2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems

Enabling Real-time In-Situ Processing of Ubiquitous Mobile-Application Workflows

Hariharasudhan Viswanathan, Eun Kyung Lee, and Dario Pompili
NSF Center for Cloud and Autonomic Computing
Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ
e-mail: {hari_viswanathan, eunkyung_lee, pompili}@cac.rutgers.edu

***Abstract*—The heterogeneous sensing and computing capabilities of sensor nodes, mobile handhelds, as well as computing and storage servers in remote datacenters can be harnessed to enable innovative mobile applications that rely on real-time in-situ processing of data generated in the field. There is, however, *uncertainty* associated with the *quality* and *quantity* of data from mobile sensors as well as with the *availability* and *capabilities* of mobile computing resources on the field. Data and computing-resource uncertainty, if unchecked, may propagate up the "raw data→information→knowledge" chain and have an adverse effect on the relevance of the generated results. A unified uncertainty-aware framework for data and computing-resource management is proposed to enable in-situ processing of application workflows on mobile sensing and computing platforms and, hence, to generate actionable knowledge from raw data within realistic time bounds. A two-phase solution that captures the propagation of data-uncertainty up the data-processing chain using interval arithmetic in the first phase and that employs multi-objective optimization for task allocation in the second phase is presented and evaluated in detail.**

***Keywords*-mobile grids; autonomic management; uncertainty.**

I. Introduction

It has been projected that, by 2015, mobile handhelds will surpass wired desktop PCs as the most preferred medium of access to the Internet and, therefore, will significantly impact distributed computing paradigms that use Internet-connected devices [1] as a source of computing power and storage. In addition, advances in the field of wireless sensors has led to the development of compact sensor nodes (also called motes) capable of communicating with other mobile handhelds and of capturing a wide variety of sensor data – from biomedical to kinematic. The heterogeneous sensing, computing, and communication capabilities of these wireless sensor motes – in conjunction with mobile handhelds as well as computing and storage servers in remote datacenters – can be harnessed to enable innovative *ubiquitous mobile applications* that rely on *real-time in-situ processing of data generated in the field.*

The emphasis on real-time in-situ processing arises out of the need to generate *meaningful* and *actionable* knowledge within *realistic time bounds* by processing raw data collected opportunistically from pervasive mobile sensing devices using computational models (the "raw data→information→knowledge" or the data-processing chain). Examples of *distributed, real-time* applications that will benefit include ubiquitous context-aware health and wellness monitoring of the elderly [2], [3], rainfall and flood-risk estimation [4], [5], estimation of pollution level using air-quality measurements, object recognition and tracking, and user-generated multimedia search and sharing [6]. There is, however, *uncertainty* associated with the *quality* and *quantity* of data from mobile sensors as well as with the *availability* and *capabilities* of mobile computing resources on the field.

Data and computing-resource uncertainty, if unchecked, may propagate up the data-processing chain and have an adverse effect on the relevance of the generated results (level of accuracy and timeliness). Prior works in the field of data-uncertainty management [7]–[9] and on computing-resource-uncertainty management [10]–[12] identify and capture the limitations arising out of either imperfect/partial data or insufficient computing resources to provide not-so-accurate yet meaningful results within specified deadlines. However, these works focus on data- or resource-uncertainty management *in isolation* to prevent those from adversely affecting the result. Uncertainty-aware computing in the mobile sensing and computing domain requires, however, a *unified approach* (for both data as well as resource management) as it is affected by uncertainties arising out of both the sources.

In this paper, we present a unified uncertainty-aware framework for management of data and computing resources to enable ubiquitous mobile-application workflows execution on mobile sensing and computing platforms and, hence, to generate actionable knowledge from raw data within realistic time bounds. Mobile-application workflows are typically made up of several data collection, computation, and result generation tasks with a pre-determined parallelism and order of execution. The relevance of the results from mobile-application workflows rely heavily on the quality and quantity of raw data coming from the underlying multi-modal sensing infrastructure as well as on the computing resources available to execute them in real time. Note that a large amount of high-quality data may not guarantee good results as it increases the computational complexity and, hence, the computing-resource requirements. A small degree of uncertainty associated with computing resources (and/or data) can, in fact, outweigh the benefits brought by data (or computing resources).

We build our novel uncertainty-aware management framework on top of the autonomic mobile grid middleware we proposed in [12]. The middleware aids in the organization of the heterogeneous sensing, computing, and communication capabilities of static and mobile devices in order to form an heterogeneous computing grid. The entities of this heteroge-

978-0-7695-5104-3/13 $26.00 © 2013 IEEE
DOI 10.1109/MASS.2013.86

neous grid may at any time play one or more of the following three *logical roles*: i) *service requester,* ii) *service provider*, which can be a Data Provider (DP, a sensing device), Resource Provider (RP, a computing device), or both, and iii) *arbitrator*. DPs and RPs use *Service Advertisements (SAs)* to notify the arbitrator(s) of their capabilities and availability. A sensing device that is also ready to share its computing resources has to advertise itself as both DP and RP. Details on the middleware are elaborated in [12]. When a user application places a service request to an arbitrator through a requester, it will specify i) the workflow that conveys the data-processing chain and ii) a constraint in terms of the acceptable level of approximation in the result (or an equivalent confidence measure for the response). We introduce a generalized yet powerful *workflow representation* to convey the different data sources and the data-processing tasks as well as to capture their relationships at the different stages of the workflow.

The arbitrator should determine *on the fly*, 1) the quantity of data to request from the DPs (referred to as the *task size*) along with 2) the most appropriate computational model (from a suite of models) to process the raw data, extract features, and generate a result (actionable knowledge) that satisfies the requester-specified constraint in terms of accuracy in the final output, and 3) the amount of computing resources to utilize from those made available by the RPs so that the result is delivered as soon as possible while ensuring fairness in battery drain at the RPs. It is evident that this is a complex combinatorial optimization problem as it involves the choice of the best combination of data, resources, and the model to generate results that satisfy requester-specified constraints. We propose a *two-phase solution* to make the aforementioned complex combinatorial problem tractable. In the *first phase*, based on the workflow, on the requester-specified constraints as well as on the knowledge of propagation of uncertainty, we decide on the computational models (tasks) to use and the corresponding task sizes. In the *second phase*, we assign the tasks at each stage to RPs. Our contributions are as follows,

- We propose a simple yet powerful generalized workflow representation scheme to construct data-processing chains (tasks and dependencies) for ubiquitous mobile applications that are composed of multiple parallelizable and sequential tasks.
- We determine the appropriate task sizes at the different stages of the workflow by estimating the propagation of uncertainty from raw data to the final result using *interval arithmetic* [13].
- We formulate the problem of assigning tasks to RPs at every stage of the workflow as a *combinatorial optimization problem with multiple bottleneck objectives* (application makespan and fairness in battery drain), and propose a polynomial-time stage-wise threshold-based heuristic called β^**-allocation algorithm* for the same.

