*Author Name*

# *Book title goes here*

# Contents

# *Chapter 1*

# A P2P Search Framework for Intelligent Mobile Crowdsourcing

**Andreas Konstantinidis**

*Department of Computer Science,University of Cyprus, Nicosia, Cyprus*

**Demetrios Zeinalipour-Yazti**

*Department of Computer Science,University of Cyprus, Nicosia, Cyprus*

## CONTENTS

## 1.1 Introduction

The advent of *social networks* and the widespread deployment of *smartphone* devices have brought a revolution in location-aware social-oriented applications and services on mobile phones. A *Smartphone Social Network* is a structure made up of individuals carrying smartphones, which are used for sharing and collaboration [1] (i.e., content, interests, comments and places.) For example, Google Latitude, Foursquare and Facebook Places enable users to check-in to favorite places, provide their location history, as well as numerous other functions. Smartphones can also unfold the full potential of *crowdsourcing* allowing users to transparently contribute to complex and novel problem solving. A crowd of smartphone users that is constantly moving and sensing providing large amounts of *opportunistic/participatory data* [4, 9, 3] can offer optimality to location-aware search and similarity services [5]. There is already a proliferation of innovative applications founded on opportunistic/participatory crowdsourcing that span from assigning tasks to mobile nodes in a given region to provide information about their vicinity using their sensing capabilities (e.g., noise-maps [38]) to estimating road traffic delay [41] using WiFi beams collected by smartphones rather than invoking expensive GPS acquisition and road condition (e.g., PotHole [17].)

Currently, the bulk of social networking services, designed for smartphone communities, rely on centralized or cloud-like architectures. In particular, in order to enable content sharing and community search over crowdsourced data, the smartphone clients upload their captured objects (e.g., images uploaded to Twitter, video traces uploaded to Youtube, etc.) to a central entity that subsequently takes care of the content organization and dissemination tasks. Although certain types of objects, such as text-based micro-blogs, will behave reasonably well under this model, significant challenges arise for captured multimedia and sensor data (e.g., data captured by the camera, microphone, WiFi RSS readings, etc.) We claim that the centralization of these object types will be severely hampered in the future due to several constraints including, (i) data-disclosure: continuously disclosing user-captured objects to a central entity might compromise user privacy in very serious ways and (ii) energy consumption: smartphones have asymmetric communication mediums with a slow up-link, thus by continuously transferring massive amounts of data to a query processor, through WiFi/3G/4G connections, can deplete the precious smartphone battery faster, increase query response time and quickly degrade the network health.

In this book chapter, we present a P2P search framework for intelligent crowdsourcing in Mobile Social Networks. Our framework uses mobile crowdsourcing primitives, which are expected to unveil the full potential of this novel computation model [5]. Additionally it uses a P2P search approach, coined SmartOpt [25], where captured objects remain local on smartphones and searches take place over an intelligent multi-objective lookup structure we compute dynamically, since the objectives to be optimized conflict with each other. In particular, the Multi-Objective

Optimization (MOO) approach utilizes the information contributed by the registered crowd to obtain a diverse and high quality set of near optimal solutions (i.e., Pareto Front) to benefit the decision making process. The book chapter also demonstrates our implemented work over *SmartLab*[1], which is a programming cloud of 40+ smartphones [26, 30].

The remainder of this book chapter is organized as follows: Section 1.2 provides the background and overviews the related work. Section 1.3, provides our system model and defines the problem in a rigorous manner. Section 1.4 introduces the SmartOpt framework and its internal modules composed of various operators. Section 1.5 details our SmartP2P prototype system and protocol as well as introduces SmartLab, our programming cloud of 40+ Smartphones. Our experimental methodology and results are presented in Section 1.6, while Section 1.7 concludes the paper and introduces potential applications of SmartOpt.

## 1.2 Background And Related Work

In this section, we provide background information and related research work that lies at the foundation of the SmartOpt framework with crowdsourcing. Initially, we introduce several taxonomies and applications of crowdsourcing, followed by research studies on Mobile P2P search and Query Routing Trees. Finally, this sections revisits the area of MOO and introduces the Multi-Objective Evolutionary Algorithms (MOEAs).

### *1.2.1 Mobile Crowdsourcing*

*Crowdsourcing* refers to a distributed problem-solving model in which a crowd of undefined size is engaged to solve a complex problem through an open call. Crowdsourcing has still not fully penetrated the mobile workforce, which will eventually unfold the full potential of this new problem-solving model. This is true due to the smartphones' usage characteristics and unique features. Smartphones are in widespread, everyday use and are always connected. Therefore, they offer a great platform for *extending existing web-based crowdsourcing applications* to a larger contributing crowd, making contribution easier and omnipresent. Furthermore, the multi-sensing capabilities (geo-location, light, movement, audio and visual sensors, among others) of smartphones, provide a new variety of efficient means for opportunistic data collection *enabling new crowdsourcing applications*.

Crowdsourcing applications on smartphones can be classified into extensions of web-based applications or as new applications. The former class expands to users that do not have access to a conventional workstation and adds the dimension of real-time location-based information to the service. Instances of such applications are

[1] Available at: http://smartlab.cs.ucy.ac.cy/

Gigwalk[2], Jana[3] and the work of Ledlie et al. [31]. The latter class includes applications for crowdsourced traffic monitoring (e.g., Waze[4]) and road traffic delay estimation (*VTrack* [41]); constructing fine-grained noise maps by letting users upload data captured by their smartphone microphone (*Ear-Phone* [38], *NoiseTube* [40]); identifying holes in streets by allowing users to share vibration and location data captured by their smartphone (*PotHole* [17]); location-based games that collect geospatial data (*CityExplorer* [33]); and real-time fine-grained indoor localization services exploiting the Radio Signal Strength (RSS) of WiFi access points (*Airplace* [29]).

Another key characteristic of mobile crowdsourcing is whether the crowd's contribution is *participatory* or *opportunistic*. Generally speaking, computations performed by users and user generated data is the input for *participatory* crowdsourcing, while the input for *opportunistic* crowdsourcing is data generated from sensors and computations performed by the crowd's devices automatically — i.e., trajectory matching, positional triangulation. The classical crowdsourcing services on the web are participatory, since they require the active participation of the users. The crowdsourcing tasks of the second category are transparent to the user as they usually run in the background using the sensors to collect readings from the environment.

Further crowdsourcing taxonomies are proposed by Geiger et al. [19] and Quinn et al. [35]. Both studies recognize that the value of the input can lie either in the *individual* or the *collective* contribution, where "*the crowdsourcing system strives to benefit from each contribution in isolation or from an emerging property resulting from the system of stimuli*", respectively. Furthermore, [19] divides applications regarding the contribution quality, which can be *homogeneous* or *heterogeneous*. In the former, each contribution has the same weight, whereas in the latter, each contribution is evaluated and can be compared to, compete against or complete other contributions. In [35], the incentives used for the crowd are also studied, which can be one or more from: *pay, altruism, enjoyment, reputation*, among others. Finally, [35] further classifies applications according to the *human skill* that is exploited including visual recognition, language understanding and communication. Notice that human skill is only required in applications with *participatory* contribution.

Smartphones feature different Internet connection modalities that provide intermittent connectivity (e.g., WiFi, 2G/3G/4G), as well as peer-to-peer connection capabilities that provide connectivity to nodes in spatial proximity (e.g., Bluetooth, Portable WiFi or the new generation NFC). Notice that each of these connection modalities comes at different energy and data transfer rate characteristics. In particular, smartphones have typically energy-expensive communication mediums with asymmetric upload/download links, both in terms of bandwidth and energy consumption, with the upload link being the weaker link.

Classical crowdsourcing applications are developed in a centralized or a decentralized manner. Centralized methods would ship the data generated and collected from the crowd to a server where the answer would be computed. Centralized methods are currently utilized by all social networking sites (such as Twitter, Youtube,

[2]Gigwalk Inc., May 2012, http://www.gigwalk.com/
[3]Jana, May 2012, http://www.jana.com/
[4]Waze Ltd., April 2012, http://www.waze.com/

Facebook, etc.) Continuously transferring data from the smartphone to the query processor can deplete the smartphone battery, increase user-perceived delays and quickly degrade the network health. In addition, it demands users to disclose their personal data with a central authority. On the other hand, decentralized methods would send the query to the smartphones, where all computations and communications would be performed locally. This approach might also perform poorly in terms of energy consumption if it invokes expensive computation tasks on all participants.

For location-dependent crowdsourcing applications, localization is usually either: i) GPS-only (fine-grained positioning, i.e., a few meters), ii) WiFi-only (semi-fine-grained, i.e., tens of meters), iii) Cellular-only (coarse-grained, i.e., tens to hundreds of meters). The latter two methods, which can be combined, require transmitting Cellular Tower and/or WiFi Received Signal Strength values over the Internet (via WiFi or 3G connection) to the localization server. GPS does not need to communicate any information over the Internet to a localization server.

