Noname manuscript No.
(will be inserted by the editor)

Using Multi-Objective Metaheuristics for the Optimal Selection of Positioning Systems

Massimo Ficco · Roberto Pietrantuono · Stefano Russo

Received: date / Accepted: date

Abstract The interworking between cellular and wireless local area networks, as well as the spreading of mobile devices equipped with several positioning technologies pave the ground to new and more favorable indoor/outdoor Location-Based Services (LBSs). Thus, Wireless Internet Service Providers (WISPs) are required to take several positioning methods into account at the same time, in order to leverage the different features of existing technologies. This would allow providing LBSs satisfying the user-required quality of position in terms of accuracy, privacy, power consumption, and often, conflicting features. Therefore, this paper presents *GlobalPreLoc*, a multi-objective strategy for the dynamic and optimal selection of positioning technologies. The strategy exploits a pattern mining algorithm for future position prediction combined with conventional multi-objective evolutionary algorithms, for choosing continuously the best location providers, accounting for the user requirements, the terminal capabilities, and the surrounding positioning infrastructures. To practically implement the strategy, we also designed an architecture based on Secure User Plane Location (SUPL) specification to provide indoor and outdoor LBSs in interworking wireless networks exploiting *GlobalPreLoc* features.

M. Ficco
Dipartimento di Ingegneria Industriale e dell'Informazione, Seconda Università degli Studi di Napoli, Via Roma 29, 81031 Aversa, Italy.
E-mail: massimo.ficco@unina2.it

R. Pietrantuono and S. Russo
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy.
E-mail: {roberto.pietrantuono, stefano.russo}@unina.it

1 Introduction

Location-Based Services (LBSs) are considered one of the drivers in the market of telecommunication services [1]. Positioning of handset devices in wireless networks (3G/4G) is crucial to many nowadays popular services, including navigation, tracking, information and advertisement, and to more advanced services, e.g., for fraud detection, location-sensitive billing, health care, and traffic congestion avoidance. The spread of wireless hotspots into public and private areas (airports, malls, hotels, offices, etc.), and the availability of mobile terminals that support several positioning technologies (*e.g.*, GPS, Wi-Fi, Bluetooth, RFID), fosters the development of integrated indoor/outdoor positioning systems. Therefore, in order to provide value added LBSs, WISPs should be able to locate their users in both indoor and outdoor scenarios exploiting the most appropriate positioning technology depending on the context.

Positioning techniques and technologies are known to have very different features from each other. For instance, techniques exploiting proximity to 802.11 wireless cells are suitable for indoor scenarios but have modest accuracy, while GPS-based methods have high accuracy, but they work well outdoor and entail a high power consumption. Therefore, in the last years, several Application Programming Interfaces (APIs) and architectural solutions have been proposed to develop *location providers*, which integrate several positioning techniques and technologies [2–4]. However, none of them is able to leverage the different features of heterogeneous positioning systems in an optimal way with respect to the user requirements. Indeed, some positioning requests might require a high accuracy regardless power consumption, or response time, or privacy, while others might prioritize a low power consumption even

accepting a modest accuracy. The architecture and implemented protocols are required *i)* to be able to use several kinds of technologies and switching among them seamlessly, *ii)* to offer mechanisms for specifying requirements on the desired quality positioning requests, and *iii)* to satisfy these multiple and possibly conflicting objectives in an optimal and dynamic way.

In this paper, we present *GlobalPreLoc*, a solution to enable WISPs to infer indoor/outdoor location information optimally, by exploiting a position prediction pattern-mining algorithm and multi-objective metaheuristics to account for different quality objectives. The solution enables the dynamic selection of the best mix of positioning technologies available at a given place and time, accounting for the current position and predicted future movements of the terminal. At architectural level, specific context-aware mechanisms are defined to transparently and adaptively choose the most suitable technology during the user movement, according to the required quality of position (QoP), to the mobile device capabilities, and to the surrounding positioning infrastructure. The optimization policy is implemented with respect to the specified objectives (e.g., minimize the power consumption or computational cost) and constraints (e.g., accuracy), and considering the features of location providers available in the current and predicted future positions.

Besides defining the locators selection algorithm, we designed the actual architecture embedding the algorithm as an extension of SUPL (Secure User Plane Location) [6]. SUPL is a collection of open standard specifications that defines how location information should be transferred between terminals and applications through wireless networks (UMTS, Wi-Fi, etc.), independently from the positioning systems. Finally, to implement the entire solution, we also defined software abstractions that offer to LBSs a uniform access to heterogeneous positioning technologies, allowing to integrate commercial and open location technologies developed using the most widespread location APIs.

The paper is an example of soft computing application to the area of mobile terminal positioning and location-based services as a practical way to implement context-aware policies by multi-objective approaches.

The remainder of the paper is organized as follows. Sec. 2 provides basic background about SUPL. Sec. 3 presents the architectural details and the adopted information model. Sec. 4 describes the implemented approach based on Multi-objective Evolutionary Algorithms (MOEAs) for the optimal selection of location providers. Sec. 5 shows the results of the evaluation. Sec. 6 surveys the related state-of-the-art about positioning systems. Conclusions are described in Sec. 7.

2 Interoperability and standard for location-aware computing

The delivery of multi-provider LBSs demands for proper standards. SUPL [6] is an interoperability open standard, defined by OMA, for location-aware computing, that groups a set of specifications defining a way for exchanging location information between SUPL Enabled Terminals (SETs) and location applications. SUPL employs user plane data bearers, requiring IP-capable networks and minimum modification to the network elements. It allows using several positioning strategies, such as Assisted GPS (A-GPS), Enhanced Cell-Id, Time of Arrival (TOA), and Time Difference of Arrival (TDOA). SUPL defines how signaling and position information should be transferred between actors through wireless networks, independently from the positioning systems. It also defines security, privacy and charging functions.

Fig. 1 shows the SUPL architecture. The architecture includes the following entities:

- *SUPL Enabled Terminal* (SET): any device able to communicate with a SUPL network;
- *Mobile Location Service* (MLS): an application requiring position information to provide to LBSs;
- *SUPL Agent*: the service access point, which accesses the network resources to obtain location information;
- *SUPL Location Platform* (SLP): the core of the platform; SLP is the entity responsible for estimating and delivering the SET position. It is composed of the *SUPL Location Center* (SLC) and the *SUPL Positioning Center* (SPC). The SLC coordinates the operations in the network and interacts with the SET. It receives positioning requests, authenticates,

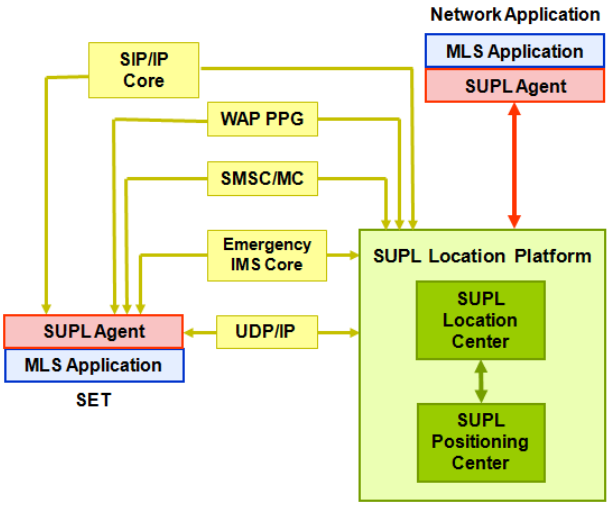


Fig. 1: SUPL architectural model

and checks that applications are authorized to request the user location. The SPC is responsible for operations needed for SET localization;
- *WAP PPG*, *SMSC/MC*, *UDP/IP*, and *SIP Core*: the communication mechanisms for conveying location request notification, used for communicating with a SET. They include: OMA Push (based on the Push Access Protocol and Push Over-The-Air Protocol), SMS, UDP/IP and SIP Push messages.
- *Emergency IMS Core*: For IMS (IP Multimedia Subsystem) emergency services, the SUPL location request notification is conveyed from the SLP to the SET via the Emergency IMS Core using SIP Push.

MLS Application and SUPL Agent can independently reside on the network or on the SET. If SUPL Agent resides in the network, the service is named *Network-Initiated*, otherwise it is named *SET-Initiated*.

In a Network-Initiated scenario, a SUPL Agent, to obtain the SET position, sends a location request to the SLP which the SET is associated with. The SLP starts a location session sending a message via SMS or WAP to the SET. Once a secure IP connection has been established, the SET determines first its coarse-grain position based on the identifier (Cell-Id) of the wireless cell it is connected to. Then, if necessary, it computes a finer position by a positioning method chosen among those supported by the SET or by the infrastructure and satisfying the required QoP.

In a SET-Initiated scenario, the SUPL Agent receives a positioning request from an application running on the SET. The SUPL Agent establishes a connection with the SLP, and then it sends a message (containing the Cell-Id) to begin a positioning session with the SLP. If the coarse-grain position computed (based on Cell-Id) meets the requested QoP, the session is finished, otherwise the communication proceeds as in the Network-Initiated scenario. Except for initial message delivery in the Network-Initiated scenario (in which an SMS or a WAP message is used), the transport is based on the TCP/IP protocol.

3 Location architecture for indoor and outdoor scenarios

In the following subsections, we first present the defined SUPL-based architectural model. We defined its requirements in [5]. Hereafter, we report the detailed design for what concerns the dynamic and optimal positioning strategy. After the high-level architecture, a description follows of: *i)* the location information model, and *ii)* the software design of handset- and network-side modules, and of the proposed API for supporting the integration with the most widespread location APIs.

3.1 High-level architecture

3.1.1 Architectural entities

The proposed architecture enables WISPs to provide LBSs that exploit dynamically both handset- and infrastructure-based location providers. To this aim, the architecture has to fulfill requirements of openness, strong WISPs interoperability, exploitation of different networks and positioning providers, and of context-related information to optimize cost-quality trade-offs. To satisfy these requirements in both Wireless Wide Area Networks (WWANs - *e.g.*, UMTS and HSDPA), and Wireless Local Area Networks (called Home Networks, HNs - *e.g.*, Wi-Fi), we devised a two-levels hierarchical SUPL-based architecture. The architecture is designed to support several kinds of networks with different location-sensing features: the classical WWAN infrastructures at the first level, and HNs at the second level. Each level includes a number of positioning cells managed by a different kind of SLP. *High level SLPs* are software components responsible for both managing user position requests and locating mobile users in the WWAN scenarios. *Low level SLPs*, called *Local SLP* (L-SLP), are software components enabled to locate mobile devices by HNs in public and private environments, without involving the WWAN provider, favoring the scalability of the architecture.

The L-SLP can be either integrated into a HN or located in a separate network entity. In the latter case, it can manage several HNs, and can locate the SETs within their coverage area. In order to locate users, the L-SLP has to manage technology-dependent data and topological information; for instance, the location-sensing technologies deployed in surrounding area, the correspondence between HN cell identifiers (*e.g.*, the SSID and the MAC address of the access point), and their geographic coordinates, as well as the topological description of the managed area. Note that using HNs in the private and public domains allows sharing indoor location information with applications and users connected to the WISPs' mobile or fixed network; thus, in order to use confidential information (*e.g.*, geographical coordinates) about public/private location-sensing systems (*e.g.*, access points, reader of RFID tags), proper (business) agreements need to be established with the public authority or the responsible for the private environment. Alternatively, several private and open geodatabase of wireless access points will be used for geolocation, such as Skyhook [46] and Geomena [49]. High level SLPs can manage multiple L-SLPs within their coverage area. Similarly to the SLP role as foreseen in the SUPL specification, they can act as *Home SLP* (H-SLP) or *Visited SLP* (V-SLP). In the former case, it is

responsible for retrieving the SET's position and interfacing with the MLSs. When a SET leaves the service area of the H-SLP of its WISP, a SUPL roaming occurs. In this case, to estimate the SET position, the H-SPL contacts the V-SLP to which the SET is connected, requesting the V-SLP to either provide just an initial position estimation (*e.g.*, based upon Cell-Id), or a fine-location estimation. To locate SETs in wide area scenarios, each high-level SLP is free to choose a positioning method according to the QoP requested by the MLS applications, the capability of the SET, and the positioning infrastructure deployed in the considered area.

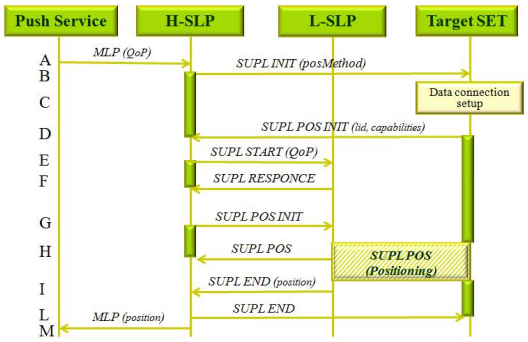


Fig. 2: Push communication scenario

3.1.2 Communication scenarios

Communication between the SLP and the SET depends on the provided service, *i.e.*, pull (i.e., SET-initiated) and push (i.e., Network-initiated) services. In any case, exchanged messages are the same foreseen by SUPL (see [6] for the exact content of SUPL messages), even though additional entities are present in our architecture (such as the L-SLP).