We also performed a detailed case study to assess how effective the proposed framework is as a key enabling technology for next-generation ubiquitous healthcare solutions (specifically, for stress and trauma assessment). Our study shows that data-uncertainty awareness helps determine the appropriate task sizes (data quantity) and, hence, control the uncertainty in the final result. Also, our multi-objective combinatorial approach for task allocation significantly outperforms the single-objective approaches in terms of makespan (15% improvement over the case whose objective is fairness in battery drain), fairness in battery drain, and network load (56% and 54%, respectively, over the case whose objective is makespan), while executing large mobile-application workflows.

The rest of the paper is organized as follows. In Sect. II, we introduce our generalized workflow representation. In Sect. III, we present our two-phase solution for handling propagation of data and resource uncertainty in dynamic mobile environments, and provide details of the two phases. In Sect. IV, we describe our experimental methodology and results. Finally, in Sect. V, we present our conclusions and plans for future work.

II. Workflows

Ubiquitous mobile applications are composed of multiple tasks whose order of execution is specified by a workflow. We have developed a simple yet powerful *generalized workflow representation scheme* to construct data-processing chains for mobile applications. First, we present our workflow representation scheme and then discuss biomedical workflows for ubiquitous health monitoring in detail.

A. Generalized Workflow Representation

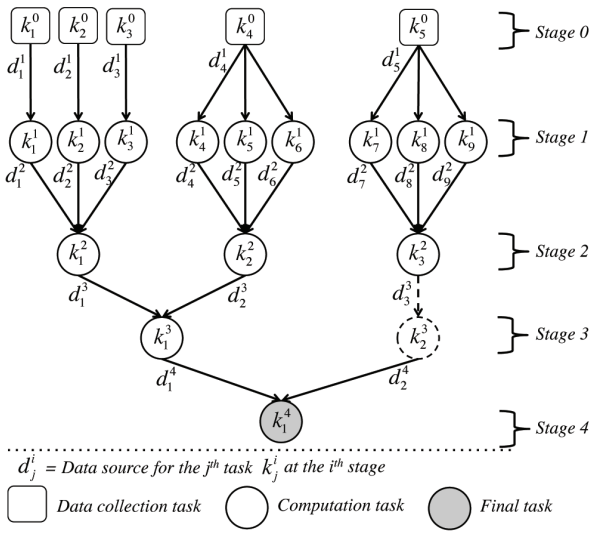

Fig. 1. Example 5-stage workflow ($M = 5$) depicted using our generalized workflow model. Note that dependencies exist only among tasks of successive stages. Dummy tasks, such as k^3_2, whose output equals the input without any propagation of uncertainty, are used to satisfy the dependency requirement.

Our generalized workflow is a *Directed Acyclic Graph (DAG)* composed of tasks (vertices) and dependencies (directed edges) as shown in Fig. 1. Tasks belong to one of the following three categories: i) data-collection task, ii) computation task, or iii) final result-generation task. These tasks are elementary and cannot be split further into *micro-tasks*, i.e., parallelization of elementary tasks does not yield any speed-up in execution time. However, tasks can be grouped together

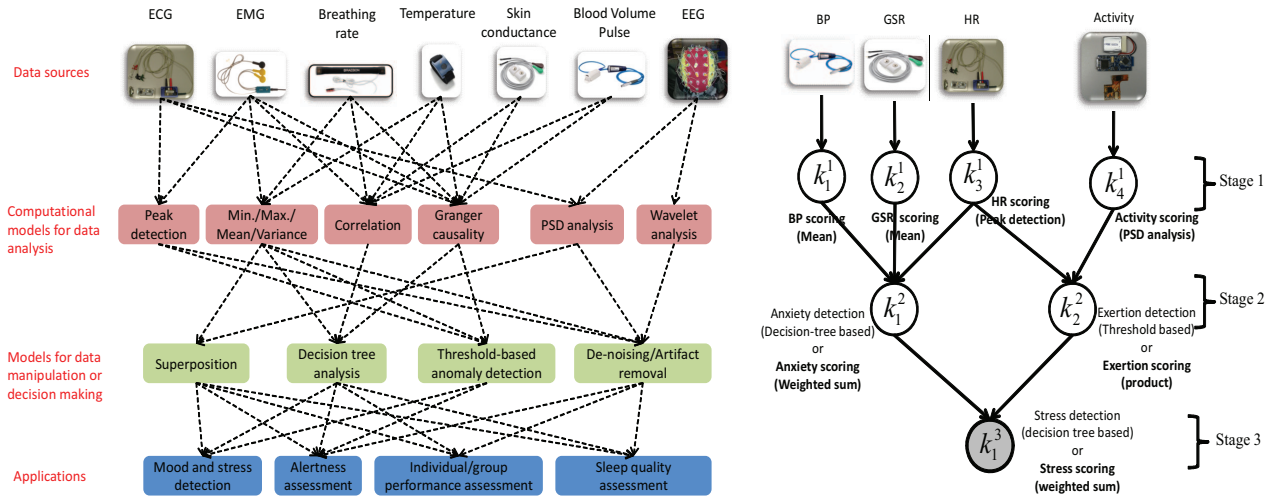

Fig. 2. (a) Sensor-based biomedical applications that constitute the larger class of ubiquitous healthcare applications; (b) A typical workflow for sensor-based non-invasive stress-detection applications.

to form *macro-tasks* (we do not study this feature as it is out of the scope of this paper). Each task is either in itself or a building block of a computational model that aids in the extraction of information from data.

The workflow is composed of multiple *stages* with a set of tasks $\mathcal{K}^i = \{k^i_j\}, j = 1, \ldots, |\mathcal{K}^i|$, to be performed at each stage i. Let M be the total number of stages, i.e., $i = 0, \ldots, M-1$. Stage 0 is composed entirely of data-collection tasks, which can be performed only at the DPs (sensing devices). The computation and final tasks (at any stage $i \geq 1$), however, can be performed at any RP (computing devices). Examples of computation and final tasks include frequency-domain analysis, estimation of Power Spectral Density (PSD), histogram analysis, linear combination, estimation of first-order statistics, rule-based (threshold-based) decision making, etc. Figure 1 shows an example 5-stage workflow ($M = 5$) represented using our generalized workflow model.

