Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 518518, 6 pages
http://dx.doi.org/10.1155/2013/518518

Research Article

An Energy-Efficient Multisite Offloading Algorithm for Mobile Devices

Ruifang Niu,[1] Wenfang Song,[2] and Yong Liu[1]

[1] *School of Electronic and Information Engineering, Henan University of Science and Technology, Luoyang 471023, Henan, China*
[2] *School of Computer Science and Engineering, Beihang University, Beijing 100191, China*

Correspondence should be addressed to Wenfang Song; iewenfangsong@163.com

Received 10 January 2013; Accepted 20 February 2013

Academic Editor: Jianwei Niu

Copyright © 2013 Ruifang Niu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Computation offloading is a popular approach for reducing energy consumption of mobile devices by offloading computation to remote servers. Most of the prior work focuses on a limited form of offloading part of computation from a mobile device to a single server. However, with the advent and development of cloud computing, it is more promising for the mobile device to reduce energy consumption by offloading part of computation to multiple remote servers/sites. This paper proposes an Energy-Efficient Multisite Offloading (EMSO) algorithm, which formulates the multiway partitioning problem as the 0-1 Integer Linear Programming (ILP) problem. Moreover, our proposed EMSO algorithm adopts the multi-way graph partitioning based algorithm to solve it. Experimental results demonstrate that our algorithm can significantly reduce more energy consumption as well as execution time and better adapt to the unreliability of wireless networks (such as the network bandwidth changes), compared with the existing algorithms.

1. Introduction

With the development of cloud computing, mobile devices have the potential to become powerful tools for information access and mobile application. Nowadays, it has become the primary computing platform for many users who expect their mobile devices, such as smart phones, to run sophisticated applications. However, the limited battery life is still a big obstacle for the further growth of mobile devices [1]. Several known power-conservation techniques [2, 3] include turning off the mobile computing devices screen when not used, optimizing I/O, and slowing down the CPU, among others. Although exponential improvements have occurred in hardware components, such developments have not come up in battery technology, and we cannot anticipate any significant changes in this field in the near future. Therefore, prolonging the battery life of mobile devices has become one of the top challenges.

One popular technique to reduce the energy consumption for mobile devices is computation offloading [4] or cyber forging [5], which means that parts of an application execute on the remote servers, with results communicated back to the local device. Most of existing work is often limited and restricted to the form of offloading computation from a mobile device to a single server [4–9]. Such scheme cannot adapt well to the cloud environment [10]. Since it is difficult to directly program the cloud-enabled application, an alternate scheme is to write a monolithic application and then automatically partition it between the mobile device and the remote sites [11, 12]. Therefore, in this paper we propose an Energy-Efficient Multisite Offloading (EMSO) algorithm, which focuses on offloading parts of an application from a mobile device to multiple remote sites. Figure 1(a) shows a monolithic mobile application totally running on the resource-limited mobile device without computation offloading, and Figure 1(b) shows the case of distributed application execution by offloading parts of the codes from the mobile device to remote servers.

Computation offloading is confronted with several key challenges. Firstly, in a multisite offloading scenario, the unreliability of the wireless network (e.g., bandwidth often changes dynamically) affects the feasibility and efficiency of computation offloading for mobile devices. Existing work relies on programmers to modify the program to deal

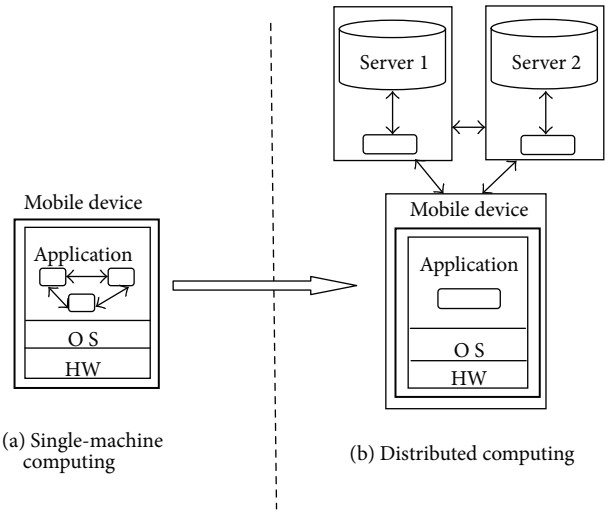


Figure 1: (a) Monolithic mobile application running on a mobile device. (b) Distributed execution by computation offloading between a mobile device and two sites.

