Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 936125, 19 pages
http://dx.doi.org/10.1155/2015/936125

Research Article

An AmI-Based Software Architecture Enabling Evolutionary Computation in Blended Commerce: The Shopping Plan Application

Giuseppe D'Aniello,[1,2] Matteo Gaeta,[1] Vincenzo Loia,[2,3] and Francesco Orciuoli[3]

[1] *Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica Applicata, Università degli Studi di Salerno, 84084 Fisciano, Italy*
[2] *Consorzio di Ricerca Sistemi ad Agenti, Università degli Studi di Salerno, 84084 Fisciano, Italy*
[3] *Dipartimento di Informatica, Università degli Studi di Salerno, 84084 Fisciano, Italy*

Correspondence should be addressed to Francesco Orciuoli; forciuoli@unisa.it

Received 1 December 2013; Accepted 25 February 2014

Academic Editor: David Taniar

This work describes an approach to synergistically exploit ambient intelligence technologies, mobile devices, and evolutionary computation in order to support blended commerce or ubiquitous commerce scenarios. The work proposes a software architecture consisting of three main components: linked data for e-commerce, cloud-based services, and mobile apps. The three components implement a scenario where a shopping mall is presented as an intelligent environment in which customers use NFC capabilities of their smartphones in order to handle e-coupons produced, suggested, and consumed by the abovesaid environment. The main function of the intelligent environment is to help customers define shopping plans, which minimize the overall shopping cost by looking for best prices, discounts, and coupons. The paper proposes a genetic algorithm to find suboptimal solutions for the shopping plan problem in a highly dynamic context, where the final cost of a product for an individual customer is dependent on his previous purchases. In particular, the work provides details on the Shopping Plan software prototype and some experimentation results showing the overall performance of the genetic algorithm.

1. Introduction

The blended commerce can be described as a commerce ecosystem in which customers do not distinguish among online, offline, and mobile interactions and use all them in the same process. In a typical example, a customer looks for a specific product by using web channels and buys this product in a physical store by using a credit card and exploiting a coupon received via SMS. U-commerce is similar to blended commerce. In [1], the author asserts that *u* stands for ubiquitous, universal, unique, and unison. In the aforementioned ecosystem, one of the main phenomena of the last years is represented by loyalty programs: points systems, coupons, and so on. The most common points system methodology foresees that frequent customers earn points that can be converted in some kind of reward. In this way, those customers work toward a certain amount of points to redeem their reward. A coupon is also considered a loyalty mechanism and is usually a ticket or a document that can be exchanged for a financial discount or rebate when purchasing a product. One of the most recent types of loyalty program foresees the establishment of a partnership among several companies to provide composite (often all-inclusive) offers. Strategic partnerships, based on coalition loyalty programs, can be extremely effective for customer retention and generate networks of business partners that take advantages each other and also offer added value to customers. In the last year, the number of coupons and points systems proposed is continuously growing. The aforementioned increase is due to specialized web sites which we subscribe to or, for instance, to the e-coupons distributed via e-mail. Definitely, a careful use of coupons and points systems and their digital forms can actually reduce the cost of a product (or service) or an entire shopping list.

On the other hand, ambient intelligence (AmI) is the next wave in computing and communications technology and can be defined as a digital environment that supports people in their daily lives by assisting them in an "intelligent way" [2]. An ambient intelligence system has a real environment and real occupants that interact with it and must be "intelligent" (i.e., it only intervenes when needed and must adapt its behaviour to current overall situations, user's preferences and needs) [3]. In this context, we claim that AmI is the right opportunity to construct a blended commerce ecosystem where the customer experience is significantly improved in terms of interactions with services (payment, product search and browsing, couponing) and cost saving. On the other hand, in this scenario, the merchants can build partnerships in order to activate coalition loyalty programs.

The paper has the following structures. Section 2 proposes an overall approach for blended commerce based on ambient intelligence and its instantiation for the shopping mall scenario where customers interact with smart environments in order to define a plan enabling them to decrease the total cost of their shopping list. Section 3 describes the proposed architecture based on the overall approach of Section 2, and a genetic algorithm to find suboptimal solutions to the minimization problem for the shopping plan. Section 4 describes an instantiation of the architecture in which the shopping plan algorithm is implemented and evaluated. Lastly, final remarks are provided in Section 5.

2. An AmI-Based Blended Commerce Framework

The objective of this section is to describe an overall approach for ambient intelligence supporting blended commerce scenarios. An ambient intelligence environment must be sensitive and responsive to the presence of people. This is achieved through the use of three enabling technologies: ubiquitous computing, ubiquitous communication, and intelligent user interfaces. Ubiquitous computing (UC), also indicated as pervasive computing, is defined as the "method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user" by Weiser [4], who invented this term in 1988. In other words, the UC vision is about embedding computing facilities in everyday objects, in order to allow them to share and to communicate information, by making them fully interconnected and constantly available to users. It is clear that in such a vision, mobile phones represent a key technology due to their communication and computing capabilities and their pervasive presence in everyday life. UC requires suitable communication infrastructures in order to permit to different devices to communicate with each other and with the users. This kind of communication is defined as ubiquitous communication (UbiCom), to indicate the ability of devices to communicate with each other and to provide services to the users in a transparent way, even without direct human interaction [5]. Devices and/or the humans can communicate everywhere and anytime, even autonomously; the communication should also be context aware, for example, it should be aware of the location, time, and so forth. The ubiquitous devices generally need to access to verified [6] and secure communication infrastructures [7] (e.g., a backbone network) to deliver the required service. In such a scenario, wireless technologies, advanced electronics, and Internet represent the main key factors to realize the *ubiquitous network*. In particular, sensors and sensor networks are fundamental to realize context-aware application in AmI scenarios. Lastly, intelligent user interfaces (IUI) are needed to enable the interaction among users and the environment. Intelligent and user-friendly interfaces are fundamental to allow a pervasive presence of computers and devices in every day life [8] in order to to improve user's quality of life. Thanks to the proliferation of inexpensive sensors, such as cameras or accelerometers embedded in smartphones, it is possible to realize novel user-experiences, such as immersive human-computer interfaces for augmented reality, gaming, assistive technologies, and other human-centric applications.

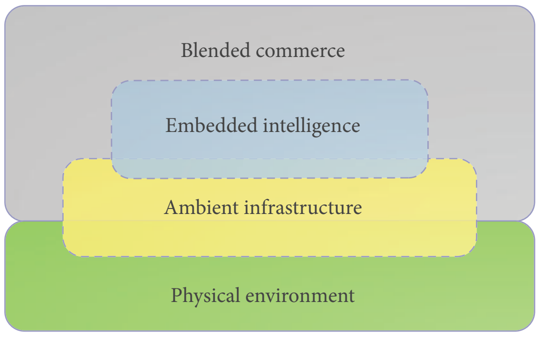


FIGURE 1: AmI-based blended commerce overall view.

The proposed approach provides the methodologies able to instantiate the aforementioned three pillars (UC, UbiCom, and IUI) in the blended commerce scenarios. This approach represents the foundations of the proposed *Mobile Software Architecture* for AmI, a software architecture for the development of mobile applications (namely, apps) that are able to exploit AmI capabilities to provide enhanced user experience in blended commerce scenarios.

2.1. Overall Approach. The aim of the proposed approach is to define the main capabilities, methodologies and technologies needed to enable blended commerce scenarios in AmI environment. Furthermore, a methodological/technological framework for the development of applications able to exploit the capabilities of devices and sensors embedded in the physical environment is provided. These applications should exhibit intelligent behaviour in order to improve user experience. The proposed overall approach is inspired to the AmI framework for u-commerce described in [9].

With respect to the framework depicted in Figure 1, *blended commerce* scenarios can be executed by different actors (customers, merchants, business partners, etc.) in *physical environment* (shopping malls, airports, high streets, etc.) that are empowered by an *ambient infrastructure* (i.e., a network of sensors and actuators, service infrastructures, communication facilities, etc.) supporting the users experience that is augmented by a set of *intelligent* capabilities known as *embedded intelligence*. The elements of the *physical environment* depend on the characteristics of the concrete scenario that is considered. The environment is enhanced with several kind of sensors and actuators through which it is possible to acquire contextual information. To achieve the goal of satisfying users with intelligent capabilities, it is necessary to interconnect all the functions of the available devices and sensors. So, protocols and technologies to enable this kind of communication are needed. Moreover, these devices have not to be connected only on physical level, but even on a logical one: this means that different devices have to exchange information and have to provide and use services offered by other devices. The *ambient infrastructure* layer provides these capabilities. Then, interconnected devices and sensors have to be properly used in order to exploit intelligent behaviour. AmI, as its name suggests, in fact, implies the presence of an intelligent component. How this component is actually implemented depends basically on the overall goal of the architecture and on the design choices of the software designer. Traditionally, AI approaches adopt machine learning algorithms, case-based reasoning techniques and so on. The counterpart of these approaches is that they are computationally expensive, thus requiring expensive and sophisticated hardware. For this reason, this paper proposes to distribute the *intelligence* in several services, algorithms and software components that take advantages of the underlying *ambient infrastructure* network for cooperating and for interacting with users. Furthermore, it is possible to implement such distributed *intelligence* as a set of simple communicating finite state machines (i.e., cellular automata) in order to be able to formally verify [10] the ambient behaviour. This intelligence is distributed in the *embedded intelligence* layer of Figure 1.

With respect to the approach depicted in Figure 1, for a concrete realization of blended commerce scenarios, it is needed that each layer provides some specific capabilities. In particular, it is needed at least in the following cases.

(i) In the *ambient infrastructure* layer, information about merchants and their offering and about users and their profile/contest. This information are mainly represented by

- (a) linked data for e-commerce based on W3C Semantic Web (http://www.w3.org/2001/sw/) technologies to model the commercial product offering of merchants;
- (b) user profiler/profiles, to capture and store information about interests, traits, habits, and preferences (and, in particular, situated preferences [11]) of the customers;
- (c) contextual information derived by sensors, like user's location and time.

(ii) In the *embedded intelligence* layer:

- (a) a set of *cooperation services*, to manage interactions across the merchants' network;
- (b) a set of *context management services* to perceiving world and users' data, represent data in a coherent context model (context perception) situated in the underlying layer, understand and evaluate the situation (context awareness) and select/execute a suitable behaviour in synergy with one or more algorithms described in the next item (context sensitivity) [12];
- (c) a set of algorithms, to provide capabilities of reasoning, recommendation, planning, and so forth, taking care of context and users information and other data provided by the environment.

(iii) In the *blended commerce* layer.

- (a) a set of mobile apps, to enable and drive interactions among customers and the environment, exploiting the underlying intelligent behaviour through intelligent user interface.

2.2. The Shopping Mall Scenario. In order to explain more clearly the several elements that are needed to instantiate the proposed approach and in order to introduce the proposed architecture, let us consider a specific blended commerce scenario set in a shopping mall. The aforementioned elements that are needed to instantiate the approach can be furthermore detailed as follows:

(i) Linked data for e-commerce: it is possible to use the GoodRelations ontology (http://www.heppnetz.de/projects/goodrelations/), to model the commercial product offering of a merchant and by populating it in order to provide the offering (products/services, etc.) of all merchants in the mall network. This linked data enables the discovery for products starting from users wish lists.

(ii) Cooperation services: a *loyalty management service* is deployed in order to support coalition loyalty programs and the generation of e-coupons that can be implemented by means of digital codes. In particular, the first function has the signature *co-couponing(productX,merchantY):List[productZ,merchantT,discountD]* (if a customer buys product$X$ from merchant$Y$ he/she receives a list of e-coupons enabling a *discountD* for *productZ* acquired by *merchantT*). The second function has the signature *co uponing(context):List[productZ,merchantT,discountD]* (given a context the function returns a list of e-coupons that can be collected by a customer and used to buy a *productZ* from *merchantT* with a *discountD*). Definitely, we have several e-coupon types, but for the sake of simplicity we consider only two e-coupons types described as follows.

