can be saved and the execution time is still acceptable. Numerous works on mobile scheduling and offloading were then studied.

A work by Yuan Zhang et al. [8] used a depth-first and linear time searching to maximize the energy saving in mobile devices. The experiment result showed excellent performance but the algorithm itself was not adaptive by the dynamic computing environment seen in the volunteered computing platform. Another widely used algorithm was Integer Linear Programming (ILP). MAUI [9], MACS [10], CloneCloud [11] adapted this algorithm within their platform. The algorithm acquired the data from the environment, e.g. networks bandwidth, mobile resources, local resources, as an input and formulated an optimization problem to create an optimum schedule. However, the result presented in [8] suggested that ILP's result accuracy was much less than the depth-first search even though the performance may be better.

In order to improve the quality of the schedules created, meta-heuristic algorithms were adopted in mobile cloud research. Qingfeng Liu et al. [12] adopted Genetic Algorithm (GA) for scheduling the tasks. However, this research ignored some factors such as CPU resources and network bandwidth. GA could obtain a near global optima solution but the algorithm itself was compute-intensive. Another efficient algorithm class is Swarm Intelligence (SI). It was suggested by [13], that SI required lower resources and computational time than GA. This claim was also supported by [14] where Ant Colony Optimization (ACO) was proposed to obtain the routing path for wireless sensor. The result illustrated that ACO was capable of execution with low processing power and resource.

From the characteristic of several algorithms discussed in this section, we are interested in adapting the ACO algorithm for use with scheduling problem in volunteered mobile sourcing platform. For better understanding of the readers, background on ACO is presented next.

III. BACKGROUND ON ANT COLONY OPTIMIZATION (ACO)

Ant Colony Optimization (ACO) is a meta-heuristic algorithm that belongs to the swarm intelligence class. The algorithm was inspired by the behavior of the ants searching for food around the colony. Initially, a set of ants starts from the starting point and randomly move around to find a food source. During their walk, the ants deposit the pheromone on the explored paths. The pheromone is a chemical which ants use to communicate with one another. Consequently, the majority of ants will travel the path with the highest pheromone concentration as it is assumed to be a path leading to the best food source. The pheromone values will be updated in each iteration due to both pheromone depositing and evaporating, which can occur concurrently.

In optimization, a path leading to each food source represents a feasible solution and the pheromone concentration in each trail is considered as a fitness value. There are two main parts in ACO: path selection and pheromone updates

ACO starts by randomly initializes the pheromone trails. The ants will then select paths according to the pheromone values (probability value) as in (1). The path with higher pheromone values will have more chance of being selected.

$$p_{i,j}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}}{\sum_{j \in N_i^k}[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}} \quad (1)$$

where:

$p_{i,j}^{k}(t)$ denotes the probability that an ant k goes from i to j at time t.

$\tau_{i,j}(t)$ denotes the pheromone value on the path from i to j at time t.

$\eta_{i,j}(t)$ denotes the heuristic information on the path from i to j at time t, which is used to select j when an ant is at i.

N_i^k denotes the feasible neighborhood of the ant$k$

α denotes the weight of pheromone

β denotes the weight of heuristic information

Next, the pheromone values on each path will be updated due to both the pheromone depositing and evaporating using (2) and (3). The most explored path will have the highest level of pheromone. The pheromone will be decreased on the less visited path.

$$\tau_{i,j}(t) = \rho\tau_{i,j}(t-1) + \sum_{k=1}^{n}\Delta\tau_{i,j}(t) \quad (2)$$

and

$$\Delta\tau_{i,j}(t) = \begin{cases} \frac{Q}{L_k(t)} & \text{, if the path from i to j is chosen by the ant k} \\ 0 & \text{, otherwise} \end{cases} \quad (3)$$

where:

ρ denotes the pheromone trail evaporation between 0 to 1

n denotes the number of ants

Q denotes a constant for pheromone updating

$L_k(t)$ denotes the cost of tour by the ant k

From the equations, more pheromone is evaporated on the long path as ants have to spend more time travelling. Therefore, shorter paths will more likely to be selected. The algorithm will run recursively until the stopping criteria is reached (number of maximum iteration). The shortest path, therefore, represents the optimal solution.