with partitioning, state migration, and even the changes in network conditions [4, 5]. Although it can save more energy for mobile devices, it increases the additional burden for programmers and cannot adapt well to network environment changes. Secondly, the granularity of offloading must be chosen appropriately. Most prior work performs offloading at the class level [6, 9, 11], which does not allow objects of the same class to be offloaded to different servers, resulting in poor partitioning performance.

This paper describes a novel approach motivated by the idea in [8] to tackle these challenges. In this paper, we introduce a multiway partition algorithm, which models the bandwidth as a random variable to better adapt to the bandwidth changes of wireless networks and allows a program to be partitioned between multiple sites. Moreover, based on the Weight Object Relation Graph (WORG) constructed by using static analysis and dynamic profiling techniques [7], we accomplish the computation offloading for a given monolithic application at the object level to perform more efficient partitioning than that at the class level. Experimental results demonstrate that our algorithm can significantly reduce energy consumption with automatic adjustment to different network conditions.

The rest of this paper is organized as follows. Section 2 provides a detailed description of our multisite offloading scheme. Experiment and analysis are presented in Section 3, followed by some concluding remarks in Section 4.

2. Energy-Efficient Multisite Offloading Algorithm

Figure 1 in Section 1 shows the multisite offloading model. It shows that, with computation offloading, a distributed application execution will be partitioned between the mobile device which must contain at least one execution module such as the user interface and one or more servers which can be used for computation offloading in order to improve the execution or reduce energy consumption for the mobile device.

Normally, determining which portions of a computation to offload is cast as a graph partitioning problem. Our proposed Energy-Efficient multisite Offloading (EMSO) models the program to be partitioned as a Weight Object Relation Graph (WORG), with nodes representing the computation module (a run time object of the application), and edges representing the interaction between modules (e.g., invocations between one object and another). In a WORG, the weight of an edge indicates communication costs (in power) of the interaction between two modules, while the weight of a node represents the computation power consumption of the object module. The goal of this paper is to minimize the energy/power consumption by computation offloading.

The total costs of the partitioning can be calculated by considering both the weights of edges for communication and the weights of nodes for computation to get the best tradeoff. The optimal partitioning scheme means the optimal choice of modules to offload [13]. The next section formalizes such problems, giving an Integer Linear Programming (ILP) formulation of the multi-site offloading problem.

2.1. Graph Construction. As for the aforementioned weights of nodes and edges of WORG, they can be estimated by either static analysis or profiling of the program. EMSO first apply constructs the initial ORG of the application by using the Soot analysis framework [14] to perform the static points to analysis. And then offline profiling [7] is performed to assign weights to the nodes and edges of the ORG to construct the WORG. Figure 2 shows a WORG which we construct by both the static analysis and offline profiling methods for an application.

2.2. Problem Formulation. Our goal is to partition a graph WORG $= (V, E)$, with vertices set V and edges set $E \in V \times V$, and a set of $k + 1$ partitions denoted as $P = \{p_0, p_1, \ldots, p_k\}$ (p_0 represents the mobile device, and $p_1, \ldots, p_k$ represent the offloading sites, k is the number of offloading sites). As shown in Figure 2, the weight of the vertex v is described as a 2-tuple $\langle t_c(v), t_s(v)\rangle$, where $t_c(v)$ indicates the CPU execution time for each object running on the client, and $t_s(v)$ is that for each object running on the server. $t_s(v)$ can be calculated by $t_c(i)/k$, where k indicates that the server is k times faster than the mobile device. Each edge $e_{v1,v2}$ is associated with a weight $\langle s(v1, v2)\rangle$ indicating the amount of the total data that need to be transmitted between two nodes. EMSO collects $\langle t_c(v), t_s(v)\rangle$ and $\langle s(v1, v2)\rangle$ metrics by offline Profiling [7] during the WORG construction described in Section 2.1.