- (a) *e-coupons type 1*
  - (1) *gathering:* these coupons, associated to a specific product (target product) of a

specific merchant can be freely gathered by means of some NFC hot spot and collected in the customer's *e-Wallet*.

(2) *redeeming:* a coupon of this type can be redeemed by the customer when he buys the target product from the right merchant. It can be considered as a discount percentage on the target product.

(b) *e-coupons type 2*

(1) *gathering:* these coupons, associated to a specific product (target product) of a specific merchant, are generated when the customer buys another product (source product) from the same or from another merchant. The customer can gather these coupons by means of some NFC hot spot and store them into his *e-Wallet*.

(2) *redeeming:* a coupon of this type can be redeemed by the customer when he buys the target product from the right merchant. It can be considered as a discount percentage on the target product.

(iii) Context management services: in a simple case, the context of a customer can be constructed by transforming his/her wish list into a list of set of products by exploiting algorithms able to match a need for a product into a set of suitable products (retrieved from linked data) able to satisfy the customers need.

(iv) Algorithms: in this scenario, the focus is on designing a *shopping plan* algorithm able to suggest a list of products $List[productZ, merchantT]$ that satisfies the customers wish list minimizing the total cost of the shopping.

(v) Mobile apps: the apps we consider in this scenario are *e-Wallet* and *e-ShoppingAssistant*. The first one stores and manages the e-coupons gathered by the customers in the shopping mall by using capabilities (e.g., NFC or camera) of their smartphone and devices dislocated across the environment. The second one enable a customer to define a wish list and to ask for the shopping sequence that satisfies the list and minimizes the total cost for the customer by using e-coupons in his/her *e-Wallet*. The *e-Wallet* app provides also the capability to browse all the daily e-coupons offered by the merchants network of the mall.

Let us consider the following example in order to explain the shopping mall scenario. Rita is at home and prepares a partial wish list by using her smartphone (by using *e-ShoppingAssistant* App). The *e-Wallet* of Rita contains some e-coupons that can be exploited for her shopping. Rita goes to her preferred shopping mall. At the entrance she uses the NFC capabilities of her smartphone to acquire a set of e-coupons by means of the Loyalty Management Service (*couponing* function). Rita initially receives the e-coupons that are related to her wish list (by invoking the context management services); subsequently, she browses the entire list of available e-coupons. The acquisition of e-coupons and the update of the wish list are performed by using an iterative and incremental process sustained by a conversational user interface provided by the mobile apps. At this point, Rita executes the *e-ShoppingAssistant* app to receive suggestions about the shopping sequence that she has to consider in order to minimize the total cost. The app interoperates with the context management services and the shopping plan algorithm to obtain the desired result (*OrderedList[productZ,merchantT]*). The *shopping plan* algorithm needs the information coming from Rita's *e-Wallet* and interacts with the loyalty management service (*co-couponing* function) to perform its task. Now, Rita is ready to start her cost saving shopping. Rita can now perform her purchases by following the generated shopping plan and by gathering and redeeming e-coupons by means of the NFC devices of merchants in her plan.

Lastly, it is important to underline that user profiles and user profiler are not used in this scenario for the sake of simplicity but they are very important elements for both blended commerce and ambient intelligence.

3. Architecture

In this section, the proposed architecture for blended commerce based on the approach defined in Section 2.1 is described. The architecture tries to combine the capabilities of smartphones with ubiquitous computing in the domain of commerce and e-commerce, for creating the vision of the blended commerce by allowing users to shop anytime and anywhere in "intelligent ways" [13] (e.g., by reducing the overall cost of the shopping) with the aid of smart environment. This could enhance the user's experience and could represent a great opportunity for the vendors to increment the whole sales volume.

Mobile devices represent a key element of this architecture. In fact, the goal of ambient intelligence is to make more seamless the boundaries among physical and digital world, through the proper use of smart sensors and advanced services. In such a scenario, mobile phones represent a key element given that they are always with user, they are widely diffused everywhere around the world, they are quite always connected and they are capable of acquiring many contextual information (e.g., location) by means of their sensors. Moreover, it is important to consider that the design of an AmI architecture must be centred on user experience; the development of AmI applications must be user-driven rather than technological-driven; this suggests to us that it is important to realize context-aware applications and personalized user interfaces able to exploit intelligent user-centric behaviour.

Building a general architecture for ambient intelligence is still a very challenging task [14], due to heterogeneity of systems and dynamics of environments. The main challenges are related to the following.

(i) *Interoperability*: the AmI environment is characterized by the presence of different sensors, devices, network protocols, software, and so forth; all these

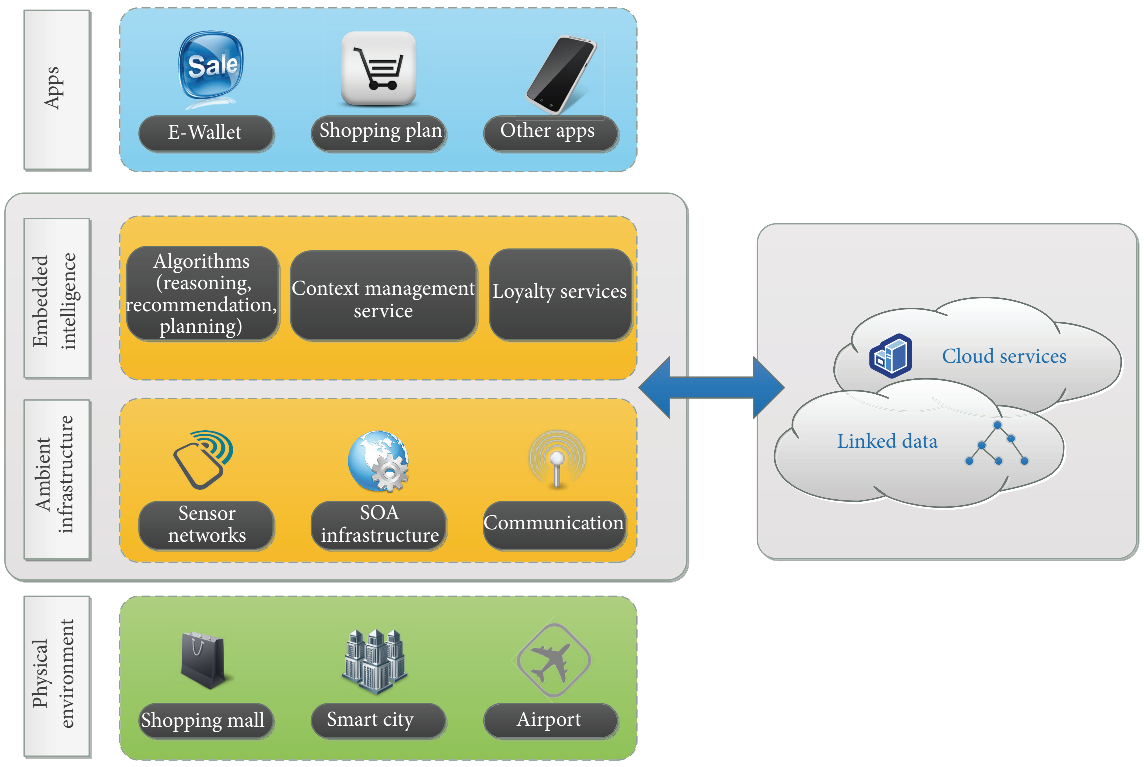


Figure 2: Architecture.

elements have to interact and share information. Making interoperable these elements represents a big issue, due to the heterogeneity of the used protocols and data formats.

(ii) *Adaptivity and personalization*: the applications have to correctly elicit user's intention, even in an autonomous and non-intrusive way; furthermore, they have to change their behaviour in order to adapt for the changing environment, by reacting to external stimulus.

(iii) *User experience*: providing intelligent user interfaces for human-computer interactions is critical for the success of AmI applications. From an architectural perspective, this means that the infrastructure has to guarantee adequate quality of service levels, high continuity and reliability even in wide, dynamic environment and with frequent interactions.

(iv) *Scalability*: the architecture has to properly scale with the growth of users, devices and physical environment, in order to guarantee an adequate service level.

The proposed architecture tries to solve the aforementioned main issues; a high level view of the architecture is depicted in Figure 2. It is constituted by four layers where each one is able to provide the needed capabilities required by the approach in Figure 1.

3.1. Ambient Infrastructure. In an AmI-based architecture, various sensors have to be integrated in order to give us the control of the environment, providing the means to access and share content and information, resulting in an immersive experience for the end user. In such a scenario, the *ambient infrastructure* layer plays a critical role in the proposed architecture, representing the backbone for the communication among sensors and services. In particular, aside from the aforementioned communication capabilities that are needed to connect sensors, devices and other systems, this layer has to adequately support services and apps:

(i) to acquire information from devices and sensors;

(ii) to exchange information with other services;

(iii) to interact with external services;

(iv) to access the (distributed) repository;

(v) to interact with users.

In order to enable these functionalities, it is needed to provide at least: (i) an *ambient network* able to connect sensors, devices and other computational resources (servers, repositories, cloud services, etc.); (ii) a software infrastructure to enable information exchange, cooperation and interoperability among services and applications.

The *ambient network* has to connect both sensors embedded into environment and users' devices that are used to access AmI applications and services. The development of large sensor networks represents a topic widely discussed in the scientific community [15, 16]. Realizing sensor networks usually implies the need to physically deploy and manage many customized sensor nodes. Consequently, in this kind of network, accessing and collecting sensor data efficiently require novel and complex networking protocols (e.g., energy-aware routing protocols, location based protocols) [16]. Moreover, with respect to a more expensive ad hoc sensor network, that requires adequate maintenance and management, the use of mobile phones for information acquisition represents an attractive and cheaper alternative. Modern smartphones are sophisticated devices with complex sensor capabilities, such as detecting location, measuring environment light, detecting orientation and so on. This allows the development of large sensor networks using cellular networks (e.g., UMTS) or Wi-Fi connections, by using apps able to collect and report sensor information to servers, in order to use these data to provide context-aware and personalized applications to users.

This kind of networks could enable users to interact with surrounding environments, using sensing and communication capabilities of their smartphones. Users, for instance, may interact with other devices of the physical environment (e.g., POS, kiosk, other mobile devices); this produces a need to interconnect these devices. The near field communications (NFC) technology was developed especially to meet this need. Section 3.1.1 describes this technology and how it could be useful for sensing the environment and sharing information in blended commerce scenarios.

Once sensor data has been collected, it must be processed in order to be available for AmI applications. Moreover, applications need to cooperate with other applications and services, by sharing information and using their capabilities. An infrastructure able to support this kind of cooperation and integration of new capabilities in the proposed architecture is required. The most suitable approach is to use service oriented architecture (SOA), an architectural style for modelling systems capabilities as a *network* of services. Section 3.1.2 gives further details on the role of SOA in the proposed architecture.

