Developing Energy-Aware Task Allocation Schemes in Cloud-Assisted Mobile Workflows

Bo Gao[†], Ligang He[†*], Xin Lu[‡], Cheng Chang[*], Kenli Li[*] and Keqin Li[*]
† Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
‡ School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China
∗ School of Computer Science and Electronic Engineering, Hunan University, Changsha, China
{bogao, liganghe}@dcs.warwick.ac.uk

***Abstract*—Mobile cloud computing is an emerging field of research which aims to provide a platform on which intelligent and feature-rich applications are delivered to the user at any time and at anywhere. When such a cloud-assisted mobile application workflow requires the cooperation of many devices, solving the task allocation problem becomes a critical step in ensuring the energy efficiency of the mobile cloud platform. In this paper, we construct a quadratic binary program to model the task allocation problem in such scenarios. In order to overcome the poor scalability of generic quadratic program solvers, we present an implementation of the simulated annealing algorithm and a greedy autonomous offload algorithm to approximate the optimal solution. Both heuristics are tailored to solve our task allocation problem efficiently. We verify and compare our algorithms against a commercial quadratic program solver in a series of simulations. Results show that both heuristics produce good solutions to the task allocation problem. Solutions provided by our greedy algorithms is consistently close to optimal and can be obtained in a more time efficient manor than our implementation of the simulated annealing algorithm.**

I. Introduction

Despite the rapid development of mobile computing technologies, battery power remains a limiting factor in the development of mobile applications. With increasingly more complex functionality required from the user, developments of mobile applications remain largely energy-constrained [1]. One solution given by researches under the broad topic of *Mobile Cloud Computing* [2], [3], also referred to as *Cloud-Based* and *Cloud-Assisted* mobile computing [4], is to utilise cloud resources as task offload or migration targets in order to reduce the energy burden imposed on the mobile devices involved in the application workflow.

A cloud-assisted mobile workflow as we discuss in this paper is an application workflow that is implemented over a group of mobile devices with access to cloud resources. The cloud resources may either be a compulsory component in the execution of the workflow, or be in an assistant role to handle computation offload requests sent from the mobile devices. Mobile application workflows can be found when a group of mobile users are to share or communicate with each other in order to accomplish a certain task. We give two example use cases of such workflows in Fig. 1 to further clarify our objective.

Ligang He is the corresponding author.

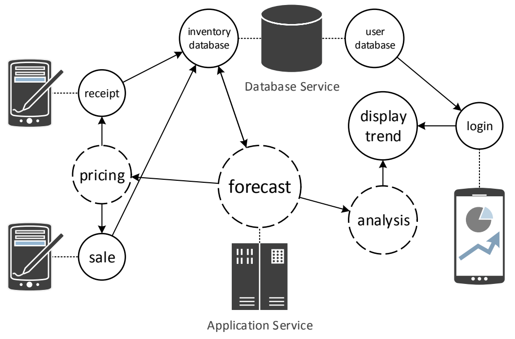
Enterprise use case: With increasing adaptation of mobile devices into enterprise business models [5], modern *enterprise applications* often include or are entirely based on mobile devices. Figure 1 (a) illustrates an application workflow involving three mobile devices and two cloud services of a supply-chain business. Two employees are concerned with the receipt and sale of goods respectively. Both activities require an up to date pricing information at runtime. A manager is concerned with the trends that are developing in the company's inventory in real-time which is produced via a series of tasks like forecast and analysis.

There are two types of tasks in this workflow: tasks like login, record sale and database queries that are fixed either on a mobile device or a cloud node; and tasks like data analysis and forecast that can be offloaded / migrated between devices and clouds. Depending on the computation size and communication size of these offload-able tasks, an energy-aware allocation scheme can help optimise the execution of the workflow in term of its overall energy cost imposed onto the mobile cloud platform.

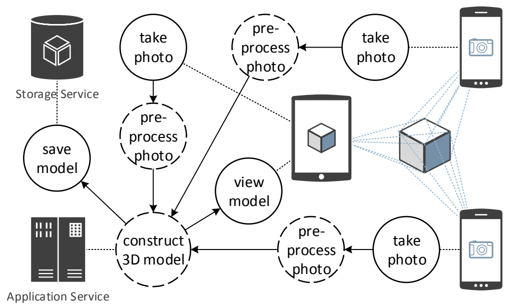
Consumer use case: Because of its portability, mobile devices encourages the development of collaborative application. In Figure 1 (a), we illustrate the collaboration of three mobile devices in scanning the 3D structure of an object. Pictures of the object is taken on all three devices at the same time and once pre-processed, these pictures are gathered to construct the 3D model of the object. Two cloud services are available to the users, one for storage and one for application hosting and computation offloading.

Similar to the enterprise use case, the tasks involved in this mobile application workflow can either be fixed or offload-able over the mobile cloud platform. Allocation of tasks is critical in deciding the energy-footprint of the workflow. For instance, once the picture has been pre-processed, the subsequent communication size may be reduced. Comparing this reduction to the computation cost of the pre-processing task on the mobile device, it may or may not be beneficial for the device to offload this task to the cloud.