In the push service the MLS application, which resides in the network, requires the position of a SET. This can be the case of an advertisement information service, which requires to be notified as soon as a SET enters within a specific area (*e.g.*, a shopping mall). Supposing that the SET is located within the H-SLP area, and it is covered by some of the L-SLP's HNs (*e.g.*, a Wi-Fi network), the scenario can be summarized as follows (Fig. 2):

- *Step A*: The MLS application requests the SET position to the H-SLP. The request includes the QoP requirements.
- *Steps B-C*: The H-SLP creates a location session with the SET sending a SUPL INIT message via a SMS or a WAP message, which contains the positioning method it intends to use. The data connection can be established through a cellular interface (*e.g.*, GPRS, UMTS, HSDPA), or through one of its wireless interfaces (*e.g.*, IEEE 802.11), depending on the device equipment and on connection availability.
- *Step D*: Once a secure IP connection is established, the SET determines its location identifier (*lid*), which is the *id* of the access point or the cell with which it is associated. Then, it sends the identifier to the H-SLP in a SUPL POS INIT message to start a positioning session, including its supported positioning methods.
- *Steps E-G*: On the base of the received identifier, a coarse-grain position can be computed, for example by using services such as Google Latitude [50]. If the estimated position does not meet the QoP requirements, the H-SLP sends a SUPL START message to the L-SLP with which the target SET is associated (selected on the base of the received *lid*), in order to inform it that a finer SET position is required. Then a POS INIT message is forwarded to the L-SLP.
- *Steps H-I*: During a SUPL POS session, on the base of the QoP required by the MLS, the L-SLP computes a SET finer position with a method chosen among those supported by the SET and by the surrounding positioning infrastructure, and selected according to the algorithm MOEA-based strategy explained in Sec. 4. The position is sent to the H-SLP.
- *Steps L-M*: The H-SLP forwards the computed position to the requesting MLS application.

In the pull service (Fig. 3), the positioning is required by an application which resides on the requester SET. An example is an information service, which requires user position to obtain information about surrounding points of interest. The scenario can be summarized as follows:

- *Step A*: A handset-resident MLS application requests the SET position. The SET establishes a data connection with the H-SLP to start a positioning session. Then, it sends a SUPL START message containing its *lid*, the supported positioning methods, and the MLS QoP.
- *Step B*: The H-SLP uses the *lid* to compute a coarse position. If the position meets the requested QoP, it sends the position to the SET in a SUPL END message, and the session is finished; otherwise, it forwards the message to the L-SLP with which the target SET is associated.
- *Steps C-F*: The L-SLP sends a SUPL RESPONSE message (including the positioning method and technology to use), and waits for the SUPL POS INT message.

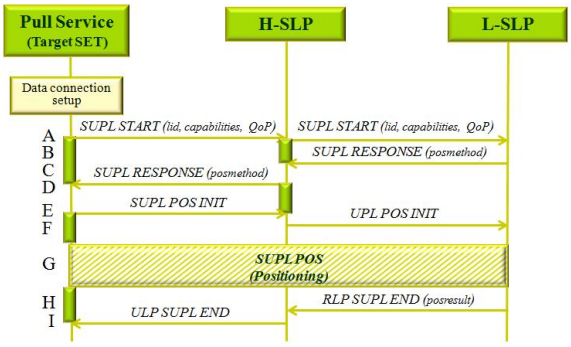


Fig. 3: Pull communication scenario

– *Step G*: During a SUPL POS session, the SET will provide its position. and the required QoP, the L-SLP computes a SET finer position.
– *Steps H-I*: The H-SLP forwards the computed position to the requesting MLS application.

The focus of this article is on the mechanism to allow MLS applications to exploit, in a transparent manner, multiple location providers, and choose the most suitable one based on their QoP requirements (e.g., accuracy, power consumption, response time). These are embedded in the SUPL POS messages. To choose the best method for a given objective function, some context-related information need to be obtained at runtime. For instance, to minimize power consumption and satisfy a QoP constraint, the terminal needs to know which techniques are available in a given area at a given time. To address such issues, a representation model of positioning information and the mechanisms for inferring dynamically such information need to be provided. These are described in the next two sections.

3.2 Information management of the covered area

A positioning system should exploit not only semantically rich spatial models – as those proposed in [51] – but also the characteristics of the available positioning infrastructure and positioning methods [1]. Specifically, to enable different positioning methods to be chosen transparently and to switch dynamically among them, some knowledge is required about the technologies available in the areas (both indoor and outdoor). In the proposed architecture, this information is managed by SLPs. The SLP adopts a hybrid location model. It expresses the location of a mobile device in terms of symbolic and geometric coordinates, representing a point or region of the physical world of interest to the end users; we call *zones* these locations. As for indoor environments, such as a building, each *zone* represents an area of interest, like a room or a hallway. As for outdoor, each *zone* could represent a coarse-grained location (*e.g.*, a garden, a street, a court). Combining the location representations of the hierarchical model and the graph-based solution, the model can take into account the interconnections and the spatial containment relationships among *zones* (*e.g.*, outdoor *zones* can be hierarchically divided into countries, states, cities and blocks; whereas indoor *zones* can be described by buildings, floors and rooms). The model includes the concept of *Area* as a set of *zones*. An *Area* groups adjacent *zones* covered by the same positioning technology. In the scenario shown in Fig. 4, four *Areas* are present: *Area* 1 is Bluetooth-based and overlapped to *Area* 2; *Area* 2 and *Area* 4 are covered by Wi-Fi technology; *Area* 3 by GPS. *RFAreas* are regions covered by RFID tags. *Areas* have *borderzones*, *i.e.*, transit *zones* among them. For instance, a building entrance hall is considered as *borderzone* of an indoor *Area* covered by WLAN infrastructure. Similarly, each outdoor *zone* close to the indoor *borderzone* is considered as an outdoor *borderzone*. *Borderzones* may be used to implement a mechanism for enabling different positioning technologies simultaneously or separately, as well as switching among them, according to the *Area* where the user is moving.

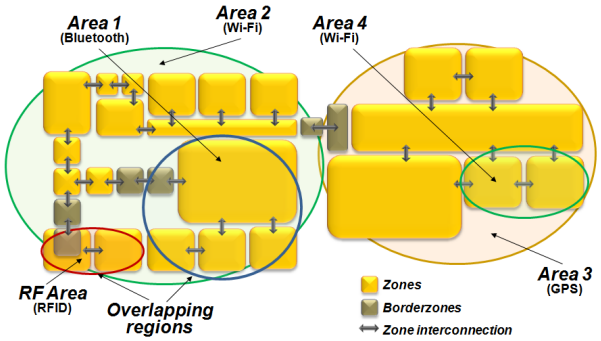


Fig. 4: Areas and borderzones scenario

In the designed architecture, each high-level SLP is responsible for managing the correspondence between the L-SLP and the associated wide area cell identifiers (Cell-Ids), whereas each L-SLP manages a fixed number of *Areas*, and the position information about the *zones* they cover. For each *Area*, it is necessary to take into account the wide area cell it belongs to, and the positioning technology therein deployed.

3.3 Low-level architectural model

At the low level, the core component of the architecture is the L-SLP, which is responsible for collecting context-

related information to choose and switch among the available location providers. A location provider may refer both to a single-source positioning approach, where one specific technology is used (e.g., GPS, Wi-FI), and, more generally, to multiple-source approaches (hereafter called multi-technology), where data from different technologies are merged together to estimate the position (also known as *data fusion*).

The L-SLP is the item that most contributes to confer flexibility (by the dynamic switch mechanism), efficiency (by supporting the optimal location provider choice according to the QoP), and extensibility (by the API definition). Fig. 5 depicts the low-level positioning architecture. It includes the applications, the SUPL entities, and the positioning subsystem equipped on the SET.

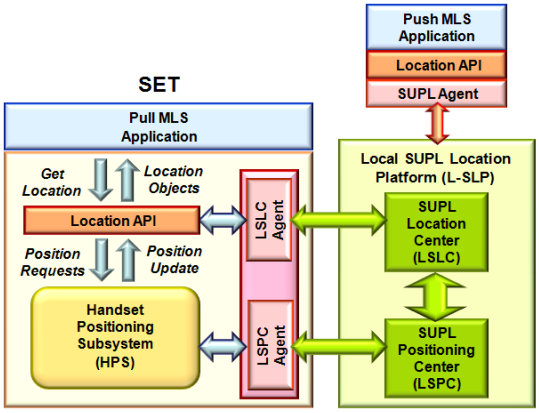


Fig. 5: The integrated architecture

The architectural model consists of:

- the *Mobile Location Service (MLS)*: it includes both pull and push applications. In order for MLSs to support and integrate different location providers, we propose a software abstraction (Location API), whose implementation can be mapped onto the most common positioning APIs (including Google Gears and JSR-179), or native APIs (*e.g.*, the S60 Location Acquisition API).
- the *Handset-based Positioning System (HPS)*: it is responsible for enabling the combined/separated use of different positioning technologies on the mobile device, according to the information received by LSLC. Moreover, it derives position data by using its location sensing equipment.
- the *Local SUPL-Location Center (LSLC)*: it coordinates the operations in the HN covered area, and interacts with the SET in order to choose the most appropriate positioning approach in a given *Area*, according to the required QoP, and to technologies supported by the mobile device and available in its coverage area.
- *Local SUPL-Positioning Center (LSPC)*: it is responsible for performing the positioning procedures. It interacts with the HPS and with the surrounding positioning infrastructure to infer the user position, and converts the positioning results into the desired format (*e.g.*, Cell-Id into location coordinates).

A more detailed description of all the conceptual entities and their interactions follows.

3.3.1 Location API

The Location API level provides application developers with a Java Application Programming Interface, which offers two main functionalities: (*i*) to infer mobile location, and (*ii*) to manage representations of location information in the mobile device. In particular, to support the integration of different location providers developed by the most widespread location APIs, we propose a high-level abstraction that includes the application functionalities provided by both JSR-179 specification and Google Gears API.

Fig. 6 represents the proposed Location API. The `LocationProvider` class provides the current location of the terminal to the application. Since a mobile device can have several location providers, the `Criteria` class allows to specify the criteria to choose the most appropriate `LocationProvider` object. The application can query the `LocationProvider` to retrieve current device location; alternatively, the application can ask it for retrieving periodic updates or the last know location.

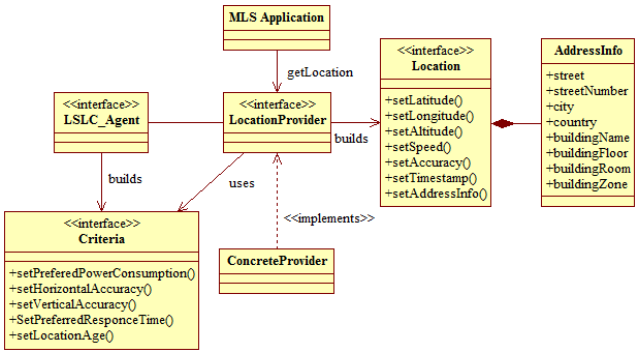


Fig. 6: Location API

The `Location` class represents the device's current location information in terms of timestamped coordinates, accuracy, speed, and information about the method used for positioning, plus optional textual address information. The logic used by applications to choose the

provider that best fits their requirements has been developed according to the SUPL specifications. In particular, the positioning method specified by the LSLC is communicated to the SET via the LSLC Agent and mapped on `Criteria` attributes, in order to identify the constraints for the location provider selection that best fits the required QoP.

3.3.2 Handset-based Positioning System

HPS is designed as a hybrid handset-based positioning system. It enables the development of pull location-aware applications, which can exploit multiple positioning technologies in the mobile device, including RFID, Bluetooth, IEEE 802.11, and GPS [47,48].

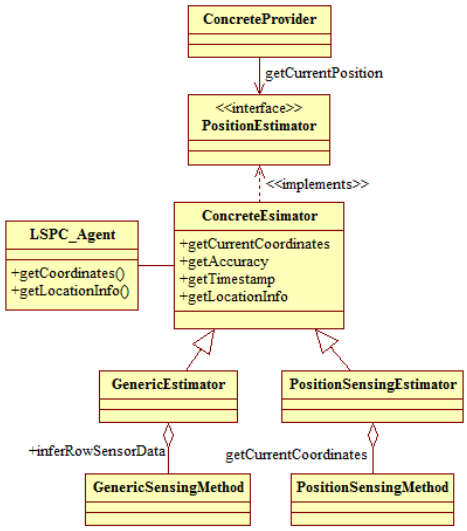


Fig. 7: The Handset-based Positioning System

Fig. 7 presents the HPS conceptual model to be implemented on the SET. The `ConcreteProvider` class implements the `LocationProvider` interface, by using specific location API (Google Gears or JSR-179). At lower level, the `ConcreteProvider` uses the `PositionEstimator` interface to retrieve geometric and symbolic position data used to build `Location` objects. `ConcreteEstimator` objects are enabled to interact with LSPC in order to associate a specific set of coordinates or symbolic representation with the current user location. The `ConcreteEstimator` uses `SensingMethod` objects to infer raw sensor data, through the following two strategies:

– *Position sensing*: If the `SensingMethod` relies on the direct availability of sensing devices (*e.g.*, a GPS terminal), the `ConcreteEstimator` can directly retrieve coordinates from it. Such an estimator is called `PositionSensingEstimator`.
– *Position inferring*: If the `SensingMethod` relies on a wireless networking infrastructure (*e.g.*, Bluetooth or IEEE 802.11), the `ConcreteEstimator` cannot directly retrieve position coordinates. Such an estimator, called `GenericEstimator`, has to interact with the LSPC to infer the current position. It sends the raw data (*e.g.*, the RSS of Bluetooth signals) obtained through the `SensingMethods`, and then it receives the position coordinates back (e.g., computed by a fingerprinting technique) [7].