3.1.1. NFC. NFC is the key to enable context awareness in an AmI environment [17]. We are surrounded by a great amount of devices and their multiple functions. Interconnecting these functions one another can generate added value to the users. NFC, a short-range wireless connectivity technology, was developed to enable this interconnection. It is based on inductively coupled proximity radio frequency identication (RFID) technology [18, 19]. The two essential elements of any NFC system are the *initiator* (reader) and the *target* (tag). The initiator starts and controls the communication with the target, which is the device that responds to the initiator's request. NFC tags (e.g., stickers or wristbands) contain small microchips with little aerials which can store a small amount of information. These tags are used to store URIs, telephone numbers, text messages, or electronic business cards. Users can access the information on a tag by simply touching it with the NFC device (e.g., a mobile phone). Usually, when users touch a tagged device with their NFC cell, a preinstalled application is executed automatically, reading necessary information from the NFC tag and sending necessary information to the context management system via Bluetooth (or Wi-Fi) to request a specific service. The application logic of the invoked service may provide the result by processing information that are possibly retrieved from one or more distributed storage systems. Thus, the service can satisfy the request directly to the NFC cell. Data on a tag is structured according to the NFC data exchange format (NDEF, [20]). NDEF is a standardized format for storing formatted data on NFC tags and for transporting data across a peer-to-peer link between two NFC devices. In fact, NFC protocol not only supports communication between an active reader and a passive tag, but also allows for peer-to-peer communication between two active readers. Thus, an NFC-capable phone can both read a tag and receive and transmit data to another NFC-capable phone [21]. The use cases for NDEF cover smart posters, business cards exchanging, and initialization of other kind, especially wireless, communication. For instance, in a shopping mall scenario, a tag attached to a product may convey an Internet address which can redirect user to a site that provides further information about the product. This represents a valid example of how NFC in the aforementioned shopping mall scenario represents a key technology for turning a shopping mall into an intelligent environment. In fact, NFC Tags can be embedded in different objects of the shopping mall, as in products, coupons, offerings, product bundles, and so forth. When the user *"touches"* the NFC Tag, she could obtain extra capabilities, that is, acquiring extra information about the product, or accessing a specific service or acquiring a coupon. In the considered shopping mall scenario, NFC tags are used to identify and gather coupons that are disseminated into the stores of the shopping mall; customers could visualize information about the coupon and its characteristics and decide to acquire it, by putting it in their wallet, using the *e-Wallet* app.

3.1.2. Service Oriented Architecture. AmI environments embed different devices from various application domains. This requires architectural paradigms enabling loose and dynamic coupling among heterogeneous capabilities. Service oriented architecture (SOA) represents an appropriate architectural paradigm able to partially solve these issues. Unfortunately, AmI environments add further issues to SOAs, both at infrastructural and at application level [17]. These issues refer, for instance, to the need for dynamically identifying available services in open network environments and to interact with them. This requires enabling interoperability of both discovery and communication protocols.

Semantic web could represent a valid solution for dealing with these issues: by using semantic web services (SWS) and reasoning algorithms it is possible to enable dynamic service discovery and composition. Nevertheless, facing semantics and behaviour heterogeneity is still a challenging issue, especially in AmI environment, considering the fact that semantic reasoning is a computing expensive task [22]. This raises serious issues about its applicability in AmI environment in which applications could run on wireless devices with limited computing resources. To solve this issue, this work proposes the adoption of cloud infrastructures for services and data deployment, thanks which it is possible to easily obtain computational resources as needed.

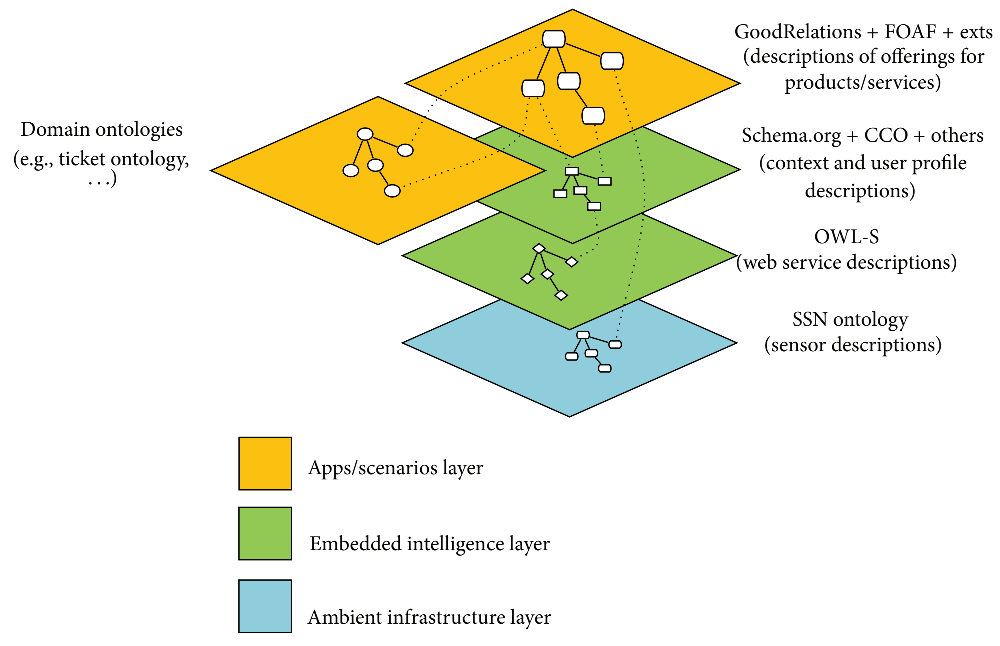

GoodRelations + FOAF + exts
(descriptions of offerings for
products/services)
Domain ontologies
(e.g., ticket ontology,
. . .)
Schema.org + CCO + others
(context and user profile
descriptions)
OWL-S
(web service descriptions)
SSN ontology
(sensor descriptions)
Apps/scenarios layer
Embedded intelligence layer
Ambient infrastructure layer

FIGURE 3: Linked data for AmI-based bended commerce.

3.1.3. Linked Data for e-Commerce. Effective data representation and management is essential for providing users with innovative and enhanced customer experiences. High quality information must be available to any user, anytime, anywhere, and on many devices, supporting the realization of new applications in the context of AmI. In particular, these applications have to be able to use also contextual information acquired by the sensors of the *Ambient Infrastructure* in order to exploit context-aware behaviours. Data representation and management have to support also the cooperation and interoperability among different applications: the cooperation of several services and applications is fundamental to provide comprehensive and innovative customer experiences. The realization of a data layer able to support the development of new Apps in the AmI environment raises a number of challenges, mainly due to the heterogeneity of data (given that those data are produced by different sensors or different applications or services or come from external data sources). Moreover, these data should be machine-interpretable in order to allow services to *understand* and process them automatically. We need efficient methods and solution to structure, annotate, share, make sense of the available data, and facilitate transforming it to *actionable knowledge*. Issues related to interoperability, automation, and data analysis suggest the adoption of a semantic approach. Applying semantic technologies to AmI promotes interoperability among resources, data models, data providers and consumers, and facilitates effective data access and integration, resource discovery, semantic reasoning and knowledge extraction. The proposed architecture uses the technologies of the semantic web and the linked data principles to address these issues and to provide a common and easily extensible data model.

Let consider the architecture depicted in Figure 2. Each level has different kind of information and resources that have to be semantically described by means of different ontologies. In particular, it is necessary to describe (i) sensor characteristics and data; (ii) service capabilities and characteristics for enabling dynamic service discovery and composition; (iii) context information that are obtained by sensor data processing; (iv) user profile for enabling personalization of the user experience; (v) domain-specific information and resources related to the particular AmI scenario.

Figure 3 depicts the main ontologies used in each layer of the architecture. The *ambient infrastructure* layer consists of *W3C Semantic Sensor Ontology* (W3C Semantic Sensor Ontology http://purl.oclc.org/NET/ssnx/ssn) for describing sensors, observations, and related concepts. It does not describe domain concepts like time, locations, temperatures, and so forth, that could be represented by using domain-specific ontologies. In the *embedded intelligence* layer, *OWL-S* (OWL-S: Semantic Markup for Web Services http://www.w3.org/Submission/OWL-S/) ontology is used for describing service capabilities and interfaces. Thanks to *OWL-S* it is possible to realize semantic matchmaking algorithms that could improve the selection of the most suitable service (with certain capabilities or characteristics) even for software agents; furthermore, it represents an enabler for semantic interoperability. The *embedded intelligence* layer provides functionalities for context and user profile management. *Schema.org* (Schema.org http://schema.rdfs.org/) and *mIo! Ontology Network* (mIo! Network Ontology http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/82-mio-ontologies), along with domain specific ontology like *Geonames* (Geonames Ontology http://www.geonames.org/ontology/documentation.html) and *Time Ontology* (Time Ontology http://www.w3.org/TR/owl-time/), are used to represent the main contextual information. User profiles are represented by means of *FOAF* (FOAF Vocabulary http://xmlns.com/foaf/spec/) and *Cognitive Characteristics*

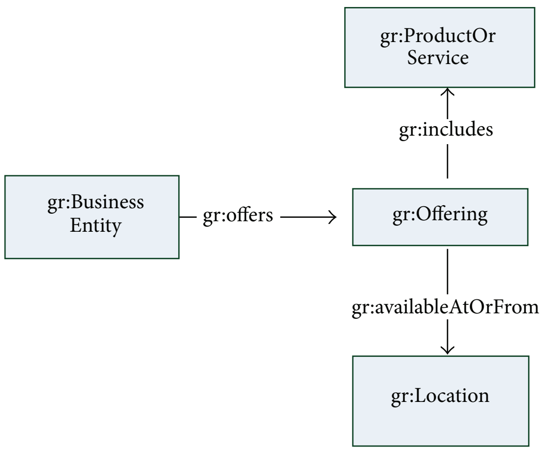


Figure 4: Describing offering by using GoodRelations.

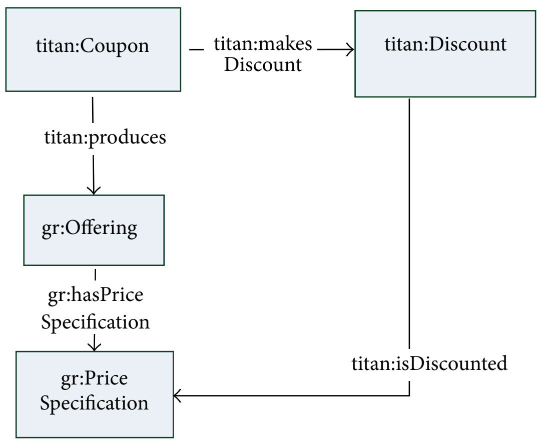


Figure 5: Conditional coupon model [24].

Ontology (*CCO*) (The Cognitive Characteristics Ontology 0.2 http://purl.org/ontology/cco/core); these ontologies allow describing information about people, like, for instance, interests, cognitive aspects, skills, affective states, and so on. The highest level of the architecture contains apps related to the defined AmI scenario.

For the aforementioned shopping mall, for instance, this work proposes to use an ontological model based on the *GoodRelations* ontology [23] and its extensions. *GoodRelations* is an ontology for describing typical e-commerce scenarios. It is based on four principal entities: the agent (an organisation or person), the object (the product or the service), the offering, and the location where it is available. These entities are described in *GoodRelations* by means of the following classes: `gr:BusinessEntity`, `gr:ProductOrService`, `gr:Offering`, and `gr:Location` (see Figure 4). The `gr:BusinessEntity` class is used to describe the information about business partners (e.g., name, category, etc.); `gr:Offering` indicates the characteristics of the offerings (e.g., availability period and SSN). Moreover, `gr:Produc-tOrService` and `gr:PriceSpecification`, respectively, describe the offered products/services and their prices.

Furthermore, in order to better describe, organize, and specialize information about products and services, it is possible to use additional domain ontologies. For instance, the *Ticket Ontology* (Ticket Ontology http://www.heppnetz.de/ontologies/tio/ns) is used for a deeper description of ticket offerings. In the Shopping Mall scenario we are also interested in modelling coupons and coalition loyalty programs. For example, in order to model a coupon offered by two merchants that realize a dynamic coalition in order to provide some discount to their users. To model such a situation, the authors in [24] propose some extensions to *GoodRelations* (see Figure 5).