Our research investigates ways to model and optimise the energy efficiency of such workflows running atop a platform of mobile and cloud devices. Our goal is to develop ways to produce energy-aware task allocation schemes and provide an energy efficient execution platform for cloud-assisted mobile



(a) Business task interaction



(b) 3D scan workflow

Figure 1: Example use cases of cloud-assisted mobile application workflows. Solid circles represent tasks that are fixed on a device or a cloud service, whereas dashed circles represent tasks that are free to offload / migrate between devices and clouds.

workflows. We discuss related work in the field of mobile cloud computing in the next section. We then model the energy-aware task allocation problem as a quadratic binary program in section III. Two heuristic algorithms are developed in section IV that are tailored to overcome the scalability issue of a standard quadratic program (QP) solver. A series of simulation results are then presented and analysed in section V to verify and analyse the solutions produced by our algorithms before conclusions are drawn in the last section.

II. Related Work

The idea of transferring computation to nearby (in terms of network connectivity) processing unit to facilitate the reduction in energy cost of mobile devices has been researched along with the maturity of mobile technologies. Pioneered by the likes of MAUI [6], CloneCloud [7] and ThinkAir [8], adaptive computation offload as a core technology in mobile cloud computing has gathered momentum in recent years and has grown from a futuristic concept to a practical means to improve and augment the user experiences of mobile applications. We recommend two excellent surveys, [3] and [9], to the interested readers for a comprehensive list of existing implementation of computation offload / migration between mobile and cloud.

Our research targets scenarios where several devices and clouds support the execution of workflows. This is an extension to existing researches in mobile cloud computing. Existing techniques like MAUI [6], CloneCloud [7] and ThinkAir [8] focuses on the cooperation between one mobile device and one cloud server. Thus a common approach towards the task allocation problem is to model the problem as a linear program [6] [10]. A linear program is suitable for modelling situations where communication time is not considered or when there are only two devices involved in the process. However, in the cases of mobile workflows, communication tasks are an essential part of the workload and occurs significant amount of energy cost [11]. Thus we construct our model's objective function as a quadratic program in order to accurately capture the communication costs between devices.

In [12], an energy-aware task allocation scheme is developed for mobile-only platforms. We extend the platform to include cloud resources which are more suitable and realistic as computation offload targets. We also develop heuristics to overcome the scalability issues of solving the quadratic program. Execution of our allocation algorithms is to be carried out by the workflow engine which oversees the execution of mobile workflows. In [13] a detailed mobile workflow engine is implemented and tested on Nokia devices. A decentralised workflow coordination architecture designed for mobile devices is presented in [14] for use in biological studies and the supply-chain industry. Authors of [15] propose a rapid application development framework based on a dynamic workflow engine for creating mobile web services.

III. Energy-Aware Mobile Cloud Platform Model

A. Hardware and Network Metrics

We consider a mobile cloud platform *MCP* consisting of a set of p processing nodes, denoted $P = \{P_1, \cdots, P_p\}$. Each processing node may either be a *mobile* node from set $P^M \subseteq P$ or a *cloud* node from set $P^C \subseteq P$, with $P^M \cap P^C = \emptyset$ and $P^M \cup P^C = P$. We characterise each processing node $P_i \in P$ with the following metrics:

s_i	CPU speed of P_i, measured in the number of clock cycles available in a second;
e_i^{cmp}	Current draw from the battery of $P_i \in P^M$ when its CPU is running, measured in mAh;
$e_i^{snd/rcv}$	Current draw from the battery of $P_i \in P^M$ when the device is sending/receiving data to/from the wireless network, measured in mAh;
e_i^{mnt}	Current draw from the battery of $P_i \in P^M$ when the device is maintaining the wireless connection alive to anticipate transmission of data, measured in mAh.

All nodes ($P_i \in P$) are interconnected via a network, and we use b_{ij} and l_{ij} to denote the bandwidth and latency between devices P_i and P_j. Thus, we have p-matrices $B = (b_{ij})_{p\times p}$

and $L = (l_{ij})_{p\times p}$ to hold all of the bandwidth and latency information of the underlying network of the MCP. When two adjacent tasks are assigned to the same device, we assume that they share the same memory address space on the device. Therefore, we assign positive infinite values to the principal diagonal elements of B, that is $b_{ii} = +\infty, i \in P$, and zeros to the principal diagonal elements of L, that is $l_{ii} = 0, i \in P$.

Our choice of hardware energy metrics is in line with researches that investigate the energy consumption of mobile devices [16]–[19] especially for the wireless module. In our model, we emphasis the difference in energy consumption between the sender and the receiver of the data, and that it cost energy to maintain a live connection in anticipation of data transmission. We refer the interested reader to [20] for a comprehensive survey in energy measurement models of mobile smart devices.

B. Application Workflow Metrics

Application workflows hosted on an MCP is represented by a directed graph $W = (T, R)$ whose vertex set $T = \{t_1, \dots t_n\}$ denotes the set of *tasks* of the workflows. An n-matrix $D = (d_{ab})_{n\times n}$ denotes the weighted adjacency matrix of W, where d_{ab} is the size of the data package that is to be sent from t_a to t_b for $(t_a, t_b) \in R$. All principle diagonal elements of D are zeros.

Each task has a profile $t_a\left(d_{(.a)}, d_{(a.)}, c_a\right)$, $a \in \{1, \dots n\}$ where $d_{(.a)}$ and $d_{(a.)}$ are the a-th column and the a-th row of D which represent the incoming and outgoing data of t_a respectively. c_a denotes the size of the workload of t_a, measured in the number of clock cycles required by t_a.