### 1.2.2 Mobile P2P Search

*Mobile Peer-to-Peer/MANET* search can be roughly classified into: i) *Blind* Search [20, 32, 44], where mobile peers propagate the query using an unsophisticated (e.g., random, TTL property) approach to as many nodes in the network as possible, and ii) *Informed* Search [6, 23, 24, 34, 39], where mobile peers use semantic or location information to forward queries to specific nodes in the network. The proposed search approach presented in this chapter belongs to the latter class with the difference that we utilize a centralized approach where mobile peers (i.e., smartphone devices) subscribe to a centralized registry. Similar to [39], we utilize a content summary mechanism (i.e., profile) for discovering mobile peers that will participate in a query $Q$ by the centralized node. However, in our setting, the content summary of each mobile peer is stored at the centralized node upon its registration thus allowing multiple query users to use this information without requiring the retransmission of the content summary to each mobile peer. In *PeerDB* [34], the authors propose an agent-assisted query processing approach that has the ability to reconfigure the network based on optimization criteria (e.g., channel bandwidth). Although, this can increase the performance of the system (e.g., minimize energy cost, increase time performance), it imposes a high cost for maintaining the agents at each mobile node. In *Location-Aided Routing (LAR)* [24], the authors take into account the physical location of a destination mobile node, reaching in this way only a set of nodes close to the query user, which maximizes the performance of a query (i.e., time, energy). In *SmartOpt*, we additionally augment each mobile node with a social profile, which further decreases the number of participating nodes as only nodes that support a given query will contribute to the results.

*Query Routing Trees (QRTs)* in smartphone networks have recently received attention in the context of people-centric applications [4]. Such applications feature continuous sharing of data that can be utilized to create a number of collaborative scenarios (e.g., BikeNet [16]). A central component to realize such scenarios is the

availability of some high-level communication structure, such as QRTs. In [42], the authors present a technique that profiles the activities of the user in order to minimize the number of communication packets transmitted in the smartphone network. In contrast to [42], which focuses on a single objective of energy, our proposed technique focuses on two additional objectives: time overhead and recall. In [18], the authors form QRTs using flooding in order to continuously track mobile events and relay data to the query user. Similarly to the BFS algorithm, presented earlier, this approach suffers from significant energy waste as all nodes continuously and actively participate in the smartphone network. QRTs have also been extensively studied in the context of unstructured P2P system (e.g., IS, >RES, RBFS, Random Walkers, APS, etc. [45]), yet none of these was taking into account the resource-constrained nature of smartphone networks. Similarly query routing structures proposed for Sensor Networks, such as TAG, ETC and MHS [2], focus on building routing trees that are near-optimal (with respect to a single objective) but expose good aggregation and data synchronization properties during continuous data percolation to a centralized sink. On the other hand, our setting deals with snapshot query cases and multi-objective query optimization for smartphone social networks.

### 1.2.3 Multi-Objective Optimization

*Multi-Objective Optimization (MOO)* (a.k.a. multi-criteria or multi-attribute optimization) [8, 13] is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints and can be mathematically formulated as

$$\begin{aligned} &\text{minimize } F(x) = (f_1(x), \ldots, f_m(x))^T \\ &\text{subject to } x \in \Omega \end{aligned} \tag{1.1}$$

where $\Omega$ is the decision space and $x \in \Omega$ is a decision vector. $F(x)$ consists of $m$ objective functions $f_i : \Omega \to \Re, i = 1, \ldots, m$, where $\Re^m$ is the objective space.

The objectives in (1.1) often conflict with each other and an improvement on one objective may lead to deterioration of another. Thus, no single solution exists that can optimize all objectives simultaneously. In that case, the best trade-off solutions, called the set of Pareto optimal (or non-dominated) solutions, is often required by a decision maker. The Pareto optimality concept is formally defined as,

**Definition 1.** A vector $u = (u_1, \ldots, u_m)^T$ is said to dominate another vector $v = (v_1, \ldots, v_m)^T$, denoted as $u \prec v$, iff $\forall i \in \{1, \ldots, m\}$, $u_i \leq v_i$ and $u \neq v$.

**Definition 2.** A feasible solution $x^* \in \Omega$ of problem (1.1) is called *Pareto optimal solution*, iff $\nexists y \in \Omega$ such that $F(y) \prec F(x^*)$. The set of all Pareto optimal solutions is called the Pareto Set (PS), denoted as,

$$PS = \{x \in \Omega | \ \nexists y \in \Omega, F(y) \prec F(x)\}.$$

The image of the PS in the objective space is called the Pareto Front (PF),

$$PF = \{F(x) | x \in PS\}.$$

**Table 1.1: Table of Symbols**

| Symbol | Description of Symbols |
|---|---|
| $C$ | (Centralized) Social Networking Service |
| $U$ | Users of the Social Network (i.e., $\{u_1, u_2, ..., u_M\}$) |
| $P$ | User Profiles stored by $C$ for $U$s (i.e., $\{p_1, p_2, ..., p_M\}$) |
| $o_{ik}$ | Object $k$ (images, videos, etc.) recorded by user $i$. |
| $G$ | Conceptual Graph connecting the users in $U$. |
| $G'$ | Social Neighborhood of some arbitrary user. |
| $Q$ | Query conducted in social neighborhood $G'$ ($G' \subseteq G$). |
| $U'$ | Users that are connected to $C$ during the execution of $Q$. |
| $X$ | Query Routing Tree constructed to answer $Q$. |

MOO has numerous applications in virtually all domains of sciences, engineering and economics. MOO is a relatively new area in mobile/wireless networks, in general, and in Smartphone Networks in particular. In MOO, it is difficult to apply an existing linear/single objective or systematic method to effectively tackle a Multi-objective Optimization Problem (MOP), giving a set of non-dominated solutions. This is mainly due to the increased complexity and high dimensionality of the search (or decision) space. Our optimizer borrows ideas from *Multi-Objective Evolutionary Algorithms (MOEAs)*, which have been shown effective in obtaining a set of non-dominated solutions in a single run. In the literature, several MOPs were proposed in the content of Wireless Sensor Networks and Mobile Networks [22, 27, 28, 37], tackled in most cases by Pareto-dominance based MOEAs (e.g., the state-of-the-art Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [14]) and in few cases by decompositional MOEAs (e.g., Multi-Objective Evolutionary Algorithms based on Decomposition (MOEA/D) [46]).

## 1.3 System Model And Problem Formulation

In this section, we outline our system model and formulate the MOP SmartOpt aims to solve. A table of respective symbols is shown in Table 1.1.

### *1.3.1 System Model*

**Overview:** Let $C$, denote a social networking service that maintains centrally the social profiles $P = \{p_1, p_2, ..., p_M\}$, for each of its $M$ subscribed mobile users (i.e., a crowd $U = \{u_1, u_2, ..., u_M\}$). The profiles record basic user details, authentication credentials, the user interests (e.g., traveling, sports, music, etc.) and friendship relations that define the conceptual social network graph $G$ among the $M$ users. In our setting, a user $u_i$ ($i \leq M$) uses a smartphone (or tablet) device to both perform its day-

to-day activities but also to opportunistically capture objects of interest at arbitrary moments (e.g., "take a picture of the Liberty Statue"). Each object $o_{ik}$ might be tentatively *"tagged"* with GPS information and other user tags (e.g., *"lat: 40.689201355, long: -74.0447998047, tags: "Statue Liberty Ellis Island"*).

**Connection Modalities:** Each $u_i$ features different Internet connection modalities that provide intermittent connectivity to $C$ (e.g., WiFi, 2G/3G/4G). Each $u_i$ also features peer-to-peer connection modalities that provide connectivity to nodes in spatial proximity (e.g., Bluetooth, Portable WiFi or upcoming NFC available in Android)[5]. We assume that when $u_i$ is connected to $C$, then $C$ is aware of $u_i$'s absolute location (e.g., GPS) or $u_i$'s relative location (e.g., the cell-ids within $u_i$'s range, WiFi RSSI indicators within $u_i$'s range or other means utilized for geo-location). Notice that each of the connection modalities comes at different energy and data transfer rate characteristics. For example, we've profiled an Android-based HTC Hero and found that WiFi consumes 39mW/byte, 3G consumes 24mW/byte and Bluetooth consumes 14mW/byte. Additionally, Bluetooth had a symmetric data rate of 864kbps, while WiFi an asymmetric data rate of 123Kbps (up) and 2Mbps (down) and 3G an asymmetric data rate of 2.7Mbps (up) and 7.2Mbps (down). The nominal data rates for the aforementioned modalities might differ significantly, as this is also validated in [21], mainly due to the deployment environment. Moreover, while the power consumption on the different kinds of radios can be comparable, the energy usage for transmitting a fixed amount of data can differ an order of magnitude because the achievable data rates on these interfaces differ significantly [36]. Finally, the availability characteristics of these kinds of modalities can vary significantly. The penetration of some form of cellular availability (e.g., WiFi or 3G) is significantly higher than Bluetooth, on average. Thus, uploading or downloading large data items using Bluetooth can be more energy-efficient than using a radio network, but Bluetooth may not always be available and it is often slower.