IV. The Proposed Scheduling Method

In this scheduling problem, we aim to assign workflow tasks to appropriate computing units of two classes: volunteered mobile sourcing (free) and computers at a datacenter (paid). The objective is to use as much free sources as possible to save cost at a data center while still maintaining an acceptable execution timeframe. The key contribution is to design a scheduling framework that will create a near-optimum schedule with two objectives: optimum cost and execution time. There are two important modules in this framework: First, the workflow construction, which is a module that converts user input files describing tasks into a workflow Direct Acyclic Graph (DAG). Secondly, the scheduling optimization, which is a module that assigns task nodes in the workflow DAG to computing units both in the data center and volunteered mobile sourcing.

A. Workflow Construction

In this module, the input workflow DAG is constructed. First, the user uploads the application workflow in the XML format. This file defines the structure of the workflow, task dependencies, input/output, and runtime of each sub-task. The XML input file consists of 2 main parts. The first part of a file gives a definition of each task, while the second part of a file shows the dependency of tasks.The example of a typical input format is illustrated in Fig. 1 (a) and (b).

```
<?xml version="1.0" encoding="UTF-8"?>
<adag xmlns="http://pegasus.isi.edu/schema/DAX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX http://pegasus.isi.edu/schema/dax-2.1.xsd"
    version="2.1" count="1" processor="2.5" memory="2GB" jobCount="25" fileCount="0" childCount="20">
  <!--Part 1-->
  <job id="ID00000" namespace="Montage" name="mProjectPP" runtime="13.67">
    <uses file="region.hdr" link="input" size="304"/>
    <uses file="2mass-atlas-ID00000s-jID00000.fits" link="input" size="4222080"/>
    <uses file="p2mass-atlas-ID00000s-jID00000.fits" link="output" size="4158210"/>
    <uses file="p2mass-atlas-ID00000s-jID00000_area.fits" link="output" size="4158210"/>
  </job>
  <job id="ID00001" namespace="Montage" name="mProjectPP" runtime="13.32">
    <uses file="region.hdr" link="input" size="304"/>
    <uses file="2mass-atlas-ID00001s-jID00001.fits" link="input" size="4222080"/>
    <uses file="p2mass-atlas-ID00001s-jID00001.fits" link="output" size="4173937"/>
    <uses file="p2mass-atlas-ID00001s-jID00001_area.fits" link="output" size="4173937"/>
  </job>
```

Fig. 1. (a) Tasks' definitions

```
<child ref="ID00005">
  <parent ref="ID00000"/>
</child>
<child ref="ID00006">
  <parent ref="ID00000"/>
  <parent ref="ID00004"/>
</child>
<child ref="ID00007">
  <parent ref="ID00001"/>
  <parent ref="ID00000"/>
</child>
```

Fig. 1. (b) Tasks' dependency

Tasks (DAG's nodes) are defined by a set of attributes including task ID, workflow name, task name, and runtime required (on a specific machine specification). In addition, the input data and output data files are also described using three attributes: file name, input or output flag, and file size in bytes.

Tasks dependencies (DAG's edges) are then defined and a DAG graph is formed. Each edge is described using "child" and "parent" elements, and the "ref" attribute that specifies the node ID. The example of XML code defines the edge between node ID00000 and ID00005 where ID00000 is a parent node is illustrated in Fig. 2.

```
<child ref = "ID00005" >
        <parent ref = "ID00000">
</child>
```

Fig. 2. XML code

Once the XML input file is uploaded by a user, the abstract workflow is constructed using the JAVA's graph data structure as shown in Fig. 3. In the abstract workflow, each vertex represents a task (T_i). The vertex also has incoming and outgoing degrees. Incoming degree refers to data input size needed for processing the task, while outgoing degree refers to data output size. The weight on the vertex is the task complexity. The weight on the edge is the amount of data transferred between tasks. The abstract workflow is then

mapped to the specific processing environment in order to create a workflow schedule.