We can formulate the multiway partitioning problem as the 0-1 ILP problem. Our goal is to minimize the energy consumption, that is, the value of the following objective function:

$$\text{Energy (WORG)} = \sum_{v \in V} \left(E\left(v_l\right) \cdot x_l + E\left(v_s\right) \cdot x_s\right) + \sum_{v1 \in V,\, v2 \in V} \left|x_l - x_s\right| \cdot E\left(e_{v1,v2}\right), \quad (1)$$

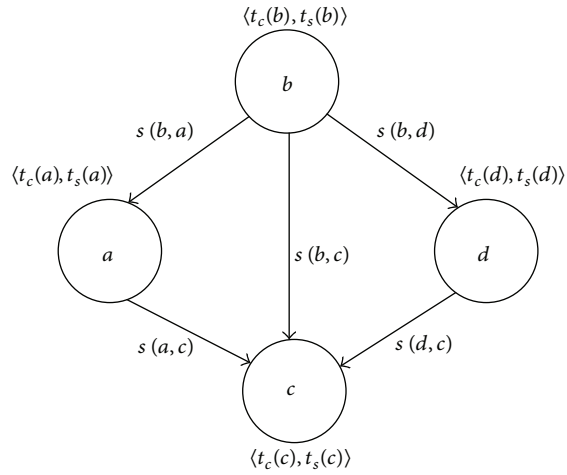


Figure 2: WORG of an application.

Input: $WORG = (V, E)$, B, b, N_L, a
Output: *Xmin*-the optimal partitioning scheme,
MinEnergy-the minimal energy consumption
(1) Compute the minimum energy consumption when bandwidth = b using the Stoer-Wagner algorithm, noted as *minE*
(2) $minE = min\ E^*(1 + a)$
(3) For v_i in V
(4) If v_i in N_L
(5) $X[i] = 1$; // vertexes running on the client
(6) Else
(7) $X[i] = -1$; //vertexes to be partitioned
(8) End if
(9) End for
(10) DFSearch(1, *minE*, *WORG*, *X*, *Xmin*, *MinEnergy*)
(11) Return {*Xmin*, *MinEnergy*}

Algorithm 1: Graph partitioning based algorithm.

where x_l and x_s indicate the assignment of each node: $x_l = 1$, $x_s = 0$ if vertex v_l is assigned to the client and v_s is assigned to servers, $x_l = 0$, $x_s = 1$ otherwise. Equation (1) is subject to the following constraint:

$$\forall v \in V : x_l + x_s = 1. \tag{2}$$

$E(v_l)$ and $E(v_s)$ are the energy consumption of vertex v running on the client and the server, respectively. They can be computed through the following (3):

$$\begin{aligned} E(v_l) &= P_c \times t_c(v), \quad v \in P_0, \\ E(v_s) &= P_s \times t_s(v), \quad v \in P_1 \ldots P_k, \end{aligned} \tag{3}$$

where $t_c(v)$ and $t_s(v)$ are the weights of vertex v when running on the client and on the servers, respectively. P_c and P_s are the power CPU of the client or the servers.

$E(e_{v1,v2})$ is the energy consumption for data transmission between vertex $v1$ and vertex $v2$ when they are not running on the same site, for example, one running on the client and the other on the servers. $E(e_{v1,v2})$ is computed by (4):

$$E(e_{v1,v2}) = \frac{s(v1, v2)}{b} \times P_{\text{wi-fi}}, \tag{4}$$

where $s(v1, v2)$ is the weight of the edge between vertex $v1$ and $v2$. b indicates the network bandwidth and $P_{\text{wi-fi}}$ is the power of the wireless Wi-Fi network interface.

To minimize the value of (1), the key is to determine the value of x_l and x_s, that is, 0 or 1. As remote servers usually executed much faster than mobile devices with powerful configuration, it can save energy and improve execution to offload part of computation to servers. However, when vertexes are assigned to different sites, the interaction between them leads to communication cost. Therefore, our problem formulation aims at the optimal assignment of vertexes for graph partitioning and computation offloading by trading off computation costs and communication costs.