C. Fixed and Constrained Tasks

Not all tasks of a mobile workflow are suitable for offload from its host. For instance, a task that authenticates the user's identity using the fingerprint reader on the smartphone has to be executed on the smartphone; a task that manages database files saved on a cloud-storage service has to run locally on that cloud; a task which senses the user's heartbeat must run on the smart-watch, etc. These tasks are $fixed$ on their hosts.

Furthermore, there may be tasks that are only allowed to be executed on a subset of P. For instance, when a task is associated with sensitive data shared between a group of users, or when the OS of each device varies in versions such that the execution of certain tasks are not supported by all of P. These tasks are $constrained$ within a group of devices.

We denote the set of devices that a task t_a may execute on with P^{t_a}. For a fixed task, P^{t_a} has a cardinality of one and contains only its host. For a task which is not at all constrained, $P^{t_a} = P$.

D. Allocation Scheme and Energy Costs

Given an allocation scheme $\psi : T \to P$, we first derive the energy cost of computing $t_a \in T$ to be

$$\mathcal{E}^{cmp}_{a\psi(a)} = e^{cmp}_{\psi(a)} \times \frac{c_a}{s_{\psi(a)}} \tag{1}$$

where $\psi(a)$ is the device to which t_a is assigned. Secondly, we have the energy cost of transferring d_{ab}, $(t_a, t_b) \in R$ as given by (2).

To represent an allocation scheme ψ, we first construct an $n \times p$ binary matrix $X^\psi = \left(x^\psi_{ai}\right)_{n\times p}$, such that

$$x^\psi_{ai} = \begin{cases} 1 & \text{if } \psi(a) = i, \quad t_a \in T,\ P_i \in P \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

We refer to matrix X^ψ as an *assignment matrix* and a valid assignment must satisfy the following constraints

$$\sum_{P_i \in P^{t_a}} x^\psi_{ai} = 1, \quad a = 1, 2, \dots, n, \tag{4}$$

$$x^\psi_{ai} \in \{0, 1\}, \quad a = 1, 2, \dots, n, \ \ i = 1, 2, \dots, p. \tag{5}$$

(4) ensures that every task must be assigned to one and only one device within the group of devices which it is able to execute. (5) states that all tasks are indivisible.

E. Quadratic Program Formulation

With (1) (2) and (3), we derive the energy cost of the MCP under allocation scheme ψ as (6). The quadratic term in (6) gives the energy cost for the communications between tasks, whereas the linear term gives the energy cost for the execution of tasks. Next we introduce $(pn)^2$ coefficients q_{aibj} as given by (7). With (7) we transform (6) to

$$\text{minimise:} \quad \mathcal{E}^\psi = \sum_{b=1}^{n}\sum_{j=1}^{p}\sum_{a=1}^{n}\sum_{i=1}^{p} q_{aibj} x^\psi_{ai} x^\psi_{bj} \tag{8}$$

as the objective function of our quadratic program. Let coefficients q_{aibj} be the entries of a $pn \times pn$ matrix Q, such that q_{aibj} is at row $(i-1)\,n + a$ and column $(j-1)\,n + b$, and $vec(X^\psi) = \left(x^\psi_{11}, x^\psi_{12}, \dots, x^\psi_{1n}, x^\psi_{21}, \dots, x^\psi_{pn}\right)^T$ be the vector formed from the columns of X^ψ. It is easy to see that equivalent formulations are given by (8) and

$$\text{minimise:} \quad \mathcal{E}^\psi = vec(X^\psi)^T \cdot Q \cdot vec(X^\psi) \tag{9}$$

Therefore (9) constrained by (4) and (5) constitutes a quadratic program whose objective is to derive a task allocation scheme ψ to minimise the energy cost of the MCP. The assignment matrix which specifies ψ is given by

$$\arg\min_{X^\psi}(\mathcal{E}^\psi) \tag{10}$$

IV. Heuristics

Amongst all combinatorial optimisation problems, a binary quadratic program is one of the hardest to solve [21]. Despite the development of modern quadratic program (QP) solvers, finding the global optimal solution of a QP remains a computational difficult task. When the problem size gets bigger ($p \times n > 200$ in our experience on a laptop with a first generation Core i7 processor and 8GB of memory), finding the exact optimal solution is time consuming. In MCP scenarios, hardware metrics like a mobile device's bandwidth changes

$$\mathcal{E}^{tran}_{ab\psi(a)\psi(b)} = \underbrace{e^{snd}_{\psi(a)} \times \frac{d_{ab}}{b_{\psi(a)\psi(b)}} + e^{mnt}_{\psi(a)} l_{ab}}_{\text{sender's cost}} + \underbrace{e^{rcv}_{\psi(b)} \times \frac{d_{ab}}{b_{\psi(a)\psi(b)}} + e^{mnt}_{\psi(b)} l_{ab}}_{\text{receiver's cost}} \tag{2}$$

$$\mathcal{E}^{\psi} = \sum_{b=1}^{n}\sum_{j=1}^{p}\sum_{a=1}^{n}\sum_{i=1}^{p} \Big((e^{snd}_i + e^{rcv}_j)\frac{d_{ab}}{b_{ij}} + (e^{mnt}_i + e^{mnt}_j) l_{ab} \Big) x^{\psi}_{ai} x^{\psi}_{bj} + \sum_{a=1}^{n}\sum_{i=1}^{p} e^{cmp}_i \frac{c_a}{s_i} x^{\psi}_{ai} \tag{6}$$