**Search Techniques:** Let an arbitrary user $u_j$ ($j \leq M$), be interested in answering a query[5] $Q$ over its social neighborhood (i.e., nodes connected to $u_j$ either directly or through intermediate nodes) $G'$ ($G' \subseteq G$). For instance, let $Q$ be a depth-bounded breadth first search query over $u_j$'s neighbors in the $G$ graph (i.e., in $G'$). This kind of conceptual query can be realized in the following manners:

1. *Centralized Search (CS):* This algorithm assumes that the multimedia objects and tags are all uploaded to $C$ prior query execution. Once $Q$ is posted, $C$ can locally derive the answers (using its local tag database) and return the answers to $u_j$. This model, which is currently utilized by all social networking sites (such as Twitter, Youtube, Loopt, etc.), performs well in terms of query response time but performs poor both in terms of *data disclosure* (i.e., $o_{ik}$ objects and tags need to be continuously disclosed to $C$) and performance (i.e., data transmission of large objects over radio links is both energy demanding and time consuming.)

2. *Distributed Breadth-First-Search (BFS):* This algorithm assumes that the ob-

[5] Without loss of generality we assume simple Boolean keyword queries over tags.

jects and tags are all stored in-situ (on their owner's smartphones.) In order to realize the search task, a querying node $u_j$ downloads from the query processor the addresses (e.g., IP:PORT address) of its first line neighboring nodes (i.e., $G'' \subseteq G'$). $u_j$ then contacts the nodes in $G''$ in order to conduct a depth-bounded breadth first-search in a P2P fashion (i.e., using a pre-specified Query Time-To-Live $Q_{TTL} > 0$). Once some arbitrary node $u_x \in G'$ receives $Q$, it both looks at its local tags, in order to identify an answer and also forwards the request further until $Q_{TTL}$ becomes zero.

Although the BFS approach improves the data-disclosure drawback of the *CS* approach, it is quite inefficient during search. In particular, $Q$ has to go over a random neighborhood rather than a neighborhood that is contextually related to the query. For instance, in our Liberty Statue query example, we would have preferred querying a friend living in lower Manhattan rather than a person living in California (as the former would have a higher probability of capturing the statue). Also, if $u_j$ had two friends, $u_x$ and $u_y$, both living in lower Manhattan, with $u_x$ being in spatial proximity to $u_j$ during the query (i.e., within a few meters), while $u_y$ being far away, would have made $u_x$ a better choice for posting the query (as $u_x$ could have been queried through a local link such as Bluetooth).

### 1.3.2 Optimization Problem Formulation

The *Multi-Objective Query Routing Tree (MO-QRT)* structure in this chapter, improves the search operation of the BFS algorithm by optimizing the neighbor selection process. In particular, a node downloads from $C$ a QRT $X$ that is optimized according to the following formulation: *Given a social network of users $U$, a list of active users $U'$ and their coordinates, the profiles $P$ of these users and a query $Q$, posted by an arbitrary user $u_j$, the query processor aims to optimize an $X$ structure using the following* **objectives**:

**Objective 1:** *Minimize the total* Energy *consumption of X*

$$Energy(X) = min \sum_{\forall (u_a,u_b) \in X (X \subseteq U')} e(u_a,u_b) \tag{1.2}$$

where, $e(u_a,u_b)$ denotes the energy consumption for transmitting one bit of data over the respective edge (WiFi, Bluetooth and 3G).

**Objective 2:** *Minimize the* Time *overhead of X*

$$Time(X) = min(max_{(u_a,u_b) \in X} t(u_a,u_b)) \tag{1.3}$$

where, $t(u_a,u_b)$ denotes the delay in transmitting one bit of data over the respective edge.

**Objective 3:** *Maximize the* Recall *rate of X*

$$Recall(X,Q) = max(\frac{Relevant(Q) \cap Retrieved(X,Q)}{Relevant(Q)}) \tag{1.4}$$

where *Relevant(Q)* denotes the set of all objects in $U'$ that are relevant to $Q$, formally as:

$$Relevant(Q) = \bigcup_{\forall u_a \forall k (u_a \in U')} (o_{ak})),$$

given that $u_a$'s profile (denoted as $p_a$) contains terms found in $Q$. On the other hand, *Retrieved(X,Q)* denotes the set of objects that have been retrieved in response to $Q$ over structure $X$, formally as:

$$Retrieved(X, Q) = \bigcup_{\forall u_a \forall k (u_a \in X)} (o_{ak})),$$

again given that $p_a$ contains terms found in $Q$.

In a MOP, there is no single solution $X$ that optimizes all objectives simultaneously, but a set of trade-off candidates. The set of trade-off solutions, commonly known as the Pareto Front (PF), is often defined in terms of Pareto Optimality. That is, considering a maximization MOP with $n$ objectives: a solution $X^*$ is considered non-dominated or Pareto optimal with respect to another solution $Y$, iff $\forall i \in \{1, ..., n\}, X_i \geq Y_i \wedge \exists i \in \{1, ..., n\} : X_i > Y_i$, this is denoted as $X \succ Y$.

## 1.4 The SmartOpt Framework

In this section, we present the SmartOpt framework (see Figure 1.1) that is composed of three modules: i) *the Optimizer Module*, which identifies a set of non-dominated QRTs (i.e., Pareto Front) based on geolocated information and social data contributed by the crowd (i.e., a social community of smartphone users); ii) *the Decision Maker module*, which selects a non-dominated QRT $X$ based on some user-preference criteria from the Pareto Front; and iii) *the Search Module*, which propagates the QRT $X$ to the P2P network to retrieve the objects of interest.

### *1.4.1 The Optimizer Module*

The SmartOpt optimizer is founded on a MOEA, during which a population of candidate solutions (a.k.a. chromosomes), evolve into better solutions (w.r.t. the objective functions), by utilizing a set of operators (e.g., selection, crossover and mutation) that are inspired by natural evolution. The given application of operators is inherently stochastic, but applications to numerous domains such as bioinformatics, computational science, engineering, economics and other fields, have shown that MOEAs can be more effective to difficult multi-objective optimization problems when domain knowledge is incorporated to the operators [27]. In the context of SmartOpt, we introduce both domain expertise into our operators as well as utilize well-known operators that have been proven accurate over the years.

Specifically, we have implemented and specialized the MOEA/D framework [46], which is the state-of-the-art of the decompositional MOEAs and the winner of the Unconstrained Multi-Objective Evolutionary Algorithm competition in

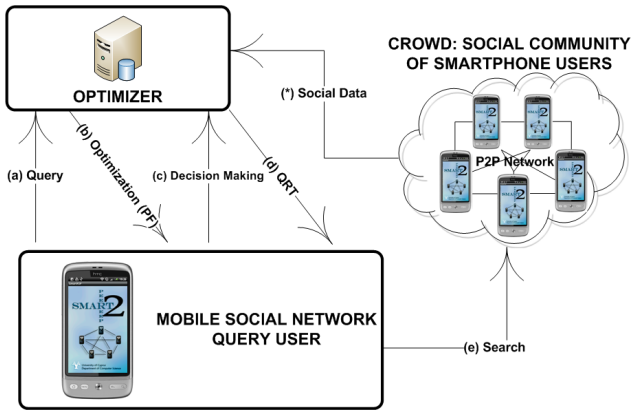



**Figure 1.1: The SmartOpt framework: (*) The crowd continuously contributes social data (i.e., text, images, videos etc.) to the optimizer. (a) A mobile social network user posts a query to the optimizer. (b) The optimizer obtains a set of non-dominated solutions (PF) and send it back to the user. (c) The user (decision maker) chooses a Pareto-optimal solution based on instant requirements and preferences. (d) The optimizer forwards the selected Pareto-optimal QRT to the user. (e) The user searches the P2P social network for objects of interest.**

the Congress of Evolutionary Computation, 2009. We initially proposed a tree-based encoding representation suitable for the MO-QRT problem and we then designed a MOEA/D composed of our M-tournament selection approach and the two-point crossover and random mutation genetic operators as originally proposed by Zhang and Li in [46]. Furthermore, we hybridized MOEA/D with a problem-specific repair heuristic for identifying infeasible solutions generated by the genetic operators and converting them to feasible. Note that the SmartOpt framework can adopt any MOEA as its Optimizer module (such as NSGA-II [14]) with minor modifications.