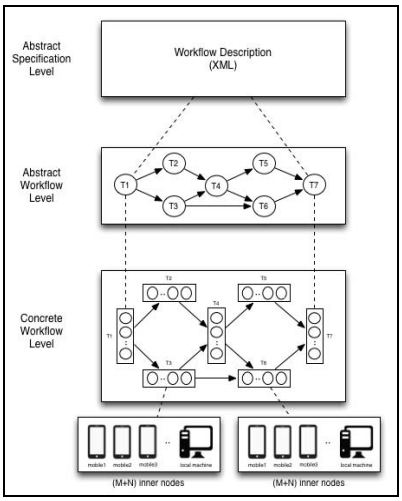

Fig. 3. Process of Workflow Construction

In this process, each task in the abstract workflow is mapped with the available machines. The weight on each task, T_i, will be converted from an abstract complexity to an actual execution time. Notice that each vertex carries a set of weights since there are various possibilities for execution time depending on the processor it is scheduled to. If the available machines consist of *M* local processors and *N* mobile processors, the weights will be a vector of *(M+N)* values.

The structure of the workflow schedule can be explained as followed, let *G = (V, E)* is the completely connected graph that represents the search space.

- *V* is a finite set of task $V= \{T_0, T_1 \dots, T_n\}$, where *n* is the number of total tasks of the input workflow. Each task $T_i(0 \le i \le n)$ has an set of available machines $S_i = \{S_i^1, S_i^2, \dots, S_i^k\}$, where $S_i^k (1 \le k \le M+N)$ represents runtime of task *i* on an available machine, *k*.
- E is a finite set of relations and dependencies of tasks. The weight on an edge represents an amount of data (in bytes) transferring between source and target nodes.

We assume that a datacenter is homogeneous. Runtimes on each machine for any task T_i will thus be the same. The runtime on mobile devices are estimated as a summation of processing time and data transfer time.Once the experiment environment is defined, the mobile device specifications and network bandwidth are known. The processing time of each vertex can be estimated based on the baseline clock speed and runtime given in the input workflow (XML file). We use the input baseline numbers and perform a table lookup in the processor benchmarking data in www.passmark.com.

The transfer time can be estimated using multiple regression technique. Here, we download a network benchmarking software on iOS, called 4Gmark, and perform data sending and receiving multiple times with different sizes of data. The collected data are then regressed in the form of $f(x_1, x_2) = \beta_o + \beta_1 x_1 + \beta_2 x_x$, where x_1 is a file size (MB) and x_2 is network bandwidth (Mb/s). We choose to perform multiple regression instead of using the simple equation ($x_1 * x_2$) in order to compensate for error in communicating which is common in a real situation. The regression model with an R^2 of 97.5% is then used to estimate the transfer time.

Finally, the cost for executing each task on a machine (local or mobile), S_i^k are used to construct a set of candidate schedules. The best or the optimum schedule will then be selected from this candidate set using a meta-heuristic optimization presented in the next section.

B. Workflow Optimization with Multi-Objective ACO

In order to solve the scheduling problem for mobile sourcing, all possible combinations of task assignments (candidate schedules) form a search space. The objective of the optimization is to look for a candidate schedule which maximize the cost saving at the data center (energy saving) and the total runtime saving of the entire workflow. Note that, if a scheduler outsources more tasks, the cost at the data center will be lower, but the runtime will be higher.

The fitness value of the candidate solution can be calculated using (4).

$$\max f(x) = w_1 * f_1(x) + w_2 * f_2(x), w_1 + w_2 = 1 \qquad (4)$$

where *f(x)* is a score, $f_1(x)$ describes the energy saving and $f_2(x)$ describes the runtime saving of a feasible solution *x*. Note that in order to normalize $f_1(x)$ and $f_2(x)$ onto the same scale, percentage of energy saved and runtime saved is used.

