Distrib Parallel Databases (2013) 31:115–149
DOI 10.1007/s10619-012-7108-0

Intelligent search in social communities of smartphone users

Andreas Konstantinidis · Demetrios Zeinalipour-Yazti · Panayiotis Andreou · George Samaras · Panos K. Chrysanthis

Published online: 11 September 2012

Abstract Social communities of smartphone users have recently gained significant interest due to their wide social penetration. The applications in this domain, however, currently rely on centralized or cloud-like architectures for data sharing and searching tasks, introducing both data-disclosure and performance concerns. In this paper, we present a distributed search architecture for intelligent search of objects in a mobile social community. Our framework, coined *SmartOpt*, is founded on an *in-situ* data storage model, where captured objects remain local on smartphones and searches then take place over an intelligent multi-objective lookup structure we compute dynamically. Our *MO-QRT* structure optimizes several conflicting objectives, using a multi-objective evolutionary algorithm that calculates a diverse set of high quality non-dominated solutions in a single run. Then a decision-making subsystem is utilized to tune the retrieval preferences of the query user. We assess our ideas both using trace-driven experiments with mobility and social patterns derived by Microsoft's GeoLife project, DBLP and Pics 'n' Trails but also using our real Android *SmartP2P* (http://smartp2p.cs.ucy.ac.cy/) system deployed over our *Smart-Lab* (http://smartlab.cs.ucy.ac.cy/) testbed of 40+ smartphones. Our study reveals that SmartOpt yields high query recall rates of 95 %, with one order of magnitude less time and two orders of magnitude less energy than its competitors.

Keywords Intelligent search · Peer to peer · Evolutionary computation · Multi-objective optimization · Smartphones · Social networks

Communicated by: Dipanjan Chakraborty.

A. Konstantinidis (✉) · D. Zeinalipour-Yazti · P. Andreou · G. Samaras
Department of Computer Science, University of Cyprus, Nicosia, Cyprus
e-mail: akonstan@cs.ucy.ac.cy

P.K. Chrysanthis
Department of Computer Science, University of Pittsburgh, Pittsburgh, USA

1 Introduction

The widespread deployment of smartphone devices and the advent of social networks have brought a revolution in social-oriented applications and services for mobile phones. A Smartphone Social Network is a structure made up of individuals carrying smartphones, which is used for sharing and collaboration [1] (i.e., content, interests, comments and places). Sites such as Google Latitude, Gowalla, Foursquare, Facebook Places and Loopt enable users to report on Who-What-When-Where events, check-in to favorite places, provide their location history, etc. For instance, users of Facebook can upload geo-located photos on-the-go and tag (i.e., comment-on) photos with the given service exceeding over 50 billion photos as of 07/2010. ABI Research[1] projects that Smartphone Social Network applications will reach almost 150 million users in 2013 while academic efforts in this direction are also underway [41].

Additionally, there is already a proliferation of innovative applications founded on crowd-sourcing (e.g., [8]) and opportunistic/participatory sensing [5, 7, 12], where applications can task mobile nodes in a given region to provide information about their vicinity using their sensing capabilities (e.g., noise-maps [39], etc.). Another example is road traffic delay estimation [45] using WiFi beams collected by smartphone devices 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, 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, accelerometer, WiFi RSS readings, etc.). We claim that the centralization of these object types will be severely hampered in the future due to the following constraints:

i. **Data-Disclosure Constraints:** Continuously disclosing user-captured objects to a central entity might compromise user privacy in very serious ways.[2] Even Google's CEO Eric Schmidt mentioned recently[3] that "… *every young person one day will be entitled automatically to change his or her name on reaching adulthood in order to disown youthful hijinks stored on their friends' social media sites.*"
ii. **Energy Constraints:** 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 both deplete the precious

[1]"Location-Based Mobile Social Networking", Market Development, Revenue Opportunities, Applications, and Key Industry Players, ABI Research3Q.

[2]"Google Apologizes for Buzz Privacy", David Coursey, PC World Business Center (online), Feb. 15th, 2010.

[3]"Google and the Search for the Future", Holman W. Jenkins Jr., The Wall Street Journal (online), Aug. 14th, 2010.

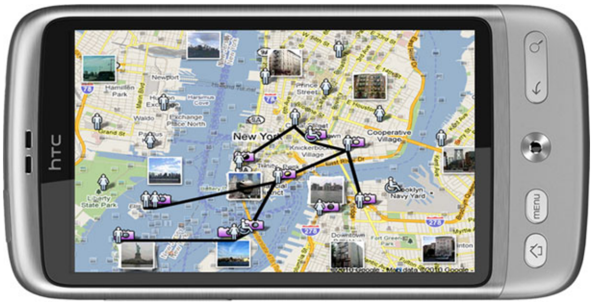


Fig. 1 A visual illustration of the *Multi-Objective Query Routing Tree (MO-QRT)* structure proposed in this work. Our SmartOpt Framework constructs MO-QRT structures optimized on several conflicting objectives (i.e., energy, time and recall). Our structure can be utilized for finding objects (e.g., images, videos, etc.) in a social neighborhood, without the necessity of having the objects disclosed to the social network provider

smartphone battery faster, increase query response times, but can also quickly degrade the network health.[4]

In this paper, we present techniques to enable smartphone users keep their data *in-situ*, for data-disclosure and performance reasons, offering at the same time high performance search capabilities over other user's data in the social community. When a user invokes a search to find an object of interest, e.g., *"Pictures of street artists performing in Manhattan"* (see Fig. 1), the user first downloads a *Query Routing Tree (QRT)* $\mathcal{X}$ from a SmartOpt server. The $\mathcal{X}$ structure resembles spanning tree structures constructed during searches in unstructured (Mobile) Peer-to-Peer (P2P) systems [19, 47, 53, 54] or aggregation trees used in sensor networks [2], but $\mathcal{X}$ is tuned to optimize several objectives concurrently during searches in a smartphone network. The tree structure $\mathcal{X}$ provides better scalability than having the query node contact all nodes that might contain an answer. This intrinsic characteristic differentiates P2P architectures from respective centralized architectures.

In particular, the MO-QRTs proposed in this work are optimized to (i) minimize energy consumption during search; (ii) minimize the query response time in conducting the search; and (iii) maximize the recall rate of the user query. Most existing works optimize the objectives (i)–(iii) individually, or optimize one and constrain the complementation. This often results in "poor" solutions since the objectives are conflicting and a decision maker needs an optimal trade-off set, commonly known as the *Pareto Front (PF)* in the context of *Multi-Objective Optimization (MOO)*. Figure 2, shows an example where each point represents a QRT solution. The x-coordinate of a point is the QRT's overall energy consumption, the y-coordinate is the QRT's overall response time in conducting the search and the z-coordinate is the QRT's overall recall to the query (i.e., percentage of relevant answers returned, shall be defined more rigorously later). A QRT $\mathcal{X}$ dominates a QRT $\mathcal{Y}$, if $\mathcal{X}$ has lower energy consumption, requires less response time and provides higher recall rate than $\mathcal{Y}$ at the same

[4]"Customers Angered as iPhones Overload AT&T", Jenna Wortham, The New York Times (online), Sept. 2nd, 2009.

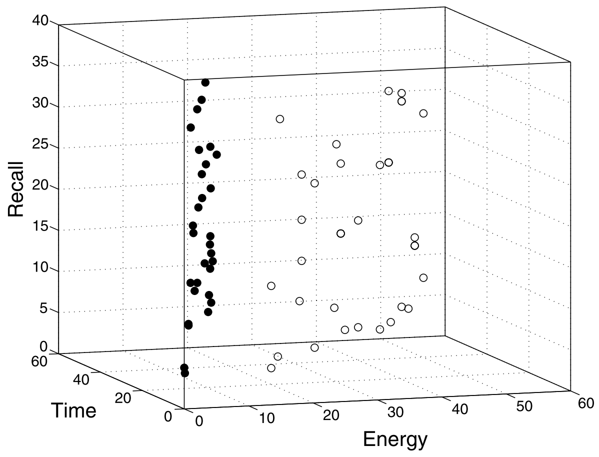


Fig. 2 A Pareto Front example of the MO-QRT problem. *Solid circles* represent non-dominated QRTs

time. The Pareto Front is composed of all QRTs that are not dominated by others, that is, the black dots in Fig. 2. A major issue in MOO is that there is no single point (called solution thereafter) that can optimize all objectives simultaneously. The literature hosts several approaches that can efficiently deal with multiple conflicting objectives and provide a set of non-dominated solutions in a single run, such as the Multi-Objective Evolutionary Algorithms (MOEAs) [14] that have been shown very effective in the past. An operator that has the same characteristics as those of MOO, but mainly received attention in the database community for disk-resident data and applied to data mining problems, is the skyline operator [6, 10, 20, 35, 58]. Skyline operators are mainly classified as centralized [10, 20, 31, 35, 44] or distributed [6, 21, 48, 49, 51, 58]. The former aims to collect all the data from multiple resources to a centralized server, which in turn retrieves the skyline (i.e., the global set of non-dominated solutions). In the distributed case, each source initially retrieves a local skyline and then attempts to obtain the global skyline. However, in most cases the skyline operators are based on systematic approaches (i.e., deterministic or exact) for dealing with disk-resident data giving in most cases exact skyline solutions. Using a systematic approach in our case is not efficient due to the high complexity and high dimensionality of the proposed problem, as discussed later in Sect. 3.

This paper builds on our previous work in [30], in which we presented the preliminary design of the SmartOpt framework. In this paper, we introduce several new improvements and extensions that are summarized as follows:

- We extend the SmartOpt framework with new features including *Decision Making*, during which a non-dominated QRT $\mathcal{X}$ can be selected from the Pareto Front based on some user-preference; and *Searching*, during which the QRT solution $\mathcal{X}$ is propagated to the network using a text-based Peer-to-Peer tree propagation protocol.
- We present a detailed description of the SmartOpt architecture, including insight information on all its components and internal procedures, its protocol, its *SmartP2P* [26] prototype system and our *SmartLab* [27] platform of 40+ real smartphones that has been utilized in our evaluation.