MOEA/D requires some pre-processing steps, which consists of representing a QRT and decomposing the problem into a set of scalar sub-problems, before executing its main part that is outlined in Algorithm 1.

**Representation:** In our approach, a solution[6] $X$ is a QRT with $|G'|$ active smartphone users that can participate in the resolution of $Q$. Without loss of generality, let $X$ be represented as a vector in which each index $i$ corresponds to a user $u_i$ and the value of that position corresponds to $u_i$'s parent. The root of the tree is the query user (for simplicity noted as $u_1$). A negative value $-1$ in any position indicates that the given user is not currently selected in the query routing tree $X$.

**Decomposition:** Initially, the MOP should be decomposed into $m$ subproblems by adopting any technique for aggregating functions, e.g., the Tchebycheff approach

[6]The terms *"solution"*, *"vector"* and *"QRT"* are utilized interchangeably.

**Algorithm 1** The SmartOpt Optimizer

**Input:**

- network parameters (e.g., $Q$, $P$, $U$, $G$);
- $m$ : population size and number of subproblems;
- $T$: neighborhood size;
- weight vectors $(w_j^1, ..., w_j^m)$, $j = 1,2,3$;
- the maximum number of generations, $gen_{max}$;

**Output:** set of non-dominated QRTs, known as the Pareto Front ($PF$).

**Step 0 (Setup):** Set $PF := \emptyset$; $gen := 0$; $IP_{gen} := \emptyset$;

**Step 1 (Initialization):** Uniformly randomly generate an initial set of QRTs $IP_0 = \{X^1, \cdots, X^m\}$, known as the initial internal population;

**Step 2:For** $i = 1, \ldots m$ **do**

**Step 2.1 (Genetic Operators):** Generate a new solution (i.e., QRT) $Y$ using the genetic operators.

**Step 2.2 (Local heuristic):** Apply a problem-specific repair heuristic on $Y$ to produce $Z$.

**Step 2.3 (Update Populations):** Use $Z$ to update $IP_{gen}$, $PF$ and the $T$ closest neighbor solutions of $Z$.

**Step 3 (Stopping criterion): If** stopping criterion is satisfied, i.e., $gen = gen_{max}$, **then** stop and output $PF$, **otherwise** $gen = gen + 1$, go to Step 2.

used here. In this case, the $i^{th}$ subproblem is in the form

$$maximize \quad g^i(X|w_j^i, z^*) = max\{w_j^i | f_j(X) - z_j^*|\} \tag{1.5}$$

where $f_j$, $j = 1,2,3$, are the objectives of our MOP formulated earlier in Subsection 1.3.2, $z^* = (z_1^*, z_2^*, z_3^*)$ is the reference point, i.e., the maximum objective value $z_j^* = max\{f_j(X) \in \Omega\}$ of each objective $f_j, j = 1,2,3$ and $\Omega$ is the decision space. For each Pareto-optimal solution $X^*$ there exists a weight vector $w$ such that $X^*$ is the optimal solution of (1.5) and each solution is a Pareto-optimal solution of the MOP in Subsection 1.3.2. For the remainder of this chapter, we consider a uniform spread of the weights $w_j^i$, which remain fixed for each subproblem $i$ for the whole evolution and $\sum_{j=1}^{3} w_j^i = 1$.

**Initialization Step 1:** In Step 1 of Algorithm 1, we adopt a random method to generate $m$ QRT solutions for the initial Internal Population (i.e., $IP_0$). Namely, a QRT solution $X$ is initiated by setting each smartphone user $u_i, i = 1 \ldots M$ as a parent. Then, mobile users $u_j, j = 1 \ldots M$ are uniformly randomly selected, and $u_i$ is set as $u_j$'s parent iff $i \neq j$ and $u_i$ is either the root or has already a parent. If $u_j$ has already a parent then we stop and we set as parent the user $u_{i+1}$. This continues until all users $u_i$ are set as parents once. Then, the $IP_{gen}$ is used to store the best QRT $X^i$ found for each subproblem $g^i$ during the search, i.e., at each generation $gen$.

**Genetic Operator Step 2.1:** The genetic operators (i.e., selection, crossover and mutation) are then invoked on $IP$ for offspring reproduction, i.e., generate a new QRT solution $Y^i$ for each subproblem $g^i, i = 1 \ldots m$. The following steps summarize the details of each operator:

- **Selection:** We utilize our M-Tournament tree selection [28] for selecting the M closest individual QRTs from the neighborhood of each subproblem $g^i$, which are then added in a tournament and the two QRTs with the best fitness are selected as parents for crossover. The given selection operator allows to easily adjust the selection pressure, is simple to implement and works in constant time.

- **Crossover (a.k.a. reproduction or recombination):** allows our algorithm to generate new solutions that share many of the characteristics found in parents, yet are different QRTs. In particular, the 2*x-point tree crossover* operator takes as an input two parent QRTs, $Pr_1$ and $Pr_2$, and subsequently generates two new QRTs $O_1$ and $O_2$, the offspring. The best offspring $O$ is finally selected as follows:
  - Two crossover points $x_1$ and $x_2$ are uniformly randomly selected from numbers 1 to M-1, where $x_1 < x_2$.
  - The pieces of the parents $Pr_1$ and $Pr_2$ falling within $x_1$ and $x_2$ are exchanged to produce two offspring, e.g., $O_1, O_2$.
  - The best offspring $O$ is then forwarded to the mutation operator, where $O = O_1$ if $g^i(O_1, w^i_j) < g^i(O_2, w^i_j)$ and $O = O_2$ otherwise.

- **Swap Mutation:** modifies an offspring $O$ to a solution $Y$ with a probability $r_m$ by uniformly randomly swapping the values of two indexes $j, z$ in $O$.

**Repair Step 2.2:** In Step 2.2 of Algorithm 1, a problem-specific local search heuristic checks a QRT solution $Y$ and calculates a QRT $Z$ iff:

- **Case #1:** there is a disconnected user $u_i$ in QRT $Y$ (i.e., $u_i$ with or without children that does not have a parent);

- **Case #2:** two or more user ids $i$ of user $u_i$ are the same in QRT $Y$;

- **Case #3:** there is an infinite loop in QRT $Y$;

In all cases, the solution $Y$ is considered infeasible. An infeasible solution can be generated during reproduction (i.e., genetic operation). A local heuristic repairs the QRT solution $Y$ to $Z$ by: uniformly randomly generating a parent for the disconnected user $u_i$ in Case #1, replacing the duplicate user $u_i$ with another user $u_j$ in Case #2, breaking the loop by connecting a random user of the loop with another user out of the loop in Case #3. The repair heuristic continuously repairs solution $Y$ until it does not fall in any of the Cases #1, #2 or #3. In particular, Step 2.2 is repeated

after each repair to check for further infeasibility, i.e., wether a new discontinuity, a duplication or a loop appears in the solution. If this is the case then the solution is repaired as before, otherwise the feasible Solution $Z$ is used to update the populations of MOEA/D as follows.

**Population Update Step 2.3:** In Step 2.3, the update phase of Algorithm 1 is processed in three steps. (1) Update $IP$, $IP/\{X^i\}$ and $IP \cup \{Z^i\}$ if $g_i(Z^i|w^i,z^*) < g^i(X^i|w^i,z^*)$, otherwise $X^i$ remains in $IP$. (2) Update the neighborhood of $Z^i$, i.e., the solutions of the $T$ closest subproblems of $i$ in terms of their weights $\{w^1,\cdots,w^m\}$ are updated. If $g^j(Z^i|w^j,z^*) < g^j(X^j|w^j,z^*)$, then $IP/\{X^j\}$ and $IP \cup \{Z^i\}$, otherwise $X^j$ remains in $IP$, where $j \in \{1,...,T\}$. (3) Update the Pareto Front $(PF)$, which stores all the non-dominated solutions found so far during the search. $PF = PF \cup \{Z^i\}$ if $Z^i$ is not dominated by any solution $X^j \in PF$ and $PF = PF/\{X^j\}$, for all $X^j$ dominated by $Z^i$. The search stops after a pre-defined number of generations, $gen_{max}$.

### 1.4.2 The Decision Maker Module

In the decision making module, the query user $u_j$ is prompt to decide its preference in terms of *Time* (i.e., Objective 2 calculated by Equation 1.3) and *Recall* (i.e., Objective 3 calculated by Equation 1.4) of the query response to receive from the Smartphone P2P Network. The decision maker module of SmartOpt then finds the QRT solution $X$ of the PF that best satisfies the user's decision and it is also the most *Energy* efficient (i.e., Objective 1 calculated by Equation 1.2) at the same time. In this way, $u_j$ is responsible to decide the user-oriented objective values (i.e., time and recall) and the decision maker module the system oriented objective value (i.e., energy), since it is assumed that Smartphone users will not be interested in conserving the overall system energy of the network. Here it is important to notice that we proposed a *posterior* approach for giving the opportunity to the user to visually choose a QRT, from the set of Pareto-optimal QRTs obtained by the MOEA/D, based on instant requirements and preferences; instead of choosing a QRT *a priori*, without any knowledge on the obtained Pareto Front, or *interactively* that consumes additional time and energy from the Smartphone users.