Different computational models may be suitable to accomplish a specific goal at a particular stage of the workflow depending on which of the following two criteria is more important: 1) the time taken to achieve the goal or 2) how the model (used to achieve the goal) propagates uncertainty from the inputs to the output. For instance, if the goal is to generate one scalar output from ten scalar inputs, the specific model may be the estimation of the maximum, minimum, median, mode, or mean (simple or weighted), depending on the time taken and/or on how each of these models propagate the uncertainty from the inputs to the output. In this paper, we assume that the computational models (i.e., the task types) in the workflow are *fixed and known a priori*.

Let $\mathcal{S}$ be the set of SPs, which includes both the data providers (DPs) and the resource providers (RPs). The SPs that perform the tasks at stage $i-1$, where $i \geq 1$), serve as data sources d^i_m, $m = 1, \ldots, |\mathcal{K}^{i-1}|$, for the tasks that have to be performed at stage i. Tasks at stage 0 are performed at the DPs themselves (as mentioned earlier) and they do not have data sources as the data is generated locally. The data sources for tasks at stage 1 are the DPs themselves (SPs where stage 0 tasks are performed). The mapping of data sources (SPs at stage $i-1$ to tasks at stage $i \geq 1$ of the workflow is represented by the matrix $\mathbf{Q^i} = \{q^i_{sk}\}$, where $s \in \mathcal{S}$ and $k \in \mathcal{K}^i$. Note that $\mathbf{Q^i}$ captures only the mapping of data sources to tasks at stage i; the actual allocation of tasks at any stage $i \geq 1$ to SPs takes multiple objectives into consideration (as detailed in Sect. III-B).

There are no dependencies between the tasks at a particular stage and, hence, they can be performed in parallel. Also, without any loss in generality, we assume that there can be dependencies only between tasks of consecutive stages in the workflow. Whenever we have dependencies between tasks of non-consecutive stages, we introduce the notion of "dummy tasks," whose output equals the input without any propagation of uncertainty and whose cost of operation (in terms of time and battery drain) is zero. In Fig. 1, task k^3_2 in stage 3 is a dummy task, which was introduced to break up the dependency between tasks k^2_3 and k^4_1 in non-consecutive stages 2 and 4, respectively. Note that, dummy tasks increase *neither* the cost of operation *nor* the computational complexity of the task-allocation problem as they need not be explicitly allocated to SPs; by default they are, in fact, allocated to the same SP that performed the corresponding parent task.

B. Biomedical Workflows

To understand the concept of mobile-application workflows and their constituent computational tasks, let us take the example of sensor-based biomedical applications (shown in Fig. 2). These applications are part of the larger class of ubiquitous mobile healthcare applications. Figure 2(a) shows the non-invasive sources (sensors) of different vital signs – Electrocardiogram (ECG), Electromyogram (EMG), breathing rate, temperature, skin conductance or Galvanic Skin Response (GSR), blood volume pulse, and Electroencephalogram (EEG) – along with the computational models for data analysis, data manipulation, and decision making. Some interesting

ubiquitous mobile biomedical applications include mood and stress detection or stress-level rating as well as assessment of alertness, cognitive performance (individual as well as group), sleep quality, etc.

Data analysis here refers to the extraction of features such as first-order statistics, PSD, degree of correlation, and causality. Data-manipulation models may refer to data-cleaning tasks such as artifact removal or tasks for extraction of derived data (like weighted combinations) from multiple independent sources. Models for decision making include threshold-based anomaly detection and decision trees. Applications employ these tasks at different stages of their workflows (the data-processing chain) so to extract actionable knowledge from the raw data acquired in the field.

Stress detection and stress-level rating: Figure 2(b) depicts the workflow for stress detection and stress-level rating [14], [15], which uses vital-sign data acquired from biomedical as well as kinematic sensors. Stage 1 of this workflow is purely composed of data-analysis tasks, whereas Stages 2 and 3 are composed of data-manipulation or decision-making tasks. In Stage 1, Heart Rate (HR), Blood Pressure (BP), GSR, and activity are each given a score $[0, 1]$ (0 corresponding to low, 1 to high) based on simple manipulations of the corresponding sensor outputs. Assigning these normalized scores requires domain knowledge and is in some cases subject specific (as in GSR and HR). Anxiety, physical exertion, and stress detection involve the use of decision-tree- or threshold-based models. Conversely, the model for anxiety-level scoring involves a weighted combination of HR, BP, and GSR scores. Similarly, physical exertion is scored by multiplying the activity and HR scores. Finally, stress level is determined using a weighted combination of scores for anxiety and physical exertion.

III. Proposed Solution

When the requester places a service request to an arbitrator (initiated by an automated application or by the user of an application), it will specify 1) the workflow that conveys the data-processing chain using the generalized workflow model and 2) the acceptable level of approximation in the result (or an equivalent confidence measure for the response). Let the user-specified acceptable level of uncertainty in the result be γ. The arbitrator should solve a *complex combinatorial problem* to determine the best combination of computational models (for tasks), quality and quantity of data (task sizes), and the computing resources to use such that the quality of the generated results is maximized. Quality of the results is determined by two factors, i) accuracy and ii) timeliness. *While the choice of tasks and task sizes has a bearing on the accuracy as well as on the timeliness of the result, the choice of computing resources has an effect only on the timeliness*. We exploit this fact and propose a *two-phase solution* to make the aforementioned combinatorial problem tractable.

In *Phase I*, the arbitrator first derives the maximum acceptable uncertainty at the output of every task at every stage of the workflow (i.e., the γ_j^i's, where $i = 1, \ldots, M-1$ and $j = 1, \ldots, |\mathcal{K}^i|$) based on the user-specified γ and on the knowledge of the workflow. Then, the arbitrator determines the sampling duration (i.e., the task sizes) so that the uncertainty bound on the output of every task at every stage is not violated. In *Phase II*, the allocation of tasks at every stage of the workflow to SPs is determined with the following two objectives in mind: minimization of total response time "and" minimization of maximum battery drain among all SPs in the heterogeneous computing grid. Even though the task allocation is determined in one shot, execution of the workflow is carried out in a *stage-wise* manner: such a stage-wise approach serves as a *natural checkpointing mechanism* to avoid the propagation of uncertainty in the event of SP failures.