3.3.3 Local SUPL Positioning Center

LSPC supports multiple heterogeneous positioning methods in HNs. In order to improve the availability and the accuracy of positioning, it integrates and correlates different data obtained through the surrounding infrastructure and the SET's positioning subsystem. Moreover, it is able to combine geometric and symbolic representations to represent the current user location. In particular, in order to transform raw sensor data into position information, it manages "*technology-dependent*" data related to the positioning infrastructure deployed in the covered area. For example, these may include: (*i*) the addresses and position of Wi-Fi access points deployed in a building, for supporting a triangulation RSS-based method; and (*ii*) the positions of wireless base stations, for Cell-Id positioning.

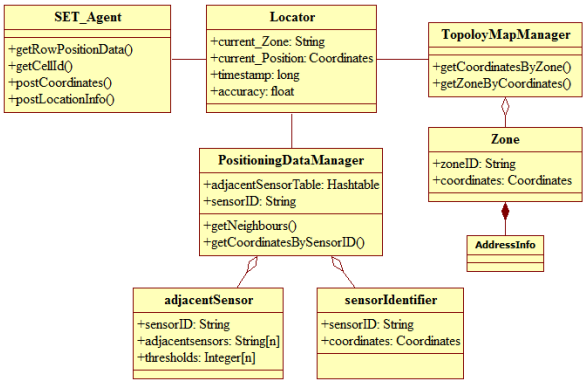


Fig. 8: The Local SUPL Positioning Center

Fig. 8 depicts the architecture of the LSPC subsystem. The `Locator`, which interacts with the SET through the SET_Agent, uses positioning techniques (*e.g.*, triangulation, proximity, fingerprinting, data fusion) to infer the current position, by combining "*static position*" information related to the covered area (*e.g.*,

the RSS radio map and the coordinates of the considered wireless access points) with raw data received by the SET or by the infrastructure (e.g., the RSS received from the SET's neighboring wireless sensors). The `sensorIdentifier` associates each wireless sensor (*e.g.*, Wi-Fi access points, RFID tags) with its identifier (*e.g.*, MAC address), whereas the `adjacentSensor` represents the list of adjacent sensors. The `Locator` uses the `TopologyMapManager` and the `PositioningDataManager` components to manage "*static position*" information and "*technology-dependent*" data respectively.

3.3.4 Local SUPL Location Center

The LSLC is the component managing positioning requests. It is in charge of selecting the optimum location provider that better fits the required QoP objectives, as well as avoid unnecessary use of resources. The *GlobalPreLoc* strategy is mainly implemented by this module. LSLC services are triggered when: (*i*) the SET performs the first position request; (*ii*) the device moves into a new *Area*[1]; (*iii*) the application changes QoP; (*iv*) the raw sensor data become unavailable (*e.g.*, for interference phenomena); and (*v*) periodically (*e.g.*, for power reduction).

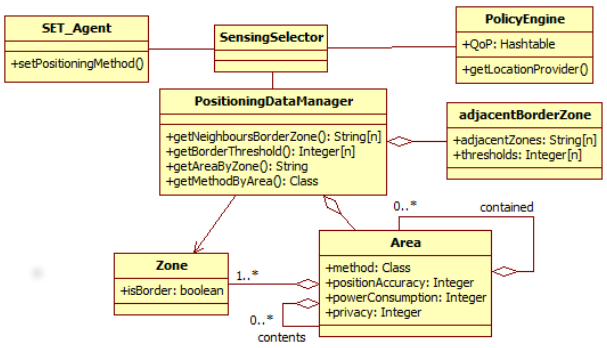


Fig. 9: The Local SUPL Location Center

Fig. 9 shows the high-level object model of the LSLC component. In order to enable context-aware control and management of the location providers, the `SensingSelector` must: (*i*) choose the right location provider according to the QoP, the current state of the SET, and the characteristics of the positioning infrastructure where the SET moves, and (*ii*) enable a new technology on the mobile device, disabling the previous one (if necessary). Therefore, it implements mechanisms and algorithms to enforce the policies for the dynamic selection of location providers and for the automatic switching among positioning technologies. The `PolicyEngine` implements specific policies used to select the location provider on the base of the application requirements, such as the required accuracy, power consumption, expected response time, or privacy level. All these features strictly depend on the adopted positioning techniques and the underlying technology that each location provider uses: for example, the power consumption depends on the kind of used interface, if it is power-saving or not, if it is integrated in the SET or available by an external device (*e.g.*, a GPS device interacts with the SET via Bluetooth), or if it uses an infrastructure-based approach. Similarly, the accuracy is tied to the technique used, as there are great differences between GPS, Bluetooth- and WiFi-based positioning techniques.

Tab. 1 shows an example of classification of technologies with respect to the considered features (accuracy, power consumption, privacy and security level, response time to serve the positioning request, and cost in terms of number of exchanged messages between the SET and the L-SLP). Location providers can be handset-based or infrastructure-based, and can adopt different technologies (e.g., Wi-Fi, Bluetooth), including "multi-technology", where sensors data fusion is used to combine measurements coming from different technologies. The column of the Table comprise all the features that we consider in our work. Additionally, we assume to use the fingerprinting positioning technique for handset-based approaches and proximity technique for infrastructure-based ones. In the following, we put the focus on the strategy we implemented for the policy engine to select the best mix of location providers.

4 Optimal selection of location providers

4.1 Overview

Each positioning request coming from an MLS application includes the policy for location provider management, expressed by the required QoP. When different policies are simultaneously required (*e.g.*, a high accuracy and a low power consumption), there may be a conflict. Thus, there is the need of coping with several possible conflicting objectives and constraints, and of choosing the best location provider satisfying the QoP requirements.

To introduce *GlobalPreLoc*, let us first consider the features of a QoP request. The `PolicyEngine` treats *power consumption, accuracy, cost*, and *response time* as numerical variables (expressed, respectively in: Ampere-hour, meters, number of messages, milliseconds); pri-

[1] An area, as defined in section 3.2, groups adjacent zones covered by the same positioning technology; the movement to a new area occurs whenever a SET enters an area covered by a positioning technology different from the current one

Table 1 Features for location provider management (H: Handset; I: Infrastructure)

Positioning Approach	Technology	Accuracy	Power Cons.	Privacy	Security	Response Time	Cost
H-based	Wi-Fi	high	high	low	high	low	medium
H-based	Bluetooth	high	low	low	high	low	medium
H-based	RFID	high	low	low	low	low	medium
H-based	GPS	medium	high	very high	very high	very low	medium
H-based	Wireless cell	very low	very low	high	high	low	very low
H-based	Multi-techn.	very high	very high	very low	medium	medium	very high
I-based	Wi-Fi	low	medium	high	high	low	very low
I-based	Wireless cell	very low	very low	high	high	low	very low

vacy and security are instead expressed by an ordinal scale from 1 to 5, with higher number indicating lower privacy/security (see [6] for a detailed description of privacy and security mechanisms offered by SUPL). A possible simple mechanism is to implement a greedy strategy, where the `PolicyEngine` selects the best location provider based on the current position and available technology. For instance, let us discretize the time in a set of time points $t_k \in \{t_0, \ldots, t_n\}$. Consider that a positioning request is issued at time t_n. A greedy approach would manage the requests by selecting, at any time point $t_{n+1}, t_{n+2}, \ldots$, the best solution satisfying the QoP at that time. To deal with possible conflicting requests, such a strategy could allow the user to specify a set of priorities among constraints, preferring the policy with the highest priority.

Such a strategy is simple to implement, but it is not able to opt for the best solution in a given temporal range. In fact, the complexity of the problem raises when one considers that the best location provider at time t_n depends not only on the characteristics of the positioning technologies available in the current position, but also on how the SET will move in the next future. For instance, if *i)* the QoP requires a high accuracy level and, secondarily, also a minimal power consumption, and *ii)* the SET is located in outdoor, than the best choices could be either "GPS" or "Multi-technology" (the latter providing a better accuracy, but higher power consumptions), depending on future positions. If the SET is going to move to an indoor area with available a Bluetooth or RFID technology (with a low power consumption), than the best choice for t_n could be "Multi-technology", as the indoor future position will allow using a low-consumption location provider. If the SET is going to move outdoor, than the GPS could be a better option, because its power consumption is lower. In the next section, we introduce the optimization strategy developed to support such kinds of policy.

4.2 The global MO evolutionary strategy

For what said, the `PolicyEngine` should consider, for an optimal choice, the prediction of future movements of the SET. In the following, the problem is formulated as a multi-objective optimization problem, with objectives depending on the *QoP*, and constraints depending both on the *QoP* and on data about movement prediction. Such a strategy is called `GlobalPreLoc` (globally-optimal, prediction-based, location provider selection). We address the described problem by two complementary approaches: *i)* a *pattern mining* approach for predicting future movements of the SET according to past positions, and *ii)* a periodic *optimization approach* that, using data on predicted future positions, chooses the best set of location providers for the current and future positions, matching the QoP objectives and constraints.

Specifically, considering a set of equally-spaced discrete time points $t_0, \ldots, t_n$, let us suppose the algorithm is run at time t_n. In the first phase, it predicts the future movements of the SET (in terms of visited *zones*), for a time window $W = \{t_{n+1}, \ldots, t_{n+N}\}$; then, based on this prediction, the algorithm looks for the best set of location providers for each future time point of W. Note that, the most important output is the future-aware choice for the immediately next time point, t_{n+1}; the choices given by the solution for all the other time points of W can be updated by running the algorithm again in any t_{n+k}, with $k > 1$. This may happen for two reasons: *i)* the algorithm is run with the goal of keeping always the most updated prediction and optimal solution with respect to the next N time points; *ii)* the algorithm is run because the prediction of future positions, made at time t_n, turned out to be wrong at time t_{n+k}. The decision on how often we run the algorithm in W is important because it regulates the performance in terms of trade-off between computational cost and the opportunity of having up-to-date solutions with respect to the problem variability. This point is further discussed in the evaluation section.

4.2.1 Prediction of future position

This phase aims at predicting locations to which the SET will move in the next set of time points $t_h \in W$. There are several approaches toward this aim, that can be distinguished in: *(i)* state-space models, *(ii)* template matching techniques, and *(iii)* data mining techniques. State space models describe the spatial movement through sequence models, such as hidden Markov models [8], Conditional Random Fields (CRFs) [9], or their extensions [10,11]. They are suitable to capture the movements as transitions among states, supporting the generation of possible future visits and the estimation of an associated probability, but they suffer from high training complexity [8]. Template matching approaches compare extracted features to pre-stored patterns or templates, using some similarity metrics specifically defined for comparison (e.g., those used for string-matching problems) [12]. These techniques have also been reported to have issues with high runtime complexity, noise intolerance, or spatial activity variation.

For our purpose, we adapt a technique falling in the category of data mining, as its runtime complexity is low compared to state-space models and template matching techniques. Data mining techniques in this field aim at extracting a set of association rules on trajectories patterns, then used for prediction. The works in this category usually define a model based on frequent patterns, acting on a trajectory defined as an ordered sequence of locations tagged with time-stamps [13–16]. Many of them are based on a modified version of the *Apriori* algorithm [17,14] and report high accuracy values, up to 80% [14]. In this work, we leverage the variant described in [17], called *WhereNext*, where patterns are equipped with temporal information and an evaluation function is defined allowing to choose the best set of patterns for the construction of a good prediction model. A brief description of the adopted algorithm follows.

The WhereNext scheme

The *WhereNext* strategy makes use of the concept of *trajectory*, which is a commonly adopted abstraction in location prediction problems. A trajectory is defined as a spatio-temporal sequence: $T =< x_0, y_0, t_0 >, \ldots, < x_n, y_n, t_n >$, where t_i $(i = 0, \ldots, n)$ denotes a timestamp such that $t_i < t_{i+1}$, $\forall \;\; 0 < i < n$, and $(x_i,\ y_i)$ are points in the bi-dimensional space (R^2). In other words, each point $l_i =< x_i, y_i, t_i >$ indicated the object in position x_i, y_i at time t_i.