2.3. Partitioning/Offloading Algorithm. We perform a multiway graph partitioning based algorithm to solve the ILP problem. First, we transform the WORG to a Directed Acyclic Graph (DAG) and perform the topologic sort. Then, we use the depth-first search to traverse the search tree and compute the $Energy(G)_B$ and $Energy(G)_b$ for each encountered nodes, where B is the current bandwidth, and b is the critical bandwidth that meets $P\{B \geq b\} > P_c$. P_c is the guaranteed probability, and $Energy(G)_b$ represents the energy consumption of the particular partitioning scheme when bandwidth is b. During the search, if $Energy(G)_b$ does not fulfill the constraints or $Energy(G)_B$ is larger than the current minimal energy (*MinEnergy*), that is, $Energy(G)_B > MinEnergy$, the subtree of the node will be cut and it traverses back to the parent node to continue searching. After traversing the whole tree of the DAG, we will get the optimal partitioning that fulfills the given constraints. The partitioning algorithm is shown as Algorithm 1, where N_L is the node collection which runs locally on the client, and a is an empirical constant to set the constraint of the minimal energy. The DFsearch function is the depth-first search algorithm described as above shown.

3. Evaluation

3.1. Experiment Setup. This section presents the experimental results of our EMSO algorithm. We evaluate the performance of EMSO by comparing it with the No Application Partitioning (NAP) case and a Static Application Partitioning (SAP) algorithm [15]. The evaluation metrics are energy consumption and execution time. We perform the comparisons on three random graphs generated by certain schemes to simulate the real-world scenarios involving a client device and two remote servers. The dataset used in experiments are listed in Table 1 as follows.

We suppose that the application is initially located on a mobile device and the bandwidth varies between the value of 10 kb/s and 100 kb/s. Other parameters, such as power consumption rates, are set as $P_i = 1.7W, P_c = 2.6W$, and