### 1.4.3 The P2P Search Module

In the final phase, the query user $u_1$ receives the Pareto-optimal tree $X$ from the decision maker module of SmartOpt and proceeds with a recursive execution of Algorithm 2 on all smartphone devices participating in the tree $X$. Recall that $X$ is a vector in which each index $i$ corresponds to a user $u_i$ (IP address and port) and the value of that position corresponds to $u_i$'s parent (IP address and port).

As soon as a smartphone device $u_j$ receives $Q$ it creates a set $O_j$ of all objects $o_{ji}$ that satisfy $Q$ (line 4). Immediately then, $u_j$ transmits these objects to the query user $u_1$ (line 6) using the most efficient communication technology (i.e., bluetooth, 3G). In the final step, the smartphone device $u_j$ forwards $Q$ to all its child nodes (lines 8-14). This is done by checking each parent entry in $X$ with its own (line 11).

**Algorithm 2** : **Search Phase**

**Input:** The Query User $u_1$, A Pareto-optimal Query Routing Tree $X$, A Query $Q$
**Output:** A set of objects $O_j = \{o_{j1} \dots o_{jk}\}$.

```
1:  procedure SEARCH(u1,X,Q)
2:      if (j ≠ 1) then
3:          //Step 1: Find a set of local objects Oj that satisfy Q
4:          Oj = ⋃∀i oji, satisfy(oji, Q)
5:          //Step 2: Send local objects Oj to query user u1
6:          Send(Oj, u1);
7:      end if
8:      //Step 3: Forward query u1 to all children smartphone devices
9:      for i = 1 to |X| do
10:         //if j is the parent of i
11:         if (X[i] == j) then
12:             Search(u1, X, Q);
13:         end if
14:     end for
15: end procedure
```

If a match $u_i$ is found, $u_j$ transmits $Q$ and $X$ to $u_i$ (line 12). This process executes recursively until all smartphone devices in $X$ receive the query.

## 1.5 The Smartphone Prototype System

In this section, we provide an overview of our prototype system, coined SmartP2P[7], developed for the ubiquitous Android Operating System to demonstrate the applicability of the SmartOpt framework. We particularly evaluated SmartP2P on our programming cloud of Smartphones, coined SmartLab testbed.

Our client-side software is developed around the SDK Tools r12 of Android 2.2 and its installation package (i.e., APK) has a size of 327KB. Our code is written in JAVA and consists of around 7500 lines of code. In particular our server-code (i.e., optimizer) uses 5000LOC and runs over JDK 6 and Ubuntu Linux, our smartphone code uses 1600 LOC plus 250 lines of XML elements. The server side also includes a Microsoft SQL server R2 and utilizes the JMATH-PLOT package for drawing the Pareto Front images.

The Graphical User Interface of our system provides a primitive interface for a user to query the active users in the community. Initially, the SmartP2P allows a user to formulate a query in order to find objects of interest. Then the user is provided a group of algorithmic choices including (i) two simple distributed choices,

[7] Available at: http://smartp2p.cs.ucy.ac.cy/

i.e., *Random Search* and *Breadth-First Search*, as well as (ii) two MOO choices, i.e., the *MOEA based on Decomposition (MOEA/D)* and the *Non-Dominated Sorting Genetic Algorithm II (NSGA-II)*. The user selects an algorithm and the optimizer calculates a QRT in case (i) or a Pareto Front in case (ii). In both cases, the result is returned to the query user. Note that alternative approaches can be easily utilized by our framework with minor modifications. For example, the results can be aggregated at each node and returned back to the user following a reverse path of the same QRT, which is called an aggregation tree [15].

The decision maker is only enabled when the query user selects an algorithm from case (ii) to perform the search. In this case, the Pareto Front is forwarded and displayed to the query user. Then the query user makes use of a slide bar below the Pareto Front image to set a desired level of time and recall of the search to be initiated. Note that if the user does not choose a desired level of those two objectives, the solution with the minimum energy consumption is automatically chosen. In any case, the decision maker finds the QRT that is closer to the user's choice and downloads it from the optimizer to the user's smartphone. Finally, the query user initiates the search using a P2P protocol and the results of the search as well as the selected QRT are both displayed on the user's smartphone.

The peer-to-peer protocol that lies at the foundation of our prototype system is a text-based protocol, as opposed to a binary protocol, for portability (i.e., endianness) reasons. We also did not chose an XML-based protocol implementation for performance reasons (i.e., minimize annotations.) At a high level, a smartphone user, denoted as *QP*, starts out by registering its obfuscated location (e.g., vector of intercepted cell tower IDs or MAC addresses of WiFi access points) to a well-known host-cache (i.e., the SmartOpt SERVER in our case.) After this exchange, the client is considered to be "connected" to the service for a pre-specified amount of time (i.e., $k$ seconds in our setting, after which the lease can be renewed). Now assume that a "connected" client *QP* wants to query the active nodes in the network. *QP* first issues a GET command to the SERVER in order to obtain a tree $T$ that captures its optimization criterions (with respect to time, energy and recall.) Notice that the SERVER is already aware of the social graph and other statistics used in the optimization process. The issued command is supplemented by a token returned during the registration. The returned tree is serialized in the following format ``NodeIP:NodePort(ParentIP:ParentPort)'', with -1 denoting no-parent but is shortened below for ease of exposition. Once $T$ is obtained by *QP*, *QP* connects to `P0=root(T)` and submits its query $Q$ (i.e., {k1,k2,...,kn}), its HOME_ADDR address (i.e., IP:PORT) as well as a hop count parameter. P0 then forwards these parameters to its own children (e.g., P10 and P15), in a recursive manner for N levels (using a predetermined Time-To-Live (TTL) value enforced by the hop count.) Any peer receiving $Q$, conducts a local search and informs *QP* directly on HOME_ADDR about possible answers. If a peer in $T$ is not responding for whatever reason the given branch of the tree is disregarded. The fact that the query tree is optimized for minimum delay, minimum energy and maximum recall provides an advantage of our approach compared to other approaches for unstructured P2P search, like Breadth-First-Search, Random Walks [32], as this is presented in our experimental evaluation.

In particular, we found that the MO-QRT structure can greatly reduce the number of search nodes, by exploiting meta-relations captured in the social networking graph and the user interests matrix.

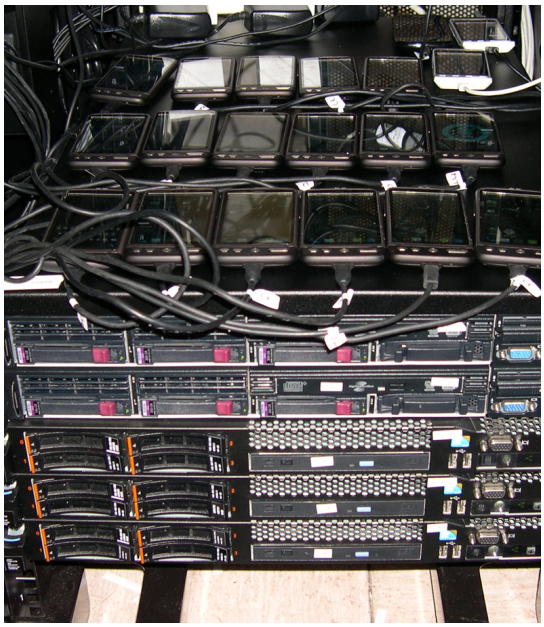


**Figure 1.2: Subset of the SmartLab Programming Cloud: Smartphone fleet connected locally to our datacenter. There are additional devices connected both remotely and wirelessly to our datacenter.**[9]

### *1.5.1 The SmartLab Programming Cloud*

Experimenting with a large number of devices can be a tedious process as each device needs to be connected to the programming station, the application needs to be installed separately and the operator needs to manually launch the instances on each device and collect the results. In order to overcome the inherent problems of this setup we have implemented *SmartLab* [26, 30], an innovative programming cloud of approximately 40+ Android smartphones and tablets, which is deployed at the University of Cyprus (see Figure 1.2). SmartLab is inspired by both PlanetLab [7] and MoteLab [43]. Its intuitive web-based interface is easy to use and provides the ability to reserve and use Android devices for a desired amount of time. Users are able to

[9] Available at: http://smartlab.cs.ucy.ac.cy/

reboot, list, transfer and remove files, change Android device settings by using the interactive Android Debugging (ADB) shell session. Additionally, registered users can upload and install executable APK files on their reserved Android devices simultaneously. The SmartLab users are also able to extract application data, output and results automatically from all reserved devices, take screenshots as well as watch the display of all reserved devices during runtime. Users are also granted access to log files for error and exception handling.