The algorithm aims at constructing a set of association rules on the locations traversed by users, in order to apply them for prediction whenever a new trajectory is observed. Specifically, the rules are constructed on a set of "regions" in which the space is divided. Regions are what we called *zones* in our information model.

The first objective of *WhereNext* is to mine the most frequent trajectories from the ones observed in the past and stored in a database of trajectories. To this aim, the notion of *T-patterns* is borrowed, as defined in [18] and used in many works. A *T-pattern* is a description of frequent trajectories that consider both the space (divided in regions) visited and the time of the visit in term of duration. A *T-pattern* is defined as: *a pair* (S, A)*, where* $S =< R_0, \ldots, R_n >$ *is a sequence of regions, and* $A = \alpha_1, \ldots, \alpha_n$ *is the temporal annotation of the sequence. A T-pattern is represented as* $(S, A) = R0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2}, \ldots \xrightarrow{\alpha_n} R_n$. The last notation indicates that the object has been in region R_0, then moved to R_1 and took α_1 time units; then it moved to R_2 in α_2 time units, and so on.

The *T-pattern* mining algorithm to extract the most frequent traversed patterns considers two values, σ and τ, which represent a spatial and a temporal minimum threshold for pattern selection, respectively. The former refers to the minimum spatial density required to a zone in order to be selected as "frequent" region; it is computed by considering each single trajectory and incrementing the density value of all the zones containing any of its points. The latter is the minimum time the object is required to be in a region.

The *T-patterns* are represented in a model called *T-pattern Tree*, which allows for an efficient querying of the rules through the tree navigation. From the tree, the association rules are, in fact, immediate. They are of the form: $R0 \xrightarrow{\alpha_1} R_1 \xrightarrow{\alpha_2} R_2 \Rightarrow^{\alpha_3} R_3$, meaning that, if the trajectory matches the pattern made up of R_0, R_1 and R_2 satisfying temporal properties depending on α_1 and α_2, then the predicted next region is R_3. The *T-pattern Tree* representation allows to compactly represent these rules and to manage the many possible rules associated with each *T-pattern* (e.g., $R0 \xrightarrow{\alpha_1} R_1 \Rightarrow^{\alpha_2} R_2 \xrightarrow{\alpha_3} R_3$ is another rule of the previous pattern). More formally, the *T-pattern Tree* is defined as a triple $PT = (N, E, Root(PT))$, where N is a finite set of nodes, E is a set of labeled edges and $Root(PT)$ is a fake node representing the root of the tree. Each edge from a node u to v is labeled with a time interval int, of the form $[time_{min}, time_{max}]$, representing the travel time interval of the transition from node u to v. The algorithm for the construction of a *T-pattern Tree* is reported in [17].

WhereNext acts in two main phases: first, the database of trajectories is queried in order to extract the set of trajectories of interest, then used to build the mentioned *T-patterns Tree*. This includes the most frequent

movements of users. In the second phase, given an observed trajectory T, the *T-pattern Tree* is used to apply the association rules giving the prediction of the next expected location of T. To perform the prediction, *WhereNext* assigns a punctual score to each node, indicating the goodness of the node r, from the prediction point of view, with respect to T (i.e., its likelihood of being reached by T, given that it already reached the parent node $r-1$). The score is assigned according to the spatio-temporal distance of the node with respect to the trajectory, regulated by a spatio-temporal window wherein the moving object will be after the time specified in the edge towards r. Thus, if this trajectory's window intersects the region of the node, the node's punctual score is maximum (the spatial density value of the region); otherwise, if the window needs to be enlarged by a temporal threshold th_t to intersect the region, the score is the maximum divided by the time distance between the intersection point and the region; if the (enlarged) window does not intersect the region, the score is the maximum divided by the weighted sum of the spatial and temporal distances between the region and the nearest point of the trajectory. Finally, if the distance between is greater than a spatial threshold th_s, the score of the node is 0.

From the punctual score assigned to each node, a path score is derived that denotes the goodness of an entire path *w.r.t.* to T. The path score is obtained from punctual scores by means of several functions; *Wherenext* defines three aggregation functions, consisting in the average, the sum, and the maximum of punctual scores on nodes. Given a trajectory T and a *T-pattern Tree*, the algorithm computes the path score relative to T for each path, according with the aggregation function selected. When the tree is visited, the predicted path (i.e., the association rule to apply) is the one with the best score; the region associated with it (i.e., the *consequent* of the association rule) is returned as the predicted next region. The *T-Patterns Tree* is of course updated from time to time.

The efficiency of the described algorithm can be improved if we can consider the personalized mobility profiles of each user, namely if we can associate user's identity to mobility patterns. In this way, the search for the most likely path can exploit the repetitiveness of mobility patterns of a single user. However, in such a case, there might more severe privacy concerns, as the needed information is about the mobility profile of individual users. The optimization could therefore be implemented whenever such a piece of information is allowed to be used by the service provider for satisfying the request of the MLS application (different MLS applications can have different privacy requirements).

To apply the algorithm for our problem, suppose that the algorithm is recalled at time t_n. This means that it is applied to the trajectory $T_n =< l_0, \ldots, l_n >$, with $l_i =< x_i, y_x, t_i >$, obtaining a prediction of trajectory T_{n+1}, namely
$\hat{T}_{n+1} =< l_0, \ldots, \hat{l}_{n+1} >$. To get predictions for t_{n+2}, the algorithm is applied again, assuming the correctness of $\hat{T}_{n+1}$, obtaining $\hat{T}_{n+2}$, and so on up to $\hat{T}_{n+N}$. Predictions of the farthest time points in the future are of course subject to a greater error probability. If a prediction error is experienced at some point t_h, *WhereNext* is run again in order to obtain an updated prediction, which the MOEA algorithm will rely on. Given the set of J known zones in which the SET can be, denoted with $Z = \{Z_1, \ldots Z_j, \ldots Z_J\}$, the output of applying *WhereNext* at time t_n is the set of zones in which the SET is expected to be in the next $t_{n+1}, \ldots, t_{n+N}$ timestamps[2]. These are used for global optimization purpose.

4.2.2 Optimization Problem Formulation

The information from prediction is used to formulate the optimization problem, presented in the following.

Notation, Representation, and Constraints

Let us introduce some notations useful in the following:

- $LP = \{lp_1, lp_2, \ldots, lp_i, \ldots, lp_M\}$: the set of location provider chosen;
- $W = \{t'_1, \ldots, t'_k, \ldots, t'_N\} = \{t_{n+1}, \ldots, t_h, \ldots, t_{n+N}\}$: time window over which the selection of location provider is carried out. We denote it with t' to distinguish the future timestamps, for which we want the prediction and the optimization, from the past timestamps used by the prediction algorithm for training purposes.
- $x_{i,k} \in 0, 1$: a binary variable denoting the choice of the location provider lp_i for the time instant t'_k.
- $PZ = \{pz_1, \ldots pz_k, \ldots pz_N\}$: set of predicted zones by the prediction algorithm in which the SET is expected to move in the next N time instants. Each zone is associated with a map of deployed location providers; this means that $x_{i,k}$ is forced to be 0 whenever the zone predicted at time t'_k (pz_k) does not deploy the location provider lp_i. We denote with ZC_k a vector with the list of constraints regarding the zones for the t'_k time instant. An entry of ZC_k is the index i for which: $x_{i,k} = 0$. Thus, we have N vectors, one per time point, each having, in general, different size.

[2] Note that, the zones predicted at each future timestamp are not necessarily different from each other.

- $F = \{Power, AccuracyError, Privacy, Cost, ResponseTime, Security\}$: set of features of location providers that the SET can use. Each location provider is associated with a table containing the information on the typical performance for each feature F (namely, the power consumption per time unit, the expected accuracy, cost, response time, security and privacy level). This information is widely available through the characterizations provided in the literature ([29]); alternatively, they can be measured on the field in a preliminary training phase. The value of a feature is denoted by its lowercased name (namely: *power*, *accuracy_error*, *privacy*, *cost*, *response_time*, *security*).
- $QoP_{obj} = \{Obj_1, Obj_2, \dots Obj_r, \dots Obj_R\}$: set of optimization objectives required by the MLS application to the `PolicyEngine`. Each objective refers to one of the features in F:
  $F_r = \{Power \mid AccuracyError \mid Privacy \mid Cost \mid ResponseTime \mid Security\}$. The value of a generic feature F_r is denoted by f_r. Each objective can be expressed as a *minimization* goal.
- $QoP_{constraints} = \{c_1, c_2, \dots c_q, \dots c_Q\}$, each QoP constraint is a bound on one of the features in F; we denote with the subscript l or u a lower or upper bound respectively, so as: $C_q = power_u$ expresses the constraint $power \leq power_u$, $C_q = privacy_u$ expresses the constraint $privacy \leq privacy_u$ and so on. Generically, $f_{r,\{l|u\}}$ denotes the lower or upper bound of the feature value f_r.

The objective is to select the best set of location providers for each discrete time point of the time window optimizing the QoP_{obj}, while satisfying the $QoP_{constraints}$. Feasible solutions are assignment of location providers lp_i to time instants t'_k; representation is therefore binary, with decision variables being $x_{i,k}$.

Objective and model

The set of objective functions associated with the objectives $Obj_1, \dots Obj_r, \dots Obj_R$ are:

$$Obj_r = \sum_{k=1}^{N} \sum_{i=1}^{M} x_{i,k} f_r \quad r = 1, \dots, R$$

where the MLS application choses the R objectives to optimize among the set of features F. The set of constraints is expressed by $QoP_{constraints}$, and by ZC_k vectors (namely, the zone constraints, indexed by j; $zc_{k,j}$ is an item of ZC_k). An additional constraint is that only one location provider can be chosen as the best one in a given time interval. This leads to the requirement: $\sum_{i=1}^{M} x_{i,k} = 1 \ \ \forall k \in W$. The formulation is therefore:

$$min \quad Obj_r = \sum_{k=1}^{N} \sum_{i=1}^{M} x_{i,k} f_r \qquad r = 1, \dots, R \tag{1}$$

subject to:

$$\begin{array}{l} -QoP \ \ constraints: \\ c_1 : f_r \ \ \{< \mid \leq \mid > \mid \geq\} \ \ f_{r,\{l|u\}} \qquad r \in [1, R] \\ c_2 : f_{r'} \ \ \{< \mid \leq \mid > \mid \geq\} \ \ f_{r',\{l|u\}} \qquad r' \in [1, R] \neq r \\ \dots \\ c_q : f_{r^{q-th}} \ \ \{< \mid \leq \mid > \mid \geq\} \ f_{r,\{l|u\}} \\ \qquad\qquad\qquad\qquad\qquad\qquad r^{q-th} \in [1, R] \neq r', \dots r^{q-th-1} \end{array}$$

$$\begin{array}{l} -ZC_k \ \ constraints \ \ sets: \\ x_{i,1} = 0 \qquad \forall i : i = zc_{1,1}, zc_{1,2}, \dots i = zc_{1,|ZC_1|} \\ x_{i,2} = 0 \qquad \forall i : i = zc_{2,1}, zc_{2,2}, \dots i = zc_{2,|ZC_2|} \\ \dots \\ x_{i,N} = 0 \qquad \forall i : i = zc_{N,1}, zc_{N,2}, \dots i = zc_{N,|ZC_N|} \end{array}$$

$$\sum_{i=1}^{M} x_{i,k} = 1 \qquad \forall k : k = 1, \dots, N.$$

Handling of Multiple Solutions

The objectives are measured on orthogonal scales, so we use Pareto optimality, wherein a solution X is said to dominate another solution Y, if X is no worse than Y in all objectives and it is strictly better than Y in at least one objective. Using Pareto optimality, several approaches can be implemented to select the knee point of a Pareto front [19]. We leave the decision on how to deal with the solution set to the MLS application requiring the optimization, allowing it prioritizing one objective over another.

We adopt the simple approach in which the MLS application (hence the user) is allowed to specify a set of weights for the pursued R objectives, $w = 1, \dots w_r, \dots w_R$ (so that $\sum_{r=1}^{R} w_r = 1$ and $0 \leq w_r \leq 1$), denoting the importance of each objective. Let us denote the set of the R fitness values (one per objective) of a solution X as: $Y(X) = \{y_{1,x}, \dots, y_{r,x}, \dots, y_{R,x}\}$. We normalize these values in [0,1] over the entire Pareto front: $y'_{r,x} = \frac{y_{r,x} - min_x(y_{r,x})}{max_x(y_{r,x}) - min_x(y_{r,x})}$. The chosen solution X^* is the one with the maximum utility function value: $U(Y'(X)) = -\sum_{r=1}^{R} w_r \cdot y'_{r,x}$.

Computational Search

We use three different metaheuristics in this study in order to infer the best performing one for the formulated problem. We consider: NSGA-II [20], SPEA2 [21],

IBEA [22]. They all are well-known evolutionary algorithms (EAs), which are among the most popular metaheuristics for solving MO problems.
Adopting three schemes leads to have 3 variants of the *GlobalPreLoc* strategy, depending on the selected MOEA algorithm. We denote them as: $GlobalPreLoc_{NSGA}$, $GlobalPreLoc_{SPEA2}$, and $GlobalPreLoc_{IBEA}$. Their performance is compared in the experimental section with respect to scenarios we are interested in for the positioning problem.