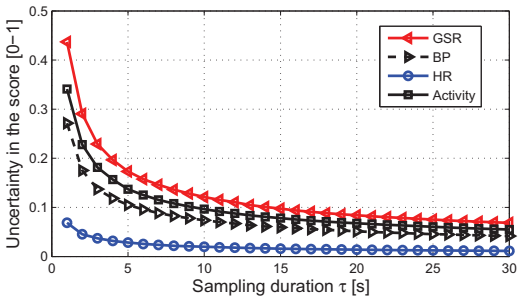

Fig. 3. The decrease in uncertainty γ (in terms of width of the interval within which the score lies as depicted in the y-axis) with increase in sampling duration τ. Heart Rate (HR), Blood Pressure (BP), Galvanic Skin Response (GSR), and Activity are each given a score [0,1] (0 -low, 1-high) based on the sensor output. Variability in such output necessitates representation of the score in terms of an interval. The higher the variability in sensor output, the larger the uncertainty in the score. Large τ decreases uncertainty in data.

A. Phase I: Propagation of Uncertainty

The end-user application conveys the acceptable level of uncertainty in the final result in terms of the "maximum width γ of the interval" within which the actual result lies. An interval (i.e., a range of possible values) is a compact way of representing the uncertainty associated with a particular quantity. The quantities of interest can be direct sensor outputs (e.g., temperature, skin conductance) or derived quantities from the raw sensor data (e.g., heart rate, blood pressure). When intervals are used to represent quantities, the mathematical computations on those quantities also need to be performed on intervals instead of fixed values. The branch of mathematics that deals with intervals instead of fixed values is known as *interval arithmetic* [13]. We leverage interval arithmetic to study the effect of "variability in sensor data" on the "variability in the final result". The objective in Phase I is to determine the most appropriate sampling duration (and, hence, the quantity of data to collect) at each sensor (or data provider) so to minimize the uncertainty in sensor data or in the derived quantity of interest and, hence, restrict the variability in the final result to γ.

The *sampling duration* τ (and, hence, data quantity) has a direct impact on the quality of the information extracted from raw sensor data. Figure 3 shows the decrease in uncertainty γ – in terms of width of the interval within which the HR, BP, GSR, and Activity scores lie – with increase in sampling

TABLE I
INTERVAL-ARITHMETIC RULES FOR BASIC MATHEMATICAL OPERATIONS AND MONOTONIC FUNCTIONS IN ONE VARIABLE

Operation	Interval of the result
$[x_1, x_2] \diamondsuit [y_1, y_2]$, where $\diamondsuit \in \{+, -, *, /\}$	$[\min\{x_1 \diamondsuit y_1, x_1 \diamondsuit y_2, x_2 \diamondsuit y_1, x_2 \diamondsuit y_2\}, \max\{x_1 \diamondsuit y_1, x_1 \diamondsuit y_2, x_2 \diamondsuit y_1, x_2 \diamondsuit y_2\}]$
$a^{[x_1, x_2]}$	$[a^{x_1}, a^{x_2}]$, $a > 1$
$\log_b [x_1, x_2]$	$[\log_b x_1, \log_b x_2]$, $\forall\ x_1, x_2 > 0$, $b > 1$
$[x_1, x_2]^n$	$[x_1^n, x_2^n]$, if odd $n \in \mathbb{N}$ or if even $n \in \mathbb{N}$, $x_1 \geq 0$ $[x_2^n, x_1^n]$, if even $n \in \mathbb{N}$, $x_2 < 0$ $[0, \max\{x_1^n, x_2^n\}]$, if even $n \in \mathbb{N}$, otherwise

duration. As detailed in the previous section, HR, BP, GSR, and Activity are each given a score (between 0 and 1) but in the form of an interval (e.g., $[0.45, 0.55]$). The higher the variability in sensor output, the larger the uncertainty in the score. It is important to note that the minimum and maximum values of the raw sensor data, which were used for computing the score, correspond to the boundaries of the 95% confidence interval of samples instead of the absolute maximum and minimum values as in the case of traditional interval arithmetic. The actual width of the 95% confidence interval is large when the sampling duration is small and viceversa. Usage of boundaries of 95% confidence intervals instead of that of the entire range of values makes our method resilient to outliers, which unnecessarily widen the intervals especially in the case of small sample sizes. Thus, determining the appropriate sampling duration and, hence, the number of samples used to calculate confidence intervals (with a priori knowledge of the sensor sampling rate), is crucial. We perform a linear search (as shown in Algorithm 1) for the most appropriate sampling duration and leverage interval arithmetic to study how the uncertainty propagates up the workflow.

Uncertainty propagation using interval arithmetic: We assume that estimates of the width of the 95% confidence intervals for different sample sizes (corresponding to different τ's) are available. This is a realistic assumption as every sensor can be calibrated offline, which needs to be done only once. These estimates can be maintained locally at the data providers and communicated to the arbitrator when requested, or can be maintained at the arbitrator. In our case, we assume that the arbitrator is in possession of the interval estimates. Every task in the workflow is represented mathematically using a combination of the basic operations listed in Table I. In Table I, $\diamondsuit$ represents the basic mathematical operations $+, -, *, /$. Once the confidence interval estimates at the different DPs are known (for a particular τ), the rules of interval arithmetic are leveraged to propagate the uncertainty up the workflow to the final result. If this estimate in the result uncertainty is smaller than the pre-specified γ, then the sizes of the tasks at all the M stages of the workflow (i.e., $\mathbf{R^i} = \{r_j^i\}, j = 1, \ldots, |\mathcal{K}^i|$, for $i = 0, \ldots, M-1$) can be calculated using sampling duration τ, knowledge of the sampling rate of the different sensors, and sensor-output data type.

B. Phase II: Stage-wise Multi-objective Optimization

Once the tasks and task sizes at all the stages of the workflow are determined in Phase I, they are allocated to SPs for execution in Phase II. Let $\mathcal{K}$ be the set of tasks that have to be completed at a particular stage of the workflow. Note that, as we address the problem of workflow-task allocation to SPs one stage at a time, we do not use an index to represent the current stage of the workflow. Service discovery at the arbitrators is achieved through voluntary service advertisements from the SPs. In addition to basic information about location, duration of availability (i.e., start t_s^{in} and end t_s^{out} times of the availability), and residual battery capacity (e_s^{adv} [Wh]) at each SP $s \in \mathcal{S}$, advertisements from DPs include types of raw data, types of sensors, and first-order statistics about the measurements; while the ones from RPs include information about the amount of computing (γ_s^{cpu} [normalized CPU cycles]), memory (γ_s^{mem} [Bytes]), and communication (γ_s^{com} [bps]) resources available for use.