$$q_{aibj} := \begin{cases} e^{cmp}_i \dfrac{c_a}{s_i} + \underbrace{(e^{snd}_i + e^{rcv}_j)\dfrac{d_{ab}}{b_{ij}} + (e^{mnt}_i + e^{mnt}_j) l_{ab}}_{=0} & \text{if } (a,i) = (b,j), \\ e^{snd}_i \dfrac{d_{ab}}{b_{ij}} + e^{mnt}_i l_{ab} & \text{a } < \text{ b}, \\ e^{rcv}_j \dfrac{d_{ba}}{b_{ij}} + e^{mnt}_j l_{ab} & \text{a } > \text{ b}. \end{cases} \tag{7}$$

by the second. It is impractical to rely on QP solvers for the development of task allocation schemes. Therefore in this section, we present two heuristics to approximate the solutions.

Note that in both heuristics, we assume that an allocation scheme ψ^0 is currently in place to schedule all tasks on the MCP. Also, some of the notations we use to present the algorithms in this section like t, T, c and p are not to be confused with the notations we used to construct the model in section III.

A. Simulated Annealing

Simulated annealing (SA) [22], [23] is a meta-heuristic algorithm commonly applied to NP-hard combinatorial optimisation problems. One of the main features of simulated annealing is that by occasionally allowing inferior solutions on its search path, the algorithm is able to perform uphill search steps so that its solution needs not get stuck at local optimal points.

The algorithm has a simple structure. As illustrated in Algorithm 1, the main procedure has two nested loops. The outer loop (line 4 - 17) iterates for a number of *cycles*. Each cycle corresponds to a temperature T which is trialed in the inner loop (line 5 - 15) and *cooled* by a cooling ratio ($0 < T_c < 1$) at the end of each cycle. At the start of the inner loop, a candidate (ψ') is randomly choosen from the local neighbourhood of the current solution (ψ^*). This candidate is then accepted as the best solution either through the fact that it improves the value of our objective function or with probability $\min\{1, exp(-|\mathcal{E}^{\psi'} - \mathcal{E}^{\psi^*}|/T)\}$ regardless of whether it improves the objective function or not.

Next, we discuss details of our implementations of the simulated annealing algorithm.

1) Cooling Schedule: The use of the exponential function (line 10) in the simulated annealing algorithm means that the probability (p) of a candidate / incumbent solution (ψ') being accepted is directly related to the temperature (T) of each cycle. The higher the temperature, the higher the chance of an inferior candidate solution may be accepted. Likewise, following the cooling (reduction) of T at the end of each cycle, the probability of an inferior solution being accepted by the algorithm is gradually reduced towards the end of the algorithm.

Because of this property, the performance of an implementation of the simulated annealing algorithm greatly depends on its choice of a *cooling schedule*. On one hand, the *initial temperature* must be high enough such that the final solution is independent from the initial solution (ψ_0). A low initial temperature constraints the development of the solution by assigning low or zero probability to inferior solutions from the start of the algorithm. On the other hand, the *exit temperature* needs to be small enough so that the development of the final solution is adequately constrained by the algorithm. A solution produced by a high exit temperature is less refined and may be randomly further away from the optimal solution.

In our implementation of the simulated annealing algorithm we set the initial and exit temperature as $T_0 = (\log(p_0))^{-1}$ and $T_e = (\log(p_e))^{-1}$ where p_0 and p_e are the initial and exit acceptance probabilities that we set out. Then we have $T_c = (T_e - T_0)^{\frac{1}{NumberOfCylces-1}}$ to complete the cooling schedule. We also normalise $|\mathcal{E}^{\psi\prime} - \mathcal{E}^{\psi *}|$ by its averages in each cycle at line 10. To preserve the essential structures of a simulated annealing algorithm, fine tunings of the algorithm are not presented in detail in Algorithm 1.

2) Calculating $|\mathcal{E}^{\psi\prime} - \mathcal{E}^{\psi *}|$: In each trial of the algorithm, the energy cost of the candidate solution ($\mathcal{E}^{\psi\prime}$) is to be calculated (line 10). Therefore its calculation is crucial to the time complexity of the algorithm as a whole. Because Q is of size $(pn)^2$, the calculation of $\mathcal{E}^{\psi\prime}$ using (9) becomes a time consuming task with increases in either p or n or both. In our experience, as the complexity of the problem increases, it becomes less feasible to apply the heuristic than that of an MIQP solver if (9) is used to calculate $\mathcal{E}^{\psi\prime}$.

To overcome this issue, we observe the following:

$$\mathcal{E}^{\psi'} - \mathcal{E}^{\psi*} = Q_{(j-1)n.}vec(X^{\psi'}) + Q_{.(j-1)n}vec(X^{\psi'})^T - Q_{(i-1)n.}vec(X^{\psi^*}) - Q_{.(i-1)n}vec(X^{\psi^*})^T \quad (11)$$

Theorem 1. *If ψ' is the allocation scheme which alters only the assignment of $a \in T$ from i to j (with $i, j \in P$) when compared with ψ^*, then $\mathcal{E}^{\psi'} - \mathcal{E}^{\psi*}$ is given by* (11) *where $Q_{(j-1)n.}$ and $Q_{(i-1)n.}$ denote the $(j-1)n$-th and the $(i-1)n$-th row of Q, $Q_{.(j-1)n}$ and $Q_{.(i-1)n}$ denote the $(j-1)n$-th and the $(i-1)n$-th column of Q respectively.*

Proof: Observe that an allocation matrix X has only binary values and that when one application changes allocation from ψ^* to ψ', only a pair of values of that allocation matrix is exchanged. Apply these observations to (9), the reader should not find it difficult to come to (11). ∎

With (11), we are able to reduce the time complexity of calculating $|\mathcal{E}^{\psi'} - \mathcal{E}^{\psi*}|$ from a complexity that is greater than $O((pn)^2)$ for multiple vector-matrix multiplications to $O(pn)$ for the sum of vector dot products.