4.2.3 Frequency of application

As mentioned in the previous sections, depending on the desired trade-off between computational cost and optimality, several policies are possible for applying *GlobalPreLoc*. The trade-off is regulated mainly by the frequency by which we run the algorithm. In fact, considering the past time units, $\{t_0, \ldots, t_n\}$, suppose the algorithm is applied at time t_n, and give the zones prediction and the corresponding optimal location provider selection for the time window $W : \{t_{n+1}, \ldots, t_{n+N}\}$. Then the algorithm can be run again after N time units, or it can be applied again after 1 time unit to update the global solution for the next N units, taking the two extremes. Let us assume that the algorithm is run every F time units. The optimal solution from the QoP point of view, but also the most expensive one, is to run the algorithm at every time unit $t_h \in \{t_{n+1}, t_{n+N}\}$, namely $F = 1$, considering the window W sliding forward of 1 time unit. In this way, the solution at any time accounts always for the position prediction in the next N time units (in this sense, it is optimal). On the other hand, the worst but cheapest solution is to run the optimization only each N time units, namely $F = N$, meaning that the solution found at t_n is not changed until t_{n+N} and is optimal only for the time window $\{t_{n+1}, t_{n+N}\}$. In other words, the solution does not consider the predictions at times successive to t_{n+N} (i.e., $t_{n+N+1}, t_{n+N+2}, \ldots$) until the next evaluation at time t_{n+N}. The parameter F may be important to decide the trade-off between optimality and execution cost. Any intermediate solution between $F = 1$ and $F = N$ makes sense. Additionally, to deal with possible prediction errors, we decided to act as follows, whenever $F > 1$ (e.g., F=5): the prediction is carried out either each F time points or whenever the actual *zone* in which the SET is at time $t_h \in [t_{n+1}, t_{n+F}]$ is different from the predicted one. In this case, the prediction at time t_n was wrong, and thus the algorithm is run again although $t_h < t_{n+F}$. In the next section, *GlobalPreLoc* is evaluated with respect to several aspects, including its sensitivity wit respect to this point.

5 Evaluation

We evaluate *GlobalPreLoc* by running it under all the mentioned MOEAs, in order to assess the performance with respect to several aspects. Specifically, the evaluation aims at addressing the following research questions:
RQ1 (*Validation*): How does the three *GlobalPreLoc* strategy perform compared to a random selection strategy? This is a typical question performed as a preliminary 'sanity check'; in fact, any intelligent computational search technique is expected to outperform random search unless there is something wrong in the formulation.
RQ2 (*MOEA solute quality comparison*): Which of the adopted MOEAs provides better solutions for the location provider selection problem? This question focuses on the comparison among MOEAs according to some common performance metrics regarding the goodness of the proposed Pareto solutions.
RQ3 (*Cost comparison*): Which of the adopted MOEAs performs better in terms of computational time? As a MOEA is not run just once in the *GlobalPreLoc* scheme, we are interested not only in the goodness of the solution from the optimality point of view, but also on how long it take, and thus how much it is computationally expensive (namely, we consider the computational time as indicator of computational "cost").
RQ4 (*Sensitivity*): What is the impact of the frequency of application of *GlobalPreLoc* on the solution cost-quality trade-off? Since, *GlobalPreLoc* foresees to run a MOEA instance from time to time, depending on the width of the time window and on the frequency of application, several solutions can be obtained depending on such tuning. We therefore assess the algorithms in various settings.

In the following, we describe: *i)* the evaluation metrics by which the algorithms are compared to each other; *ii)* the scenarios under which we compare the algorithms; *iii)* the method used to tune the parameters of the MOEAs; *iv)* the sensitivity analysis with respect to the algorithm application frequency. From this, the total number of runs comes out, which turned out to be 180 combinations each one repeated 100 times. Algorithms implementation and experimental settings have been carried out by using jMetal [23], an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for multi-objective optimization problems[3].

[3] jMetal is freely available at *http://jmetal.sourceforge.net/*

5.1 Evaluation Metrics

Unlike single-objective evaluation, multi-objective algorithms cannot be evaluated by a direct comparison among solutions. To provide a quantitative assessment, we employ two well-known indicators: the *Hypervolume* (HV) [24], and the *generational distance* (GD) [25]. To compute these, we normalize fitness values to avoid unwanted scaling effects [23] and compute a reference Pareto front of the solutions RF, by considering the union of the reference fronts of the approaches compared. The HV is one of the most accurate indicators; it calculates the volume, in the objective space, covered by members of a non-dominated set of solutions from an algorithm of interest. The larger this volume, the more it covers of the non-dominated solution space, hence the better the algorithm. It provides a measure taking into account the convergence and diversity of the obtained approximation set, and allows measuring the performance of different algorithms as compared to each other. The GS is the average distance between the set of solutions S, and the reference front RF. It is computed as the average n-dimensional Euclidean distance between each point in S and its nearest neighbour in RF. Finally, other than measuring the quality of provided solutions, we need also to care about their computational cost. We consider the average time (T) for one run of the algorithm (namely, for computing one solution set), as indirect measure of computational cost.

As evolutionary algorithms are of stochastic nature, we need to adopt inferential methods to interpret results correctly. To this aim, we perform 100 independent runs for each combination of algorithm and scenario. The HV indicator and GD indicators are then computed for each run. They are compared by means of the non-parametric Wilcoxon test [26], being it robust to non-normality, where the Bonferroni correction is applied to account for multiple comparison protection. All the statistical tests are performed with a confidence level of 95%.

5.2 Setting of scenarios

We analyze a number of scenarios in which the location provider selection problem may occur. Each scenario is characterized by how many objectives are required to optimize (and the related constraints), and on the characteristics of the zones in which the SET moves. To this aim, we consider the following parameters to generate scenarios:
1) Number of objectives: we consider three different situations, where we want to optimize 2, 3, or 4 objectives altogether, and also set a bound for their value. They are: power consumption and accuracy (for the 2 objective case), plus privacy level (for the 3 objective cases), plus response time (for the 4 objective cases).
2) Number of Positioning Infrastructure Objects (PIO): PIOs represent fixed objects, which can be used to localize the terminals (e.g., Bluetooth or WiFi antennas, RFID tags). In the scenarios, we vary the number of PIOs (and therefore the number of technologies available in each zone), to represent a scenario with larger and smaller number of available location providers. We consider a basic scenario in which the number of zones is fixed to 20, and the number of data points in the window W is fixed to N=10. Based on this, we take four different number of PIOs: 40, 60, 80, 100, leading to an average number of location providers per zone of 2, 3, 4, and 5 respectively. On the other hand, we determine the type of PIO by randomly selecting among the available types (i.e., Bluetooth, WiFi, GPS etc.), with equal probability of selecting each type. The available types of PIO are the ones of Table 1, which are 8. Note that, for the purpose of evaluating the MOEA algorithms, the characteristics of the zones are not relevant, because the needed parameters to set up the optimization problem are those describing the deployment of technologies in each zone. This configuration generates 12 scenarios. On each scenario, we run the three algorithms ending up to 36 *instances*, each one executed 100 times for statistically relevant analysis.

5.3 MOEA parameters

The tuning of algorithmic parameters is a necessary, and often unreported, information for allowing replication of experiments. We adopted the following procedure for parameters tuning. Specifically, we set up a maximum number of fitness evaluations for all the algorithms (500,000), in order to make a fair comparison among different parameter configurations. Keeping this value fixed, we varied the population size and number of generations as in Table 2. We differentiate five settings, similarly to [27]: *very small* (VS), *small* (S), *medium* (M), *large* (L), and *very large* (VL) population size. All configurations are allowed an identical budget of fitness evaluations (500,000), hence ensuring that all require approximately the same computational effort. Each algorithm is run 100 times with each configuration; results are compared in terms of HV and GD by the Wilcoxon test with a confidence level of 95%, and the best configuration is selected[4]. In particular, the

[4] *p-values* are adjusted by the Bonferroni correction, in order to account for multiple comparison protection.

Table 2 Configurations of MOEA algorithms

Configuration	*Population Size*	*Generations*
Very Small (VS)	50	10,000
Small (S)	100	5,000
Medium (M)	200	2,500
Large (L)	500	1,000
Very Large (VL)	1,000	500

Table 3 Parameters setting

Scenario #	**#Obj.**	**#PIOs**	**MOEA Configuration**		
			NSGA	*SPEA2*	*IBEA*
1	2	40	S	VS	S
2	3	40	VS	VS	S
3	4	40	S	S	S
4	2	60	VS	VS	VS
5	3	60	VS	VS	VS
6	4	60	S	VS	S
7	2	80	S	VS	VS
8	3	80	VS	VS	S
9	4	80	VS	VS	VS
10	2	100	VS	VS	VS
11	3	100	VS	VS	VS
12	4	100	VS	VS	VS

configuration having the highest HV with a confidence of 95% is preferred; if there is no configuration statistically better than the others, the configuration with the lowest GD (again at 95% of confidence) is selected. If no configuration is significantly better than the others in both HV and GD measure – a case never occurred in our trials– than the *medium* configuration is selected. This evaluation step is performed for each scenario, resulting in a total number of runs of 12 scenarios * 3 algorithms * 5 configurations * 100 repetitions = 18,000. The chosen configuration for each MOEA in each scenario is in Table 3. The "very small" configuration turned out to be the best configuration in the majority of cases, for all the algorithms. The successive analyses on each scenario are conducted under such "best settings". Additionally, we adopted the single point crossover operator, with crossover probability of 0.9, a bit flip mutation operator, with probability of mutation 1 over the number of bits [28]. The selection is done via binary tournament. The archive size is 100.

5.4 Sensitivity analysis

The previous settings allow for a static analysis of algorithms with respect to several independent instances of the optimization problems. We carry out a further analysis to evaluate the trade-off between cost and solution quality under different values of the application frequency F. Given a window of length N=10, we consider different values of F. To run this analysis in a concise way, we take the best instance for each MOEA algorithm from the previous section's evaluation and run the following procedure:

1. We first consider the best situation in terms of expected solution quality, namely $F = 1$, in which the algorithm is run N times at each time unit, having always the most updated optimal solution. For this, we collect the total cost to run the algorithm N times, denote it as C_1, and take the sets of the N solutions (x_j^*, with $j = 1, \ldots, N$) as reference solutions (where x_j^* is the one with the best utility function value).
2. We consider the other extreme, $F = N$, in which the algorithm is run just once and the solution obtained is assumed to be the best one for all the time points regardless of future predictions in t_{N+k} ($k = 1, \ldots, N$). This is equivalent to say that, for each time point j, we have N identical solutions $x_j = x_{F=N}$. For each j, we evaluate the number of time unitss in $x_{F=N}$ in which the solution differs from x_j^*, representing how many times running the algorithm at time j differed from the optimal solution (i.e., the number of different chromosomes). This value is denoted as ε_j; we take the sum over j, $\epsilon = \sum_j \varepsilon_j$ and consider the percentage with respect to all the possible changes, $(N-1)+(N-2)+\cdots+1 = N(N-1)/2$, denoted with $\epsilon_{\%_N}$. We also consider the cost of running the algorithm in this case (which is run just once), $C_N < C_1$.
3. Similarly, we consider all the intermediate case, for $1 < F < N$. When F takes an intermediate value, e.g., $F = 3$, we expect that the solutions may differ from the reference one in $1 < j < F$, in $F < j < 2F$, $\ldots$, and in $kF < j < N$ (with k representing the number of times we run the algorithm before reaching t_N). However, the number of changes are expected to be lower compared to the case $F = N$, because the re-computation of the optimal solution at some intermediate time point is expected to reduce the difference with the optimal case $F = 1$. Also in this case, the $\epsilon_{\%_F}$ and the cost C_F to run the algorithm k times are collected.

Letting F varying between 1 and N, we analyze the output $\epsilon_{\%_F}$ and cost C_F values. The analysis is repeated 30 times, taking the average $\bar{\epsilon}_{\%_F}$ and $\bar{C}_F$ values.

5.5 Results

This section presents the results with respect to research questions 1-4.

Results for RQ1 - Validation: the first research question is aimed at establishing if the proposed strategy

is worth with respect to a random selection. The second is implemented in a way to satisfy the constraints on location providers availability in each zone (namely, it can select only the location providers available, at a given time point, in the zone in which the SET is moving) and of selecting only one location provider at a time. Considering the 12 scenarios and 3 MOEA algorithms (hence 36 cases), the random search has always been statistically worse than any MOEA algorithm for both quality indicators, according to the Wilcoxon test, with a confidence of 95%. Because of very low solution quality (HV is always less than 0.1), the random search is not considered a possible alternative to a MOEA, but just as a means to validate the choice of adopting a MOEA. We do no longer consider it in the next research questions.