– We introduce an elaborate experimental study and solid experimental evidence for the motivation and efficiency of our propositions using both a trace-driven experimental methodology with mobility and social patterns derived by Microsoft's GeoLife project, DBLP and Pics 'n' Trails, but also using our SmartP2P real system developed in Android and deployed over our SmartLab testbed of 40+ smartphone devices. We also assess the optimality of our multi-objective optimizer (MOEA/D and NSGA-II) and different peer-to-peer search techniques (breadth-first-search, random walkers and SmartOpt trees).
– We provide background and related work on the following four areas related to the scope of this paper: *Mobile P2P Search*, *Query Routing Trees (QRTs)*, *skyline queries* (centralized, distributed) and *Multi-Objective Optimization*. We also qualitatively explain the differences and similarities of the referenced techniques compared to the SmartOpt framework.

The overall contributions of this paper to the state-of-the-art are the following:

– We propose the *Multi-Objective Query Routing Tree (MO-QRT)* problem for Smartphone Social Networks and formulate it as a Multi-objective Optimization Problem (MOP), which minimizes the energy consumption and time overhead during searches but also concurrently maximizes the recall rate of answers.
– We propose a principled framework, coined SmartOpt, for designing an efficient algorithm for the MO-QRT problem composed of an optimizer, which is based on a specialized Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) combined with a posterior decision maker and a Peer-to-Peer search approach. We propose several complementary techniques for designing an efficient and effective approach as introduced in Sect. 4. We have also developed a real prototype system, named SmartP2P, for the ubiquitous Android Operating System that shows how the proposed framework can be utilized in real conditions.
– We evaluate our *SmartOpt Framework* using mobility and social behavior patterns derived from GeoLife [57], DBLP [13] and Pics 'n' Trails [42, 43] using both a trace-driven experimental methodology and a real execution of our SmartP2P prototype over our SmartLab testbed.

The remainder of the paper is organized as follows: Sect. 2, provides our system model and defines the problem in a rigorous manner. Section 3 provides the background and overviews the related work. Section 4 introduces the SmartOpt framework and its internal modules composed of various operators. Section 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 Sect. 6, while Sect. 7 concludes the paper.

2 System model and problem formulation

In this section, we outline our system model and formulate the problem SmartOpt aims to solve. A table of respective symbols is shown in Table 1.

Table 1 Table of symbols

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

2.1 System model

Overview Let $\mathcal{C}$, denote a social networking service that maintains centrally the profiles $\mathcal{P} = \{p_1, p_2, \dots, p_M\}$, for each of its M subscribed users (i.e., $\mathcal{U} = \{u_1, u_2, \dots, u_M\}$). 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 $\mathcal{G}$ 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 capture objects of interest at arbitrary moments (e.g., "take a picture of the Statue of Liberty".) 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 $\mathcal{C}$ (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) [8]. We assume that when u_i is connected to $\mathcal{C}$, then $\mathcal{C}$ 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 39 mW/byte, 3G consumes 24 mW/byte and Bluetooth consumes 14 mW/byte. Additionally, Bluetooth had a symmetric data rate of 864 kbps, while WiFi an asymmetric data rate of 123 kbps (up) and 2 Mbps (down) and 3G an asymmetric data rate of 2.7 Mbps (up) and 7.2 Mbps (down). The nominal data rates for the aforementioned modalities might differ significantly, as this is also validated in [22], 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] $\mathcal{Q}$ over its social neighborhood (i.e., nodes connected to u_j either directly or through intermediate nodes) $\mathcal{G}'$ ($\mathcal{G}' \subseteq \mathcal{G}$). For instance, let $\mathcal{Q}$ be a depth-bounded breadth first search query over u_j's neighbors in the $\mathcal{G}$ graph (i.e., in $\mathcal{G}'$). 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 $\mathcal{C}$ prior query execution. Once $\mathcal{Q}$ is posted, $\mathcal{C}$ 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 $\mathcal{C}$) 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 objects 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., $\mathcal{G}'' \subseteq \mathcal{G}'$). u_j then contacts the nodes in $\mathcal{G}''$ in order to conduct a depth-bounded breadth first-search in a P2P fashion (i.e., using a pre-specified Query Time-To-Line, i.e., $\mathcal{Q}_{TTL} > 0$). Once some arbitrary node $u_x \in \mathcal{G}'$ receives $\mathcal{Q}$, it both looks at its local tags, in order to identify an answer and also forwards the request further until $\mathcal{Q}_{TTL}$ becomes zero.

Although the BFS approach improves the data-disclosure drawback of the *CS* approach, it is quite inefficient during search. In particular, $\mathcal{Q}$ 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).

2.2 Optimization problem formulation

The *Multi-Objective Query Routing Tree (MO-QRT)* structure, proposed in this paper, improves the search operation of the BFS algorithm by optimizing the neighbor selection process. In particular, a node downloads from $\mathcal{C}$ a QRT $\mathcal{X}$ that is optimized

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

according to the following formulation: *Given a social network of users $\mathcal{U}$, a list of active users $\mathcal{U}'$ and their coordinates, the profiles $\mathcal{P}$ of these users and a query $\mathcal{Q}$, posted by an arbitrary user u_j, the query processor aims to optimize an $\mathcal{X}$ structure using the following* **objectives**:

Objective 1: Minimize the total Energy *consumption of $\mathcal{X}$*

$$Energy(\mathcal{X}) = \min \sum_{\forall(u_a,u_b)\in\mathcal{X}(\mathcal{X}\subseteq\mathcal{U}')} e(u_a,u_b) \tag{1}$$

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 $\mathcal{X}$*

$$Time(\mathcal{X}) = \min\Bigl(\max_{(u_a,u_b)\in\mathcal{X}} t(u_a,u_b)\Bigr) \tag{2}$$

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 $\mathcal{X}$*

$$Recall(\mathcal{X},\mathcal{Q}) = \max\left(\frac{Relevant(\mathcal{Q}) \cap Retrieved(\mathcal{X},\mathcal{Q})}{Relevant(\mathcal{Q})}\right) \tag{3}$$

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

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

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

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

again given that p_a contains terms found in $\mathcal{Q}$.

In a MOP, there is no single solution $\mathcal{X}$ 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 $\mathcal{X}^*$ is considered non-dominated or Pareto optimal with respect to another solution $\mathcal{Y}$, iff $\forall i \in \{1,\dots,n\}, \mathcal{X}_i \geq \mathcal{Y}_i \wedge \exists i \in \{1,\dots,n\} : \mathcal{X}_i > \mathcal{Y}_i$, this is denoted as $\mathcal{X} \succ \mathcal{Y}$.

In our previous works [3, 4, 53], we have studied each of the individual objective functions in isolation. For example, in [3] and [4] we have computed the energy consumption based on the energy model of the TelosB sensor device and the

CC2420 RF Transceiver including all its power modes (i.e., receive, transmit, idle, etc.). More specifically, the energy formula used was the following: Energy(Joules) = Volts × Amperes × Seconds (e.g., the energy required to transmit 30 bytes at 1.8 V is: 1.8 V × 23 × 10^{-3} A × 30 bytes × 8 bits/250 kbps = 39 J). The experimental results of these works were validated using PowerTossim, which is a well known tool for realistically measuring energy in various embedded devices. Furthermore, in [4] the critical path objective, which is similar to the time objective of this work, was calculated using an in-network recursive algorithm that took into account a number of real properties such as the link activity and the number of collisions at the MAC layer of each node. In the same manner, these works utilized a number of other objectives (e.g., network lifetime, query response time, quality of data (accuracy, recall)) calculated in a realistic manner and validated through well-established simulators. In [53], we have tackled the recall objective in P2P systems by developing a real system, coined PeerWare.

3 Background and related work

In this section, we provide related research work that lies at the foundation of the SmartOpt Framework.

Mobile Peer-to-Peer/MANET search can be roughly classified into: (i) *Blind* Search [19, 32, 52], 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 [9, 24, 25, 34, 40, 47], where mobile peers use semantic or location information to forward queries to specific nodes in the network. The proposed search approach presented in this paper 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 [40], we utilize a content summary mechanism (i.e., profile) for discovering mobile peers that will participate in a query $\mathcal{Q}$ 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)* [25], 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 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 [7]. 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 [46], 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 [46], 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. [53]), 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 (in 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.

Skyline operators are mainly used by the database community [35] to retrieve a global set of non-dominated solutions, i.e., the *skyline* similar to the Pareto Front of MOO, of a skyline (or Pareto) query in a centralized or a distributed manner. The literature, which focuses on centralized databases, aims at collecting all information from all resources to a centralized node, which in turn retrieves the global skyline using systematic approaches. For example, Tan et al. [44] adopted a Bitmap-based approach to retrieve the skyline using binary operations in a bit-string representation, as well as a B-tree based algorithm that further improves its predecessor response speed. Block Nested Loop and Sort-filter skyline (SFS) [10] approaches search the points in the data set exhaustively and retrieve the skyline points based on their domination ranking, having as a main difference that the latter initially sorts the points of the data set in an ascending order before the BNL approach is applied. Similarly, Godfrey et al. [20] extended the work in [10], by proposing the Linear-estimation-sort (LESS) algorithm that reduces the cost of SFS by eliminating a portion of the database during sorting. Papadias et al. [35] and Kossman et al. [31] have tackled the centralized skyline retrieval challenge considering R-tree nodes, using a branch-and-bound and a NN progressive approaches, respectively.

Moreover, several attempts have been made in distributed skyline retrieval, often using approaches for locally partitioning the data sets either vertically or horizontally and then retrieving the global skyline after collecting all local skylines in a central node. For example, Balke et al. [6] have proposed a vertically partitioned distributed skyline algorithm, that performs a round-robin based sorting until finding all non-dominated points for each particular database. Examples of distributed algorithms that are based on horizontally partitioning a database include [48, 49, 51] and [58]. Particularly, Wu et al. [51] separates the database region into rectangular spaces and maps each server to a region. Each server is therefore responsible for finding the skyline of their local data region and then by considering some precedence relations the global skyline is obtained. Similarly, Wang et al. [49] and Vlachou et al. [48] proposed an algorithm for skyline retrieval in P2P networks by organizing the peers in an overlay network and subspaces, respectively. Zhu et al. [58] has recently proposed a feedback-based distributed skyline (FDS) algorithm that supports horizontal partitions of the data sets of geographically distant servers. All aforementioned studies, however, process the local skylines in servers having no issues in memory and energy consumption, a crucial resource in mobile smartphone devices that considered in our case.