*SmartLab* implements several modes of user interaction with connected devices using either *Websocket-based interactions* (for high-rate utilities) or *AJAX-based interactions* (for low-rate utilities). In particular, SmartLab supports: i) *Remote Control Terminals (RCT)*, a Websocket-based remote screen terminal that mimics touchscreen clicks and gestures among other functionalities; ii) *Remote File Management (RFM)*, an AJAX-based terminal that allows users to push and pull files to the devices; iii) *Remote Shells (RS)*, a Websocket-based shell developed in-house enabling a wide variety of UNIX commands issued to the Android Linux kernels of allocated devices; iv) *Remote Scripting Environment (RSE)*, a Websocket-based RCT recording utility that translates user clicks into automation scripts for repetitive tasks; v) *Remote Debug Tools (RDT)*, a Websocket-based debugging extension to the debugging information available through the Android Debug Bridge (ADB); and vi) *Remote Mockups (RM)*, a Websocket-based mockup subsystem for feeding ARDs and AVDs with GPS or sensor data traces encoded in XML, in cases we want to carry out trace-driven experiments with those measurements. A more detailed description of SmartLab can be found in [30].

## 1.6 SmartP2P Prototype Evaluation On SmartLab

### *1.6.1 Experimental Setup*

Our experimental methodology relies on a *Trace-driven Real Deployment*, during which we deploy our SmartP2P real prototype system implemented in Android over up to 138 users using SmartLab and the traces described next.

**Datasets and Queries:** In our experimental studies, we have constructed two mobile social scenarios from the following three real datasets:

*GeoLife* [47] (*mobility*): This real dataset by Microsoft Research Asia includes 1,100 trajectories of a human moving in the city of Beijing over a life span of two years (2007-2009). The average length of each trajectory is $190,110 \pm 126,590$ points, while the maximum trajectory length is 699,600 points. Notice that 95% of the GeoLife dataset refers to a granularity of 1 sample every 2-5 seconds or every 5-10 meters.

*DBLP* [10] (*social*): This real dataset by the DBLP Computer Science Bibliography website, includes over 1.4 million publications in XML format. In particular, the dataset records the paper titles, paper urls, co-authors, links between papers and

authors and other useful semantics. In order to map this dataset to our problem, we assume that each object is an author's paper. We also assume that each object is "tagged" by the keywords found in the paper title.

*Pics 'n' Trails* [12, 11] (*mobility and social*): This is a real dataset composed of around 75 GPS traces of a user moving in Tokyo, Japan during 2007 and a collection of geotagged photos taken along with a short description. In particular, the dataset is comprised of 4179 photos in Japan as well as trajectories with a granularity of 1 sample every 10-15 seconds.

In order to link the above datasets we have constructed two mobile social scenarios:

*Mobile-Social Scenario 1 (MSS-1)*: uses the DBLP social dataset and GeoLife mobility dataset. The DBLP dataset is used to construct a social graph $G$ of authors that are related based on their research interests (i.e., keywords of their articles' titles) as well as their co-authorships that are attributes of the DBLP dataset. Then we have mapped each DBLP author to a trajectory of the Geolife dataset. Particularly, we have extracted 1,100 authors from the DBLP dataset and we have mapped them to the 1,100 trajectories of the Geolife dataset using a 1:1 correspondence. This resulted in a social graph with 1,100 mobile DBLP authors moving in the city of Beijing, China.

*Mobile-Social Scenario 2 (MSS-2)*: uses the Pics 'n' Trails social and mobility dataset. The Pics 'n' Trails dataset is initially used to construct a social graph $G$ of 75 users that are connected based on their interest in taking photos of sightseeing in Japan (i.e., similar tags on their photos taken). Each user is, therefore, carrying a random number of photos tagged with a short description that describes a particular sightseeing in Japan and is associated with a GPS trajectory from the Pics 'n' Trails dataset. This resulted in a social graph with mobile users that carry photos with tags and move in the city of Tokyo, Japan.

In our experiments, we utilize the following three queries:

```
-- Query 1
SELECT S.title, S.url
FROM SmartphoneUsers S, Query Q
WHERE (distance(S.x,S.y,Q.x,Q.y) < 10 KM)
      AND S.Title LIKE '%optimization%';

-- Query 2
SELECT S.title, S.url
FROM SmartphoneUsers S, Query Q
WHERE (distance(S.x,S.y,Q.x,Q.y) < 10 KM)
      AND S.Title LIKE '%networks%';

-- Query 3
SELECT S.title, S.url
FROM SmartphoneUsers S, Query Q
WHERE (distance(S.x,S.y,Q.x,Q.y) < 10 KM)
      AND S.Title LIKE '%Kyoto%';
```

where "S.x,S.y" represent the $(x, y)$ coordinates of a Smartphone user in $S$ and "Q.x,Q.y" represent the $(x, y)$ coordinates of the query user.
We execute nine different test instances using the two Mobile-Social Scenarios and the three queries, Query 1, Query 2 and Query 3 as shown on Table 1.2. Our scenarios are executed for various time periods (i.e., during the morning, during noon and during night), in order to capture different mobility patterns that are inherent in the GeoLife and Pics 'n' Trails datasets.

**Table 1.2: Experimental Execution Scenarios and Test Instances.**

| Scenario | Test# | $Q$ | Time | $G'$ | # of Objects | Relevant Objs. |
|---|---|---|---|---|---|---|
| MSS-1 | T1 | Query1 | Morning | 49 | 3877 | 82 |
| | T2 | Query1 | Noon | 58 | 5504 | 73 |
| | T3 | Query1 | Night | 95 | 8884 | 121 |
| | T4 | Query2 | Morning | 49 | 3877 | 319 |
| | T5 | Query2 | Noon | 58 | 5504 | 477 |
| | T6 | Query2 | Night | 95 | 8884 | 695 |
| MSS-2 | T7 | Query3 | Morning | 26 | 744 | 43 |
| | T8 | Query3 | Noon | 66 | 1877 | 115 |
| | T9 | Query3 | Night | 47 | 1456 | 92 |

**Search Algorithms:** We have implemented and evaluated the following search algorithms: i) the *Centralized Search* algorithm (*CS*), presented in Section 1.3.1, which collects all data and metadata tags at the centralized query processor prior query execution; ii) the *Distributed Breadth-First-Search Search* (*BFS*), which conducts a distributed search using a random tree that is generated with a BFS process which visits all nodes in the network, as presented in Section 1.3.1; iii) the *Random Walker (RW) Search* [32], which conducts a distributed search using a list structure that captures a randomly chosen neighbor on each step but that eventually visits all nodes in the network and iv) the SmartOpt Search, which conducts a distributed search using an optimized QRT obtained from the application of ideas presented in this paper. SmartOpt trees are inherently smaller in size, than their other alternatives, as this structure visits with a higher probability the nodes having more relevant objects (i.e., based on the social graph and the metadata stored for each node.) We evaluate the search algorithms using the following metrics: *Time*, *Energy* and *Recall*, as these were defined in Section 1.3.2 by using wall clock time along with the PowerTutor power (energy) measuring tool by the University of Michigan, USA. In particular, PowerTutor is a component power management and activity state introspection based tool that uses an automated power model construction technique for accurate online power estimation in Android.