In particular, two new classes have been defined. The `titan:Coupon` class for describing a conditional coupon offered by a coalition composed by two merchants, and the `titan:Discount` class, used for describing the discount generated by redeeming a coupon are defined; moreover, the following set of properties is added:

(i) `titan:produces`, describing that a coupon is produced by a specific offering;

(ii) `titan:makesDiscount`, indicating that a coupon generates a discount;

(iii) `titan:isDiscounted`, describing that a discount refers to an offering of a specific business partner.

Figure 6 shows an instance of the aforementioned model. In this case, there is a coalition between two merchants, who propose, respectively, two offerings: the first one for trouser and the second one for a t-shirt. The agreement between the two merchants specifies that the offering for the trouser generates a coupon which gives a discount of 10% for the offering of the t-shirt. The two offerings are modelled as instances of `gr:Offering`, `titan:t-ShirtCoupon` and `titan:t-ShirtDiscount` are instances of `titan:Coupon` and `titan:Discount`.

Linked Data in AmI environment requires scalable storage and retrieval infrastructure, due to the huge size of data and to the high rates of lookups. This huge amount of data raises several issues: their storage and management require a lot of IT resources; these large datasets have to be processed by means of computational-expensive algorithms, like reasoning algorithm; furthermore, they have to be properly managed by technically skilled IT professionals. In addition, the amount of requests from users for this data can be very variable, with the presence of short periods in which there are many requests. This variability may result in the need to oversize the IT infrastructure to cope with sudden bursts of requests from users. The actual trend to overcome these issues is the adoption of cloud-based solutions. The cloud computing model is a good choice for linked data since it provides unlimited resources on demand for collecting, analysing, visualizing, and processing huge amount of data. Cloud storage, in fact, provides two clear advantages. Firstly, it lets companies analyse massive data sets without making a significant capital investment in hardware to host the data internally, thanks to the "pay per use" business model.

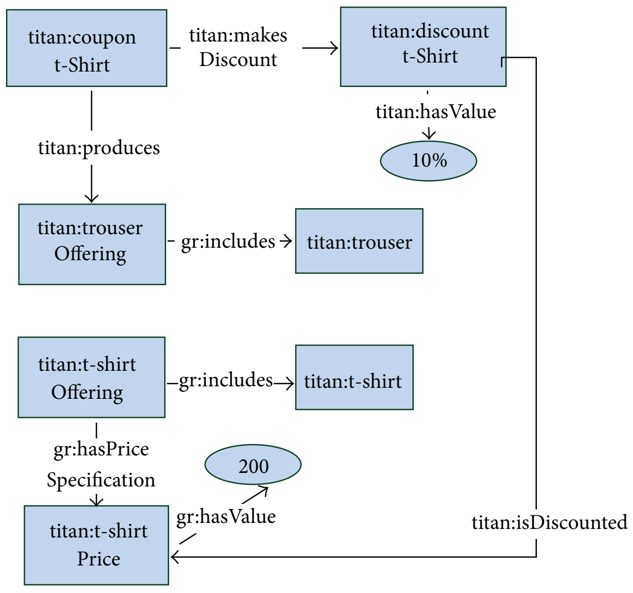


Figure 6: Conditional coupon example [24].

Secondly, since linked data hosting platforms require new skill training activities for IT staff, a hosted model can enable a more immediate deployment of linked data technology.

Given the benefits of a cloud-based solution, this paper proposes the deployment in a cloud infrastructure of linked data and of the data access and reasoning services. A possible approach for storing and processing linked data in cloud is described in [25].

This is just one of the possible deployment models of the architecture. In fact, it is still possible to deploy the linked data on traditional data center, assuming you have adequate computing power. At the other extreme, instead, it is possibly to deploy linked data and also all the services (even those belonging to *embedded intelligence* layer) in cloud.

3.2. Embedded Intelligence. The *embedded intelligence* layer represents the core of the architecture. This layer supports the realization of novel, improved user experiences in ambient intelligence. In particular, this layer hosts (i) context management capabilities that provide applications the necessary information realizing context-aware behaviours and (ii) a set of algorithms and services providing context-aware capabilities of reasoning, recommendation, planning, and so forth. In this section, we present two services of the *embedded intelligence* layer: *context management service* in Section 3.2.1 and *shopping plan algorithm* in Section 3.2.3, to solve shopping plan problem described in Section 3.2.2.

3.2.1. Context Management Service. Several definitions of *context* could be provided; for our purposes, *context* is any information that can be used to characterize the current situation of the user's environment [26]. This information needs to be described in a structured and easily extendible model to facilitate the sharing of collected information among many applications and services. In the proposed architecture, this model is represented by means of several ontologies, as described in Section 3.1.3. In this section, *context management service* architecture is described. This service has the aim to collect data of sensors and represent them using the ontologies, offering these data to apps.

In particular, the main capabilities of *context management service* are (i) perceiving world and users' data through sensors of the *ambient infrastructure* layer; (ii) representing this data in a coherent *context* model (context perception) by means of *context* ontologies deployed in cloud, as described in Section 3.1.3; (iii) understanding and evaluating the situation (context awareness); (iv) selecting/executing a suitable behaviour possibly in synergy with one or more algorithms/services of the *embedded intelligence* layer (context sensitivity).

In Figure 7 the main modules of the *context management service* is depicted. The *information acquisition* module acquires data from sensors, using different policies: acquisition of data at regular intervals, acquisition of data on particular events (e.g., when the value of a sensor exceeds a certain threshold), and so on. Subsequently, data is processed and aggregated appropriately by the *context manager* module: this module will translate the heterogeneous data collected by different sensors into a common format, by using the context ontologies for the mapping process. This allows apps to use and share contextual information that in origin are represented in different formats. Thanks to the ontological representation of *context*, the *context manager* could also apply a reasoning algorithm to infer new *context* information by deducing higher quality information (like the activity of a user) from lower level data (like the location of a user). Apps could be aware of the state of the user's *context* in two different modalities:

(i) *Pull*: the app requires specific information about context from the *context manager*;

(ii) *Push*: *context manager* alerts apps in case some context parameters are changed; apps should indicate in which context parameter they are interested.

In order to provide more flexibility for applications with respect to the ability of sensing and reacting to the changes in the *context*, the paper proposes a rule-based approach to context-awareness. In this case, the *awareness* can be described by a widely-used automation control pattern: the IF-THEN pattern. This pattern allows a specific action to be taken once a predetermined condition is satisfied [27]. The general format of IF-THEN pattern is: IF *condition* THEN *do something*, where the *condition* is a logic predicate related to some contextual information; if *condition* is true some *event* or *notification* may be sent to interested apps. These IF-THEN rules are evaluated and executed by *rule engine* module, depicted in Figure 7. All the rules are stored in a rule base; the *rule engine* executes them and send a notification to apps (if the action requires sending an informative message to user) or it triggers a specific event (if the verification of the condition requires the execution of some action by apps). Further details on rule engine and context manager modules are out of the scope of this work.

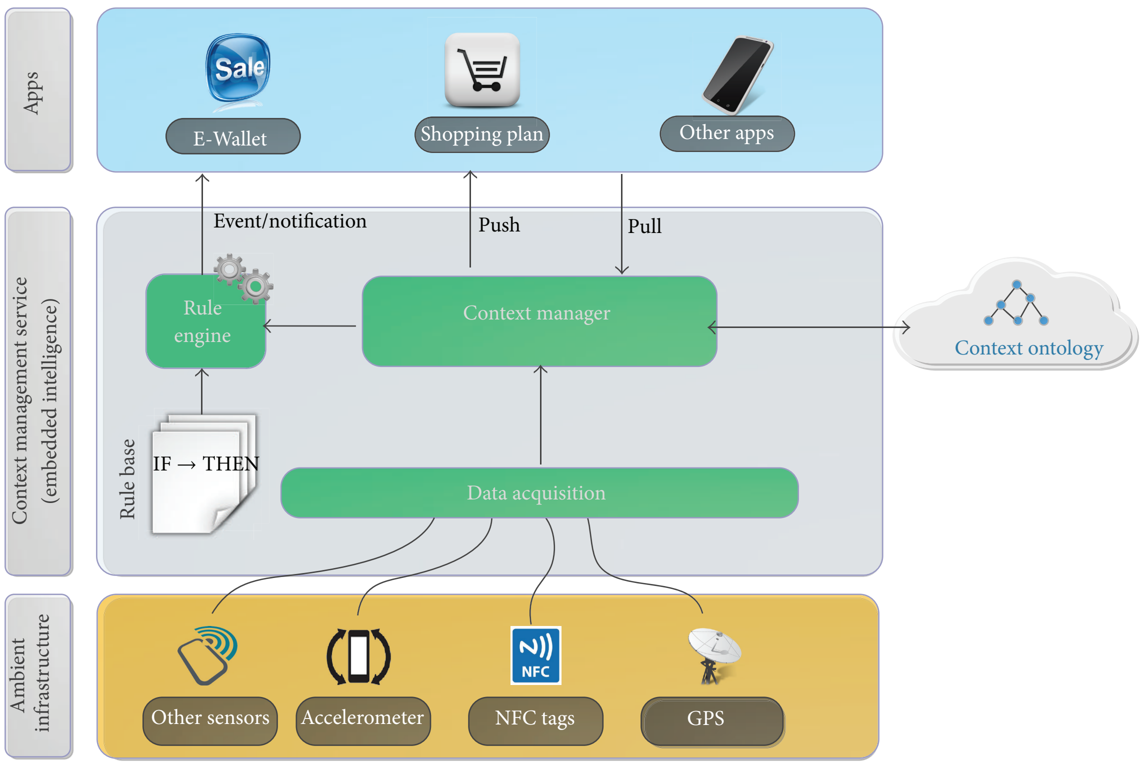


FIGURE 7: Context management service.

3.2.2. Shopping Plan Problem. This plan briefly describe the shopping plan problem, also presented in [28]. Let us assume that a customer has the wish list $\{w_1, w_2, w_3\}$ and that w_1 can be satisfied by the product set $\{a_1\}$, w_2 by $\{b_1\}$, and w_3 by $\{c_1, c_2\}$. The product sets matching the items in the customer's wish list are obtained by using discovery capabilities on the semantic layer constituted by the Linked Data for e-Commerce (see Section 3.1.3). Now, let us assume that the costs of the products are

$$\operatorname{cost}\left(\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ c_2 \end{bmatrix}\right) = \begin{bmatrix} 2 \\ 5 \\ 3 \\ 4 \end{bmatrix}. \tag{1}$$

In this case, binding the customer's wish list with a_1, b_1, c_1 has total cost 10, while adopting the solution a_1, b_1, c_2 has total cost 11. It is clear that the first solution is better than second one in terms of cost saving for the customer. Changing the sequence of products does not modify the aforementioned considerations given that the acquisition of a product, in this case, is independent from the acquisition of other products. But, if we include in our scenario the use of e-coupons (in particular those of type 2), we have some surprises. For instance, let us assume that the purchase of the product c_2 enables the customer to collect an e-coupon providing a discount of 40% on the product b_1. Thus, the solution a_1, b_1, c_1 with cost 10 is no longer the best solution. Now, the best solution is the ordered sequence $[a_1, c_2, b_1]$ with cost 9. Note that the order of purchases is relevant, because the customer can redeem his e-coupons only if he buys c_2 before b_1. In fact, the sequence $[a_1, b_1, c_2]$ has cost 11. Indeed, the sequences $[c_2, b_1, a_1]$ and $[c_2, a_1, b_1]$ are also best solutions.