The percentage of energy saved at a data center is calculated using (5).

$$f_1(x) = \left(\frac{E_{local} - \sum_{i=0}^{n} (Eexe_i + Erecv_i + Esent_i)}{E_{local}} \right) * 100 \qquad (5)$$

where

- $Eexe_i$ – Energy used in executing task *i*. It is calculated by using (6), where R_i is runtime of task T_i and $P_{execute}$ refers to execution power
$$Eexe_i = R_i * P_{execute} \qquad (6)$$
- $Erecv_i$ – Energy used in receiving data for task *i*. This energy is spent when the data is being transferred to the local machine. It is calculated by (7), where $P_{transfer}$ refers to data transfer power and $inputFileSize_i$ is a total file size.
$$Erecv_i = inputFileSize_i * P_{transfer} \qquad (7)$$
- $Esent_i$ – Energy used in sending data for task i. This energy is spent when data is being transferred out to mobile devices. It is calculated by (8), where $outputFileSize_i$ refers tooutput file size.
$$Esent_i = outputFileSize_i * P_{transfer} \qquad (8)$$
- E_{local} – Energy spent when all tasks are executed on one local machine.
- *n* – total tasks

The percentage of runtime saved for the entire workflow can be calculated by (9).

$$f_2(x) = \left(\frac{T_{local} - \sum_{i=0}^{n} (Twait_i + Ttransfer_i + Texe_i)}{T_{local}} \right) * 100 \qquad (9)$$

- $Texe_i$- Runtime of task i
- $Twait_i$ - Wait time before executing task i. Note that a task may have to wait for a machine to be available or the arrival of input data
- $Ttransfer_i$– Data transferring time
- $Tlocal_i$ – Overall runtime when all tasks are executed on one local machine
- n - total tasks

$w_1$and w_2 are the weights of each function. The value is between 0.0 and 1.0.

$$\begin{cases} w_1 > w_2 & ,f_1(x) \text{ is more important than } f_2(x) \\ w_1 < w_2 & ,f_2(x) \text{ is less important than } f_2(x) \\ w_1 = w_2 & ,f_1(x) \text{ and } f_2(x) \text{ are equally important} \end{cases}$$

A candidate solution is represented by a one-dimensional array, which represents path or trail that an ant explores. Fig. 4 illustrates the solution.

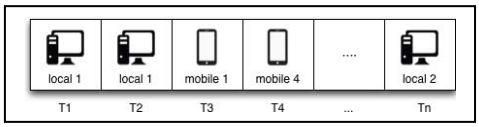

Fig. 4. Solution representation

The size of the array equals to the size of the total tasks of the workflow (n). Each value in the array represents where task i should be executed. For example, task 1 and task 2 are executed on local machine #1, task 3 is executed on mobile device #1 and so on.

To construct a solution, the ant used the information of pheromone in order to select the next node. The ant explores every task assignment until it reaches the end node (E). The process is illustrated in Fig. 5.

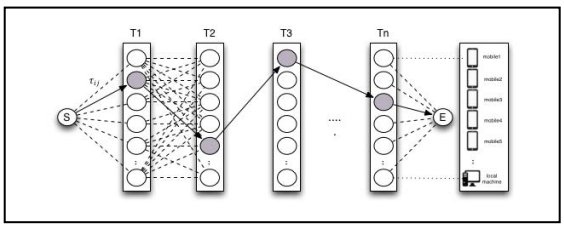

Fig. 5. Process of creating a new solution

The Ant Colony Optimization algorithm presented in section 3 is adopted in order to search for the optimal solution. In each iteration, a group of P ant agents is attempting at finding the optimal solution. The solution represents where the task T_i should be executed. Each node in the graph represents task T_i. Each T_i consists of a set of available machine S_i. Each node in the set S_i connects to every node in the set S_{i+1}. At the beginning, every ant is assigned at the starting point S. Then, the ant selects the available machine for each task based on the probability value which has been calculated from the pheromone values. In order to select the appropriate machine for task T_i, we apply roulette wheel selection method adapted from parent selection in genetic algorithms (GA). The path with higher probabilistic value will have higher chance of being selected.