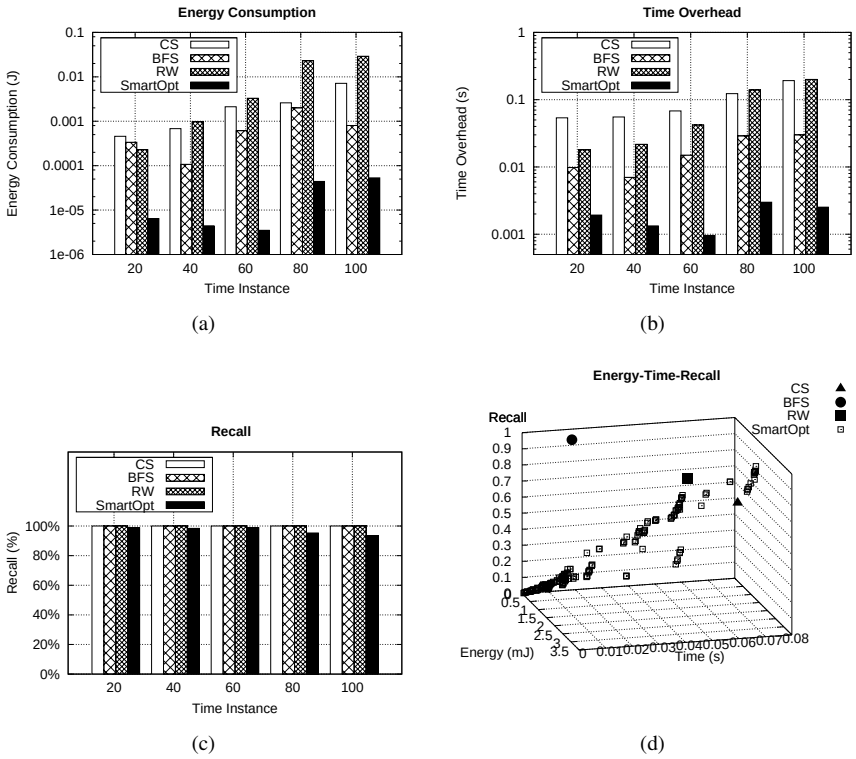



**Figure 1.3: Evaluation of the CS, BFS, RW and SmartOpt search algorithms using the energy, time and recall performance. The bottom/right figure shows SmartOpt compared to the solutions of CS and BFS in the objective space at timestamp $ts$=70 of Mobile-Social Scenario-1 (MSS-1).**

### *1.6.2 Experimental Results*

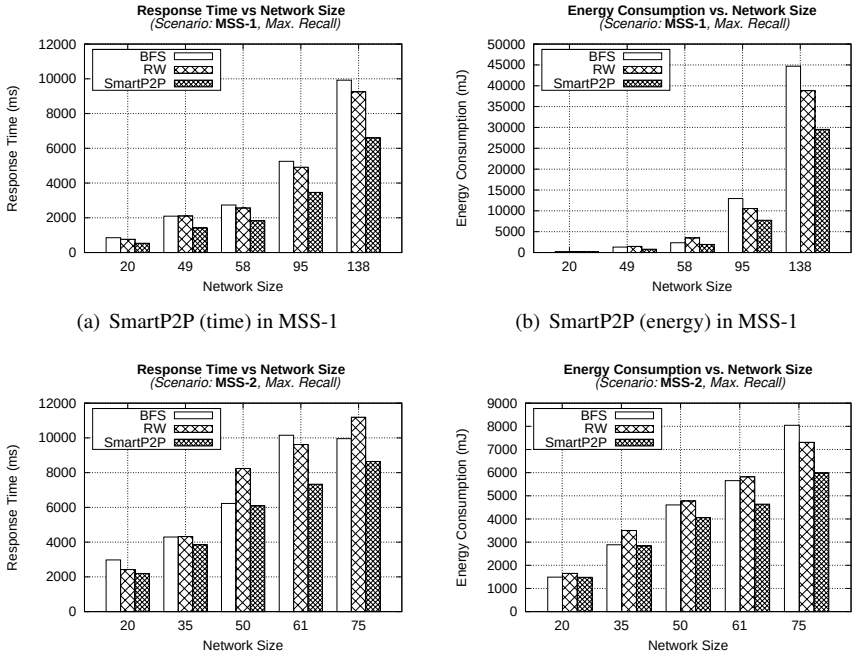
Initially, we evaluate the performance of the SmartOpt against the CS, BFS and RW on 100 consecutive timestamps in Mobile-Social scenario 1 (GeoLife+DBLP) using our model-driven simulator. At each *timestamp (ts)* we compare the energy consumption, time overhead and recall of all algorithms.

Figure 1.3 illustrates the results of our experiment for all performance metrics. In Figure 1.3 (top/left) we observe that the energy consumption of SmartOpt is one to two orders of magnitude smaller than its competitor CS, BFS and RW in all timestamps. BFS seems more efficient than CS as it does not communicate all metadata to the centralized query node. On the other hand, RW is worse than all approaches as the sequential visit to all nodes in the network drains considerable energy (i.e., in each communication only 1 message is sent, as opposed to the rest of the techniques that communicate with several nodes in a single round.) Similar observations apply


**Figure 1.4: A screenshot of SmartP2P on SmartLab using real-time screen capture**

for Figure 1.3 (top/right) where we demonstrate the time overhead for all algorithms. This happens as the energy is proportional to the time interval the communication transceiver is in active mode. Moreover in Figure 1.3 (bottom/left), we show that the recall rate for the SmartOpt framework is close to 95%, consistently. The slight decrease of recall with respect to the increase of timestamp is due to the variations in the network. In particular, we noticed an increase of the number of users in the last timestamps, which increase the complexity of the search space, making it harder to find the solutions with maximum recall without increasing the computational effort of the algorithm. In conclusion, the SmartOpt framework consumes less time and less energy and it is able to identify most expected answers at the same time. In Figure 1.3 (bottom,right), we demonstrate the results for a single timestamp ($ts$=70) for all algorithms. The various solutions generated by SmartOpt optimizer are represented by open squares. The single solutions supplied by the CS, RW and BFS algorithms are represented by a solid triangle, a solid square and a solid circle, respectively. We observe that the solution provided by the CS algorithm is the worst w.r.t. BFS and RW in all three performance metrics. However, the CS algorithm demonstrates higher recall (10%) than all solutions provided by the SmartOpt framework. This occurs because, CS dictates global participation by all smart objects in the network (i.e., all smart objects forward their results to the query user). However, this has a significantly negative impact on both energy and performance. Specifically, compared to

the SmartOpt best solutions, CS, BFS and RW feature an increase of two orders of magnitude in energy and one order of magnitude in time.




(a) SmartP2P (time) in MSS-1

(b) SmartP2P (energy) in MSS-1

(c) SmartP2P (time) in MSS-2

(d) SmartP2P (energy) in MSS-2

**Figure 1.5: Evaluating our SmartP2P prototype system in Android using the SmartLab Testbed for different network sizes in both Mobile Social Scenarios 1 (GeoLife+DBLP) and Mobile Social Scenarios 2 (Pics 'n' Trails) with respect to time and energy.**

Finally, we evaluate our SmartP2P prototype Android implementation, presented in Section 1.5, over our distributed SmartLab infrastructure as illustrated in Figure 1.4. For the evaluation, we focus only on the distributed search algorithms: *BFS, RW* and *SmartP2P*. We present the query response time, measured in seconds and energy consumption, measured with PowerTutor in Watts and presented in Joules. We utilize five different network sizes in Mobile Social Scenario 1 (MSS-1): 20, 49, 58, 95 and 138 and five different network sizes in Mobile Social Scenario 2 (MSS-2): 20, 35, 50, 61 and 75 to show the scalability aspects of the different search algorithms. In order to accommodate these instances over a physical infrastructure, which was considerably smaller (i.e., 40+ smartphones), we had to run several instances on each of the available physical smartphones (using separate socket servers). For example,

an HTC Desire smartphone could easily host tens of instances without any particular performance penalty (recall that these are 1GHz smartphones with 512MB of RAM) while the lower-end HTC Hero devices (with a 512MHz processor and 288MB of RAM) were excluded from our experiments as they were considerably slower and could not host tens of instances. For practical reasons we did not utilize the Bluetooth connection between instances and considered as a local link the socket communication of instances on the same physical smartphone host.

Figure 1.5 (a), presents the response time for the different executions given that all algorithms obtain the complete result set (i.e., maximum recall) in mobile-social scenario 1. We observe that SmartP2P obtains the expected answer in little anywhere between 1.5 seconds and 6 seconds while both BFS and RW require in many cases as much as 10 seconds. The competitive advantage of SmartP2P over both BFS and RW is considerably better for larger network sizes. This is very encouraging as Smartphone Networks might consist of thousands of nodes in an area of interest (i.e., within the spatial boundary of a query.) Figure 1.5 (b), presents the energy consumption in mobile-social scenario 1 as this was measured by PowerTutor (i.e., only the energy related to CPU and Networking without taking into account costs related to LCD utilization.) The given figure shows that SmartP2P manages to locate the complete answer set utilizing 25% and 33% less energy than RW and BFS, respectively. We also noticed that by bringing down the recall expectation to $\approx$80%, would allow us to obtain great energy savings considerably faster ($\approx$50%). Similarly, Figures 1.5 (c) and (d) show that the SmartP2P search approach is more efficient than the BFS and the RW in MSS-2 as well. In particular, SmartP2P conserves up to 30% time and 25% energy for max recall. Here it is important to notice that the relative performance of the algorithms in terms of energy is the same in both the simulations and the testbed experiments (i.e., SmartP2P consumes less energy than both BWS and RW), although the actual values in the real setting is lower than that of the simulations. This is mainly due the fact that the algorithms are implemented in different computational platforms using different network parameter settings (e.g., the network sizes) with additional constraints of the physical infrastructure.