In a more general form, we have a customer's wish list represented by a set of items:

$$\{w_1, w_2, \ldots, w_n\}. \tag{2}$$

Moreover, by means of matching operations with the linked data for e-commerce we are able to construct n sets of products:

$$P^1, P^2, \ldots, P^n, \tag{3}$$

where

$$P^i = p_1^i, p_2^i, \ldots, p_{k_i}^i. \tag{4}$$

P^i is the set of all products that bind the need for item w_i in the wish list. Let us assume that a set of functions are available. In particular,

(i) *merchant*(p) returns the information about the merchant selling p;

(ii) *cost*(p) returns the cost of p;

(iii) *wallet*(p) returns the discount percentage for product p obtained by applying coupons of type 1 taken from the customer's *e-wallet* (typically it is possible to have at most one valid coupon of type 1 for each product);

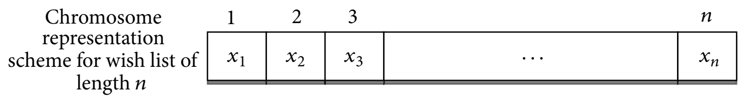

Chromosome representation scheme for wish list of length n
1
2
3
n
x_1
x_2
x_3
...
x_n

FIGURE 8: Chromosome structure.

(iv) *coupon*(p_1, p_2) returns the discount percentage for product p_2 obtained by applying a coupon of type 2 gathered by purchasing product p_1 before of product p_2.

Now, with $P = P^1 \cdot P^2 \cdot \ldots \cdot P^n$ being the cartesian product of $P^1, P^2, \ldots, P^n$, then $\overline{P}$ is the set constructed by the unions of all permutations of all elements in P. For instance, if $n = 3$, $P^1 = \{a_1, a_2\}$, $P^2 = \{b_1\}$, and $P^3 = \{c_1, c_2\}$ then $\overline{P}$ is constructed as follows:

$$\begin{aligned}\overline{P} = \{ & \\ & \{a_1, b_1, c_1\}, \{a_1, b_1, c_2\}, \{a_2, b_1, c_1\}, \{a_2, b_1, c_2\}, \\ & \{b_1, a_1, c_1\}, \{b_1, a_1, c_2\}, \{b_1, a_2, c_1\}, \{b_1, a_2, c_2\}, \\ & \{c_1, b_1, a_1\}, \{c_1, b_1, a_2\}, \{c_2, b_1, a_1\}, \{c_2, b_1, a_2\}, \\ & \{c_1, a_1, b_1\}, \{c_1, a_2, b_1\}, \{c_2, a_1, b_1\}, \{c_2, a_2, b_1\} \\ & \}.\end{aligned} \tag{5}$$

Finding a solution for shopping plan means finding the element $x = [x_1, x_2, \ldots, x_n]$ of set $\overline{P}$ with the minimal total cost.

The cost function for the element x is defined as

$$c(x) := \sum_{i=1}^{n} \gamma_i(x_i), \tag{6}$$

where $\gamma_i(x_i)$, namely, the cost associated with x_i, ith component of x, is written as

$$\gamma_i(x_i) := \text{cost}(x_i)\left(\text{wallet}(x_i) + \sum_{j=1}^{i-1} \text{coupon}(x_j, x_i)\right). \tag{7}$$

Let us underline that (6) and (7) foresees that possible discounts provided by several coupons are cumulable; indeed, it is also possible to add some further constraints and/or conditions to model situations in which coupons are either partially cumulable or not cumulable.

Searching the minimum of (6) is in this case not trivial for the following reasons: the cost function is of polymorphic type as its coefficients are highly variable for possible discounts of products and variations of the users' wallets; there is a so called "memory effect" because the cost associated to x_i can be related to the application of a coupon gathered by purchasing a product x_j, $j \neq i$, before of product x_i. Another important remark concerns suitable methods for obtaining numerical results. For this task, some studies have already been made. For instance, shopping optimization through minimization of costs associated to the items to buy is described as a NP-hard problem in [29], where a suboptimal solution is found via an opportune heuristic. A similar problem for the Internet shopping optimization problem (ISOP) is also better formalized in [30], where NP-hardness is rigorously proved and polynomial time algorithms are proposed for searching solutions. In general, the models described above are highly static, as they focus properly neither on possible dynamic fluctuations of items prices by coupons or users' wallets. This suggests the adoption of alternative ways for solving our problem. In particular, this paper proposes a canonical genetic algorithm.

3.2.3. Shopping Plan Algorithm. This paper proposes a genetic algorithm to solve the shopping plan problem described in Section 3.2.2; this algorithm is also described in [28]. This algorithm is part of the *embedded intelligence* layer.

First of all, let us to describe the chromosome representation as depicted in Figure 8. The chromosome represents an admissible ordered sequence of products to purchase with respect to the n items in the customer's wish list. So, the product x_i is the ith product to be purchased. More in details, the ith element x_i of the chromosome can be written as p_k^j, that is, the kth product of the jth product set satisfying the jth item in the customer's wish list (see the definition of wish list in formula (2) and sets of products satisfying it in formulas (3) and (4)). Now, we can assert that a chromosome is admissible if it respects the following conditions:

(i) for each element w_j in the customer's wish list there exist exactly one element x_h (in the chromosome) containing a product $p_i^j \in P^j$ that satisfies w_j;

(ii) if w_j and w_k (with $j \neq k$) are elements in the customer's wish list and w_j is satisfied by element x_{h_1} and w_k is satisfied by element x_{h_2} in the chromosome, then $h_1 \neq h_2$;

(iii) if x_{h_1} and x_{h_2} (with $h_1 \neq h_2$) are elements in the chromosome and x_{h_1} contains product $p_i^j \in P^j$ and x_{h_2} contains product $p_u^k \in P^k$, then $j \neq k$.

The shopping plan algorithm follows the canonical genetic algorithm structure reported as follows.

Choose an initial population.

Determine the fitness of each individual.

Perform selection.

Repeat.

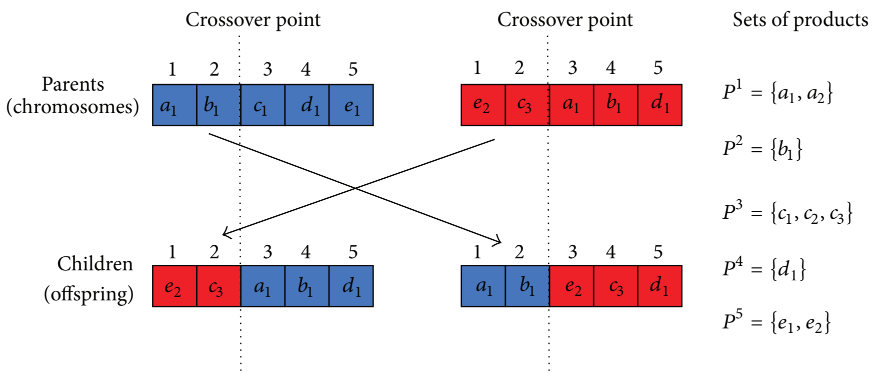


FIGURE 9: Example of crossover.

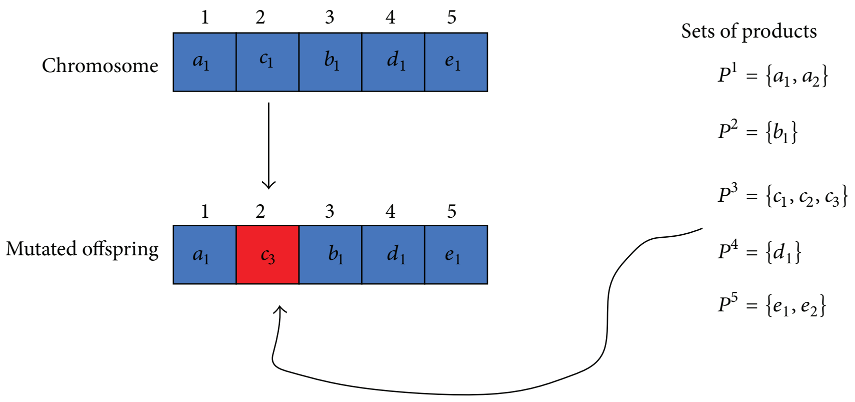


FIGURE 10: Example of mutation.

Perform crossover.
Perform mutation.
Determine the fitness of each individual.
Perform selection.

Until some stopping criterion applies.

Let us go more in details for the main elements of the algorithm. Firstly, the initial population is randomly generated by using uniform probability distribution. Secondly, the fitness function exploits the cost function described in the formulas (6) and (7). Thirdly, the selection is performed by exploiting the *Rank Selection* approach where every chromosome receives a rank by considering its fitness measure. The worst has rank 1, the best has rank M (if the cardinality of the population is M). In this case, the selection probability depends on the relative fitness rather than the absolute fitness. Fourthly, the crossover operator is implemented by starting from the *single point crossover* approach. An example of the crossover behaviour in the shopping plan algorithm is reported in Figure 9. The point of crossover is randomly identified in the chromosome. The proposed crossover consists in inverting the elements of the chromosomes before the crossover (chromosome 1 for offspring 2 and chromosome 2 for offspring 1) and maintaining the admissibility of the offsprings by selecting adequate elements of the parents (chromosome 1 for offspring 1 and chromosome 2 for offspring 2).

Fifthly, an *artificial, chromosome-level mutation* [31] is applied: the mutation operator is implemented by randomly identifying an element in the chromosome and replacing it by using an element randomly selected in the same set of products (Figure 10).

Lastly, the stopping criterion we adopt is to stop the computation when no better fitting chromosomes are generated after σ (experimentally calculated) iterations of the main algorithm cycle.

3.2.4. Shopping Plan Algorithm Evaluation.

3.2.4. Shopping Plan Algorithm Evaluation. The evaluation and tuning of the *shopping plan* algorithm parameters is achieved by using a performance measure, namely, *likelihood of optimality*; it represents the quality of the execution of the algorithm with respect to the optimal solution. Likelihood of optimality Lopt(k) is defined in [32] by considering that the algorithm was executed for *k generations* in each of *n runs*. Let m be the number of runs which produced an optimal solution within k generations. The likelihood of optimality Lopt(k) at the kth generation is the estimated probability m/n.

This performance measure allows deciding a *cut-off* generation K, that is, how many generations the algorithm should be executed in each run. Let $C = kr$ be the total computation cost given to execute the algorithm, where r is

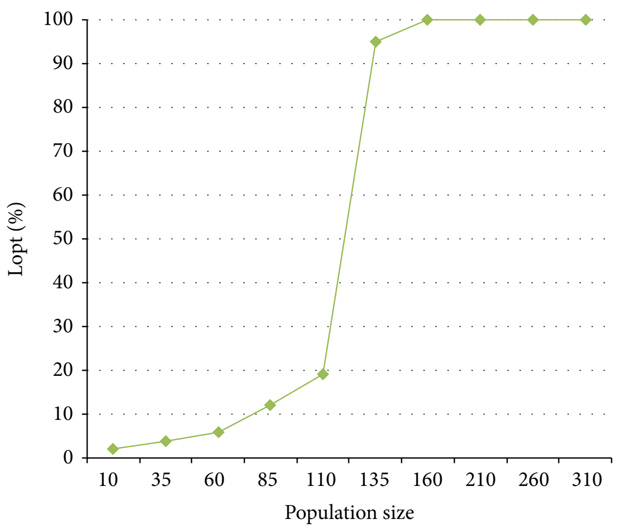


Figure 11: Lopt(k) and population size.