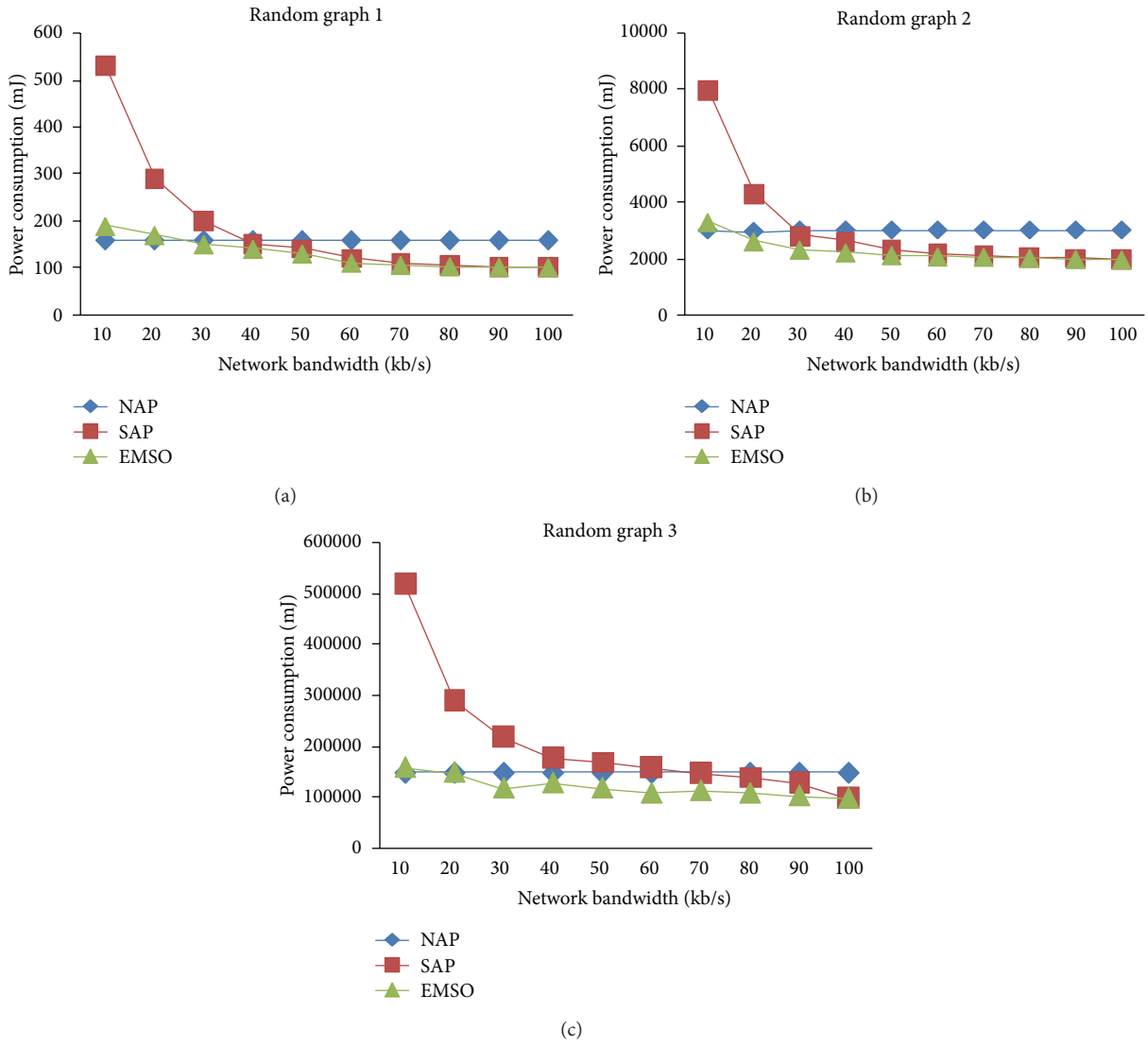


Figure 3: Energy (power) consumption comparisons of different algorithms with network bandwidth variation.

Table 1: The size of graphs.

Random Graph	Number of nodes	Number of edges
Graph 1	15	60
Graph 2	30	350
Graph 3	100	3238

$P_{\text{wi-fi}} = 2.3W$. $k = 5$ indicates that servers execute 5 times faster than the mobile device.

3.2. Energy Consumption Evaluation. From (1), we can see that the communication cost (i.e., $E(e_{v1,v2})$) is critical to partitioning decision, and it is directly related with the network bandwidth. However, in most cases, the network bandwidth changes dynamically, especially in wireless networks of mobile devices. The bandwidth is considered as a variable to improve the dynamic of partitioning in our EMSO.

To evaluate the adaption of EMSO to bandwidth changes, we compare the energy consumption of three algorithms with bandwidth changes. The experimental results with bandwidth varying in steps of 10 kb/s are presented in Figure 3. As shown in Figure 3, from Random Graph 1 with fewer nodes and edges to Random Graph 3 with the most nodes and edges, the energy consumptions of all three approaches increase, because the computation become larger and more complex as the nodes and edges grow. NAP consumes the constant energy with increasing bandwidth because the whole application keeps running on the mobile device without offloading and without energy costs of communication. As the bandwidth changes between 10 kb/s and 100 kb/s, the energy consumption of SAP varies more severely than that of EMSO. When the bandwidth becomes lower, SAP still maintains the former partitioning scheme, resulting in great increase of communication costs. However, our EMSO algorithm can find the better partitioning assignment when network bandwidth changes. Particulary, when bandwidth >20 kb/s,