Table 4 Number of times when algorithm x (in row) was significantly better than y (in column), with 95% confidence over the 12 scenarios. In the remaining times, x and y do not differ significantly.

(a) HV Indicator

	NSGAII	SPEA2	IBEA
NSGAII	–	3/12	1/12
SPEA2	2/12	–	2/12
IBEA	7/12	10/12	–

(b) GD Indicator

	NSGAII	SPEA2	IBEA
NSGAII	–	0/12	1/12
SPEA2	12/12	–	5/12
IBEA	10/12	4/12	–

Results for RQ2 - MOEA quality comparison: MOEA are compared with respect to the HV and GD indicators. We use the best configuration of each algorithm as in Table 3. Table 4 shows, for each scenario, the outcome of the Wilcoxon test for both HV and GD indicators. We have the IBEA algorithm performing better than NSGAII 7/12 times, and better than SPEA2 10/12 times according to the HV indicator. While NSGAII and SPEA2 are quite similar, since 3/12 times NSGAII outperforms SPEA2, 2/12 times the opposite occurs, and the remaining 7/12 times the HV is statistically the same. Results of the GD indicator tells that IBEA has still better performance than NSGAII, while it has similar performance as SPEA2 (4/12 against 5/12); SPEA2 always outperforms NSGAII.

A better comprehension of results is given by Tables 5-6, showing in detail the outcomes of the Wilcoxon tests, and then by the box plots in Figure 10-11, where the extent of the difference among algorithms is also visible. From box plots, IBEA turns out to provide less variable results across scenarios for the HV indicator; the GD indicator plots highlight the good performance of SPEA2, which is comparable with IBEA. NSGAII overcomes IBEA only in one scenario, the number 4, while it always performs worse than the others. Taking the mean among the median value of each scenario, we notice that NSGAII is slightly better than SPEA2 for the HV indicator (they are: 0.448, 0.446, and 0.477 for NSGAII, SPEA2, and IBEA, respectively), while for the GD indicator they are: 0.0192, 0.0134, and 0.0146 for NSGAII, SPEA2 and IBEA. In other words, regarding HV, IBEA confirms to exhibit the best behaviour, NSGAII is slightly better than SPEA2, as the former outperforms the latter 3/12 against 2/12 and the mean of medians is slightly better (even when they are statistically the same, NSGAII has slightly higher values in the average). As for GD, SPEA2 and IBEA are comparable to each other, with SPEA2 having a lower average value. Considering the standard deviations of medians across scenarios, the variability of results of the three algorithms is approximately the same (0.150, 0.140, and 0.132 for HV; 0.008, 0.006, and 0.007 for GD, considering, in the order, NSGAII, SPEA2, and IBEA).

Results for RQ3 - Cost comparison: we took the average execution time for computing a solution for each algorithm. Experiments are run on a machine equipped with 3 GHz Intel Core i7, dual core, 8GB of RAM, and a 256 MB SSD HD, running Mac OS X 10.8.5.

Table 7 Number of times when algorithm x (in row) was significantly better than y (in column), with 95% confidence over the 12 scenarios for the TIME indicator. In the remaining times, x and y do not differ significantly.

	NSGAII	SPEA2	IBEA
NSGAII	–	12/12	12/12
SPEA2	0/12	–	0/12
IBEA	0/12	12/12	–

Table 7 reports a concise snapshot of the results, which are extensively reported for each scenario in Table 8. Compared to quality indicators, results on computation time are much clearer. There is a net outcome: NSGAII is the best one in every scenario, followed by IBEA, then by SPEA2. Looking at the ex-

Table 5 Wilcoxon tests w.r.t. HV indicator

(a) Scenario 1

	SPEA2	IBEA
NSGAII	–	▲
SPEA2		▲

(b) Scenario 2

	SPEA2	IBEA
NSGAII	–	▽
SPEA2		▽

(c) Scenario 3

	SPEA2	IBEA
NSGAII	–	–
SPEA2		▽

(d) Scenario 4

	SPEA2	IBEA
NSGAII	▲	–
SPEA2		▽

(e) Scenario 5

	SPEA2	IBEA
NSGAII	–	▽
SPEA2		▽

(f) Scenario 6

	SPEA2	IBEA
NSGAII	▽	–
SPEA2		▲

(g) Scenario 7

	SPEA2	IBEA
NSGAII	▲	–
SPEA2		▽

(h) Scenario 8

	SPEA2	IBEA
NSGAII	–	▽
SPEA2		▽

(i) Scenario 9

	SPEA2	IBEA
NSGAII	–	▽
SPEA2		▽

(j) Scenario 10

	SPEA2	IBEA
NSGAII	▲	▽
SPEA2		▽

(k) Scenario 11

	SPEA2	IBEA
NSGAII	–	▽
SPEA2		▽

(l) Scenario 12

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▽

Table 6 Wilcoxon tests w.r.t. GD indicator

(a) Scenario 1

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▽

(b) Scenario 2

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▲

(c) Scenario 3

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▲

(d) Scenario 4

	SPEA2	IBEA
NSGAII	▽	▲
SPEA2		▲

(e) Scenario 5

	SPEA2	IBEA
NSGAII	▽	–
SPEA2		▲

(f) Scenario 6

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▽

(g) Scenario 7

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▽

(h) Scenario 8

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▽

(i) Scenario 9

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		–

(j) Scenario 10

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		–

(k) Scenario 11

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		▲

(l) Scenario 12

	SPEA2	IBEA
NSGAII	▽	▽
SPEA2		–

Table 8 Wilcoxon tests w.r.t. TIME indicator

(a) Scenario 1

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(b) Scenario 2

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(c) Scenario 3

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(d) Scenario 4

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(e) Scenario 5

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(f) Scenario 6

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(g) Scenario 7

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(h) Scenario 8

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(i) Scenario 9

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(j) Scenario 10

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

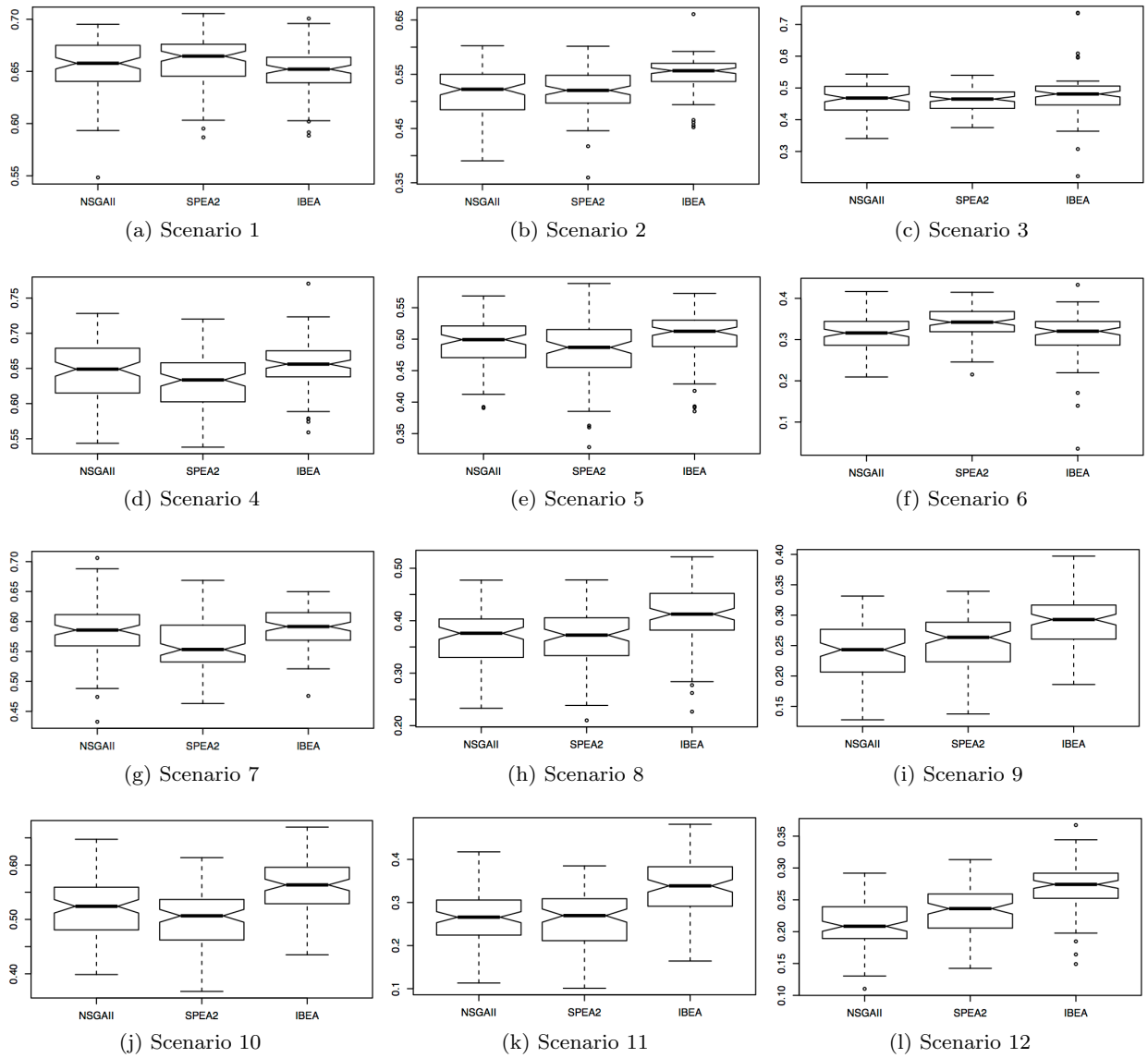
(k) Scenario 11

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

(l) Scenario 12

	SPEA2	IBEA
NSGAII	▲	▲
SPEA2		▽

tent of the differences, Figure 12 highlights, through box plots, that there is very high difference between NSGAII and SPEA2. NSGAII takes, in the average, 111ms, whereas SPEA2 takes 829 ms; IBEA takes an intermediate value, 439ms. Box plots highlight very tight confidence intervals for all the algorithms, especially for NSGAII, i.e., very low variability in the single scenario. Standard deviation of median values across sce-



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

(d) Scenario 4 (e) Scenario 5 (f) Scenario 6

(g) Scenario 7 (h) Scenario 8 (i) Scenario 9

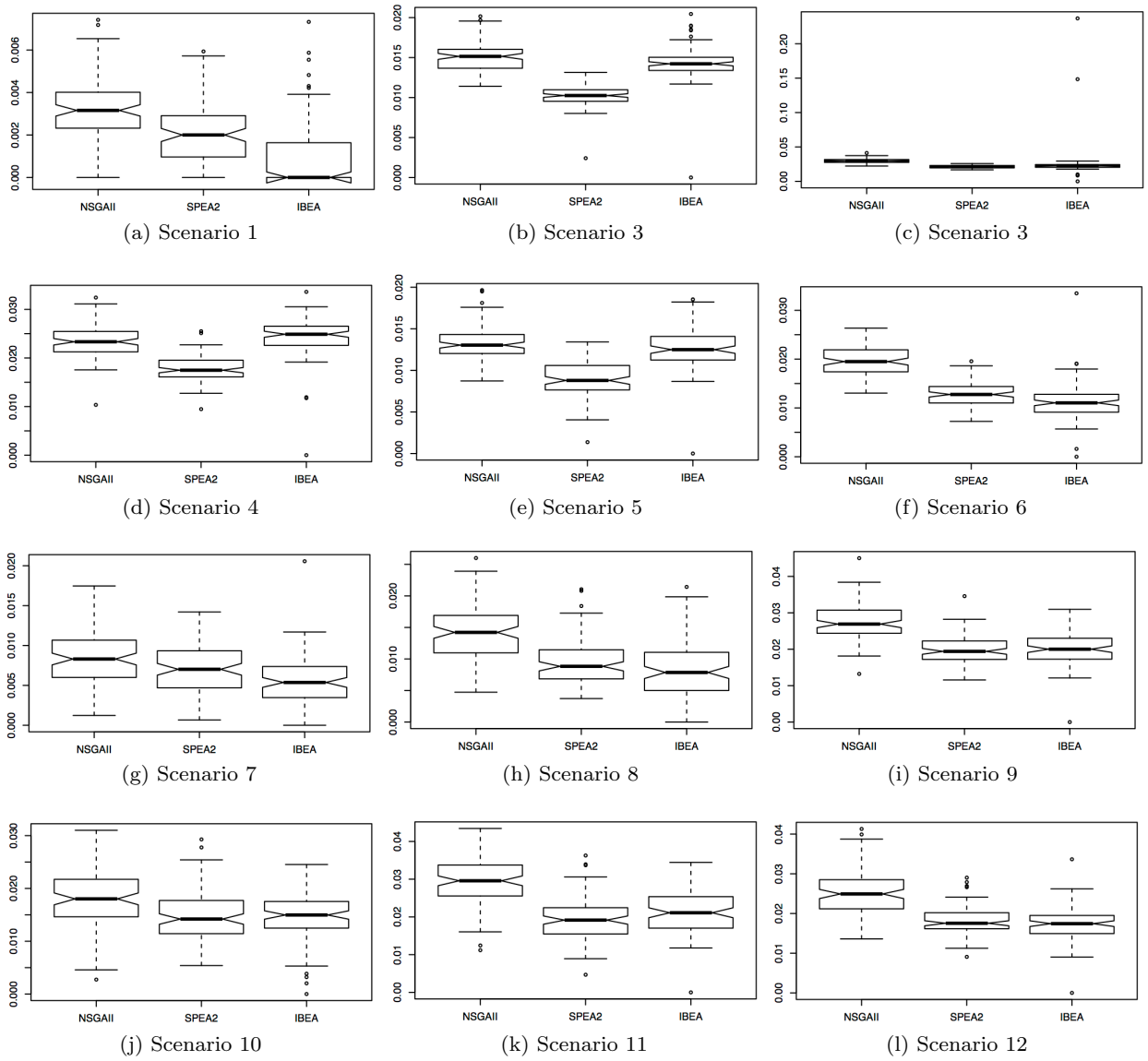
(j) Scenario 10 (k) Scenario 11 (l) Scenario 12

Fig. 10: Boxplots for the HV quality indicator the 12 scenarios

narios also confirm the superiority of NSGAII: 27.03ms, 68.40ms, 63.66ms for, respectively, NSGAII, SPEA2, IBEA.