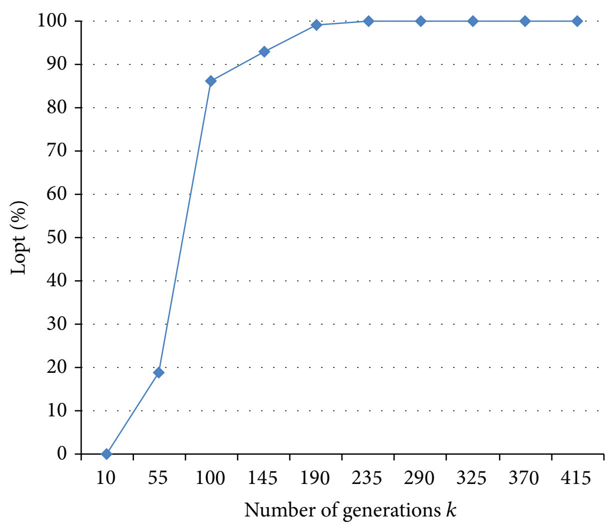


Figure 12: Lopt(k) and number of generations k.

the number of repeated runs. If C is fixed, we want to find k maximizing $(1 - p(k))^r$, where $p(k)$ denotes the probability that the algorithm produces an optimal solution within k generations. It is well-known that diversity in a population plays a key role to achieve an optimal solution. On the other hand, approaching to convergence decreases the diversity. Hence, there is a trade-off between convergence speed and solution quality. Furthermore, for the shopping mall scenario, the computational time is bounded *a priori*, in order to compute a solution in a suitable time for the user who is interacting with a mobile app. In this case, the convergence of a population is not appropriate to decide the termination of the algorithm. Therefore, we focus on direct correlations between solution quality and computational cost without considering the convergence. We have planned and executed different experimentation scenarios to evaluate algorithm performances with respect to the variation of population size, number of generations and crossover rate, in order to identify the best set of parameters for solving shopping plan problem.

The simulations have been executed on the same dataset by varying the algorithm parameters; in each experimentation scenarios the algorithm was executed for 100 runs. The dataset for the evaluation has been generated randomly with the following characteristics:

(i) number of items in the wish list = 10;

(ii) average number of product for each set of products = 100 and variance = 10 (generated by using a Gaussian probability distribution);

(iii) probability to use a coupon for a single product = 0.5;

(iv) probability to generate a coupon for a single product = 0.5.

The first experimentation consists of calculating the Lopt(k) when the population size varies from 10 to 300, assuming that the number k of generations is 200, the crossover rate is 0.8 and the mutation rate is 0.3. Figure 11 shows the results of this experimentation. It is possible to observe that, for population size bigger than 150, the Lopt(k) assumes values near 100%.

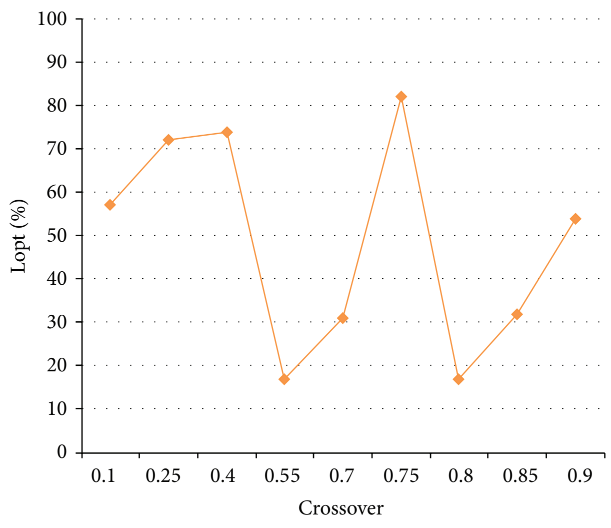


Figure 13: Lopt(k) and crossover rate.

Figure 12 shows Lopt when k varies from 10 to 500, assuming that population size is 150, the crossover rate is 0.8 and the mutation rate is 0.3. This figure clearly indicates saturation as the number of generations increases (in particular, after $k = 200$).

Figure 13 shows the likelihood of optimality Lopt when the crossover rate varies from 0.1 to 0.9, assuming that population size is 150, generations k is 200 and mutation rate is 0.3. The above experimentations results suggest that the best parameter setting is population size = 150, crossover rate = 0.75, mutation rate = 0.3, and number of generations = 200. This is used as the standard configuration for solving shopping plan problem.

With this set of parameters, we have conducted other two experimentations. The first one is focused on comparing the performances, in terms of total discount for the customer, of the genetic algorithm with respect to a good solution that can be rapidly found without considering the analysis of the space of admissible solutions. The second one is focused on comparing time and average error performances of the genetic algorithm with respect to those of a brute force algorithm. Let us describe the first experimentation scenario. The used dataset has been generated randomly with the following characteristics:

(i) number of items in the wish list = 20;

(ii) average number of product for each set of products = 1000 and variance = 100 (generated by using a Gaussian probability distribution);

(iii) probability to use a coupon for a single product = 0.06;

(iv) probability to generate a coupon for a single product = 0.04.

Moreover, the genetic algorithm is configured as follows:

(i) crossover probability = 0.8;

(ii) mutation probability = 0.3;

(iii) number of individuals in the initial population = 300;

(iv) number of generations = 150.

Figure 14 reports the results of the genetic algorithm executions. In particular, the obtained results are

(i) cost of the shopping calculated by using lower prizes products and available coupons in the wallet (no alternative plans are considered) = 1726.95€;

(ii) value of the best solution found by the genetic algorithm = 1260.56€;

(iii) total discount obtained by the genetic algorithm = 466.39€ (27.01%);

(iv) genetic algorithm execution time = 4 seconds.

The above scenario demonstrates that the genetic algorithm computes a good solution for the shopping plan problem, providing a discount of the 27% of the total cost of the shopping (calculated without considering the coupons), in a reasonable execution time.

In the second scenario, the used dataset is configured as follows:

(i) number of items in the wish list = 5;

(ii) average number of product for each set of products = 10 and variance = 2 (generated by using a Gaussian probability distribution);

(iii) probability to use a coupon for a single product = 0.5;

(iv) probability to generate a coupon for a single product = 0.5.

The genetic algorithm has been executed with the following parameters:

(i) crossover probability = 0.8;

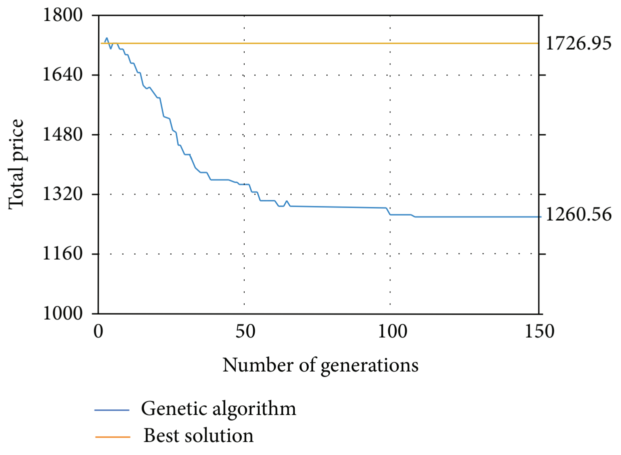


Figure 14: Results of the first experimentation scenario.

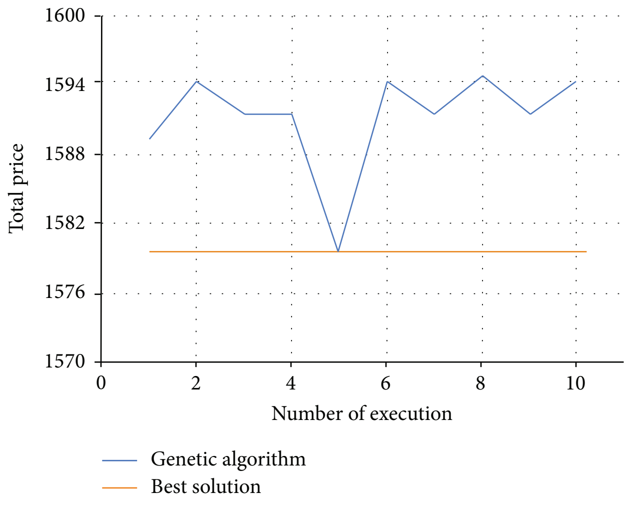


Figure 15: Results of the second experimentation scenario.

(ii) mutation probability = 0.3;

(iii) number of individuals in the initial population = 100;

(iv) number of generations = 200;

(v) number of executions = 50 (10 blocks of 5 executions).

Figure 15 reports the results of the genetic algorithm executions compared with the brute force algorithm results. In particular, brute force algorithm generates all possible solutions and exhaustively calculates the cost of each solutions returning the best (that with the minimum cost):

(i) cost of the optimal solution (brute force algorithm) = 1579.69€;

(ii) costs of the solutions for each 5 executions of the genetic algorithm = 1589.31€, 1594.19€, 1591.43€, 1579.69€, 1594.19€, 1591.43€, 1594.8€, 1591.43€, 1594.19€;

(iii) average advantage of brute force algorithm with respect to the genetic algorithm = 0.78%;

(iv) brute force algorithm execution time = 176281 milliseconds;

(v) genetic algorithm execution time = 109 milliseconds.

This second scenario demonstrates that the genetic algorithm computes a good solution for the shopping plan problem, with an error of 0.78% with respect to the optimal solution computed by the brute force algorithm. Moreover, the genetic algorithm has an execution time that is much lower than the one of the brute force algorithm.

3.3. Apps. The *apps* layer of the architecture contains the applications that allow users to interact with the surrounding environment, in order to exploit the intelligent behaviours provided by the *embedded intelligence* layer. The applications in this layer strongly depend on the specific AmI scenario. In the aforementioned blended commerce scenario, for instance, we could consider a set of Apps that enable users:

(i) to store and manage her credit cards and to use them for mobile payments;

(ii) to acquire, store, and redeem e-coupons ad e-vouchers;

(iii) to discover products and merchants able to satisfy her wish list;

(iv) to obtain a list of products and merchants that try to minimize the total cost of her wish list.

These functionalities could be provided by the Apps described in the following, namely, by *e-Wallet App* and *e-ShoppingAssistant App*.

3.3.1. E-Wallet App. The *e-Wallet App* implements a *digital wallet* that enables users to acquire, manage, and redeem e-coupons and e-vouchers. A *digital wallet* is a service allowing the holder to securely access, manage, and use identification and payment instruments in order to initiate payments. This service may reside on a device owned by the holder (e.g., a mobile phone or a PC) [33]. A *digital wallet* is mainly designed for financial transactions, but recently some applications used it as a means to provide further capabilities, as loyalty programs. In fact, one of the main obstacles to a wider adoption of *digital wallets*, and especially *mobile wallet*, lies in the lack of perceived usefulness by users. For instance, while paying by a *mobile wallet* is easier than paying with cash, it is not easier than paying with credit cards. So, it is necessary to add novel capabilities to digital wallet in order to increase their diffusion. For that reason, in our work we propose a mobile digital wallet allowing

(i) storing and using credit cards;

(ii) acquiring, managing, and using e-coupons and e-vouchers through NFC capabilities of the smartphone.

The *e-Wallet* application is composed of two parts: one, the *e-Wallet App*, represents the client thanks to which user can acquire and redeem coupons; two, the *e-Wallet-ManagementService*, is a Web service managing coupons acquisition policies and the server-side redemption process of coupons.

Although coupons can be distributed to the user in different ways (e.g., by web site or e-mail), we are mainly interested in their acquisition through physical environment. In an AmI scenario, in fact, user can obtain coupons by the interaction with the intelligent environment. Let us consider again the shopping mall scenario. The stores in the shopping mall can offer different discounts to customers by means of coupons, delivered through NFC tags attached to products or to other elements into the mall. Every NFC tag contains *URI* that identifies a single coupon. When users touch the tag, the *e-Wallet App* installed on his/her device is launched. The *e-Wallet App* communicates with the *e-WalletManagementService* that manages coupons of the shopping mall. This service checks if the user has the rights to acquire that coupon; if yes, user can put the coupon in his digital wallet for further uses.