Algorithm 1 Simulated Annealing

```
1: procedure SAnnealing($\psi^0, Q$)
2:     $T \leftarrow T_0$
3:     $\psi^* \leftarrow \psi^0$
4:     for $c \leftarrow 1, NumberOfCycles$ do
5:         for $t \leftarrow 1, NumberOfTrials$ do
6:             $\psi' \leftarrow local(\psi^*)$
7:             if $\mathcal{E}^{\psi'} < \mathcal{E}^{\psi^*}$ then
8:                 $\psi^* \leftarrow \psi'$
9:             else
10:                $p = exp(-|\mathcal{E}^{\psi'} - \mathcal{E}^{\psi^*}|/T)$
11:                if $p > rand(0, 1)$ then
12:                    $\psi^* \leftarrow \psi'$
13:                end if
14:            end if
15:        end for
16:        $T = T \times T_c$
17:    end for
18:    return $\psi^*$
19: end procedure
```

B. Greedy Autonomous Offload

Heuristics that are used in the literature to approximate QPs like simulated annealing (SA) often share a common evolution-like structure which iteratively improves on a best-known result. The optimality of the final solution is often dependent on the number of iterations the algorithm is allowed to run. In an MCP environment, workflows need to be nimble and adaptable to the constantly changing network conditions of mobile devices. It is often not practical to let the algorithm running for a large number of iterations. Therefore in the design of our second heuristic, we take a step back from the established algorithms and aim to build an algorithm that is most practical to the MCP.

Algorithm 2 Greedy Autonomous Offload

```
1: procedure GAO-Main($\psi^0$, $Q$)
   This procedure is executed by the workflow engine, triggered by the changes in network conditions or periodically.
2:     $\psi' \leftarrow \psi^0$
3:     repeat
4:         $\psi^* \leftarrow \psi'$
5:         for $d \leftarrow 1, |P^M|$ do
6:             $\psi'_d \leftarrow$GAO-Device($d$, $\psi'$)
7:         end for
8:         $\psi' \leftarrow$ Reduce ($\psi'_d$)
9:     until $MaxIterations$ or $\psi' == \psi^*$
10:    return $\psi'$
11: end procedure

12: procedure GAO-Device($d$, $\psi'$)
    This procedure may either execute on the mobile devices or on the workflow engine.
13:    $c \leftarrow$BestConnectedCloud
14:    $\Delta\mathcal{E}^* = 0$
15:    $\psi'_{Me.ID} \leftarrow \psi'$
16:    for all $a \in Me.Offloadables$ do
17:        $\psi'' \leftarrow \psi'_{Me.ID}$
18:        $\Delta\mathcal{E} = \mathcal{E}^{\psi''(a)=c}_{Me.ID}$ - $\mathcal{E}^{\psi''}_{Me.ID}$
19:        if $\Delta\mathcal{E} > \Delta\mathcal{E}^*$ then
20:            $\psi'_{Me.ID}(a) \leftarrow c$
21:        end if
22:    end for
23:    $\psi'(a) \leftarrow c$
24:    return $\psi'_{Me.ID}$
25: end procedure
```

In the design of this heuristic, as shown in Algorithm 2, we emphasis on the core feature of an MCP which is computation offload (or migration) from mobile to cloud. On a workflow level, the adjustment to the initial allocation scheme is carried out in rounds (line 3 to 9) triggered either by changes in MCP or periodically. On a device level, we first associate each mobile device with the cloud which it has the best connection with (line 13). Then all tasks currently located on this device and are not fixed to this device is measured against each other in terms of the energy savings that may occur if it is offloaded to this designated cloud (line 16 to 22). This constitutes the device-level decision making process of our algorithm.

The first procedure of the algorithm is the main function of GAO. All devices from $|P^M|$ are allowed to offload computation in each iteration until no device is able to reduce its energy cost any further.

We refer this heuristic as the Greedy Autonomous Offload

(GAO) algorithm. This algorithm is *greedy* in that each device offload the one most "profitable" task to the most "promising" location known to it. This means that the algorithm is quick and cheap (in terms of energy cost) to execute on the device. Although it does not apply exhaustive search methods for the optimal offload scheme, it produces good result which we demonstrate in the next chapter. As a possible extension to this algorithm we could model the per-device offload decision as an integer program as in [6] to obtain the optimal solution.

This algorithm is *autonomous* since it allows each device to make its own offload decisions independently. This is due to the fact that the device-level procedure (line 6) of the algorithm may execute locally on mobile devices. Note that although ψ' is requested by the procedure as input, this does not create any extra communication workload for the devices. The communication of ψ' between the devices and the workflow engine is requested to guide the execution of the workflow regardless of any offload requests. Once an offload decision has been made on the device, only the difference between ψ' and $\psi'_{Me.ID}$ is to be returned at line 24. This autonomous behaviour also mimics the cooperation of mobile devices when each is equipped with one-to-one mobile cloud computing offload schemes as suggested by [6], [7], [24].