More closer to our work is [21], in which the authors study skyline retrievals on mobile devices of Mobile Ad-hoc NETworks (MANETs). The authors propose an algorithm that iteratively probes the mobile devices to construct a local subtree skyline, i.e., points that are in the skyline, rooted at each individual device. The global skyline is retrieved after probing all mobile devices. Generally, the dimensionality of the problems tackled in all aforementioned cases is low. For example, finding properties to optimize the distance from the beach and the price of a property is a common problem tackled in several skyline cases [31, 35, 58]. One solution in these kinds of problems in the decision space (e.g., the location (2-D) of a property) has a one-to-one mapping with one solution in the objective space (e.g., distance from the beach and price). In these cases, one can easily adopt systematic approaches, search all solutions and find the real skyline. However, in our case, obtaining a QRT by selecting some or all N active smartphones, increases the dimensionality (e.g., can be up to $2 \times N$, considering only the x, y coordinates of the smartphones) of the search space for obtaining a single QRT and thereinafter a solution in the objective space (e.g., the energy, time overhead and recall of that particular QRT). This increases the complexity of these kinds of problems, including the fact that in most cases there is not even any knowledge about the real Pareto Front that should be obtained. Therefore, it is nearly impossible to use a systematic approach and search all QRTs (i.e., all combinations of smartphones) for dealing with the proposed problem. This is the major reason why a stochastic approach, such as an Evolutionary Computation approach, might be more appropriate.

Multi-Objective Optimization (MOO) (a.k.a. multi-criteria or multi-attribute optimization) is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. 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 [23, 28, 29, 38], tackled in most cases by Pareto-dominance based MOEAs (e.g., the state-of-the-art Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [15], Evolutionary Multi-objective Crowding-based Algorithm (EMOCA) [37], etc.) and in few cases by decompositional MOEAs (e.g., Multi-Objective Evolutionary Algorithms based on Decomposition (MOEA/D) [56]).

4 The SmartOpt framework

In this section, we present the SmartOpt framework (see Fig. 3) that proceeds in three phases: (i) *the Optimization phase*, during which a set of non-dominated QRTs (i.e., Pareto Front) is identified; (ii) *the Decision Making Phase*, during which a non-dominated QRT $\mathcal{X}$ is selected based on some user-preference criteria from the Pareto Front; and (iii) *the Search Phase*, during which the QRT solution $\mathcal{X}$ is propagated to u_j and the search process is initiated.

Our framework 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 [28]. In the context of SmartOpt,

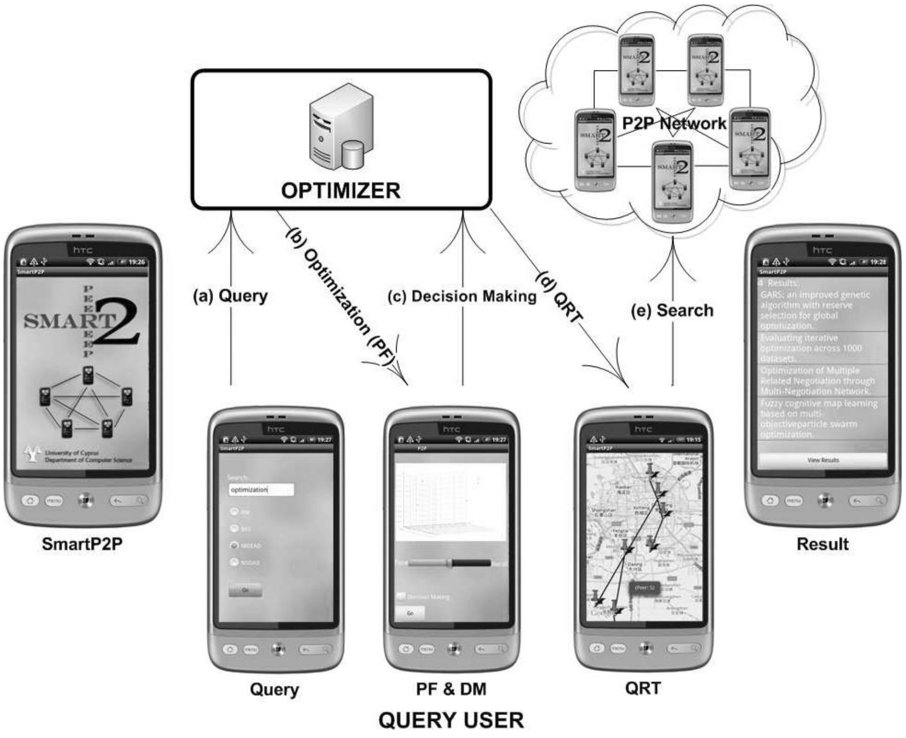


Fig. 3 The SmartOpt framework with our SmartP2P prototype system. (**a**) A user posts a query to the optimizer. (**b**) The optimizer obtains a set of non-dominated solutions (PF) and send it back to the user. (**c**) The user (decision maker) chooses a Pareto-optimal solution based on instant requirements 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

we introduce both domain expertise into our operators as well as utilize well-known operators that have been proven accurate over the years.

Specifically in the Optimization Phase, we have implemented and specialized the MOEA/D framework, which is the state-of-the-art of the decompositional MOEAs and the winner of the Unconstrained Multi-Objective Evolutionary Algorithm competition in 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 [56]. 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. In the Decision Making Phase, 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. Finally in the Search Phase, our framework uses a fast text-based Peer-to-Peer tree propagation protocol to retrieve objects of interest from the social network.

4.1 Pre-processing steps of SmartOpt optimizer

The pre-processing steps consists of representing a QRT and decomposing the problem into a set of scalar sub-problems.

Representation In our approach, a solution[6] $\mathcal{X}$ is a QRT with $|\mathcal{G}'|$ active smartphone users that can participate in the resolution of $\mathcal{Q}$. Without loss of generality, let $\mathcal{X}$ 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 $\mathcal{X}$. Figure 4 illustrates a query routing tree $\mathcal{X}$ representation as well as $\mathcal{X}$ in a smartphone network.

Decomposition Initially, the MOP should be decomposed into m subproblems by adopting any technique for aggregating functions, e.g., the Tchebycheff approach used here. In this paper, the ith subproblem is in the form

$$\text{maximize } g^i\left(\mathcal{X}|w^i_j, z^*\right) = \max\left\{w^i_j\left|f_j(\mathcal{X}) - z^*_j\right|\right\} \tag{4}$$

where f_j, $j = 1, 2, 3$, are the objectives of our MOP formulated earlier in Sect. 2.2, $z^* = (z^*_1, z^*_2, z^*_3)$ is the reference point, i.e., the maximum objective value $z^*_j = \max\{f_j(\mathcal{X}) \in \Omega\}$ of each objective f_j, $j = 1, 2, 3$ and Ω is the decision space. For each Pareto-optimal solution $\mathcal{X}^*$ there exists a weight vector w such that $\mathcal{X}^*$ is the optimal solution of (4) and each solution is a Pareto-optimal solution of the MOP

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

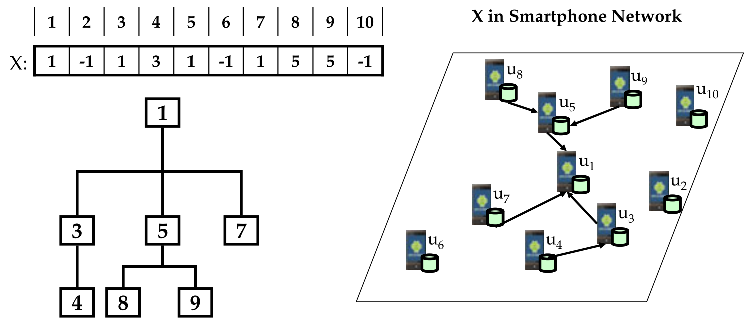


Fig. 4 The query routing tree $\mathcal{X}$ representation (*left*) and conceptual structure (*right*)

Algorithm 1 The SmartOpt Optimizer

```
Input:
    • network parameters (e.g., $\mathcal{Q}, \mathcal{P}, \mathcal{U}, \mathcal{G}$);
    • $m$: population size and number of subproblems;
    • $T$: neighborhood size;
    • weight vectors $(w_j^1, \ldots, w_j^m)$, $j = 1, 2, 3$;
    • 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 $IP_0 = \{\mathcal{X}^1, \ldots, \mathcal{X}^m\}$, 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) $\mathcal{Y}$ using the genetic operators.
    Step 2.2 (Local Heuristic): Apply a problem-specific repair heuristic on $\mathcal{Y}$ to produce $\mathcal{Z}$.
    Step 2.3 (Update Populations): Use $\mathcal{Z}$ to update $IP_{gen}$, $PF$ and the $T$ closest neighbor solutions of $\mathcal{Z}$.
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.
```

in Sect. 2.2. For the remainder of this paper, we consider a uniform spread of the weights w_j^i, which remain fixed for each subproblem i for the whole evolution and $\sum_{j=1}^{3} w_j^i = 1$.

4.2 Optimization phase

In this phase, SmartOpt optimizes in an online manner the solution space using a set of genetic operators. An outline of this phase is provided in Algorithm 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 $\mathcal{X}$ 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

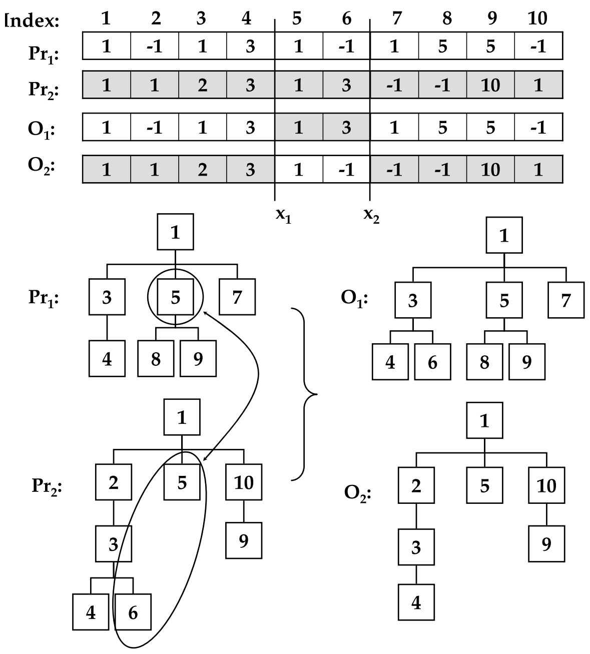

Index:
1 2 3 4 5 6 7 8 9 10
Pr1: 1 -1 1 3 1 -1 1 5 5 -1
Pr2: 1 1 2 3 1 3 -1 -1 10 1
O1: 1 -1 1 3 1 3 1 5 5 -1
O2: 1 1 2 3 1 -1 -1 -1 10 1
x1
x2
Pr1:
Pr2:
O1:
O2:

Fig. 5 The Crossover operator of SmartOpt optimizer

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. Thereinafter, the IP_{gen} is used to store the best QRT solution $\mathcal{X}^i$ found for each subproblem g^i during the search, i.e., in 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 $\mathcal{Y}^i$ 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 [29] 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 $2x$*-point tree crossover* operator takes as an input two parent QRT solutions, Pr_1 and Pr_2, and subsequently generates two new QRTs O_1, 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.