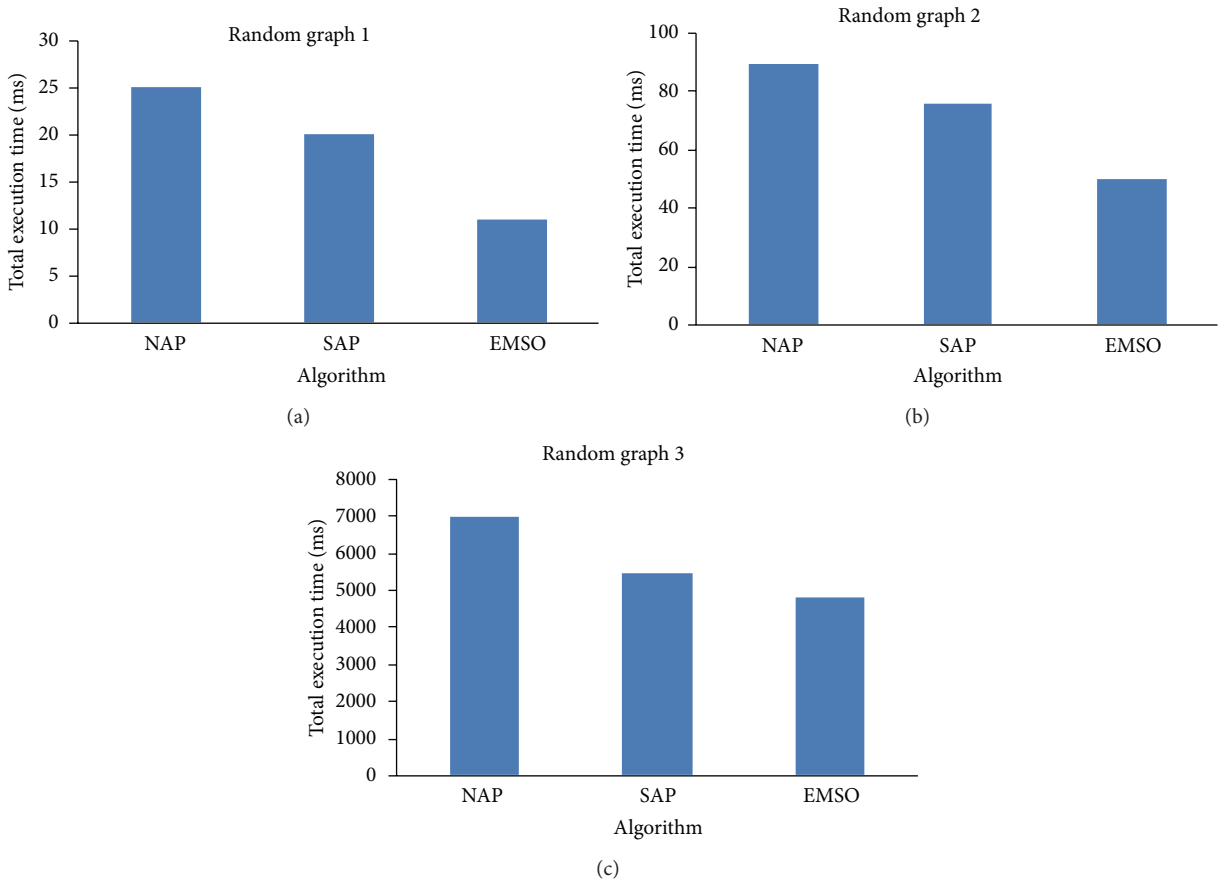


Figure 4: Total execution time comparisons of different algorithms.

EMSO algorithm saves about 25% energy compared to SAP. Meanwhile, when bandwidth >20 kb/s, our proposed EMSO algorithm also outperforms NAP due to partitioning approaches. The results demonstrate that EMSO is effective and beneficial to perform the partitioning for mobile devices.

3.3. Execution Time Evaluation. To further evaluate the performance of our EMSO algorithm, we estimate the total execution time of different algorithms as another evaluation metric to perform the partitioning in Figure 4. If it takes less time to execute a mobile application, it is beneficial for energy conversation of mobile devices and also improves the user experience with high-efficiency execution. As shown in Figure 4, we can see that our EMSO executes the computation offloading much faster than NAP without code offloading and faster than SAP with static partitioning, which demonstrates that, compared to NAP and SAP, our proposed EMSO can significantly improve execution time and reduce energy consumption for resource-restricted mobile devices. Besides, the result that execution time for Random Graph 3 is much larger than the other two (Random Graph 1 and Random Graph 2) also meets our expectations because of its more nodes and edges compared with the other two.