Another benefit of the greedy autonomous structure is that the task allocation decision engine is able to react to the changing network conditions more efficiently. For instance, when a device is temporarily cut-off from the MCP network, the workflow engine may pause the procedure and wait for the device to come back online. Depending on the new connection speed that device has to the MCP when it recovers, the workflow engine can decide whether to restart the whole procedure or continue the existing procedure. Likewise, when a new cloud resource become available on the MCP, the device can adjust its favourite offload destination and revise its decisions.

V. Simulations

In this section, we carry out simulation studies to verify and compare the results produced by the proposed algorithms. We give details of the hardware and software parameters used in the simulations first before simulation results are presented.

A. MCP Construction

While it is intractable to cover all possible use cases of mobile cloud platforms, we aim to base our simulation closely to the characteristics of an average modern mobile device with a wireless connectivity typically ranged within the capacities of existing wireless technologies (e.g. WiFi, 3G and LTE). On a hardware level, we construct our simulation with two building blocks: a *typical mobile device* and a *typical wireless connection*:

Definition 1. A *typical mobile device* has a battery capacity of 2000mAh, draws a current of 200-300mA during data transmission (with uplink drawing 20% more current than downlink) and 100-200mA when executing local computation tasks.

Table I: Comparison of Algorithms - Time

Simulation groups[†]			Algorithm time in seconds (ratio)		
ID	$\|P\|(\|P^C\|)$	$\|T\|/\|R\|$	SA	GAO	Optimal
S0	10(2)	60/90	0.46 (1.86)	0.04 (0.14)	0.24 (1)
M0	20(2)	120/180	0.69 (0.25)	0.12 (0.04)	2.77 (1)
L0	30(4)	180/270	1.22 (0.17)	0.39 (0.05)	6.82 (1)
X0	40(4)	240/360	2.73 (0.13)	0.96 (0.04)	20.28 (1)

[†] - Each group contains 100 tests the average of which is presented.

Definition 2. A *typical wireless connection* has an uplink bandwidth of 2-10Mbps, and a latency of 10-50ms.

The values in Definition 1 are based on the data presented in recent researches [11], [20], [25]–[27]. Characterisation of energy consumptions in smart devices and wireless networks is a challenging research topic. Because of the rapid development of new devices and emerging network standards, it is unrealistic to associate exact quantities to activities on mobile smart devices. Therefore, we use value ranges to characterise the energy consumptions of devices and networks in Definition 1 to simulate the variety of devices and power characteristics which may exist in a mobile cloud computing environment. We also used the tools presented in [18] to verify the values used in the definition. The network data in Definition 2 is based on the combination of 3G and LTE data presented in a recent report produced by Ofcom [28].

Likewise, on an application workflow level, we construct our simulation with two basic unit workloads:

Definition 3. A task, $t_a \in T$, has a *unit computation workload* if its execution, c_a, takes 1 second to complete on a typical device.

Definition 4. Two tasks, $\{t_a, t_b\} \subseteq T,\ (t_a, t_b) \in R$, have a *unit communication workload* if the size of the data sent from t_a to t_b, d_{ab}, takes 1 second to complete on a typical wireless connection.

In our simulation, we specify each task's workload size using multiples (uniformly distributed over [1, 20]) of a unit computation workload and a unit communication workload. The size of W of each simulation group is as specified in the first half of Table I.

To produce an initial baseline allocation scheme (ψ^0) we apply a baseline algorithm which attempts to reduce the total energy cost by distributing the number of tasks evenly across the MCP including both mobile and cloud nodes. This algorithm provides a good baseline value because although it does not seek the benefit of using an energy efficient device, its chance of being able to take that advantage is consistent.

B. Results and Analysis

1) Solution Quality: We now compare the quality of the solutions produced by both of our heuristics, SA and GAO, against the optimal solutions produced by a commercial QP solver (CPLEX v12.6.1). We plot the results from S0, M0, L0