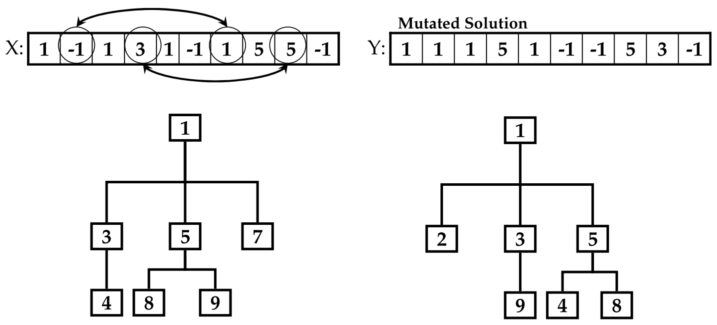


Fig. 6 The swap mutation operator of the SmartOpt optimizer

- 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.

  The procedure of the 2x-point crossover is illustrated in Fig. 5 for $M = 10$.

– **Swap Mutation:** modifies an offspring O to a solution $\mathcal{Y}$ with a probability r_m by uniformly randomly swapping the values (i.e., parents in the tree) of two indexes j, z of the QRT $\mathcal{Y}$. Figure 6 shows an example where a solution $\mathcal{X}$ of size $M = 10$ is processed by the mutation operator and based on the probability parameter r_m the indexes $X_2 = -1$ and $X_9 = 5$ are modified by swapping their values with those of indexes $X_7 = 1$ and $X_4 = 3$, respectively, creating solution Y. This results in assigning a parent to Y_2, i.e., the root 1, and changing the parent of X_9 to 3. Note that, X_2 had no parent, and therefore was not included in the tree, before mutation. Mutation operator is often used for improving exploration and consequently the diversity of the obtained solutions.The modified QRT solution $\mathcal{Y}$ is then forwarded to the repair heuristic.

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

– **Case #1:** there is a disconnected user u_i in QRT $\mathcal{Y}$ (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 $\mathcal{Y}$;
– **Case #3:** there is an infinite loop in QRT $\mathcal{Y}$.

In all cases (illustrated in Fig. 7), the solution $\mathcal{Y}$ is considered infeasible. An infeasible solution can be generated during reproduction (i.e., genetic operation). A local heuristic repairs the QRT solution $\mathcal{Y}$ to $\mathcal{Z}$ 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. All repair techniques are shown with dotted lines in Fig. 7. The repair heuristic continuously repairs solution $\mathcal{Y}$ until it does not fall in any of the Cases #1, #2 or #3. Solution $\mathcal{Z}$ is then used to update the populations of MOEA/D.

Population Update Step 2.3: In Step 2.3, the update phase of Algorithm 1 is processed in three steps. (1) Update IP, $IP/\{\mathcal{X}^i\}$ and $IP \cup \{\mathcal{Z}^i\}$ if $g_i(\mathcal{Z}^i|w^i, z^*) <$

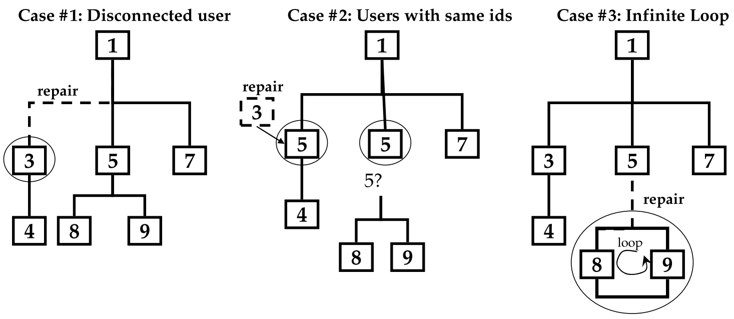


Fig. 7 The repair operator of SmartOpt optimizer

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

4.3 Decision making phase

In the posterior decision making phase used in this paper, the query user u_j is prompt to decide its preference in terms of *Time* (i.e., Objective 2 calculated by Eq. (2)) and *Recall* (i.e., Objective 3 calculated by Eq. (3)) of the query response to receive from the Smartphone Network. The decision maker module of SmartOpt then finds the QRT solution $\mathcal{X}$ of the PF that best satisfies the user's decision and it is also the most *Energy* efficient (i.e., Objective 1 calculated by Eq. (1)) at the same time. By 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.

For example, consider that the SmartOpt optimizer has obtained the PF of Fig. 8 in Phase 1. The slidebar at the bottom of the figure is the query user's decision, where $w_1 = 0.3$ and $w_2 = 0.7$, s.t. $w_1 + w_2 = 1$, correspond to the user's preference in terms of time and recall, respectively. Then, the decision maker module calculates and obtains the solution $\mathcal{X}$ that is closer (in terms of Euclidean distance) to the user's decision in the objective space and provides Pareto optimal energy consumption (i.e., E' in Fig. 8) at the same time. In cases where there are more than one solution that equally satisfy the user's decision then the most energy efficient is selected to be searched. Figure 8 also shows solutions A, B and C that represent the extreme solutions of the PF. That is, solution $\mathcal{A}$ represents the best Pareto optimal solution in terms of time, in case that the query user is only interested in receiving the results fast, fully ignoring recall, i.e., $w_1 = 1$, $w_2 = 0$. Solution $\mathcal{B}$ represents the best Pareto

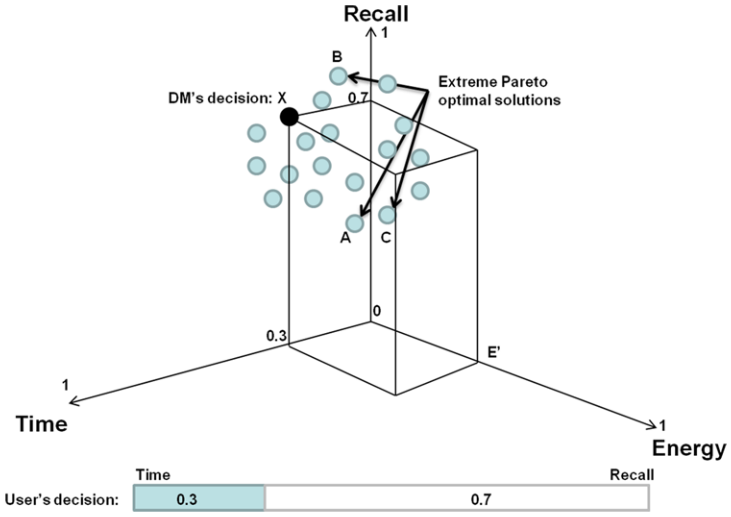


Fig. 8 Decision making example

optimal solution in terms of recall, in case that the query user is only interested in the amount of information (recall), fully ignoring the time, i.e., $w_1 = 0$, $w_2 = 1$. Finally, solution C represents the best Pareto optimal solution in terms of energy that the decision maker module automatically selects, in case that the query user does not have a preference with respect to the other two objectives. The Pareto optimal QRT $\mathcal{X}$ is then propagated to u_j and the search process is initiated in the following phase.

4.4 Search phase

In the final phase, the query user u_1 receives the Pareto-optimal tree $\mathcal{X}$ from the decision maker module of SmartOpt and proceeds with a recursive execution of Algorithm 2 on all smartphone devices participating in the tree $\mathcal{X}$. Recall that $\mathcal{X}$ 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 $\mathcal{Q}$ it creates a set O_j of all objects o_{ji} that satisfy $\mathcal{Q}$ (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 $\mathcal{Q}$ to all its child nodes (lines 8–14). This is done by checking each parent entry in $\mathcal{X}$ with its own (line 11). If a match u_i is found, u_j transmits $\mathcal{Q}$ and $\mathcal{X}$ to u_i (line 12). This process executes recursively until all smartphone devices in $\mathcal{X}$ receive the query.

4.5 Summary of SmartOpt architecture

The proposed SmartOpt framework aims at obtaining a diverse and high quality set of non-dominated QRT solutions (PF) by using a MOEA in the Optimization Phase (detailed in Sect. 4.2). Then it opts for the best suited Pareto-optimal QRT $\mathcal{X}^* \in PF$ based on instant requirements and preferences of the query user u_j (decision maker)

Algorithm 2 Search Phase

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

```
1:  procedure SEARCH($u_1$,$\mathcal{X}$,$\mathcal{Q}$)
2:      if ($j \neq 1$) then
3:          //Step 1: Find a set of local objects $O_j$ that satisfy $\mathcal{Q}$
4:          $O_j = \bigcup_{\forall i} o_{ji}, satisfy(o_{ji}, \mathcal{Q})$
5:          //Step 2: Send local objects $O_j$ to query user $u_1$
6:          $Send(O_j, u_1)$;
7:      end if
8:      //Step 3: Forward query $u_1$ to all children smartphone devices
9:      for $i = 1$ to $|\mathcal{X}|$ do
10:         //if $j$ is the parent of $i$
11:         if ($\mathcal{X}[i] == j$) then
12:             $Search(u_1, \mathcal{X}, \mathcal{Q})$;
13:         end if
14:     end for
15: end procedure
```

in the Decision Making Phase (Sect. 4.3). The query user u_j then downloads and utilizes QRT $\mathcal{X}^*$ to search the mobile social network and find objects of interest o_{ik} recorded by users $u_i \in \mathcal{X}^*$ and related to query $\mathcal{Q}$ in the Search Phase (detailed in Sect. 4.4).