As a conclusion, as shown in Figure 3 on energy consumption and Figure 4 on total execution time, it is obvious that our proposed EMSO algorithm effectively saves the most energy and takes the least execution time to perform the partitioning, which is significantly beneficial for energy conservation of mobile devices.

4. Conclusion

This paper proposes an Energy-Efficient Multisite Offloading (EMSO) algorithm for computation offloading to save energy of mobile devices. This is a multi-site partitioning approach, which supports multiple differentiated offloading sites and assigns appropriate objects between mobile devices and servers dynamically to minimize energy consumption as the network bandwidth changes. EMSO models the application partitioning as a 0-1 ILP problem by using the multiway graph partitioning based algorithm to get the best tradeoff between computation costs and communication costs. With the constructed Weight Object Relation Graph (WORG), EMSO performs partitioning at the object level to achieve more precise offloading. Our evaluation demonstrates that EMSO is efficient in computation partitioning/offloading for mobile devices and outperforms the static algorithm in prior work with respect to both energy consumption and execution time.

Acknowledgments

This work was supported by the Research Fund of the State Key Laboratory of Software Development Environment

under Grant no. BUAA SKLSDE-2012ZX-17, the National Natural Science Foundation of China under Grant no. 61170296 and 61190120, and the Program for New Century Excellent Talents in University under Grant no. NECT-09-0028.

References

[1] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 3, pp. 277–324, 2003.

[2] K. Lahiri, S. Dey, D. Panigrahi et al., "Battery-driven system design: a new frontier in low power design," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 261–267, Bangalore, India, 2002.

[3] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery modeling for energy-aware system design," *IEEE Computer*, vol. 36, no. 12, pp. 77–87, 2003.

[4] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in *Proceedings of the 4th ACM international Conference on Compilers, Architecture and Synthesis for Embedded systems (CASES '01)*, pp. 16–19, Atlanta, Ga, USA, 2001.

[5] R. Balan and J. Flinn, "The case for cyber foraging," in *Proceedings of the 10th ACM SIGOPS European European Workshop (Sigcom '02)*, pp. 160–165, Saint-Emilion, France, July 2002.

[6] N. Geoffray, G. Thomas, and G. Folliot, "Transparent and Dynamic Code Offloading for Java Applications," in *Proceedings of the OTM Confederated International Conferences CoopIS, DOA, GADA, and ODBASE, CO*, pp. 57–66, 2006.

[7] L. Wang and M. Franz, "Automatic partitioning of object-oriented programs for resource-constrained mobile devices with multiple distribution objectives," in *Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS '08)*, pp. 369–376, Melbourne, Australia, December 2008.

[8] E. Cuervoy, A. Balasubramanian, D. K. Cho et al., "MAUI: making smartphones last longer with code offload," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 49–62, San Francisco, Calif, USA, June 2010.

[9] K. Yang, S. Ou, and H. H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 56–63, 2008.

[10] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: can offloading computation save energy?" *Computer*, vol. 43, no. 4, Article ID 5445167, pp. 51–56, 2010.

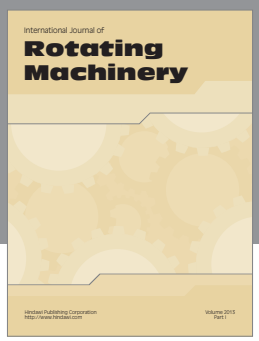
[11] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

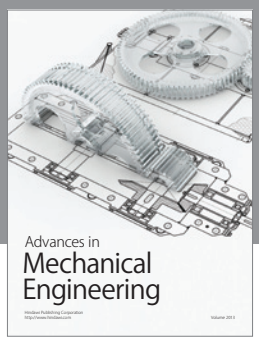
[12] B. G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proceeding of the 8th Workshop on Hot Topics in Operating Systems (HotOs '09)*, pp. 100–109, Monte Verità, Switzerland, 2009.