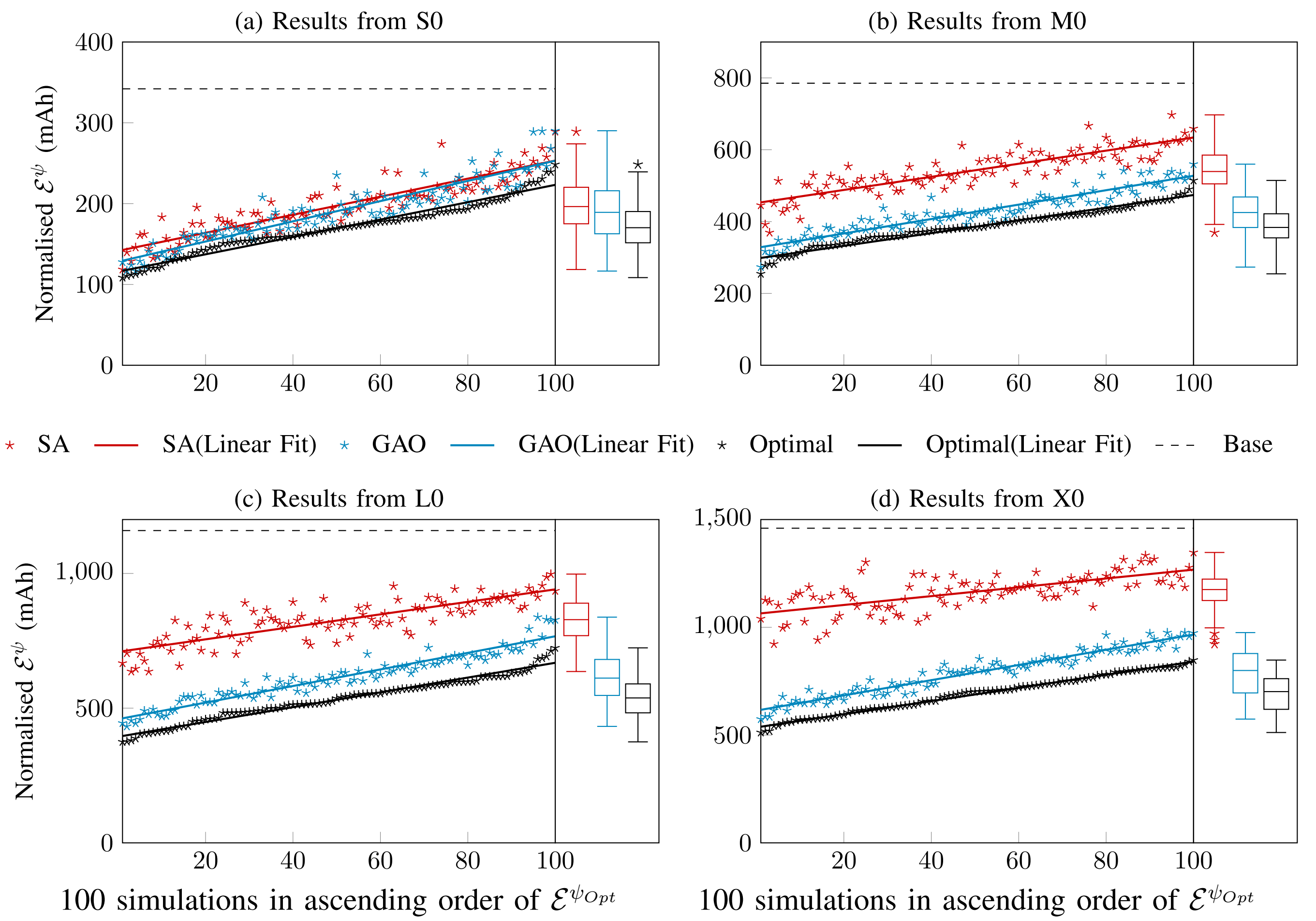


Figure 2: Comparison of algorithms: Solution quality.

and X0 in Fig. 2. Each of the four plots in the figure illustrates the quality of the allocation schemes ($\mathcal{E}^{\psi}$) produced by our heuristics and the optimal allocation schemes produced by the QP solver in each of the four simulation groups. Note that we plot the results in ascending order of its optimal (minimum) energy cost as given by the QP solver so that the results are easier to read. This does not indicate any trends in our simulation settings. We also normalised the results so that the results from each simulation may be compared to each other.

When developing energy-aware task allocation schemes, our objective is to minimise the amount of energy cost of the MCP. From all four plots of Fig. 2, we see that in all four (S0, M0, L0 and X0) groups of simulations, the proposed algorithms are able to reduce $\mathcal{E}^{\psi}$ to good extent. When the scale of the problem is small, as in S0 and M0, the differences between our heuristics and the solver is relatively small when compared with simulations of larger scales, as in L0 and X0.

From Fig. 2, we see that the quality of the solutions produced by GAO is consistently close to that of the optimal solution in all four simulation groups. In comparison, the solution quality of SA is gradually reduced as the scale of the MCP increases from S0 to X0. This is largely due to the fact that we keep the number of cycles and trials as a constant in SA when producing allocation schemes for all simulations. The quality of results produced by SA would have been improved if larger values were applied to simulations of larger sizes. However, such improvement would also increase the execution time of the SA algorithm. As shown in Table I, the SA algorithm takes longer to execute than GAO in all simulations, therefore we opt to not increase the number of cycles and trials in SA.

2) Algorithm Time: We give each algorithm's average execution time and their ratios to the QP solver's execution time in Table I. We see that GAO is superior than both other algorithms in terms of execution time. In M0, L0 and X0, GAO only takes 5% of the execution time of the QP solver. More importantly, the execution time is kept under one second which is acceptable for scheduling decisions in mobile application workflows of the simulated scale. Our implementation of the SA algorithm also has good scalability when compared to the QP solver.

For small scale MCPs, as in S0 and M0, the time costs of QP solver are also acceptable given that it produces the optimal allocation schemes.

As a summary of our findings, we see that GAO is able to produce good allocation schemes that are close to optimal within a reasonable time frame. Its solution quality and algorithm efficiency are significantly better than traditional heuristics like SA in solving the energy-aware task allocation problem in cloud-assisted mobile application workflows. It is also possible to deploy both GAO and QP solvers at the same time so that a good solution can be quickly produced and adopted from GAO and the optimal solution may be adopted at a later stage, provided no hardware or software metrics has been modified significantly.