5 The SmartP2P prototype system

In this section, we describe our prototype system, coined SmartP2P,[7] developed for the ubiquitous Android Operating System to demonstrate the applicability of the SmartOpt framework. We particularly overview the GUI and protocol of the framework as well as its evaluation on our programming cloud of Smartphones, coined SmartLab testbed.

5.1 Overview

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 327 KB. Our code is written in JAVA and consists of around 7500 lines of code. In particular our server-code (i.e., optimizer) uses 5000 LOC 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.

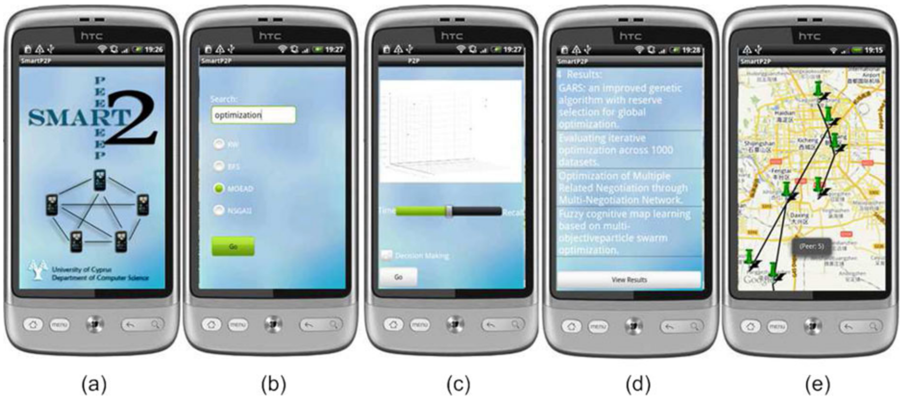
[7] Available at: http://smartp2p.cs.ucy.ac.cy/.

5.2 Graphical user interface

The Graphical User Interface of our system provides a primitive interface for a user to query the active users in the community (the details of the protocol are presented in the next paragraph). Figure 9(b) shows the GUI through which a query is formulated in order to find objects of interest. The group of algorithmic choices provided by the SmartP2P framework is shown below the search box. SmartP2P provides (i) two simple distributed choices, 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 presses the "Go" button. Then the SmartP2P 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. 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 as shown in Fig. 9(c) (note that the image can zoom in/out for better visualization). Then the query user makes use of the 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. By pressing the "Go" button, 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. The results of the search as well as the selected QRT are both displayed on the user's smartphone as shown in Figs. 9(d) and (e), respectively.

5.3 Protocol

We shall next provide an abstraction of the peer-to-peer protocol that lies at the foundation of our prototype system. We chose to implement a text-based protocol, as opposed to a binary protocol, for portability (i.e., endianness) reasons. We also did not



GARS: an improved genetic algorithm with reserve selection for global optimization.
Evaluating iterative optimization across 1000 datasets.
Optimization of Multiple Related Negotiation through Multi-Negotiation Network.
Fuzzy cognitive map learning based on multi-objectiveparticle swarm optimization.

Fig. 9 The SmartP2P Android GUI. (**a**) The intro screen. (**b**) The keyword search optimization with four algorithmic choices screen and (**c**) the resulted Pareto Front screen for decision making. (**d**) The results retrieved after initiating a P2P search on the smartphone network. (**e**) The QRT selected by the query user and utilized to retrieve the objects of interest

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 in Step 1 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). The above function is carried out using the following message exchange:

```
- STEP 1: REGISTRATION
SERVER: +OK READY - welcoming message
QP CLIENT: REGISTER APPROX_LOCATION
SERVER: +OK 8734e604-0f79-45ee-9126-f71eaee540f5
<close connection to SERVER>
```

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). The Globally Unique Identifier (i.e., GUID or UUID) returned by the server provides an easy mechanism to enforce registration with the host-cache prior to any other function as explained next and requires only minimal state on the server.

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 step 2, in order to obtain a tree 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 GUID token returned during the registration step 1. The returned tree is serialized in the following format `"NodeIP:NodePort(ParentIP:ParentPort)"`, with -1 denoting no-parent but is shorten below for ease of exposition. The message exchange proceeds as follows:[8]

```
- STEP 2: TREE RETRIEVAL
SERVER: +OK READY
QP CLIENT: GET T 8734e604-0f79-45ee-9126-f71eaee540f5
SERVER: P0(-1), P10(P0), P15(P0), P30(P10), NULL
<close connection to SERVER>
```

Once T is obtained by QP, QP connects to `P0=root(T)` in step 3 and submits its query $\mathcal{Q}$ (i.e., $\{k1, k2, \ldots, kn\}$), its HOME_ADDR address (i.e., IP:PORT) as well as a hop count parameter. P0 then forwards these parameters to its own children (i.e., P10 and P15), in a recursive manner for N levels (using a predetermined Time-To-Live (TTL) value enforced by the hop count). The messaging for the first three hops (assuming Depth-First-Search propagation), is as follows:

```
- STEP 3: P2P SEARCH
P0 CLIENT: +OK READY
QP CLIENT: SEARCH 0 | HOME_ADDR | k1,k2,...,kn |
P10(-1)|P30(P10)|NULL
- next hop
P10 CLIENT: +OK READY
```

[8] Our system also supports command pipelining as opposed to utilizing separate connections for each step.

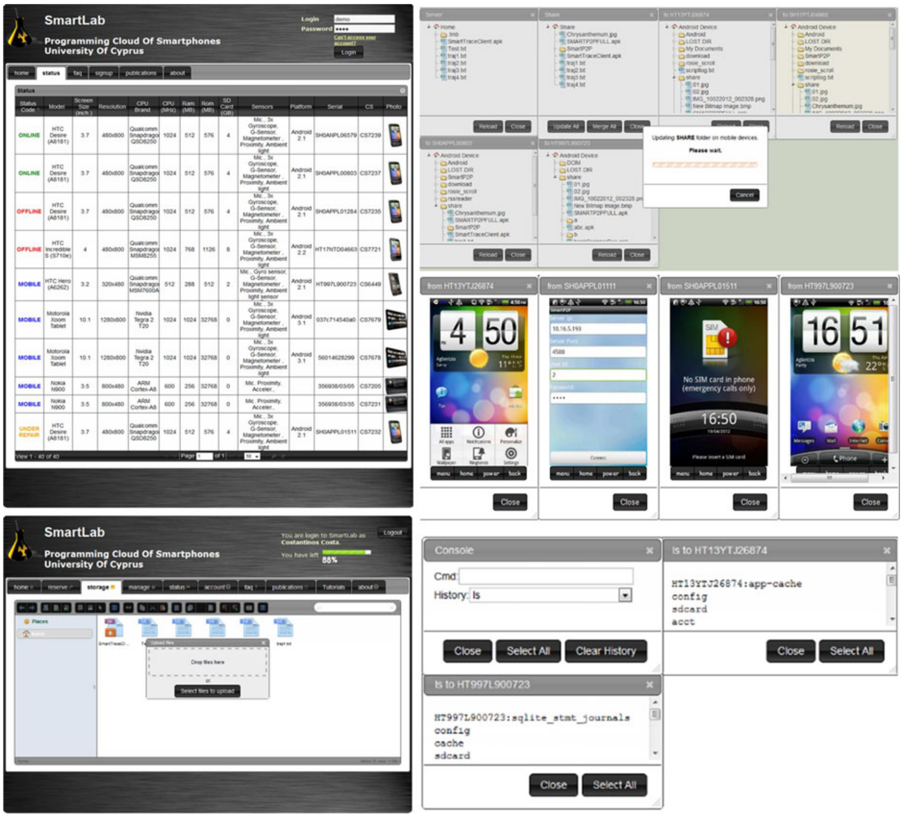
```
P0 CLIENT: SEARCH 1 | HOME_ADDR | k1,k2,...,kn |
P30(-1)|NULL
- next hop
P30 CLIENT: +OK READY
P10 CLIENT: SEARCH 2 | HOME_ADDR | k1,k2,...,kn |NULL
```

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 [33], 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.

5.4 SmartLab: a programming cloud of smartphones

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* [27], an innovative programming cloud of approximately 40+ Android smartphones and tablets, which is deployed at the University of Cyprus (see Fig. 10). SmartLab is inspired by both PlanetLab [11] and MoteLab [50]. 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 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 supports four (4) modes of user interaction: (i) *Remote Control Terminals (RCT)*, which support our in-house implementation of an ajax-based web-based remote screen terminal for Android that can mimic user clicks and gestures such as sliding in order to unlock devices and conduct other functionalities, (ii) *Remote Shells (RS)*, which supports our in-house implementation of an ajax-based web-based shell that can be utilized to issue a wide variety of known UNIX commands (e.g., `ls, ps, df, pwd, date`, etc.) to the Linux kernels that are found at the core of each Android device; (iii) *Remote Scripting Environment (RSE)*, which allows users to author Android MonkeyRunner automation scripts (written in python) in order to quickly perform repetitive tasks on selected devices; and (iv) *Remote Debug Tools (RDT)*, which provides web-based debugging extensions to the Android Debug Bridge (ADB) that are used during development. In this work, we have used SmartLab to evaluate our SmartOpt framework under real conditions. A more detailed description of SmartLab can be found in [27].



Status Code	Model	Screen Size (inch)	Resolution	CPU Brand	CPU (MHz)	Ram (MB)	Rom (MB)	SD Card (GB)	Sensors	Platform	Serial	CS	Photo
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 2.1	SH0ANPL06579	CS7239	
ONLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 2.1	SH0APPL00803	CS7237	
OFFLINE	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 2.1	SH0APPL01284	CS7235	
OFFLINE	HTC Incredible S (S710e)	4	480x800	Qualcomm Snapdragon MSM8255	1024	768	1126	8	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 2.2	HT17NTD04663	CS7721	
MOBILE	HTC Hero (A6262)	3.2	320x480	Qualcomm Snapdragon MSM7600A	512	288	512	2	Mic., Gyro sensor, G-Sensor, Magnetometer, Proximity, Ambient light sensor	Android 2.1	HT997L900723	CS6449	
MOBILE	Motorola Xoom Tablet	10.1	1280x800	Nvidia Tegra 2 T20	1024	1024	32768	0	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 3.1	037c714540a0	CS7679	
MOBILE	Motorola Xoom Tablet	10.1	1280x800	Nvidia Tegra 2 T20	1024	1024	32768	0	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 3.1	56014628299	CS7678	
MOBILE	Nokia N900	3.5	800x480	ARM Cortex-A8	600	256	32768	0	Mic. Proximity, Acceler.,		356938/03/05	CS7205	
MOBILE	Nokia N900	3.5	800x480	ARM Cortex-A8	600	256	32768	0	Mic. Proximity, Acceler.,		356938/03/35	CS7231	
UNDER REPAIR	HTC Desire (A8181)	3.7	480x800	Qualcomm Snapdragon QSD8250	1024	512	576	4	Mic., 3x Gyroscope, G-Sensor, Magnetometer, Proximity, Ambient light	Android 2.1	SH0APPL01511	CS7232	

View 1 - 40 of 40

```
HT13YTJ26874:app-cache
config
sdcard
acct
```

```
HT997L900723:sqlite_stmt_journals
config
cache
sdcard
```

Fig. 10 The SmartLab programming cloud of smartphone devices. Available at: http://smartlab.cs.ucy.ac.cy/

6 Experimental evaluation

In this section we present the experimental methodology and results of our evaluation.