Algorithm 1 determining sampling duration

Input: γ and the workflow
Output: Sampling duration τ

```
for τ = τ_min to τ_max do
    Estimate the width of the 95% confidence interval
    Propagate uncertainty up the workflow to estimate γ'
    {γ' is the estimated uncertainty in the final result}
    if γ' ≤ γ then
        return τ
    else
        Continue
    end if
end for
```

Let the time taken to complete a *unit task size* of task $k \in \mathcal{K}$ at SP s be t_{sk}^{comp} [h] and the time taken to transfer a unit block of data (in our case 10 KB) from one SP s_1 to another s_2 be $t_{s_1 s_2}^{com}$ [h]. These parameters, together with the task sizes, give the total amount of time required to execute the different tasks at the different SPs. Let the instantaneous power drawn by every workflow task k when running on a specific SP s be w_{sk}^{comp} [W] and the power drawn by the network interface at the SPs be w_s^{com} [W]. We assume that the arbitrator is aware of these parameters as there is only a finite number of types of SPs, and these types can be known in advance.

Multi-objective Combinatorial Bottleneck Problem (M-CBP): Based on all the aforementioned information, the arbitrator determines the optimal allocation matrix $\mathbf{A} = \{a_{sk}\}$, i.e., the optimization variable (1), that conveys whether a task k has been allocated to a SP s or not. In this optimization problem, there are two objectives, as in (2): i) *minimization of the maximum battery drain at all the SPs* (in terms of percentage) and ii) *minimization of the maximum time taken by SPs to complete all the tasks that have been assigned to them.* If the problem were to have only the first objective, it would imply minimization of the maximum battery drain. On the other hand, if the problem were to have only the second objective, it would imply minimization of the total response time. However, the two-objective problem strives to minimize response time while ensuring fairness in terms of battery drain at the SPs. This fairness maintains the heterogeneity of the

resource pool for longer periods by maximizing the lifetime of every single SP. This is a combinatorial optimization problem with multiple bottleneck objectives, which in short is referred to as a Multi-objective Combinatorial Bottleneck Problem (M-CBP) [16].

$$\textbf{Find:} \quad \mathbf{A} = \{a_{sk}\},\ s \in \mathcal{S}, k \in \mathcal{K}; \tag{1}$$

$$\textbf{Min:} \quad \begin{cases} \max_{s\in\mathcal{S}} \sum_{k\in\mathcal{K}} c^1_{sk} \cdot a_{sk}, \\ \max_{s\in\mathcal{S}} \sum_{k\in\mathcal{K}} c^2_{sk} \cdot a_{sk} \end{cases} \tag{2}$$

$$\textit{where, } c^1_{sk} = \frac{r_k}{e^{adv}_s} \cdot (w^{comp}_{sk} \cdot t^{comp}_{sk} + \\ + w^{com}_s \cdot \tilde{t}^{com}_{sk}), \tag{3}$$

$$c^2_{sk} = r_k \cdot (t^{comp}_{sk} + \tilde{t}^{com}_{sk}); \tag{4}$$

$$\textbf{S.t.:} \quad \sum_{s\in\mathcal{S}} a_{sk} = 1,\ \forall k \in \mathcal{K}. \tag{5}$$

Here, $\mathbf{C}^1 = \{c^1_{sk}\}$ and $\mathbf{C}^2 = \{c^2_{sk}\}$ are the cost matrices corresponding to the two objective functions and the sole constraint (5) ensures that a task be allocated to *one and only one* SP. The cost matrix $\mathbf{C}^1$ captures the battery drain (for communication as well as computation) incurred while executing a particular task at a particular SP. Similarly, cost matrix $\mathbf{C}^2$ captures the time spent (for communication as well as computation) when a particular task is performed at a particular SP. Also,

$$\tilde{t}^{com}_{sk} = \sum_{s'\in S} t^{com}_{ss'} \cdot q_{s'k},\ \forall s \in \mathcal{S}, k \in \mathcal{K}. \tag{6}$$

Here, the product of the communication delay matrix and the $\mathbf{Q}$ matrix (which captures the dependencies between tasks across successive stages) allows us to extract only the information about the communication delays when a certain task is assigned to a certain SP.

β^*-allocation: This algorithm is a polynomial-time heuristic that we propose to solve the M-CBP. This heuristic follows a threshold-based approach to determine the best possible allocation that achieves Pareto optimality (where one objective cannot be improved further without adversely affecting the other). Let $\mathcal{E}$ be the set of all possible combinations of the different thresholds. In our case, every $\varepsilon = \{\epsilon_1, \epsilon_2\} \in \mathcal{E}$ is two dimensional as we have only two cost matrices, one for the battery drain ($\mathbf{C}^1$) and one for the time taken ($\mathbf{C}^2$). With the aid of the thresholds, the two cost matrices are collapsed into one cost matrix $\mathbf{C}^*$ with binary weights (as shown in Algorithm 2). Then a simple minmax objective, shown in Algorithm 3, is used to find a feasible allocation using $\mathbf{C}^*$. Feasibility is not only determined by whether every task has been successfully allocated to a SP or not. The β^*-allocation algorithm searches for a feasible solution, an allocation $\mathbf{A}$ such that each task is allocated to one and only one SP, and the maximum number of tasks at a particular SP does not exceed β^*.

The arbitrator derives β^* from the knowledge of the following statistics about the SPs (maintained at the arbitrator) at a particular locality: the average arrival rate $\widetilde{\lambda}$ of SPs, their average sojourn duration $\widetilde{T}$ (whose inverse is called *churn rate*), and the average number of SPs $\widetilde{N}$. The relationship between these three numbers is given by the Little's Theorem, $\widetilde{N} = \widetilde{\lambda} \cdot \widetilde{T}$. When $\widetilde{T}$ is comparable to the average task execution time, then β^* is set to 1 in order to minimize the number of incomplete tasks (when SPs leave). However, when $\widetilde{T}$ is larger than the average task execution time (e.g., a few multiples), a higher number of tasks can be packed at fewer SPs with minimal risk of task incompletions. As the algorithm is threshold based, all the feasible solutions it provides will be Pareto optimal. Therefore, instead of finding all possible Pareto optimal solutions (the so-called *Pareto front*), we force an exit once the first feasible solution is found. This along with a careful choice of the granularity of the discrete thresholds can drastically improve runtime performance.