When the customer goes to the store for purchasing the discounted product, she can obtain a discount by redeeming the coupon with the *e-Wallet* app. At the same time, she can pay by using her credit card and she can redeem the coupon to obtain a discount with the same App. On the other side, the merchant has a NFC reader that allows a P2P communication with NFC-enabled smartphone of customer. This NFC reader is supported by an app that manage the coupons by communicating with the service *E-WalletManagementService*. Coupon redemption is performed by NFC, using the NPP (NDEF Push Protocol) for managing the communication between the user's device and the NFC Reader of the merchant.

3.3.2. E-ShoppingAssistant App. The *e-ShoppingAssistant App* allows users to execute the *shopping plan* algorithm described in Section 3.2.3. In particular, this app allows

(i) defining a wish list and to obtain a list of suitable products able to satisfy the customer's needs;

(ii) obtaining a list of products that satisfy the customer's wish list, trying to minimize the total cost of the shopping.

These two functionalities are realized by means of two web services. The former, namely, *WishListMatching*, has the signature *wishListMatching(List[Goals]):List[ProductX,MerchantY]*. This service, given a wish list composed of user's goals (e.g., defined in natural language), is able to transform it in a list of *ProductX* sold by *MerchantY* that satisfy this wish list. In this process, the web service exploits algorithms (of the *embedded intelligence* layer) able to match a need for a product in a set of suitable products by retrieving them in linked data. Furthermore, the algorithm could use the information about user's context (collected by *context management service*) to identify those most suitable products for the customer according to her preferences or to the actual context.

The second web service, namely, *ShoppingPlanService*, has the signature *shoppingPlan(List[GoalN,ProductX,MerchantY], UserID):OrderedList[ProductZ,MerchantT]*. This service,

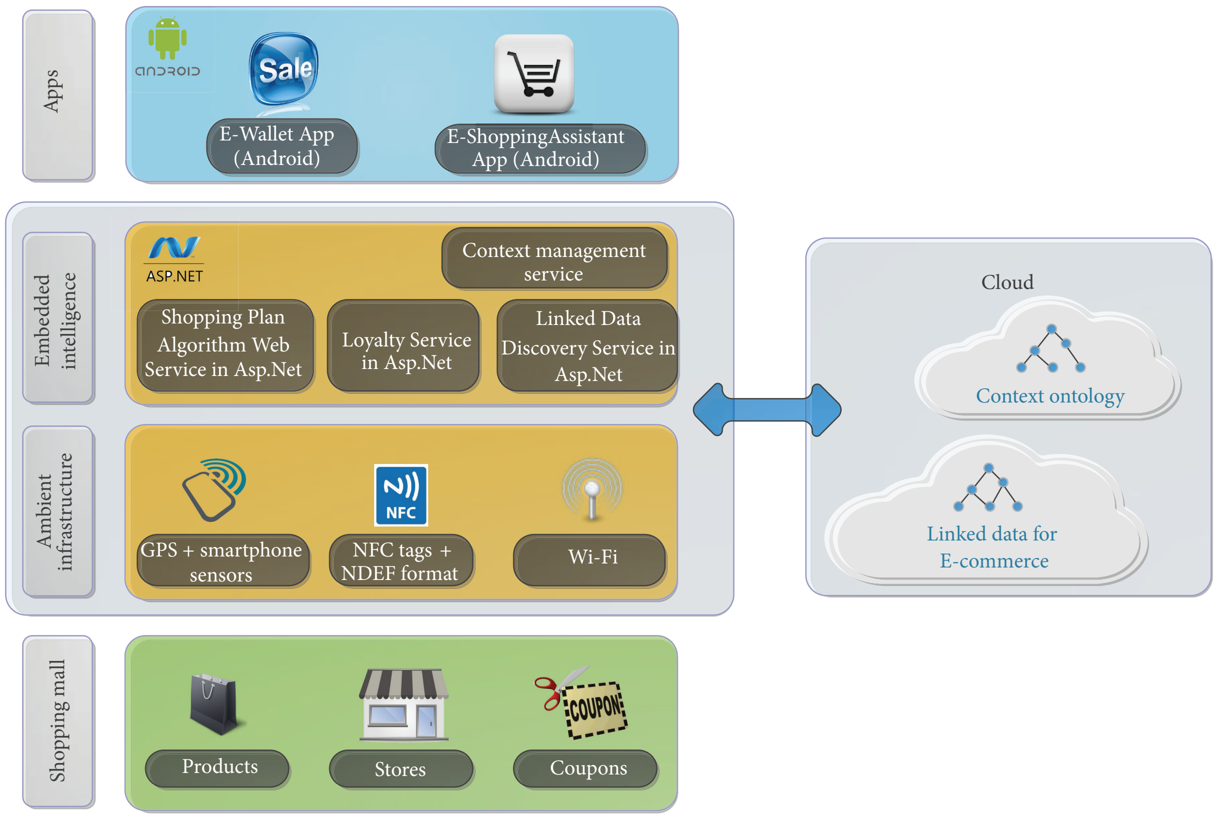

Apps
E-Wallet App (Android)
E-ShoppingAssistant App (Android)
Embedded intelligence
ASP.NET
Context management service
Shopping Plan Algorithm Web Service in Asp.Net
Loyalty Service in Asp.Net
Linked Data Discovery Service in Asp.Net
Ambient infrastructure
GPS + smartphone sensors
NFC tags + NDEF format
Wi-Fi
Cloud
Context ontology
Linked data for E-commerce
Shopping mall
Products
Stores
Coupons

Figure 16: AmI-based architecture for the shopping mall scenario.

given a list of *ProductX* sold by *MerchantY* that satisfies a *GoalN*, is able to identify an ordered sublist composed by *ProductZ* (sold by *MerchantT*) that satisfies every *GaolN* of the input list and that tries to minimize the total cost of the shopping; this service uses the coupons contained in the *e-Wallet* of the user (identified by the input parameter *UserID*) and those obtained during the purchase of products, for minimizing the overall cost. The output list is obtained by means of the shopping plan algorithm described in Section 3.2.3.

4. Prototype

This section describes the instantiation (a possible one) of the proposed architecture in the context of the aforementioned shopping mall scenario, depicted in Figure 16. In particular, the section describes the technologies and the services that are needed to implement this scenario; furthermore, a prototype of the *e-Wallet* and *e-ShoppingAssistant* apps are described.

The *physical environment* consists of the shopping mall, with all its facilities like stores, restaurants, open spaces, and so on; moreover, for this scenario, it is important to consider the products and the coupons offered by the merchants of the mall. The *ambient infrastructure* layer consists of the Wi-Fi network of the shopping mall, that it is supposed to be free and open to all the customers. The sensors are represented by those belonging to customers' smartphones. Lastly, this layer contains the NFC tags that represent the e-coupons that could be acquired by customers through the *e-Wallet* app. The *embedded intelligence* layer is mainly constituted by the *shopping plan* algorithm and by the *context management service*. Moreover, this layer contains the service for manage the e-coupons that are acquired and redeemed by users. This is implemented as a web service (namely, the *Loyalty Service*); this service decides if a user owns the rights for acquiring a specific coupon. Each coupon is an instance of the `titan:Coupon` class in linked data commerce and thus is identified by an URI. The *app* layer contains the *E-ShoppingAssistant* app and the *e-Wallet* app. The *e-Wallet* app is developed for Android mobile operating system. The app allows users to acquire new coupons by using the NFC capabilities of their smartphone. To achieve this, each NFC tag is formatted in NDEF format and contains the e-coupon URI and the AAR (Android Application Record) for identifying the App on the user' smartphone that has to be launched for managing the e-coupon. We have used the *NDEF Editor* (NDEF Editor: http://ndefeditor.com/) and the *NFC Developer App* (NFC Developer App: https://play.google.com/store/apps/details?id=com.antares.nfc) to write URI and AAR in the NFC tags. When a user touches the NFC tag formatted in NDEF, the *tag dispatch system* of Android tries to locate applications that are interested in the scanned data. In particular, the *tag dispatch system* creates an *intent* that encapsulates the NFC tag and

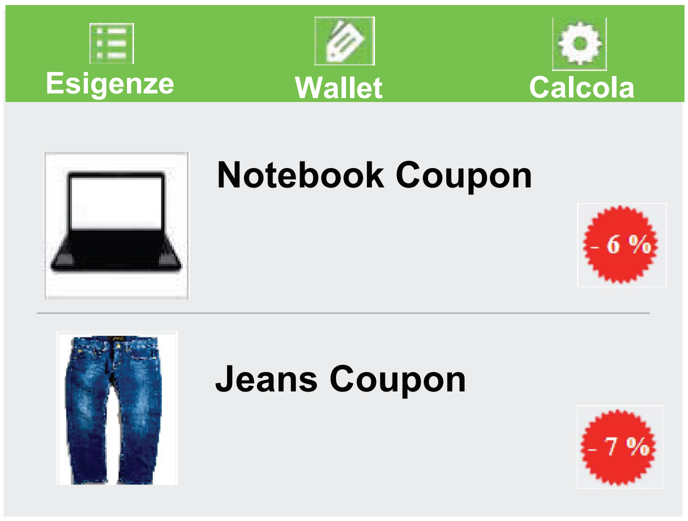


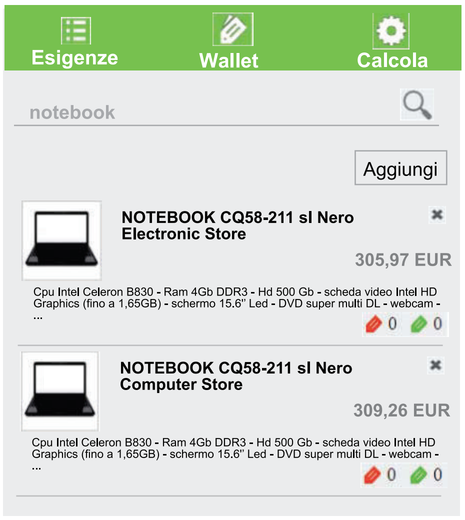
Figure 17: *E-wallet* app.

its identifying information and then sends it to an interested application that filters for the intent (NFC in Android: http://developer.android.com/guide/topics/connectivity/nfc/nfc.html). So, if user has installed *e-Wallet* app on its smartphone, this app is automatically launched when he/she touches a tag in the shopping mall. The *e-Wallet* app, after reading the content of the NFC Tag, sends a request to *loyalty web service* to check if user can obtain that coupon. If yes, the coupon is added in the *e-Wallet* of the user. In Figure 17 is shown a screenshot of the *e-Wallet* app: user has acquired two coupons, one for obtaining a discount on a notebook and another for obtaining a discount on jeans.

The *e-ShoppingAssistant* app allows user to indicate a list of goals and to obtain a list of products/merchants that satisfy these goals by minimizing the total cost of the shopping, using the coupons that are into the *e-Wallet*. For products/merchants identification phase, the app sends a request to the *linked data discovery* service of *embedded intelligence layer*. This service is able to selecting all the suitable products/merchants by matching user's request with the products/merchants descriptions in the linked data. The obtained list is then used by the shopping plan algorithm (wrapped into the *shopping plan algorithm* web service) to compute the ordered list that tries to minimize the overall costs. The communication among apps and web services use JSON notation (JSON http://www.json.org/) for representing the exchanged data. The web services of *embedded intelligence* are all implemented using Asp.Net Framework 3.5 (Asp.net Framework http://www.asp.net/).

Figure 18 shows the *e-ShoppingAssistant* when the user is adding a new goal to his/her shopping lists. In particular, the figure shows that the user is searching a notebook.

In Figure 19 is shown the list of products computed by the shopping plan algorithm; in particular, this example considers that the user wants to buy a notebook and a cordless phone. The list of products obtained give to the user a discount of 5.55 Euro, thanks to the use of one coupon and to the order in which the products are purchased.