6.1 Evaluation methodology

Our experimental methodology consists of two distinct scenarios: (i) *Trace-driven Simulation*, during which we assess the quality of the SmartOpt optimization process and also assess the quality of the SmartOpt search algorithm; and (ii) *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 [57] (*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 [13] (*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 [42, 43] (*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 $\mathcal{G}$ 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 $\mathcal{G}$ 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%';
```

Table 2 Experimental execution scenarios and test instances

Scenario	Test instance	$\mathcal{Q}$	Time	$\mathcal{G}'$	# Objects	Relevant objects
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

```
- 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 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.

Algorithms and evaluation metrics We have implemented both the (i) optimization and (ii) search algorithms, analyzed earlier in this paper, as described next:

– **Search Algorithms:** We have implemented (i) the *Centralized Search* algorithm (CS), presented in Sect. 2.1, 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 Sect. 2.1; (iii) the *Random Walker (RW) Search* [33], 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, in Experimental Series 1 (simulation) and Series 4 (real deployment), using the following metrics: *Time*, *Energy* and *Recall*, as these were defined in Sect. 2.2. For the simulation we use the time and energy profiles for our Smartphone devices we have presented in Sect. 2.1. For the real deployment, we utilize wall clock time along with the PowerTutor [55] 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.

– **Multi-Objective Optimization Algorithms:** In order to assess the efficiency of the tree construction process, we have implemented SmartOpt using two alternative approaches: (i) the MOEA/D approach, as this was described in Sect. 4; and (ii) the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [15], which maintains a population IP_{gen} of size m at each generation gen, for $gen_{\max}$ generations. NSGA-II adopts the same evolutionary operators for offspring reproduction as SmartOpt-MOEA/D. The key characteristic of SmartOpt-NSGA-II is that it uses a fast non-dominated sorting and a crowded distance estimation for comparing the quality of different solutions during selection as well as to update the IP_{gen} and the PF. The optimization algorithms are evaluated with respect to: (i) *Execution Time for Generating* $\mathcal{X}$ (Experimental Series 2); and (ii) *Multi-Objective Optimization Quality for Generating* $\mathcal{X}$ (Experimental Series 3).
  For the former case (execution time), we measure the CPU time required for the optimizer to derive $\mathcal{X}$ using both MOEA/D and NSGA-II. For the latter case (quality), we use the following combination of metrics:
  • $C(A, B)$*-metric* [60] (quality), which calculates the ratio of the non-dominated solutions in set B dominated by the non-dominated solutions in A, divided by the total number of non-dominated solutions in B. Hence,

$$C(A, B) = \frac{|\{x \in B | \exists y \in A : y \succ x\}|}{|B|}.$$

  Therefore, $C(A, B) = 1$ means that all non-dominated solutions in B are dominated by the non-dominated solutions in A. Note that $C(A, B) \neq 1 - C(B, A)$.
  • $S(A)$*-metric* [59] (diversity), which measures the diversity of A's solutions, formally the hyper-volume in the objective space that is dominated by the Pareto optimal solutions of set A. Again a lower $S(A)$, denotes that algorithm A has better diversity.
  • $NDS(A)$*-metric* (cardinality), which measures the number of non-dominated solutions in A's PF. The higher the NDS(A) score, the better algorithm A is.

6.2 Series 1: Evaluating SmartOpt search

In the first experimental series, we evaluate the performance of the SmartOpt search phase 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 11 illustrates the results of our experiment for all performance metrics. In Fig. 11 (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

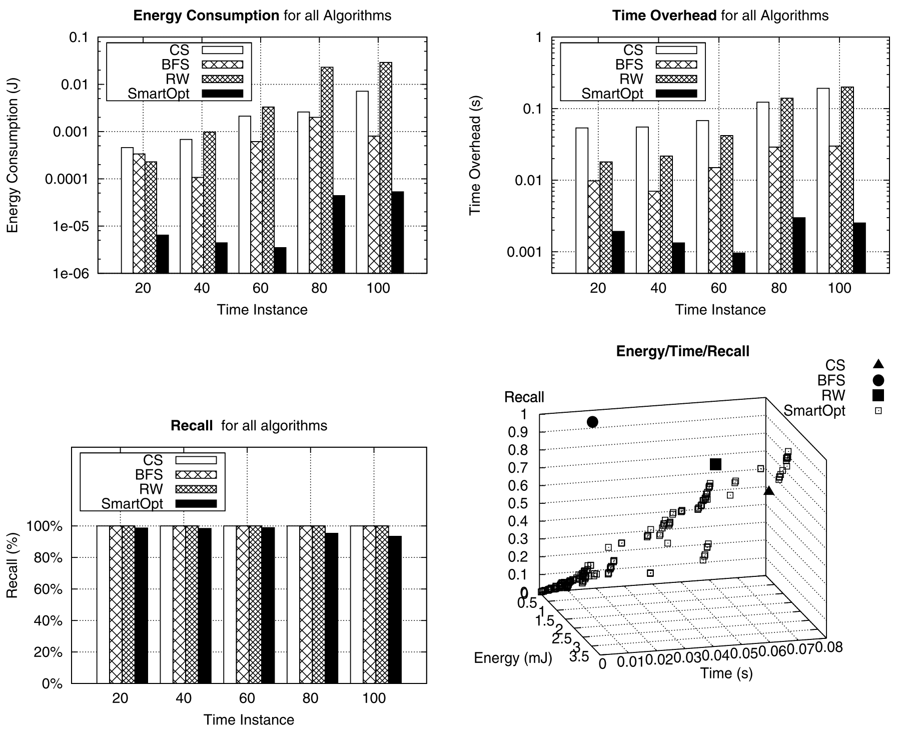


Fig. 11 Evaluation of the CS, BFS, RW and SmartOpt search algorithms using the energy, time and recall performance. The bottom/right figure shows SmartOpt compared to the solutions of CS and BFS in the objective space at timestamp $ts = 70$ of Mobile-Social Scenario-1 (MSS-1)

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 techniques that communicate with several nodes in a single round).

Similar observations apply for Fig. 11 (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 Fig. 11 (bottom/left), we show that the recall rate for the SmartOpt framework is close to 95 % consistently. Consequently, although we consume less time and less energy, we are able to identify all expected answers.

In Fig. 11 (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

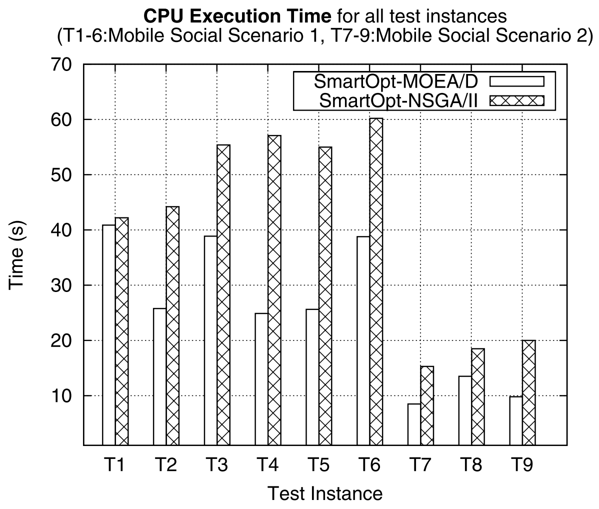


Fig. 12 Evaluation of SmartOpt-MOEA/D vs. SmartOpt-NSGA-II in terms of CPU performance in all nine test instances of MSS-1 (GeoLife+DBLP) and MSS-2(Pics 'n' Trails)

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.

6.3 Series 2: Execution time for generating $\mathcal{X}$s (PF)

In the second experimental series, we aim to identify whether SmartOpt generates the expected tree solutions quickly enough. We consequently evaluate the performance of the SmartOpt-MOEA/D and SmartOpt-NSGA-II approaches in terms of CPU time. We benchmark each algorithm by recording the time required to execute the four steps of the optimization phase described in Sect. 4.2, on all nine test instances. The results of our experiment are illustrated in Fig. 12.

We observe that SmartOpt-MOEA/D always outperforms SmartOpt-NSGA-II in terms of CPU performance. This is more evident in test instances T4, T5, T6 where the performance increases of SmartOpt-MOEA/D reaches as high as 56 %. This is because the decomposition of SmartOpt-MOEA/D naturally maintains the diversity of the population, thus balancing the effort required for generating solutions in optimal areas of the objective space. In contrast, the crowding distance mechanism of SmartOpt-NSGA-II used for maintaining diversity, may result in additional effort for obtaining solutions in the optimal areas of the objective space. Moreover, the overall CPU effort required in mobile social scenario 2 (i.e., T7, T8 and T9) is lower for both MOEAs, since the size of the Pics 'n' Trails dataset is smaller than the GeoLife+DBLP datasets.

6.4 Series 3: Multi-objective optimization quality of generated $\mathcal{X}$s (PF)

In the third experimental series, we study the quality of the QRT trees generated with the SmartOpt-MOEA/D and SmartOpt-NSGA-II optimizers. In our experiments, we have used the following algorithm setting: population size $m = 200$, crossover rate $r_c = 1$, mutation rate $r_m = 0.1$, $gen_{\max} = 250$ and $T = 12$.