When every ant reaches the end node (E), the trail each ant explored represents one solution. Then, all the candidate solutions are evaluated by the fitness function as the function of weighted scores between energy saved and runtime saved. Then, the fitness values of P ants are compared to find the best candidate of the current iteration. The solution is then kept for comparison in the next iteration. P ants repeatedly search for the optimal solutions until the number of maximum iteration is reached.

V. Experiments and Results

This work aims at creating a near-optimum schedule for scientific applications as well as identifying the optimal number of volunteered devices deployed. In this section, we present an experiment with scientific workflows downloaded from www.pegasus.isi.edu. The workflows were embedded with the workflow structure, tasks dependency, number and name of input/output of each task and the execution time of each task. We chose the Montage workflow, which was designed to deliver science-grade mosaics of the sky. Details of workflows are summarized in Table x.

TABLE I. Input Workflow

# of tasks (n)	25
Energy used	459.34 kJ
Execution time (1 machine)	30.62 min

The following assumptions were made in our experiment.

- When the mobile device is not in used, it is always being charged. Thus, the battery will never run out.
- There is no application running on the mobile. The CPU is always in the idle state before the execution of any task.
- The memory on the mobile is sufficient.
- The network connectivity is in a stable condition.
- Every mobile device connects to the home Wi-Fi

We implemented the MOACO algorithm with Java and jGraph (Directed Acyclic Graph manipulation plugin). The parameters of MOACO were set as follows: 10 ants, alpha (α) =3, beta (β) = 2, rho (ρ) = 0.01, Q = 2.0. For the fitness function, we set w_1 and w_2 to 0.3 and 0.7 respectively and the number maximum iteration is 10,000.

The computing environment consists of 1 local machine with similar specification as given in Montage data profiles (Intel Xeon 2.4 GHz processor with 2GB of memory). The number of mobile devices used is varied from 1 to 25. For the device specifications, we considered the top 4 smart phones available in the market: iPhone5s (type I), Samsung Galaxy S4 (type II), Samsung Galaxy Note3 (type III) and HTC one (type IV). The number of mobile devices of each type was divided equally in the experiment.

The example results from the execution time estimation is presented in Table II. Montage workflow has 9 different classes of sub-tasks. Table II shows the estimation the workflow. The baseline values are given along with the workflow profile. Type I, II, III and IV columns refer to the

execution time of sub-tasks when run on four different mobile devices.

TABLE II. ESTIMATION OF RUNTIME ON MOBILE DEVICES

Task	Baseline	Type I		Type II	
	Time(s)	Time(s)	Relative	Time(s)	Relative
mProject	13.32	35.83	x1.7	41.15	x2.1
mDiffFit	10.60	28.51	x1.7	32.75	x2.1
mAdd	1.38	3.71	x1.7	4.26	x2.0
Task	**Baseline**	**Type III**		**Type IV**	
	Time(s)	Time(s)	Relative	Time(s)	Relative
mProject	13.32	36.36	x1.7	36.36	x1.7
mDiffFit	10.60	28.94	x1.7	28.94	x1.7
mAdd	1.38	3.75	x1.7	3.75	x1.7

For the deadline, we set a constraint that the entire workflow must be finished within 1.5 time of the execution time on the local machine. The overall execution time of each mobile device must not be over the volunteered time. We assume that every user donates their resources for 3 hours.

TABLE III. PERCENTAGE OF ENERGY AND TIME SAVED

# of mobile devices	% Energy Saved	%Time Saved	# of mobile devices	% Energy Saved	%Time Saved
1	24.27	-2.93	8	76.42	-0.56
2	50.44	-7.72	9	79.06	0.87
3	53.44	-7.87	10	79.65	4.11
4	68.26	-9.91	11	79.97	5.88
5	74.25	-10.50	15	82.18	9.95
6	74.52	-6.96	20	84.15	17.01
7	72.42	-0.64	25	87.56	17.24

The result of a 25-node Montage workflow is shown in Table III. This table shows the results of the percentage of energy and time saved when various numbers of volunteered mobile devices were used. The trend of the energy saved was significantly increased when a number of mobile devices increased. In other words, as more tasks were outsourced, the cost at the data center was reduced. Using MOACO, the schedule selected by the ants produced up to 87.56% energy saving.