Results of RQ4 - Sensitivity: the last research question is about sensitivity of results against the frequency of application, F. Taking the best instances, Figure 13 shows the trade-off between computational time and the percentage of error committed in not running the algorithm at every time unit. The latter percentage is related to the goodness of the solution with respect to the variability of the problem (e.g., how much robust are the solutions to variations in terms of SET movements, where higher variation would require higher application frequency F); the indicator is not related to the quality of the provided solutions with respect to the reference front (i.e., the extent to which they are optimal), which is already captured by previous indicators.

Results show that, for NSGAII, applying the algorithm each time unit for 10 time units (we considered a window of width $N = 10$) costs about 600 ms; up to $F = 3$ the error is still around 10%, for a total time of 200ms; then the error increases considerably. A similar sensitivity is observed by the other two algorithms, with execution times much greater (from about 8000 to 1000 ms for SPEA2, and from 3500 to 500 ms for IBEA). A balance point for the three algorithms seems to be $F = 3$. However, the choice depends mainly on



(a) Scenario 1 (b) Scenario 3 (c) Scenario 3

(d) Scenario 4 (e) Scenario 5 (f) Scenario 6

(g) Scenario 7 (h) Scenario 8 (i) Scenario 9

(j) Scenario 10 (k) Scenario 11 (l) Scenario 12

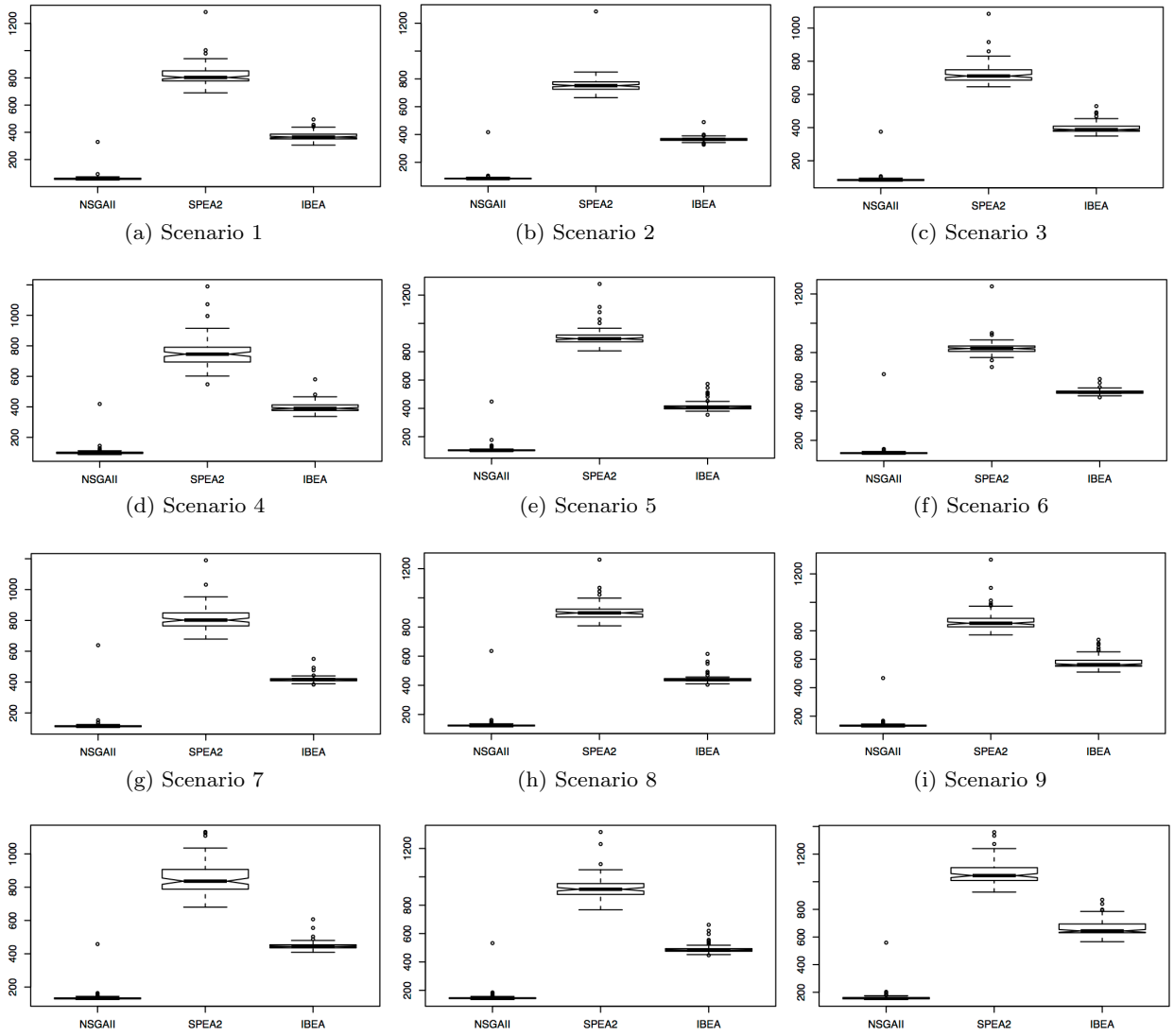
Fig. 11: Boxplots for the GD quality indicator the 12 scenarios

the user requirements (e.g., it has a computational capability of 1000ms, then he can choose $F = 1$ for the best results).

In summary, in our problem instance we observed that, regarding quality indicators, IBEA has the best balance between HV and GD (it has the best HV values, and GD values comparable to SPEA2 ones); regarding time, NSGAII is by the best one, SPEA2 the worst one, and IBEA in the middle. Sensitivity to F is roughly the same for the three algorithms. Therefore, if the user has strict requirements on the computational capability, then NSGAII with $F = 3$ is the best configuration; if a higher computational time can be afforded, than it can turn to IBEA and choose F based on available computational capability; if computational time is not a problem at all, than it can opt between IBEA and SPEA2 (depending on the importance he gives to HV and to GD), and of course $F = 1$.

5.6 Remarks on tuning of further parameters

In the previous sections, we evaluated *GlobalPreLoc* with respect to several parameters. Besides those ones, there are other parameters that need to be tuned by observing field data, or by a preliminary experimentation before deploying the system. For instance, the frequency of application of the algorithm may depend also on the chosen width of the time interval $Width = t_k - t_{k-1}$. Its



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

(d) Scenario 4 (e) Scenario 5 (f) Scenario 6

(g) Scenario 7 (h) Scenario 8 (i) Scenario 9

(j) Scenario 10 (k) Scenario 11 (l) Scenario 12

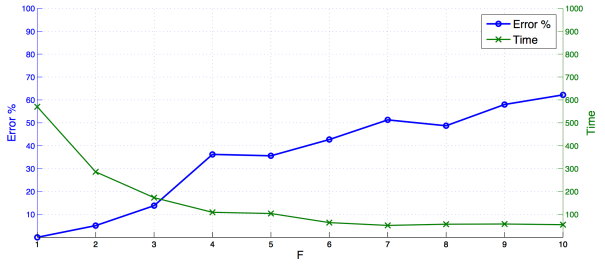
Fig. 12: Boxplots for the TIME quality indicator the 12 scenarios

best tuning needs to be done by collecting preliminary field data, where the velocity of the SETs' movements and regularity of movement patterns play an important role. In fact, the frequency of application of the algorithm and the correctness of proposed solutions depend mainly on how fast the SET moves and on how quickly and unpredictably it changes direction. A high speed and irregularity requires a tight $Width$ and small F values, incurring higher computation cost to get the same optimal solutions. Further analysis will be devoted to this in our future work. Other choices regard the parameters of *WhereNext*; our experiments are not intended to validate *WhereNext*, whose performance are already assessed in [17], but we were interested in *WhereNext* as end users and its experimentation is out of the scope of this work.

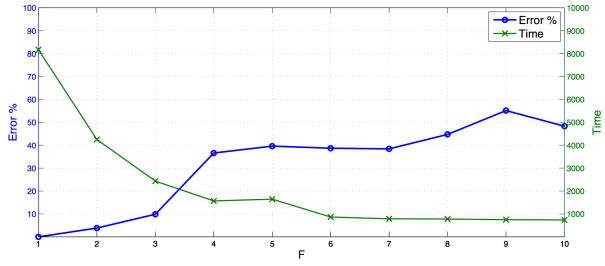
6 Related work

This section reviews some relevant works that propose software architectural solutions for positioning.

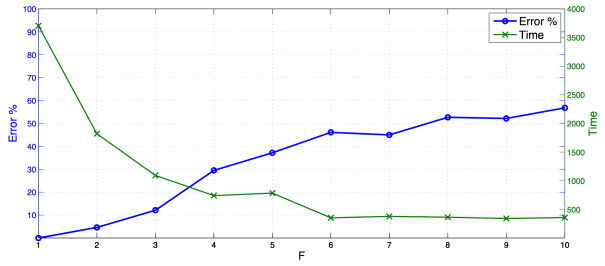
La Marca et *al.* [31] propose a software architecture for indoor and outdoor positioning, which provides a Java package to develop pull LBSs exploiting handset-based positioning. Mobile devices can infer their own position by looking for Cell-Ids of radio beacons of the surrounding wireless equipments, such as IEEE 802.11 access points, Bluetooth devices, and GSM wireless cells.



(a) NSGA



(b) SPEA2



(c) IBEA

Fig. 13: Sensitivity of solutions to the frequency of application of the MOEA algorithms

It only supports the proximity positioning technique (the position accuracy is related to the coverage area of the wireless networking beacons). It does not support policies for automatic switching among providers.

Several works propose suitable abstractions to structure the components of a generic positioning architecture. Nord and Synnes [32] describe how different location providers can be integrated through appropriate wrappers to provide LBSs with a uniform API. They use the GPS for outdoor scenarios, and the WaveLan positioning servers for indoor. Hosokawa et *al.* in [33] propose an architecture for seamless navigation, whose goal is to reduce the need of knowing details about the positioning systems and maps, as well as to support the dynamic switching among the positioning systems. Other middleware solutions include: *i)* the Platform for LBSs (PoLoS) [34], whose main feature is the "extensibility", via a modular architecture that can be easily extended with additional functionalities; *ii)* the Framework for Location Aware ModElling (FLAME), an architecture that aims at abstracting from the adoption of different positioning technologies [35]; *iii)* the Location Operating REference model (LORE), proposed by IBM, for location-aware services development, supporting heterogeneous positioning techniques and industry-standard location APIs, location modeling, location-dependent query processing, and intelligent location-aware message notification [36].

More advanced solutions include the possibility of accessing low-level details for improving the context-awareness features of LBSs and, more in general, of positioning management operations and management decisions. An example is the *Middelwhere* solution, proposed by Ranganathan et *al.* [37], which supports infrastructure-based approaches and push LBSs. It introduces a distributed middleware infrastructure for positioning, which separates applications from location sensing technologies. Like our solution, it enables the integration of different location technologies, but it does not support automatic switching and QoP policies for the optimal selection of providers. Moreover, it uses a CORBA-based technology to enable distributed communication among components, the LBSs, and the wrapper of the location technologies. Hightower et *al.* propose a layered location stack [38]. They describe how to partition the layers of positioning architectures and how to pass parameters between them. An API is designed to integrate indoor and outdoor technologies. Hohl et *al.* [39] propose a software platform for development of a global infrastructure based on computer representations of regions of the physical world, augmented by virtual objects. Recognizing that basic services are usually re-implemented by spatial-aware application developers, the main purpose of that work is to propose a uniform access to the platform for different spatial-aware applications. Pfeifer [40] combines heterogeneous positioning data from different sources, by exploiting principles of data fusion. The goal is to obtain more precise and more reliable results according to the various needs of LBSs. The authors in [29] present POSIM (Positioning System Integration and Management), a middleware that enables integration and management of multi-positioning systems on the same device. POSIM enables to access and control positioning information at a high level of abstraction via context-awareness management decisions, as well as to access to low-level positioning details when needed. Like our solution, POSIM allows the access to heterogeneous positioning systems in a uniform way, and chooses and merges location information on the base of the characteristics depending on the applicable context. Specifically, POSIM allows implementing policies to guide the positioning procedure based on LBS requirements (*e.g.*, minimize power

consumption), user preferences, device and infrastructure characteristics in a dynamic way, through its Policy Manager.