Algorithm 2 *β^*-allocation*: a heuristic for M-CBP

Input: $\mathbf{C}^1$, $\mathbf{C}^2$, $\mathcal{E}$, β^*
Output: An optimal solution $\mathbf{A}^*$ on the Pareto front

```
for every ε ∈ E do
  {Construct the bi-adjacency matrix C*}
  for every s ∈ S and k ∈ K do
    if c¹_sk ≤ ε1 and c²_sk ≤ ε2 then
      c*_sk ← 1
    else
      c*_sk ← 0
    end if
  end for
  {Find a fair allocation in bipartite graph G[C*ᵀ]}
  [Feasibility, A] ← MINMAX_ALLOCATE(β*, G[C*ᵀ])
  if Feasibility == False then
    Continue
  else
    A* ← A
    Break
  end if
end for
return A*
```

Algorithm 3 MINMAX_ALLOCATE: fair task allocation

Input: β^* and $G[\mathbf{C}] = \{\mathcal{K}, \mathcal{S}; E\}$, where $\mathbf{C} = \{c_{ks}\}$, $k \in \mathcal{K}$, $s \in \mathcal{S}$
Output: $Feasibility$ and $\mathbf{A} = \{a_{sk}\}$

```
Find the optimal allocation A s.t. the maximum no. of tasks
assigned to SP s is minimized
β = max_{s∈S} Σ_{k∈K} a_sk
if β ≥ β* then
  return False
else
  return [True, A]
end if
```

IV. Performance Evaluation

We have implemented a small-scale prototype of the proposed framework and performed an empirical evaluation. We have also used simulations to show the scalability of the proposed framework. In the following sections, firstly, we present details about our experiment methodology. Then, we discuss specific experiment scenarios and the results that demonstrate the benefits of data-uncertainty awareness (Phase I) as well as the performance of our proposed heuristic for the multi-objective optimization problem (Phase I and Phase II).

TABLE II
AVERAGE UNIT-TASK EXECUTION TIMES AND BATTERY DRAIN OF COMPUTING DEVICES WITH DIFFERENT BATTERY CAPACITY USED IN THE TESTBED

Task	Dell Inspiron	Dell Netbook	Samsung Galaxy Tab	Samsung Galaxy S3	Motorola Atrix 2	HTC Desire HD	LG Optimus
Battery capacity [Wh/V]	55.5/11.1	50/11.1	28/4	5.7/3.8	6.6/3.8	5.3/3.8	5.7/3.7
Peak detection [s/mA]	1/23	2.5/42	3.6/36	3.1/40	6.5/45	6.8/48	10.1/40
Correlation [s/mA]	1.2/25	2.5/42	3.68/38	3.3/40	6.8/47	6.8/47	10.3/42
PSD [s/mA]	1.5/25	2.9/50	4.5/47	3.7/50	7.2/50	7.5/49	14.8/56
Granger causality [s/mA]	2.8/50	6.1/85	9.6/78	6.8/85	12.7/80	13.3/83	23.6/75
ICA [s/mA]	14.4/80	28.5/130	30.6/128	33.5/120	45.1/125	47.7/130	61.2/135

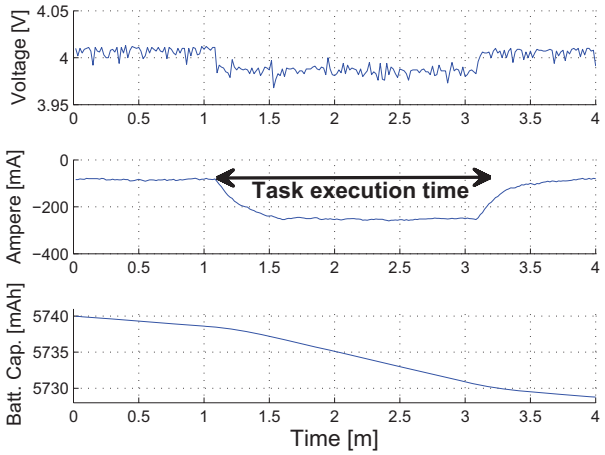

Fig. 4. Profile over time [min] of the current [mA] drawn by an application workflow task (at $\sim$ constant voltage [V]) running on a Samsung Galaxy Tab.

Devices and application profiling: Our testbed consists of Android- and Linux-based mobile devices with heterogeneous capabilities (summarized in Table II). As the objectives of the optimization problem are concerned with fairness in battery drain and makespan, the amount of battery drain in the RPs as a result of running workflow tasks as well as the task execution times need to be profiled in advance. However, the usage of actual Watt-hour [Wh] values will result in an unfair usage of those RPs with a higher battery capacity. Hence, in order to deal with the heterogeneity of mobile devices with different battery capacities (also shown in Table II), we use the *percentage battery drain* to make *fair* allocation decisions. In order to optimize allocation decisions, it is important to get a good estimate of the instantaneous power drawn in the mobile device while running a workflow task. This is obtained as follows: we ran the different workflow tasks of our biomedical applications on the individual mobile devices to determine the instantaneous power drawn (i.e., to determine the current drawn in mA as the voltage drop remains constant) and the total time taken for the task completion. Figure 4 shows the current drawn (negative values as it is current drained from the battery) and the time taken for task completion when executed in a Samsung Galaxy Tab. The voltage values did not show significant variability. From these observations, we determined the average time taken to complete a task of "unit" size, i.e., the time taken to process input data of size 10 KB.

TABLE III
THE THREE DIFFERENT WORKLOADS USED FOR THE EVALUATION OF THE PROPOSED UNCERTAINTY-AWARE MANAGEMENT FRAMEWORK

Workflow	no. of stages	no. of tasks	no. of task types
1 (Small)	4	20	5
2 (Medium)	8	40	10
3 (Large)	11	60	15

The workloads: We used different workflows to evaluate the performance of the two phases of the proposed solution. The 4-stage biomedical workflow for stress detection (shown in Fig. 2(b)) is used to illustrate the benefits of data-uncertainty handling, whereas three differently sized workflows (in terms of number of stages, total number of tasks, and number of task types as shown in Table III) are used to compare the performance of our heuristic for multi-objective optimization with competing single-objective optimization approaches.

A. Uncertainty vs. Non-uncertainty Awareness

To understand the benefits of data-uncertainty awareness, i.e., Phase I of our proposed solution, we studied the propagation of uncertainty in sensed data to the results of intermediate and final tasks in the workflow. The uncertainty in the sensed data is controlled by the sampling duration τ. As τ increases, the width of the 95% confidence interval of collected samples decreases, and so does the uncertainty in the score for HR, BP, GSR, and Activity, as shown in Fig. 3. The uncertainty in the output of Stage 1 of the workflow is propagated up the data-processing chain. Stage 2 of the 4-stage biomedical workflow is composed of simple computational models for anxiety-level and physical-exertion-level ratings. Anxiety score is a *weighted sum* of HR, BP, and GSR scores; while the physical-exertion score is the *product* of HR and Activity scores. At the final Stage 3, the stress score is a *weighted sum* of anxiety and physical-exertion scores.