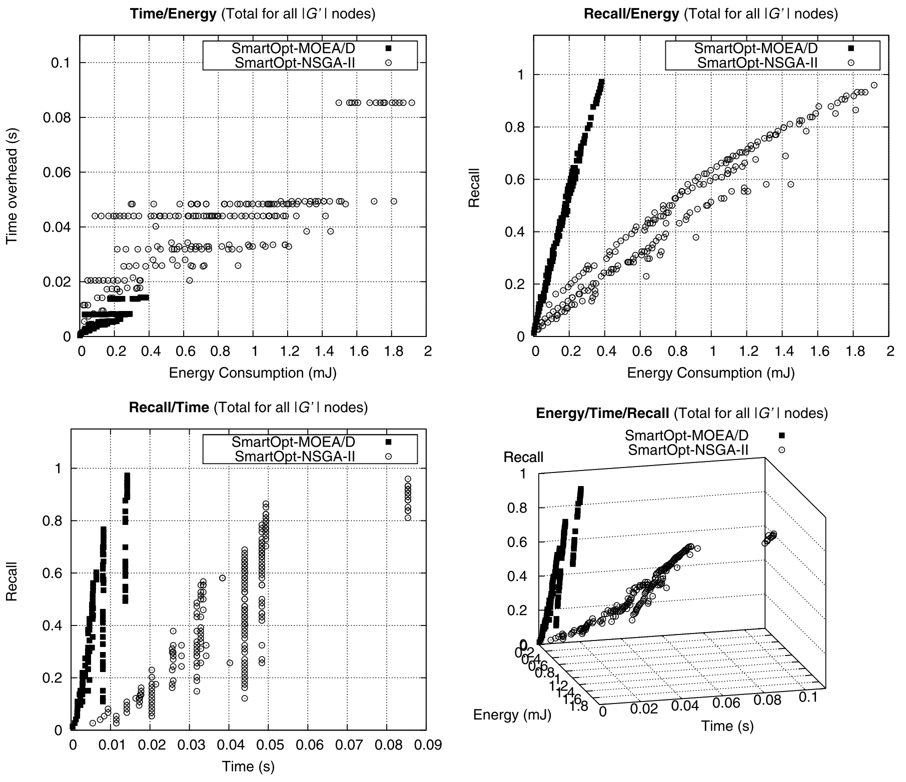


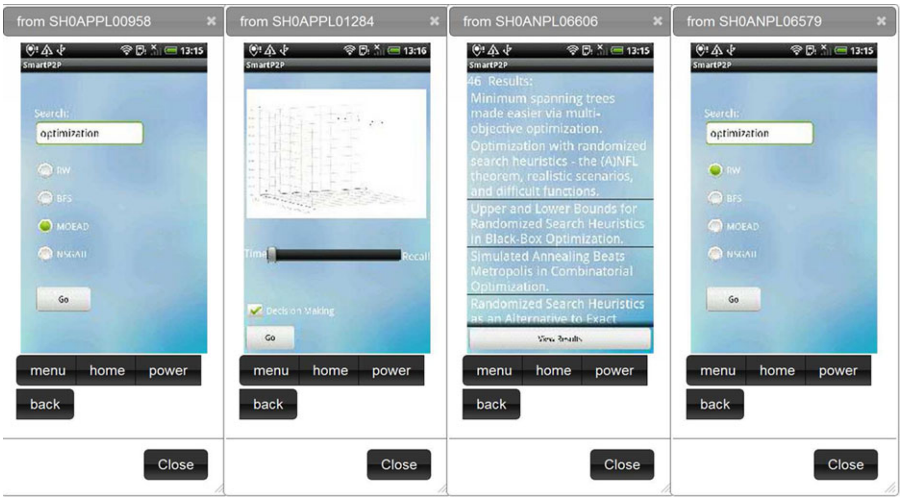
Fig. 13 Evaluation of MOEA/D vs. NSGA-II using the energy, time and recall performance metrics in Mobile-Social Scenario-1(MSS-1)

Figure 13 compares the performance of the two algorithms in combinations of two of the three objectives as well as all together in a 3D view. The results indicate the superiority of SmartOpt-MOEA/D along the direction of all the three objectives, giving non-dominated solutions of higher recall, of lower energy consumption as well as of lower time overhead. Besides, the 3-D subfigure (d) of Fig. 13 indicates that SmartOpt-MOEA/D searches the space more efficiently giving better diversity. Moreover, we observe that SmartOpt-NSGA-II obtains a higher number of NDS compared to SmartOpt-MOEA/D. However, these solutions are of inferior quality due to its low convergence speed.

Furthermore, the statistical results summarized in Table 3 compare the two approaches in all nine test instances of Table 2, supporting the observations just mentioned. That is, the non-dominated solutions obtained by SmartOpt-MOEA/D are of higher quality (i.e., C-metric) compared to those obtained by SmartOpt-NSGA-II in all cases. For example, none of the solutions obtained by SmartOpt-MOEA/D are dominated by those of SmartOpt-NSGA-II's and all of the solutions in SmartOpt-NSGA-II's PF are dominated by those of SmartOpt-MOEA/D's (C-metric) in MSS-1. Furthermore, the hyper-volume S-metric indicates that SmartOpt-MOEA/D searches the objective space more effectively and provide a more diverse

Table 3 SmartOpt-MOEA/D (denoted as M) versus SmartOpt-NSGA-II (denoted as N) in terms of quality, diversity and cardinality of the PF, in all nine test instances of Table 2. The best performance in each case is given in bold. The mean and the standard deviation (SD) is provided for each metric

TIs	C(M,N)	C(N,M)	NDS(M)	NDS(N)	$S(M) \times 10^{-4}$	$S(N) \times 10^{-4}$
T1	**1.00**	0.00	112	**188**	**0.10**	3.62
T2	**1.00**	0.00	125	**200**	**0.01**	0.57
T3	**1.00**	0.00	172	**200**	**0.07**	6.31
T4	**1.00**	0.00	200	200	**0.70**	15.02
T5	**1.00**	0.00	200	200	**3.45**	81.17
T6	**1.00**	0.00	200	200	**0.80**	127.25
T7	**0.96**	0.04	49	**95**	**0.294**	0.304
T8	**0.95**	0.04	95	**200**	**0.4403**	1.47
T9	**0.89**	0.1	58	**200**	**0.523**	1.77
Mean (μ)	0.98	0.02	134.56	187.00	0.71	26.39
Stddev (σ)	0.04	0.03	60.82	34.73	1.06	45.82



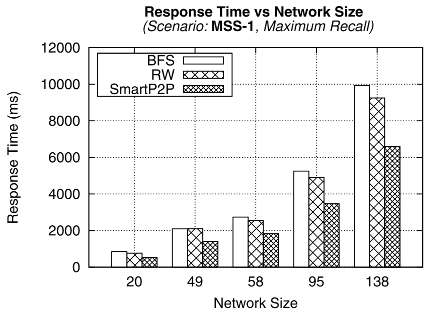
46 Results:
Minimum spanning trees made easier via multi-objective optimization.
Optimization with randomized search heuristics - the (A)NFL theorem, realistic scenarios, and difficult functions.
Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization.
Simulated Annealing Beats Metropolis in Combinatorial Optimization.
Randomized Search Heuristics as an Alternative to Exact

Fig. 14 A screenshot of the SmartP2P on SmartLab

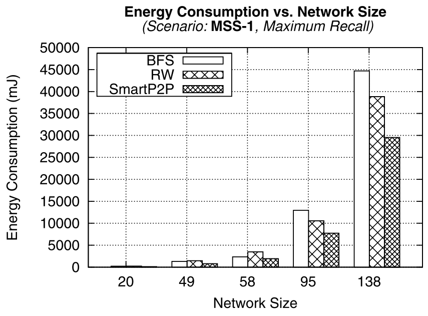
PF in all nine test instances. NSGA-II, however, provides a higher number of NDSs for the decision maker to choose, but they are of inferior quality.

6.5 Series 4: SmartP2P prototype evaluation on SmartLab

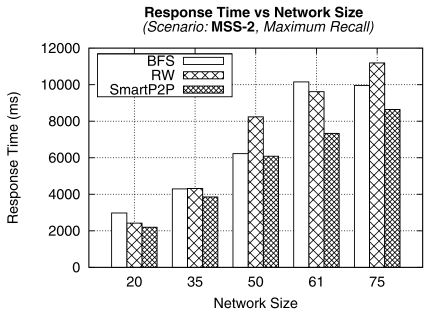
In the last experimental series 4, we evaluate our SmartP2P prototype Android implementation, presented in Sect. 5, over our distributed SmartLab infrastructure as illustrated in Fig. 14. For the evaluation, we focus only on the distributed search algorithms: *BFS, RW* and *SmartP2P*. We present the query response time, measured in seconds and energy consumption, measured with PowerTutor in Watts and pre-



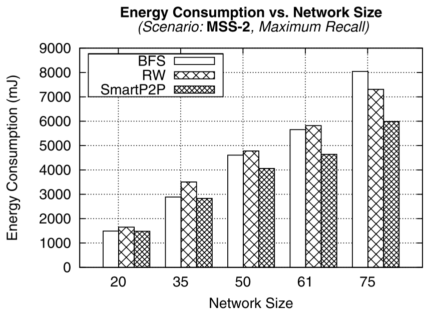
(a) SmartP2P (time) in MSS-1



(b) SmartP2P (energy) in MSS-1



(c) SmartP2P (time) in MSS-2



(d) SmartP2P (energy) in MSS-2

Fig. 15 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

sented 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 1 GHz smartphones with 512 MB of RAM) while the lower-end HTC Hero devices (with a 512 MHz processor and 288 MB 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 Blue-tooth connection between instances and considered as a local link the socket communication of instances on the same physical smartphone host.

Figure 15(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 15(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 ≈80 %, would allow us to obtain great energy savings considerably faster (≈50 %). Similarly, Figs. 15(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.

7 Conclusions

In this paper, 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 propose in this paper. 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, but also uses 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 of 95 %, with one order of magnitude less time and two orders of magnitude less energy than its competitors. Additionally, our study reveals that the MO-QRT structure is highly appropriate for content search and retrieval in Smartphone Networks. In the future, we plan to fine-tune our peer-to-peer search application and experiment with larger communities of users.