## 1.7 Conclusions And Potential Applications

In this book chapter, we present the SmartOpt framework for searching objects captured by the users in a mobile social community. Our framework, is founded on an *in-situ* data storage model and searches then take place over the MO-QRT structure we present in this chapter. Our structure concurrently optimizes several conflicting objectives (i.e., energy, time and recall). Our experimental assessment uses a trace-driven experimental methodology with mobility and social patterns derived by Microsoft's Geolife project, DBLP and Pics 'n' Trails using our real SmartP2P system developed in Android and deployed over our SmartLab testbed of 40+ smartphone devices. Our study reveals that our framework yields high query recall rates and consumes less energy than its competitors. Additionally, our study reveals that the MO-QRT structure is highly appropriate for content search and retrieval in Mobile Social Networks.

The *SmartOpt* framework can be easily adapted and used for several real-life crowdsourcing applications. For example, it can be used as a recommender system where the social crowd of smartphones generates instant information for certain events/places. A querying user can then use *SmartOpt* to generate a query routing tree to retrieve accurate and spatially proximate information about an event/place of interest (e.g. concert, football match, hospital, police station). Furthermore, *SmartOpt* can be utilized for crowdsourcing call assignment tasks, in which the crowd shares its area of expertise as well as technical skills and a query user may retrieve an optimal tree of close-by users that can perform the task more quickly and efficiently. Finally, it can be used as an instant emergency/news system. For example, considering recent disasters such as the Sandy hurricane in New York or the earthquake (and consequently the tsunami) in Japan, the users would have preferred querying and retrieving instant information from the crowd near the areas of disaster than reading arbitrary (and maybe out-dated) information on the web.

# References


[1] S. M. Allen, G. Colombo, and R. M. Whitaker. Cooperation through self-similar social networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 5(1), 2010.

[2] P. Andreou, D. Zeinalipour-Yazti, A. Pamboris, P.K. Chrysanthis, and G. Samaras. Optimized query routing trees for wireless sensor networks. *Information Systems (InfoSys)*, 36(2):267–291, 2011.

[3] M. Azizyan, I. Constandache, and R.-R. Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *MobiCom*, 2009.

[4] A.T. Campbell, S.B. Eisenman, N.D. Lane, E. Miluzzo, R.A. Peterson, X. Zheng H. Lu, M. Musolesi, K. Fodor, and G.S. Ahn. The rise of people-centric sensing. *In IEEE Internet Computing*, 12(4):12–21, July-August 2008.

[5] G. Chatzimiloudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 2012.

[6] S.-K. Chen and P.-C. Wang. Design and implementation of an anycast services discovery in mobile ad hoc networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 6(1):2, 2011.

[7] B. N. Chun, D. E. Culler, T. Roscoe, A. C. Bavier, L. L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *Computer Communication Review*, 33(3):3–12, 2003.

[8] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 5. Kluwer Academic Publishers, 2002.

[9] T. Das, P. Mohan, V.N. Padmanabhan, R. Ramjee, and A. Sharma. Prism: platform for remote sensing using smartphones. In *MobiSys*, 2010.

[10] DBLP. DBLP Computer Science Bibliography, http://dblp.uni-trier.de/xml/, 2010.

[11] G. C. de Silva and K. Aizawa. Retrieving multimedia travel stories using location data and spatial queries. In *The 17th ACM International Conference on Multimedia*, pages 785–788. ACM, 2009.

[12] G. C. de Silva, T. Yamasaki, and K. Aizawa. Sketch-based spatial queries for retrieving human locomotion patterns from continuously archived gps data. *IEEE Trans. on Multimedia*, 11(7):156–166, 2009.

[13] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley and Sons, 2002.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[15] A. Deligiannakis, Y. Kotidis, V. Stoumpos, and A. Delis. Building efficient aggregation trees for sensor network event-monitoring queries. In *GeoSensor Networks*, volume 5659 of *Lecture Notes in Computer Science*, pages 63–76. Springer Berlin Heidelberg, 2009.

[16] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Seop-Ahn, and A.T. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN'09)*, 6(1), December 2009.

[17] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys*, pages 29–39, 2008.

[18] A. Gahng-Seop, M. Musolesi, H. Lu, R. Olfati-Saber, and A.T. Campbell. Metrotrack: Predictive tracking of mobile events using mobile phones. In *DCOSS*, pages 230–243, 2010.

[19] D. Geiger, M. Rosemann, and E. Fielt. Crowdsourcing information systems : a systems theory perspective. In *22nd Australasian Conference on Information Systems (ACIS'11)*.

[20] Gnutella. Gnutella peer-to-peer network, 14 March 2000. http://gnutella.wego.com.

[21] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov. TCP over Second (2.5G) and Third (3G) Generation Wireless Networks. RFC 3481 (Best Current Practice), February 2003.

[22] J. Jia, J. Chen, G. Chang, Y. Wen, and J. Song. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. *Computers and Mathematics with Applications*, 57(11–12):1767–1775, 2009.

[23] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *11th International Conference on Information and Knowledge Management (CIKM'02)*, pages 300–307, McLean, Virginia, USA, 2002.

[24] Y. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wirel. Netw.*, 6(4):307–321, 2000.

[25] A. Konstantinidis, C. Aplitsiotis, and D. Zeinalipour-Yazti. Multi-objective query optimization in smartphone social networks. In *12th International Conference on Mobile Data Management (MDM'11)*.

[26] A. Konstantinidis, C. Costa, G. Larkou, and D. Zeinalipour-Yazti. Demo: A programming cloud of smartphones. In *10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12)*.

[27] A. Konstantinidis and K. Yang. Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific MOEA/D. *Applied Soft Computing*, 11(6):4117–4134, 2011.

[28] A. Konstantinidis, K. Yang, Q. Zhang, and D. Zeinalipour-Yazti. A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks. *New Network Paradigms, Elsevier Computer Networks*, 54:960–976, 2010.

[29] C. Laoudias, G. Constantinou, M. Constantinides, S. Nicolaou, D. Zeinalipour-Yazti, and C. G. Panayiotou. The airplace indoor positioning platform for android smartphones. In *13th International Conference on Mobile Data Management (MDM'12)*.

[30] G. Larkou, C. Costa, P. Andreou, A. Konstantinidis, and D. Zeinalipour-Yazti. Managing smartphone testbeds with smartlab. In *Proceedings of the 27th USENIX Large Installation System Administration Conference*, LISA'13, 2013.

[31] J. Ledlie, B. Odero, E. Minkov, I. Kiss, and J. Polifroni. Crowd translator: on building localized speech recognizers through micropayments. *ACM SIGOPS'10 Operating Systems Review*.

[32] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th international conference on Supercomputing (ICS'02)*, pages 84–95, New York, USA, 2002.

[33] S. Matyas, C. Matyas, C. Schlieder, P. Kiefer, H. Mitarai, and M. Kamata. Designing location-based mobile games with a purpose: collecting geospatial data with cityexplorer. In *International Conference on Advances in Computer Entertainment Technology*, 2008.

[34] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. *Data Engineering, International Conference on*, 0:633, 2003.

[35] A. Quinn and B. B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Annual Conference on Human Factors in Computing Systems (CHI'11)*.

[36] M. Ra, J. Paek, A. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely. Energy-delay tradeoffs in smartphone applications. In *MobiSys*, pages 255–270, 2010.

[37] R. Rajagopalan, C. K. Mohan, P. K. Varshney, and K. Mehrotra. Multi-objective mobile agent routing in wireless sensor networks. In *Proc. IEEE CEC'05*, Edinburgh, Scotland, September 2005.

[38] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *IPSN*, pages 105–116, 2010.

[39] T. Repantis and V. Kalogeraki. Data dissemination in mobile peer-to-peer networks. In *6th International Conference on Mobile Data Management (MDM'05)*, pages 211–219, Ayia Napa, Cyprus, 2005.

[40] M. Stevens and E. D. Hondt. Crowdsourcing of pollution data using smartphones. *Ubiquitous Computing (UbiComp'10)*.

[41] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98, New York, NY, USA, 2009. ACM.

[42] H. Tomiyasu, T. Maekawa, T. Hara, and S. Nishio. Profile-based query routing in a mobile social network. In *Mobile Data Management, 2006. MDM 2006. 7th International Conference on*, pages 105 – 105, May 2006.

[43] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 483 – 488, april 2005.

[44] B. Xu, O. Wolfson, and C. Naiman. Machine learning in disruption-tolerant manets. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(4), 2009.

[45] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos. Exploiting locality for scalable information retrieval in peer-to-peer systems. *Information Systems (InfoSys), Elsevier*, 30(4):277–298, 2005.

[46] Q. Zhang and H. Li. MOEA/D: A multi-objective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[47] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW*, 2008.