The rules of interval arithmetic (shown in Table I) were applied to propagate the uncertainty in the scores from Stage 1 to the final result in Stage 3. Let the uncertainty in the final result for a particular τ be γ_τ. The sampling duration τ^* for which the uncertainty in the final result γ_{τ^*} is not higher than the pre-specified γ, i.e., $\gamma_{\tau^*} \leq \gamma$, is the optimal sampling duration, which we find using linear search. A longer sampling duration results in unnecessary energy expenditure

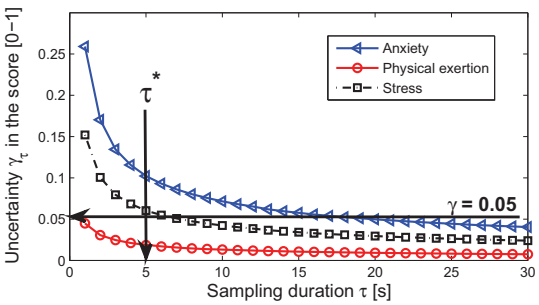

Fig. 5. Decrease in uncertainty in the outputs of the tasks (in terms of width of the interval within which the score lies) with increase in τ. Large sampling duration results in less uncertain scores for anxiety and physical exertion. The pre-specified $\gamma = 0.05$ and the optimal sampling duration, which results in $\gamma_{\tau^*} \leq \gamma$ is $\tau^* = 5$ s. Uncertainty awareness helps determine the most appropriate task size for the workflow tasks (based on τ^*), which not only has an effect on the accuracy but also on the timeliness of the result.

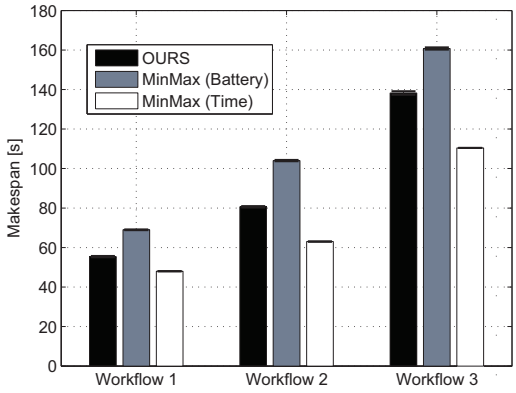

Fig. 6. Average makespan [s] of three different workflows achieved by the three competing task-allocation mechanisms.

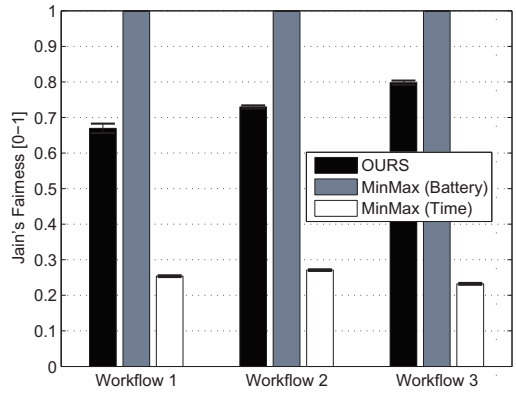

Fig. 7. Average Jain's fairness (ranging in $[0, 1]$) in battery drain achieved by the three competing task-allocation mechanisms.

for processing additional data, while a shorter duration results in higher uncertainty in the result, as shown in Fig. 5 where $\gamma = 0.05$ and $\tau^* = 5$ s. Note that a $\tau > 5$ s does not decrease the uncertainty greatly, while a $\tau < 5$ s fails to meet the pre-specified $\gamma = 0.05$. Uncertainty awareness helps determine the most appropriate task sizes for the workflow tasks (based on τ^*), which not only has an effect on the accuracy but also on the timeliness of the final result. Note that γ is just a requester-specified constraint on the quality of the result and has no effect on the performance of the proposed solutions.

B. Performance of the β^*-allocation Algorithm

Competing strategies: We assessed the performance of our heuristic for multi-objective optimization by comparing it against two popularly-used single-objective approaches: the first one, called *MinMax(Battery)*, tries to achieve fairness in battery drain; whereas the second one, called *MinMax(Time)*, tries to minimize the overall makespan of the mobile-application workflows by minimizing the time taken at every stage of the workflow. Note that, as the information regarding the underlying resource pool is unavailable to the end-user application, constraints cannot be provided to the single-objective approaches. Also, even conservative guesses for constraints may result in unfeasibility. Three different workflows differing in terms of number of stages, tasks, and heterogeneity of tasks were used to analyze the performance of all the approaches. As a single run of the workflow does not result in significant battery drain, the workflows were executed consecutively 200 times. Results presented are averages over 40 such runs (with different initial conditions in terms of battery capacity of the SPs) for statistical significance. The metrics of interest are makespan, fairness in battery drain, total battery drain, and the amount of data injected into the network.

Observations: Figure 6 shows that our heuristic for multi-objective optimization outperforms MinMax(Battery) by 15% *in terms of makespan*, but is outperformed by MinMax(Time). Conversely, as shown in Fig. 7, our heuristic outperforms MinMax(Time) by 56% *in terms of fairness*, but is outperformed by MinMax(Battery). From these two figures, it is clear that while MinMax(Time) and MinMax(Battery) perform very well in terms of their corresponding objective metric (i.e., makespan and fairness in battery drain, respectively), their performance in terms of the other metric is *very poor*. On the contrary, our multi-objective approach *finds a balance* in terms of both metrics. Figure 8 shows the total battery drain at all the SPs. MinMax(Time) seems to have incurred the least total battery drain; however, as already seen in Fig. 7, this single-objective approach drains the fastest and most the SP's batteries as it does not concern itself with the longevity of the entire resource pool. Our multi-objective optimization heuristic incurs a similar total battery drain as the energy-efficient MinMax(Battery). Finally, Fig. 9 shows the network load (in terms of kilo Bytes injected into the network) incurred by the three approaches: our heuristic incurs only slightly more load than MinMax(Battery) while significantly outperforming MinMax(Time). This is because in MinMax(Time) most of the tasks are allocated to the fastest device, which results in significant network load.