Acknowledgements This work was supported in part by the second author's Startup Grant, funded by the University of Cyprus, EU's FP7 CONET project, EU's FP6 Marie Curie TOK "SEARCHiN" project and EU's FP7 "MODAP" projects and US NSF IIS-10503. We would like to thank Mr. Christos Aplitsiotis for helping out with the development of SmartP2P and its experimentation on SmartLab.

References

1. Allen, S.M., Colombo, G., Whitaker, R.M.: Cooperation through self-similar social networks. ACM Trans. Auton. Adapt. Syst. **5**(1), 1–29 (2010)
2. Andreou, P., Zeinalipour-Yazti, D., Pamboris, A., Chrysanthis, P., Samaras, G.: Optimized query routing trees for wireless sensor networks. Inf. Syst. **36**(2), 267–291 (2011)
3. Andreou, P., Zeinalipour-Yazti, D., Chrysanthis, P.K., Samaras, G.: Power efficiency through tuple ranking in wireless sensor network monitoring. Distrib. Parallel Databases **29**(1–2), 113–150 (2011)
4. Andreou, P., Zeinalipour-Yazti, D., Pamboris, A., Chrysanthis, P.K., Samaras, G.: Optimized query routing trees for wireless sensor networks. Inf. Syst. **36**(2), 267–291 (2011). doi:10.1016/j.is.2010.06.001
5. Azizyan, M., Constandache, I., Choudhury, R.R.: Surroundsense: mobile phone localization via ambience fingerprinting. In: MobiCom (2009)

6. Balke, W.T., Güntzer, U., Zheng, J.X.: Efficient distributed skylining for web information systems. In: EDBT, pp. 256–273 (2004)
7. Campbell, A., Eisenman, S., Lane, N., Miluzzo, E., Peterson, R., Lu, H., Musolesi, M., Fodor, K., Ahn, G.: The rise of people-centric sensing. IEEE Internet Comput. **12**(4), 12–21 (2008)
8. Chatzimilioudis, G., Konstantinidis, A., Laoudias, C., Zeinalipour-Yazti, D.: Crowdsourcing with smartphones. In: IEEE Internet Computing, IEEE Press, New York (2012)
9. Chen, S.K., Wang, P.C.: Design and implementation of an anycast services discovery in mobile ad hoc networks. ACM Trans. Auton. Adapt. Syst. **6**(1), 2 (2011)
10. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting: theory and optimization. In: Int. Inf. Sys. Conference, pp. 593–602. Springer, Berlin (2005)
11. Chun, B.N., Culler, D.E., Roscoe, T., Bavier, A.C., Peterson, L.L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. Comput. Commun. Rev. **33**(3), 3–12 (2003)
12. Das, T., Mohan, P., Padmanabhan, V., Ramjee, R., Sharma, A.: Prism: platform for remote sensing using smartphones. In: MobiSys (2010)
13. DBLP: DBLP Computer Science Bibliography (2010). http://dblp.uni-trier.de/xml/
14. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, New York (2002)
15. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
16. Eisenman, S., Miluzzo, E., Lane, N., Peterson, R., Seop-Ahn, G., Campbell, A.: Bikenet: a mobile sensing system for cyclist experience mapping. ACM Trans. Sens. Netw. **6**(1), 1–39 (2009)
17. Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The pothole patrol: using a mobile sensor network for road surface monitoring. In: MobiSys, pp. 29–39 (2008)
18. Gahng-Seop, A., Musolesi, M., Lu, H., Olfati-Saber, R., Campbell, A.: Metrotrack: predictive tracking of mobile events using mobile phones. In: DCOSS, pp. 230–243 (2010)
19. Gnutella: Gnutella peer-to-peer network (14 March 2000). http://gnutella.wego.com
20. Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05), VLDB Endowment, pp. 229–240 (2005)
21. Huang, Z., Jensen, C.S., Lu, H., Ooi, B.C.: Skyline queries against mobile lightweight devices in manets. In: Proc. of ICDE (2006)
22. Inamura, H., Montenegro, G., Ludwig, R., Gurtov, A., Khafizov, F.: TCP over second (2.5G) and third (3G) generation wireless networks. RFC 3481 (Best Current Practice) (Feb 2003). http://www.ietf.org/rfc/rfc3481.txt
23. Jia, J., Chen, J., Chang, G., Wen, Y., Song, J.: Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. Comput. Math. Appl. **57**(11–12), 1767–1775 (2009)
24. Kalogeraki, V., Gunopulos, D., Zeinalipour-Yazti, D.: A local search mechanism for peer-to-peer networks. In: 11th International Conference on Information and Knowledge Management (CIKM'02), McLean, VA, USA, pp. 300–307 (2002)
25. Ko, Y.B., Vaidya, N.H.: Location-aided routing (lar) in mobile ad hoc networks. Wirel. Netw. **6**(4), 307–321 (2000)
26. Konstantinidis, A., Aplitsiotis, C., Zeinalipour-Yazti, D.: SmartP2P: a multiobjective framework for finding social content in P2P smartphone networks. In: 13th International Conference on Mobile Data Management (MDM'12) (2012)
27. Konstantinidis, A., Costa, C., Larkou, G., Zeinalipour-Yazti, D., Demo: a programming cloud of smartphones. In: 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12), pp. 465–466 (2012)
28. Konstantinidis, A., Yang, K.: Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific MOEA/D. Appl. Soft Comput. **11**(6), 4117–4134 (2011)
29. Konstantinidis, A., Yang, K., Zhang, Q., Zeinalipour-Yazti, D.: A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks. New Netw. Paradig., Elsevier Comput. Netw. **54**, 960–976 (2010)
30. Konstantinidis, A., Zeinalipour-Yazti, D., Andreou, P., Samaras, G.: Multi-objective query optimization in smartphone social networks. In: 12th International Conference in Mobile Data Management (2011)
31. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: VLDB, pp. 275–286 (2002)

32. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: 16th International Conference on Supercomputing (ICS'02), New York, USA, pp. 84–95 (2002)
33. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS, pp. 84–95 (2002)
34. Ng, W.S., Ooi, B.C., Tan, K.L., Zhou, A.: Peerdb: a p2p-based system for distributed data sharing. In: International Conference on Data Engineering, p. 633 (2003)
35. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03), pp. 467–478. ACM, New York (2003)
36. Ra, M.R., Paek, J., Sharma, A., Govindan, R., Krieger, M.H., Neely, M.J.: Energy-delay tradeoffs in smartphone applications. In: MobiSys, pp. 255–270 (2010)
37. Rajagopalan, R., Mohan, C.K., Mehrotra, K.G., Varshney, P.K.: Emoca: an evolutionary multi-objective crowding algorithm. J. Intell. Syst. (2006)
38. Rajagopalan, R., Mohan, C.K., Varshney, P.K., Mehrotra, K.: Multi-objective mobile agent routing in wireless sensor networks. In: Proc. IEEE CEC'05, Edinburgh, Scotland, September 2005
39. Rana, R.K., Chou, C.T., Kanhere, S.S., Bulusu, N., Hu, W.: Ear-phone: an end-to-end participatory urban noise mapping system. In: IPSN, pp. 105–116 (2010)
40. Repantis, T., Kalogeraki, V.: Data dissemination in mobile peer-to-peer networks. In: 6th International Conference on Mobile Data Management (MDM'05), Ayia Napa, Cyprus, pp. 211–219 (2005)
41. Sarigöl, E., Riva, P., Alonso, G.: A tuple space for social networking on mobile phones. In: ICDE (2010)
42. de Silva, G.C., Aizawa, K.: Retrieving multimedia travel stories using location data and spatial queries. In: The 17th ACM International Conference on Multimedia, pp. 785–788. ACM, New York (2009)
43. de Silva, G.C., Yamasaki, T., Aizawa, K.: Sketch-based spatial queries for retrieving human locomotion patterns from continuously archived gps data. IEEE Trans. Multimed. **11**(7), 156–166 (2009)
44. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01), pp. 301–310. Morgan Kaufmann, San Francisco (2001)
45. Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., Eriksson, J.: 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, pp. 85–98. ACM, New York (2009)
46. Tomiyasu, H., Maekawa, T., Hara, T., Nishio, S.: Profile-based query routing in a mobile social network. In: 7th International Conference on Mobile Data Management, May 2006, p. 105 (2006)
47. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for peer-to-peer networks. In: Third International Conference on Peer-to-Peer Computing (P2P'03), 1–3 September 2003, pp. 102–109 (2003)
48. Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M.: Skypeer: efficient subspace skyline computation over distributed data. In: International Conference on Data Engineering, pp. 416–425 (2007)
49. Wang, S., Ooi, B.C., Tung, A.K.H.: Efficient skyline query processing on peer-to-peer networks. In: IEEE International Conference on Data Engineering (ICDE), pp. 1126–1135 (2007)
50. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: a wireless sensor network testbed. In: Information Processing in Sensor Networks. Fourth International Symposium on IPSN 2005, pp. 483–488 (2005)
51. Wu, P., Zhang, C., Feng, Y., Zhao, B.Y., Agrawal, D., Abbadi, A.E.: Parallelizing skyline queries for scalable distribution. In: EDBT'06, pp. 112–130 (2006)
52. Xu, B., Wolfson, O., Naiman, C.: Machine learning in disruption-tolerant manets. ACM Trans. Auton. Adapt. Syst. **4**(4), 23 (2009)
53. Zeinalipour-Yazti, D., Kalogeraki, V., Gunopulos, D.: Exploiting locality for scalable information retrieval in peer-to-peer systems. Inf. Syst. **30**(4), 277–298 (2005)
54. Zeinalipour-Yazti, D., Kalogeraki, V., Gunopulos, D.: Pfusion: an architecture for internet-scale content-based search and retrieval. IEEE Trans. Parallel Distrib. Syst. **18**(6), 804–817 (2007)
55. Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R.P., Mao, Z.M., Yang, L.: Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Proceedings of the eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS'10), pp. 105–114. ACM, New York (2010)

56. Zhang, Q., Li, H.: MOEA/D: a multi-objective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
57. Zheng, Y., Liu, L., Wang, L., Xie, X.: Learning transportation mode from raw gps data for geographic applications on the web. In: WWW (2008)
58. Zhu, L., Tao, Y., Zhou, S.: Distributed skyline retrieval with low bandwidth consumption. IEEE Trans. Knowl. Data Eng. **21**, 384–400 (2009)
59. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Case Study, pp. 292–301. Springer, Berlin (1998)
60. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**, 257–271 (1999)