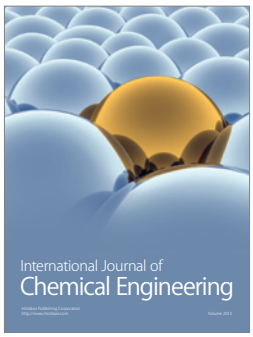
[13] S. Han, S. Zhang, Y. Zhang, and J. Cao, "Dynamic software allocation algorithm for saving power in pervasive computing," *Journal of Southeast University*, vol. 23, no. 2, pp. 216–220, 2007.

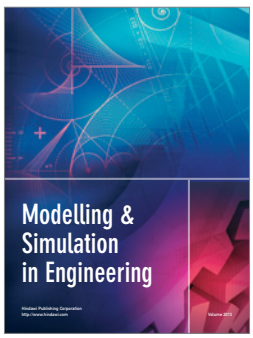
[14] Soot, November 2012, http://www.sable.mcgill.ca/soot/.

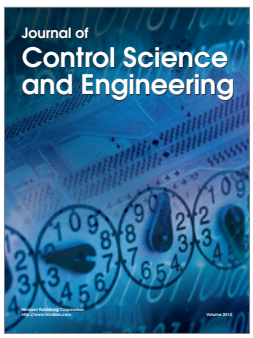
[15] S. Ou, K. Yang, and A. Liotta, "An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems," in *Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '06)*, pp. 116–125, Pisa, Italy, March 2006.

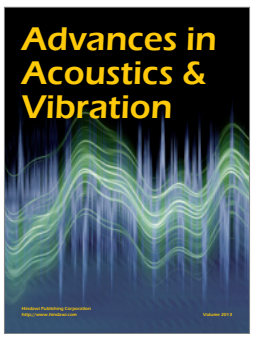


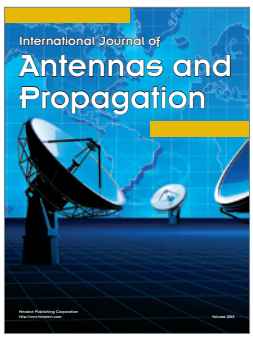


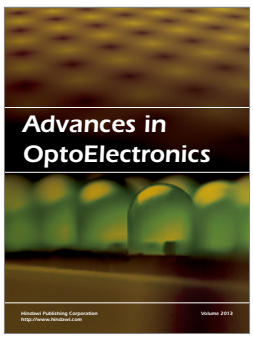


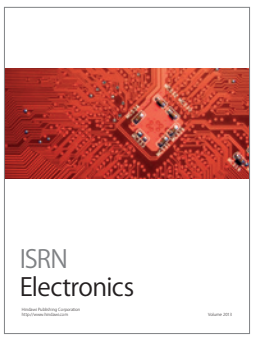


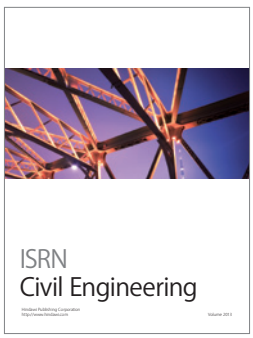


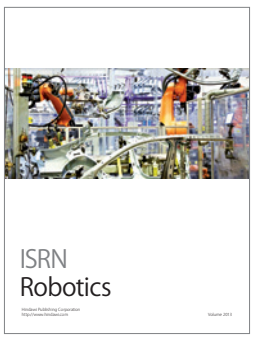


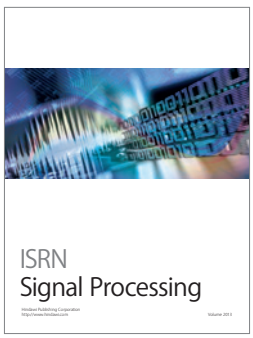


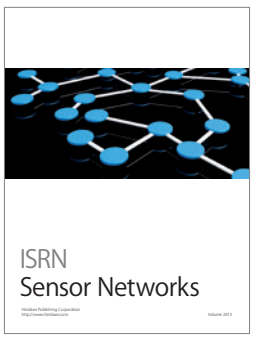












Hindawi

Submit your manuscripts at
http://www.hindawi.com