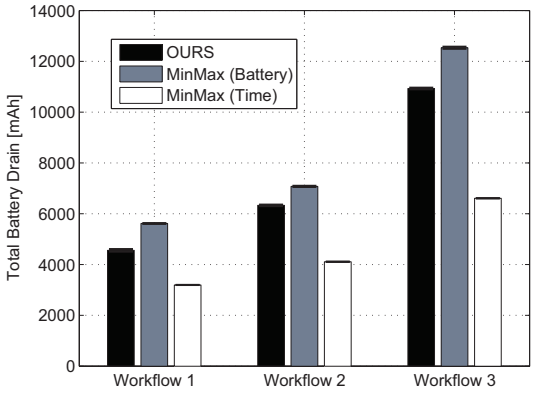

Fig. 8. Total battery drain [mAh] achieved by the three competing task-allocation mechanisms.

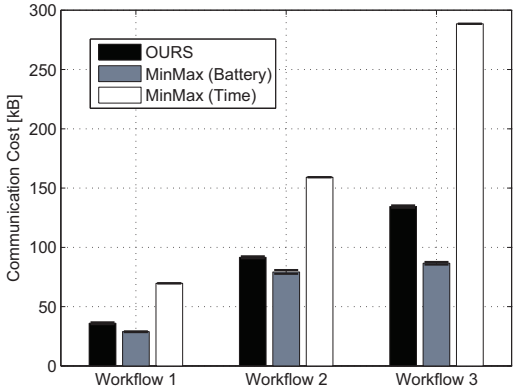

Fig. 9. Average communication cost or network load [KB] (injected into the network for a *single* run of three different workflows) incurred by the three competing task-allocation mechanisms.

V. Conclusions and Future Work

The ability to process raw data – collected opportunistically from pervasive mobile sensing devices – real time and in situ on mobile computing platforms will enable a host of innovative ubiquitous mobile applications. However, data and computing-resource uncertainty in mobile sensing and computing platforms, if unchecked, may propagate up the "raw data→information→knowledge" chain, and have an adverse effect on the relevance of the generated results. We proposed a unified uncertainty-aware framework for data and computing-resource management to enable in-situ processing of application workflows in mobile computing platforms. We presented a two-phase solution that captures the propagation of data-uncertainty up the data-processing chain using interval arithmetic in the first phase and that employs multi-objective optimization for task allocation in the second phase. Our analysis showed that data-uncertainty awareness helps determine the appropriate task sizes (data quantity) and, hence, control the uncertainty in the final result. Also, our multi-objective combinatorial approach to task allocation significantly outperforms single-objective approaches in terms of makespan, fairness in battery drain, and network load while executing large mobile application workflows. Presently, the framework is capable of executing workflows that can be modeled as directed acyclic graphs. In future, we will enhance the framework so to also support execution of cyclic workflows, which typically characterize gossip algorithms for consensus building in mobile distributed computing systems.

Acknowledgement

This work was supported in part by the Office of Naval Research - Young Investigator Program (ONR-YIP) grant no. 11028418. The authors would like to thank Prof. Jonathan Boyd, West Virginia University, for providing vital-sign data, and Devendra Desai, Department of Computer Science, Rutgers University, for his constructive suggestions and feedback.

References

[1] Volunteer Computing. [Online]. Available: http://www.gridcafe.org/volunteer-computing.html?PHPSESSID=af0ca7e6d4ad7b5bc2f4ac314e190f95

[2] H. Viswanathan, B. Chen, and D. Pompili, "Research Challenges in Computation, Communication, and Context Awareness for Ubiquitous Healthcare," *IEEE Communications Magazine*, vol. 50, no. 5, pp. 92–99, May 2012.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, Oct.-Dec. 2009.

[4] D. Hill and F. Farzan, "Downscaling Radar-Rainfall Data via Ubiquitous Sensing and Data Fusion," in *Proc. of World Environmental and Water Resources Congress*, Albequerque, NM, May 2012.

[5] Urban Flooding Preparedness Research Powered by Mobile Cloud. [Online]. Available: http://computeinmotion.com/2011/02/urban-flooding-preparedness-research-powered-by-mobile-cloud/\#more-68

[6] Hyrax: Cloud Computing on Mobile Devices using MapReduce. [Online]. Available: http://www.dtic.mil/docs/citations/ADA512601

[7] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working Models for Uncertain Data," in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, Atlanta, GA, 2006.

[8] Y. Diao, B. Li, A. Liu, L. Peng, and C. Sutton, "Capturing Data Uncertainty in High-Volume Stream Processing," in *Proc. of Conf. on Innovative Data Systems Research (CIDR)*, Asilomar, CA, Jan. 2009.

[9] R. Cheng, J. Chen, and X. Xie, "Cleaning Uncertain Data with Quality Guarantees," *Proc. of the VLDB Endow.*, vol. 1, no. 1, pp. 722–735, Aug. 2008.

[10] P. J. Darby and N. F. Tzeng, "Peer-to-peer Checkpointing Arrangement for Mobile Grid Computing Systems," in *Proc. of Intl. Conf. on High-Performance Parallel and Distributed Computing (HPDC)*, Jun. 2007.

[11] K. Lee and R. Figueiredo, "MapReduce on Opportunistic Resources Leveraging Resource Availability," in *Proc. of Intl. Conf. on Cloud Computing Tech. and Science (CloudCom)*, Taipei, Taiwan, Dec. 2012.

[12] H. Viswanathan, E. K. Lee, and D. Pompili, "An Autonomic Resource Provisioning Framework for Mobile Computing Grids," in *Proc. of Intl. Conf. on Autonomic computing (ICAC)*, San Jose, CA, Sep. 2012.

[13] J. G. Rokne, "Interval Arithmetic and Interval Analysis: An Introduction," in *Granular Computing*, W. Pedrycz, Ed. Physica-Verlag GmbH, 2001.

[14] F.-T. Sun, C. Kuo, H.-T. Cheng, S. Buthpitiya, P. Collins, and M. L. Griss, "Activity-Aware Mental Stress Detection Using Physiological Sensors," in *Proc. of Intl. Conf. on Mobile Computing, Application, and Services (MobiCASE)*, Santa Clara, CA, 2010.

[15] A. De Santos, C. Sanchez-Avila, J. Guerra-Casanova, and G. Bailador-Del Pozo, "Real-time Stress Detection by means of Physiological Signals," in *Recent Application in Biometrics*, J. Yang and N. Poh, Eds. InTech, 2011.

[16] J. Gorski, K. Klamroth, and S. Ruzika, "Generalized Multiple Objective Bottleneck Problems," *Operations Research Letters*, vol. 40, no. 4, pp. 276 – 281, 2012.