VI. Conclusion

Cloud-assisted mobile application workflows are becoming ubiquitous in our everyday lives. Developing energy-aware task allocation schemes in such a mobile cloud platform is a critical step in ensuring the efficiency of such platforms. In this paper, we first modelled the energy costs of a mobile cloud platform with a quadratic binary program. We further developed two heuristic algorithms to overcome the scalability issues of quadratic program solvers. We tailored both algorithms so that allocation schemes can be produced efficiently. We verified and demonstrated the effect of our algorithms with simulations of different scales. Simulation results show that solutions that are close to optimal are consistently produced by our greedy algorithm GAO efficiently. Our modelling technique is also applicable to other energy critical scenarios.

Acknowledgement

This work is sponsored by the Research Project Grant of the Leverhulme Trust (Grant No. RPG-101), the National Natural Science Foundation of China (NSFC) (Project No. 61401123), the Fundamental Research Funds for the Central Universities (Grant No. HIT.NSRIF.201617), the Harbin Science and Technology Bureau (Project No. 2014RFQXJ116) and the Key Program of National Natural Science Foundation of China Grant (No. 61133005, 61432005).

References

[1] K. Pentikousis, “In Search of Energy-Efficient Mobile Networking,” *IEEE Communications Magazine*, vol. 48, no. 1, pp. 95–103, Jan. 2010.

[2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[3] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[4] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, “Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 337–368, 2014.

[5] Gartner Research, “Gartner Reveals Top Predictions for IT Organizations and Users for 2012 and Beyond,” 2011. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1862714

[6] 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 *MobiSys’10 The 8th International Conference on Mobile Systems, Applications, and Services*, Jun. 2010.

[7] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “CloneCloud: elastic execution between mobile device and cloud,” in *Proceedings of the sixth conference on Computer systems - EuroSys ’11*, 2011, p. 301.

[8] S. Kosta, A. Aucinas, and R. Mortier, “ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading,” in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 945–953.

[9] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A Survey of Computation Offloading for Mobile Systems,” *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2012.

[10] H. Shachnai and T. Tamir, “On Two Class-Constrained Versions of the Multiple Knapsack Problem,” *Algorithmica*, vol. 29, no. 3, pp. 442–467, Mar. 2001.

[11] J. Sharkey, “Coding for life - Battery Life, that is.” *Google IO Developer Conference 2009*, 2009.

[12] B. Gao and L. He, “Modelling Energy-Aware Task Allocation in Mobile Workflows,” in *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous’13)*, vol. 131, no. December, 2013, pp. 89–101.

[13] L. Pajunen and S. Chande, “Developing Workflow Engine for Mobile Devices,” in *EDOC’07 11th IEEE International Enterprise Distributed Object Computing Conference*, Oct. 2007.

[14] J. Balasooriya, J. Joshi, S. K. Prasad, and S. Navathe, “Distributed Coordination of Workflows over Web Services and Their Handheld-Based Execution,” in *ICDCN’08 Proceedings of the 9th International Conference on Distributed Computing and Networking*, Jan. 2008, pp. 39–53.

[15] A. Mnaoue and A. Shekhar, “A Generic Framework for Rapid Application Development of Mobile Web Services with Dynamic Workflow Management,” in *SCC’04 IEEE International Conference on Services Computing*, 2004.

[16] L. Feeney and M. Nilsson, “Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment,” in *INFOCOM’01. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society*, 2001.

[17] A. Rahmati and L. Zhong, “Context-for-wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer,” in *Proceedings of the 5th international conference on Mobile systems, applications and services - MobiSys ’07*, Jun. 2007, p. 165.

[18] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, “Accurate online power estimation and automatic battery behavior based power model generation for smartphones,” in *CODES+ISSS’10 Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2010.

[19] G. Ananthanarayanan and I. Stoica, “Blue-Fi: Enhancing Wi-Fi Performance using Bluetooth Signals,” in *Proceedings of the 7th international conference on Mobile systems, applications, and services - Mobisys ’09*, New York, New York, USA, 2009, p. 249.

[20] N. Vallina-Rodriguez and J. Crowcroft, “Energy Management Techniques in Modern Mobile Handsets,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 179–198, 2013.

[21] C. A. Floudas and P. M. Pardalos, Eds., *Encyclopedia of Optimization*, 2009.

[22] R. Burkard and F. Rendl, “A thermodynamically motivated simulation procedure for combinatorial optimization problems,” *European Journal of Operational Research*, vol. 17, no. 2, pp. 169–174, Aug. 1984.

[23] M. R. Wilhelm and T. L. Ward, “Solving Quadratic Assignment Problems by Simulated Annealing,” *IIE Transactions*, vol. 19, no. 1, pp. 107–119, Mar. 1987.

[24] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, “Cuckoo : a Computation Offloading Framework for Smartphones,” in *MOBICASE 2010 IEEE Computer Society*, 2010.

[25] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, “A close examination of performance and power characteristics of 4G LTE networks,” in *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys ’12*, 2012, p. 225.

[26] M. Dong and L. Zhong, “Self-Constructive High-Rate System Energy Modeling for Battery-Powered Mobile Systems,” in *MobiSys’11 The 9th International Conference on Mobile systems, Applications, and Services*, 2011.

[27] A. Pathak, Y. C. Hu, and M. Zhang, “Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof,” in *EuroSys’12 7th ACM european conference on Computer Systems*. ACM Press, Apr. 2012.

[28] Ofcom, “Ofcom publishes 4G and 3G mobile broadband speeds research,” 2014. [Online]. Available: http://media.ofcom.org.uk/news/2014/3g-4g-bb-speeds/