However, there was a trade-off between time and energy saved. When mobile devices were used, communication overhead affected the overall performance. The observed performance degradation was as much as 10% from the experiment. As the number of devices increases, the concurrent computation can overcome the communication overhead. Computation and communication overlapping also occurred when enough number of computing units was deployed (10 units or more in this case).

VI. CONCLUSION

In this paper, we introduce a multi-objective ant colony optimization to solve the mobile sourcing scheduling problem. The objectives are to reduce the energy consumption of the data center and minimize the execution time of the overall process. Execution time estimation for mobile devices was performed. Based on the estimation, MOACO was able to produce a good schedule, which satisfies the deadline constraints. The created schedule also allows the workflow applications to be distributed efficiently to appropriate volunteered mobile sources. The algorithm can select suitable mobile units for each workflow. The interesting future direction is to improve the algorithm in order to select the appropriate numbers of volunteered device for dynamic workflows. Several constraints in the experiment should also be relaxed in order to make the model more realistic.

ACKNOWLEDGMENT

This work has been supported by NSTDA under project University Industry Research Collaboration (NUI-RC).

REFERENCES

[1] A. Litke, D. Skoutas, and T. Varvarigou, "Mobile grid computing: Changes and challenges of resource management in a mobile grid environment," in 5th International Conference on Practical Aspects of Knowledge Management, 2004

[2] T. Phan, L. Huang and C. Dulan. "Challenge: Integration mobile wireless devices into the computational grid," in Proceedings of the 8th annual international conference on Mobile computing and networking, 2002, pp. 271-278.

[3] E. Marinelli. "Hyrax: cloud computing on mobile devices using MapReduce." Carnegie-mellon univ Pittsburgh PA school of computer science, 2009.

[4] G. Huerta-Canepa and D. Lee. "A virtual cloud computing provider for mobile devices," in Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, 2010.

[5] Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D., "SETI@ home: an experiment in public-resource computing," Communications of the ACM, 2002, pp. 56-61.

[6] Beberg, A. L., Ensign, D. L., Jayachandran, G., Khaliq, S. and Pande, V. S., "Folding@ home: Lessons from eight years of volunteer distributed computing," in Parallel & Distributed Processing, IPDPS 2009. IEEE International Symposium, 2009, pp. 1-8

[7] Cunsolo, V. D., Distefano, S., Puliafito, A. and Scarpa, M., Cloud@ home: Bridging the gap between volunteer and cloud computing. In Emerging Intelligent Computing Technology and Applications, Springer Berlin Heidelberg, 2009, pp. 423-432.

[8] Zhang, Y., Liu, H., Jiao, L. and Fu, X. "To offload or not to offload: an efficient code partition algorithm for mobile cloud computing in Cloud Networking ," in IEEE 1st International Conference, 2012.

[9] Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R. and Bahl P., "MAUI: making smartphones last longer with code offload," in Proceedings of the 8th international conference on Mobile systems, applications, and services, 2010, pp. 49-62.

[10] Kovachev, D.; Tian Yu; Klamma, R., "Adaptive Computation Offloading from Mobile Devices into the Cloud," Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium, 2012, pp.784-791.

[11] Chun, B. G., Ihm, S., Maniatis, P., Naik, M. and Patti, A., "Clonecloud: elastic execution between mobile device and cloud," in Proceedings of the sixth conference on Computer systems, 2011, pp. 301-314.

[12] Liu, Q., Jian, X., Hu, J., Zhao, H. and Zhang, S., "An optimized solution for mobile environment using mobile cloud computing," in Wireless Communications, Networking and Mobile Computing, 5th International Conference, 2009, pp. 1-5.

[13] Wang, J., Osagie, E., Thulasiraman, P. and Thulasiram, R. K., "HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network", Ad Hoc Networks, Vol. 7, No. 4, 2009, pp. 690-705.

[14] Saleem, M., Di Caro, G. A., and Farooq, M., "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions", Information Sciences, Vol.181, No.20, 2011, pp. 4597-46.