Proprietary solutions have also been proposed. Alcatel [3] proposes a software architecture to coordinate positioning requests from a common location server across public and private access environments. On the other hand, it does take into account neither the interoperability among different WISPs, nor standard issues related to application scenarios, protocols, and architectures. IBM [41] proposes a middleware that adopts Open Location Service standard [42] to specify the interfaces to access to positioning services of an infrastructure-based server. Other examples of commercial solutions are provided by TomTom [43], Ekahau [44], and Cisco Systems [45]. They offer Software Development Kit (SDK) for the development of personal applications. TomTom is a GPS-based automotive navigation system used to provide road assistance to mobile users. Ekahau and Cisco Systems adopt the infrastructure-based approach, based on Wi-Fi network.

The architecture proposed in this paper also supports, like some of the more recent solutions presented above, the integration in a uniform way of both indoor and outdoor location providers and exploits both handset- and infrastructure-based positioning approaches. The main distinguishing point, that we presented in this paper, is the strong focus on optimality of policies for location provider selection. Context-awareness is significantly enforced in our solution, by a mechanism that supports the dynamic and transparent switch among heterogeneous indoor/outdoor location providers during the user movement, accounting for the application requirements, the device capabilities, and the surrounding positioning infrastructure. It is the first solution exploring position prediction algorithms for implementing optimal policies that satisfy multiple application requirements and constraints at the same time. Moreover, from the architectural point of view, our solution presents further advantages. By implementing the architecture and protocols on top of open standard solutions, we provide several further benefits. First, the inclusion of new positioning methods is facilitated by the SUPL paradigm, in a transparent and standard way for WISPs and location provider developers. Second, a standard-based solution fosters the WIPSs interoperability, favoring the LBS multi-vendor integration, with benefits for both WISPs, in terms of positioning solutions availability and infrastructure cost, and for users as well, in terms of wider and better availability of LBSs. Additionally, the solution, as it is conceived, enables WISPs to locate their end-users in public and private environments.

7 Conclusion

This paper presented a solution to optimally select location providers in a newly defined positioning architecture. The solution exploits a pattern-mining position prediction algorithm and MOEAs to solve the selection problem under multiple QoP objectives and constraints, representing the application requirements.

The SUPL-based positioning architecture has been first discussed; the main benefits with respect to existing solutions have been highlighted, here summarized:

- *Openness*: it is based on open standards, in order to encourage WISPs interoperability, and hence multi-vendors LBSs integration in the new generation of wireless networks;
- *Interoperability*: software abstractions have been specified in order to support the integration of location providers implemented by the most widespread location APIs;
- *Extensibility and evolution*: due to the independence between the communication and the positioning infrastructure, the positioning methods and the communication technology can evolve independently, keeping the integration between them simple in this architecture;
- *Standard compliance*: the architecture and protocol are kept compliant with SUPL specification, so as to favor an easy integration with existing SUPL-based solution and an incremental implementation of a solution based on our architecture (L-SLP can be incrementally added to the infrastructure).

Then, the focus is put on the dynamic *location provider selection* and switching. In fact, in order to improve the availability and accuracy of positioning, the architecture allows using measurements from the handset and the infrastructure, and dynamically choosing the most appropriate positioning technologies, according to the required QoP, the terminal capabilities, and the positioning infrastructures that are available at a given time and place. The usage of a position prediction algorithm allows basing this choice on current and future expected user movements. The solution as designed offers the possibility to instantiate the concrete approach by using other prediction algorithms and/or MOEAs besides the one adopted in this paper.

Currently, we are working to add further functionalities. Mainly, we are focusing *i)* on implementing new prediction algorithms in order to infer most likely user movements from tracked data (e.g., RSSIs) and enable further optimization policies, as well as on *ii)* extending the support to new location providers and additional optimization criteria. At higher level, we intend

to conceive solutions to further promote interoperability among WISPs and ease the formulation of agreements about the mentioned business privacy issues.

References

1. Ilarri S., Illarramendi A., Mena E., and Sheth AP. Semantics in Location-Based Services, *IEEE Internet Computing*, vol. 15, no. 6, 2011, pp.10-14.
2. Liu H., Darabi H., Banerjee P., and Liu J. Survey of Wireless Indoor Positioning Techniques and Systems, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 37, no. 6, Nov. 2007, pp. 1067-1080.
3. Faggion N., Leroy S. Alcatel Location-based Services Solution, Sep. 2005, Alcatel Telecommunication.
4. Google Gears, the Google Gears Geolocation API, available at http://code.google.com/intl/it-IT/apis/gears/api_geolocation.html. Last update Feb. 2011.
5. Ficco M., Pietrantuono R., Russo S. Supporting ubiquitous location information in interworking 3G and wireless networks, *Communications of the ACM*, vol. 53, no. 11, 2010, pp. 116-123.
6. The OMA Secure User Plane Location (SUPL) - v.3, available at: www.openmobilealliance.org/Technical/ release_program/supl_v3_0.aspx. Last Release 2011.
7. Ficco M., Esposito C., and Napolitano A. Calibrating Indoor Positioning Systems With Low Efforts, IEEE Transactions on Mobile Computing, vol. 13, no. 4, April 2014, pp. 737 - 751.
8. Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12). ACM, New York, NY, USA, 911–918.
9. D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, pages 1–8. ACM, 2007.
10. T. G. Dietterich. Machine Learning for Sequential Data: A Review. In Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, pages 15–30. Springer-Verlag, 2002.
11. N. Nguyen and Y. Guo. Comparisons of sequence labeling algorithms and extensions. In Proceedings of the 24th International Conference on Machine Learning, pages 681–688. ACM, 2007.
12. O. Ossama and H. M. O. Mokhtar. Similarity Search in Moving Object Trajectories. In Proceedings of the 15th International Conference on Management of Data, pages 1–6. Computer Society of India, 2009.
13. M. Morzy. Prediction of moving object location based on frequent trajectories. ISCIS, volume 4263 of LNCS, pages 583–592. Springer, 2006.
14. M. Morzy. Mining Frequent Trajectories of Moving Objects for Location Prediction. In Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition, pages 667–680. Springer-Verlag, 2007.
15. G. Yavas, D. Katsaros, O. Ulusoy, and Y. Manolopoulos. A data mining approach for location prediction in mobile environments. D.K.E., 54(2) pages 121–146, 2005.
16. H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. ICDE 2008 pages 70–79.
17. A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. WhereNext: a location predictor on trajectory pattern mining. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 637646. ACM, 2009.
18. F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. KDD 2007: 330–339.
19. Deb Kalyanmoy and Gupta Shivam. Understanding knee points in bicriteria problems and their implications as preferred solution principles, Engineering Optimization, 43(11), pages 1175-1204, 2011.
20. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.
21. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithms. In EUROGEN 2001, pages 95–100, 2002.
22. Zitzler E, Knzli S. Indicator-based selection in multiobjective search. In: Yao X et al., editors. Parallel problem solving from nature (PPSN VIII). Berlin, Germany: Springer Verlag; 2004. p. 832–42.
23. J.J. Durillo, A.J. Nebro jMetal: a Java Framework for Multi-Objective Optimization. Advances in Engineering Software 42 (2011) 760-771.
24. Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 1999;3(4): 257–71.
25. Van Veldhuizen DA, Lamont GB. Multiobjective evolutionary algorithm research: A history and analysis, Tech. Rep. TR-98-03, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst.Technol., Wright-Patterson, AFB, OH; 1998.
26. J. Cohen, Statistical power analysis for the behavioral sciences, 2nd ed. Lawrence Earlbaum Associates, 1988.
27. Filomena Ferrucci, Mark Harman, Jian Ren, and Federica Sarro. 2013. Not going to take this anymore: multi-objective overtime planning for software engineering projects. In Proceedings of the 2013 International Conference on Software Engineering (ICSE '13), 462-471.
28. Francisco Chicano, Francisco Luna, Antonio J. Nebro, and Enrique Alba. 2011. Using multi-objective metaheuristics to solve the software project scheduling problem. In Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11), Natalio Krasnogor (Ed.). ACM, 1915-1922.
29. Bellavista P., Corradi A., Giannelli C. The PoSIM middleware for translucent and context-aware integrated management of heterogeneous positioning systems, *Computer Communications*, vol. 31, 2008, pp. 1078-1090.
30. Chen JC, Maa CS, Wang YC, and Chen JT. Network-Side Mobile Position Location Using Factor Graphs, *IEEE Trans. on Wireless Communications*, vol. 5, no. 10, Oct. 2006, pp. 2696-2704.
31. La Marca A., et *al*. Place Lab: Device Positioning Using Radio Beacons in the Wild, in Proc. of the 3^{rd} *Int. Conf. on Pervasive Computing*, LNCS, vol. 3468, 2005, pp. 116-133, Springer-Verlag.
32. Nord J., Synnes K., and Parne P. An Architecture for Location Aware Applications, in Proc. of the 35^{nd} *Int. Conf. on System Sciences*, IEEE CS Press (2002).
33. Hosokawa Y., Takahashi N., Taga H. A system architecture for seamless navigation, in Proc. of the *Int. Conf. on Distributed Computing Systems*, Mar. 2004.

34. Spanoudakis M., Batistakis A., Priggouris I., Ioannidis A., Hadjiefthymiades S., and Merakos L. Extensible platform for location based services provisioning, in Proc. of the *Int. Conf. on Web Information Systems Engineering*, Dec. 2003.
35. Coulouris G., Naguib H., and Samugalingam K. FLAME: an open framework for location-aware systems, *Ubiquitous Computing*, Oct. 2002.
36. Chen Y., Chen XY., Rao FY., Yu XL., Li Y., and Liu D. LORE: an infrastructure to support location-aware services, in *IBM Journal of Research and Development*, vol. 48, no. 5, 2004, pp. 601-615.
37. Ranganathan A., Al-Muhtadi J., Chetan S., Campbell R., and Mickunas D. MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications, in Proc. of the 5^{th} *Int. Conf. on Middleware*, LNCS, vol. 3231, 2004, pp. 397-416, Springer-Verlag.
38. Hightower J., Brumitt B., and Borriello G. The location Stack: Layered Model For Location in Ubiquitous Computing, in Proc. of the 4^{th} *IEEE Int. Workshop on Mobile Computing System and Applications*, IEEE CS Press (2002).
39. Hohl F., Kubach U., Leonhardi A., Rothermel K., and Schwehm M. Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications, in Proc. of the *ACM International Mobicom Conference*, 1999, pp. 249-255, ACM Press.
40. Pfeifer T. Redundant positioning architecture, *Computer Communications*, vol. 28, no. 13, Aug. 2005, pp. 1575-1585. Elsevier Press.
41. Lee S., Cheng S., Hsu J.Y., Huang P., and C. You. Emergency care management with location-aware services, in Proc. of the *Pervasive Health Conference and Workshops*, 2006, pp. 1-6. IEEE CS Press.
42. Hansen S., Richter K., and Klippel A. Landmarks in OpenLS: A Data Structure for Cognitive Ergonomic Route Directions, LNCS, vol. 4197, 2006, pp. 383-393. Springer-Verlag.
43. TomTom International BV: Software development, available at: www.tomtom.com/pro/page.php?ID=2
44. Ekahau, Inc.: Ekahau Positioning Engine 2.0, available at: www.ekahau.com.
45. Appear Network, Inc.: Appear Context Engine, available at: www.appearnetworks.com.
46. Skyhook Wireless, Skyhook CEO undaunted by mobile giants, available at www.crunchbase.com/company/skyhook-wireless. Last update June 2011.
47. Ficco M., and Russo S. A hybrid positioning system for technology-independent location-aware computing, in *Software: Practice and Experience*, vol. 39, Feb. 2009, pp. 1095-1125.
48. Di Flora C., Ficco M., Russo S., and Vecchio V. Indoor and outdoor location based services for portable wireless devices, in Proc. of the *IEEE Int. Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'05)*, Jun. 2005, pp. 244-250, IEEE CS Press.
49. Geomena, an open geo database of Wi-Fi access points, available at http://geomena.org/. Last update Sept. 2011.
50. Google Latitude enables users to update and read their current location, and their location history, available at www.google.it/mobile/latitude/. Last update Oct. 2010.
51. Karam R., Favetta F., Kilany R., and Laurini R. Location and Cartographic Integration for Multiproviders Location-Based Services, in *Advances in Cartography and GIScience*, LNCS, vol. 1, Springer, 2011, pp. 365-383.