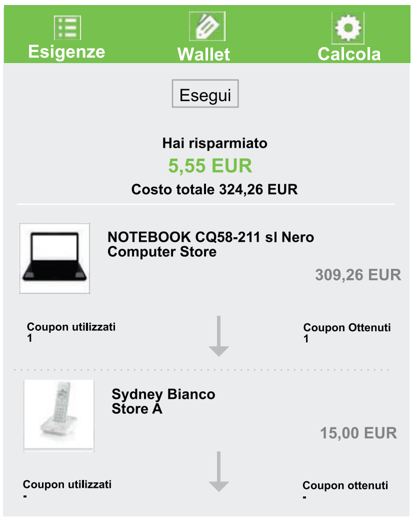
NOTEBOOK CQ58-211 sl Nero
Electronic Store
305,97 EUR
Cpu Intel Celeron B830 - Ram 4Gb DDR3 - Hd 500 Gb - scheda video Intel HD Graphics (fino a 1,65GB) - schermo 15.6'' Led - DVD super multi DL - webcam - ...
0 0

NOTEBOOK CQ58-211 sl Nero
Computer Store
309,26 EUR
Cpu Intel Celeron B830 - Ram 4Gb DDR3 - Hd 500 Gb - scheda video Intel HD Graphics (fino a 1,65GB) - schermo 15.6'' Led - DVD super multi DL - webcam - ...
0 0

Figure 18: *E-ShoppingAssistant* App: creation of the list of goals.

5. Final Remarks

The paper proposes a framework and an architecture for AmI-based blended commerce. The architecture has been instantiated in the shopping mall scenario taking care of e-couponing as a customers' tool to save costs for their shopping. In particular, the focus of the architecture is on mobile devices (e.g., smartphones, tablets). Moreover, the work defines a genetic algorithm to face the shopping plan problem with a heuristic approach. The problem can be solved by exhaustively searching, from all the admissible shopping plans, the one with the minimum cost. The main difficulty is due to the memory effect; that is, the cost of a product is dependent from the previous purchases in the plan. As shown by the performed experimentations, the genetic algorithm (we have defined) overperforms the brute force algorithm in terms of execution time. From the quality viewpoint, the



Hai risparmiato
5,55 EUR
Costo totale 324,26 EUR

NOTEBOOK CQ58-211 sl Nero
Computer Store
309,26 EUR
Coupon utilizzati 1
Coupon Ottenuti 1

Sydney Bianco
Store A
15,00 EUR
Coupon utilizzati -
Coupon ottenuti -

Figure 19: *E-ShoppingAssistant* App: resulting products list.

genetic algorithm provides good and acceptable results with respect to the optimum solution. Thus, the genetic algorithm is appropriate for AmI scenarios, as the execution time has a significant impact on the user experience. Lastly, this work proposes the implementation description of two Android apps (*e-Wallet* and *e-ShoppingAssistant*). In future works, we aim at comparing the defined genetic algorithm with other heuristic approaches, like Swarm Intelligence algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research is partially supported by the Italian Ministry of University and Research under the Project *TITAN "System for e-Money and Multi-Channel Value Added Service"*, PON RC, no. PON01_02136.

References

[1] R. T. Watson, "U-commerce the ultimate," ACM Ubiquitous Information Everywhere.

[2] A. Vasilakos and W. Pedrycz, *Ambient Intelligence, Wireless Networking, and Ubiquitous Computing*, Artech House, Norwood, Mass, USA, 2006.

[3] M. Gaeta, G. R. Mangione, F. Orciuoli, and S. Salerno, "Ambient e-Learning: a metacognitive approach," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 1, pp. 141–154, 2013.

[4] M. Weiser, "The computer for the 21st century," *Scientific American*, 1991, http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[5] A. R. Prasad, P. Schoo, and H. Wang, "An evolutionary approach towards ubiquitous communications: a security perspective," in *Proceedings of the 2004 Symposium on Applications and the Internet-Workshops (SAINT 2004 Workshops) (SAINT-W '04)*, p. 689, IEEE Computer Society, Washington, DC, USA, 2004.

[6] M. Napoli, M. Parente, and A. Peron, "Specification and verification of protocols with time constraints," *Electronic Notes in Theoretical Computer Science*, vol. 99, pp. 205–227, 2004.

[7] F. Palmieri, U. Fiore, and A. Castiglione, "Automatic security assessment for next generation wireless mobile networks," *Mobile Information Systems*, vol. 7, no. 3, pp. 217–239, 2011.

[8] A. Amato, V. Di Lecce, and V. Piuri, "A new graphical interface for web search engine," in *Proceedings of the IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems (VECIMS '07)*, pp. 42–46, June 2007.

[9] S. Keegan, G. M. P. O'Hare, and M. J. O'Grady, "Easishop: ambient intelligence assists everyday shopping," *Information Sciences*, vol. 178, no. 3, pp. 588–611, 2008.

[10] S. La Torre, M. Napoli, M. Parente, and G. Parlato, "Verification of scope-dependent hierarchical state machines," *Information and Computation*, vol. 206, no. 9-10, pp. 1161–1177, 2008.

[11] G. D'Aniello, M. Gaeta, V. Loia, F. Orciuoli, and S. Tomasiello, "A dialogue-based approach enhanced with situation awareness and reinforcement learning for ubiquitous access to linked data," in *Proceedings of the 6th International Conference on Intelligent Networking and Collaborative Systems (INCoS '14)*, pp. 249–256, 2014.

[12] A. Kofod-petersen and J. Cassens, "Explanations and context in ambient intelligent systems," in *Modeling and Using Context*, vol. 4635 of *Lecture Notes in Computer Science*, pp. 303–316, Springer, Berlin, Germany, 2007.

[13] K.-J. Lin, T. Yu, and C.-Y. Shih, "The design of a personal and intelligent pervasive-commerce system architecture," in *Proceedings of the 2nd IEEE International Workshop on Mobile Commerce and Services (WMCS '05)*, pp. 163–173, IEEE Computer Society, Munich, Germany, July 2005.

[14] G. Pan, L. Zhang, Z. Wu et al., "Pervasive service bus: smart SOA infrastructure for ambient intelligence," *IEEE Intelligent Systems*, vol. 29, no. 4, pp. 52–60, 2014.

[15] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[16] H. Turner, J. White, B. Dougherty, and D. Schmidt, "Building mobile sensor networks using smartphones and web services: ramifications and development challenges," *Handbook of Research on Mobility and Computing: Evolving Technologies and Ubiquitous Impacts*, pp. 502–521, 2011.

[17] G. Chavira, S. W. Nava, R. Hervás et al., "Services through nfc technology in ami environment," in *Proceedings of the 10th International Conference on Information Integration and Web-Based Applications and Services (iiWAS '08)*, pp. 666–669, ACM, November 2008.

[18] Ecma International, *Near Field Communication Interface and Protocol (NFCIP-1)*, ECMA-340, Rev. 2, Ecma International, 2006.

[19] ECMA, "Near field communication interface and protocol-2 (NFCIP-1)," Ecma International Standard ECMA-352, Rev. 1, 2003.

[20] NFC Data Exchange Format (NDEF), "NFC forum technical specification," Rev. 1.0, NDEF, 2006.

[21] R. Want, "Near field communication," *IEEE Pervasive Computing*, vol. 10, no. 3, pp. 4–7, 2011.

[22] V. Issarny, N. Georgantas, and S. B. Mokhtar, "Networking semantic services for pervasive computing," *ERCIM News*, no. 67, pp. 36–37, 2006.

[23] M. Hepp, "Goodrelations: an ontology for describing products and services offers on the web," in *Knowledge Engineering: Practice and Patterns: 16th International Conference (EKAW '08), Acitrezza, Italy, September 29–October 2, 2008*, vol. 5268 of *Lecture Notes in Computer Science*, pp. 329–346, Springer, Berlin, Germany, 2008.

[24] M. Gaeta, F. Orciuoli, P. Ritrovato, G. D'Aniello, and A. De Vivo, "A collective knowledge system for business partner co-operation," in *Proceedings of the 5th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS '13)*, pp. 757–762, September 2013.

[25] K. Anyanwu, H. Kim, and P. Ravindra, "Algebraic optimization for processing graph pattern queries in the cloud," *IEEE Internet Computing*, vol. 17, no. 2, pp. 52–61, 2013.

[26] C. Biancalana, F. Gasparetti, A. Micarelli, and G. Sansonetti, "An approach to social recommendation for context-aware mobile services," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 1, article 10, 2013.

[27] D. Zeng, S. Guo, Z. Cheng, and A. T. Pham, "IF-THEN in the internet of things," in *Proceedings of the 3rd International Conference on Awareness Science and Technology (iCAST '11)*, pp. 503–507, September 2011.

[28] G. D'Aniello, F. Orciuoli, M. Parente, and A. Vitiello, "Enhancing an AmI-based framework for U-commerce by applying memetic algorithms to plan shopping," in *Proceedings of the 6th International Conference on Intelligent Networking and Collaborative Systems (INCoS '14)*, pp. 169–175, 2014.

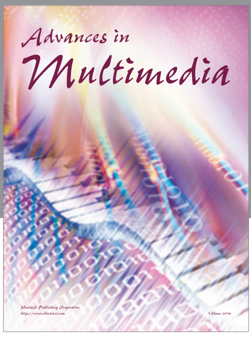
[29] A. Wojciechowski and J. Musial, "Towards optimal multi-item shopping basket management: heuristic approach," in *On the Move to Meaningful Internet Systems: OTM 2010 Workshops: Proceedings of the International Conference on On the Move to Meaningful Internet Systems (OTM '10), Hersonissos, Greece, October 25–29, 2010*, vol. 6428 of *Lecture Notes in Computer Science*, pp. 349–357, Springer, Berlin, Germany, 2010.

[30] J. Blazewicz, M. Kovalyov, J. Musial, A. Urbanski, and A. Wojciechowski, "Internet shopping optimization problem," *International Journal of Applied Mathematics and Computer Science*, vol. 20, no. 2, pp. 385–390, 2010.

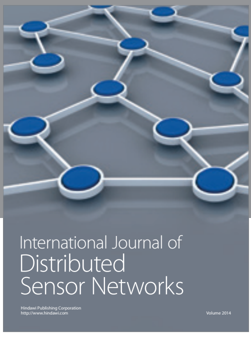
[31] I. De Falco, A. D. Cioppa, and E. Tarantino, "Mutation-based genetic algorithm: performance evaluation," *Applied Soft Computing*, vol. 1, no. 4, pp. 285–299, 2002.

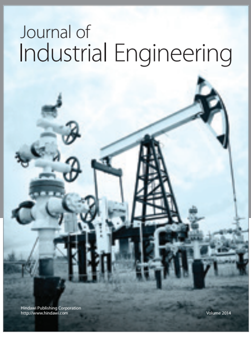
[32] K. Sugihara, "Measures for performance evaluation of genetic algorithms (extended abstract)," in *Proceedings of the 3rd Joint Conference on Information Sciences (JCIS '97)*, pp. 172–175, 1997.

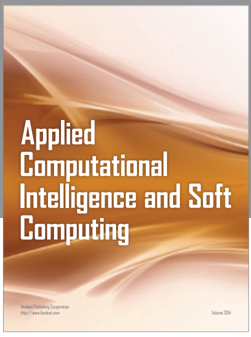
[33] European Payments Council, "Epc white paper on mobile payments," Tech. Rep. 1, European Payments Council, Brussels, Belgium, 2010.

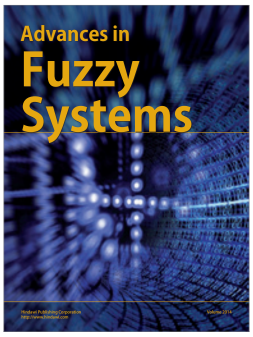

Advances in
Multimedia

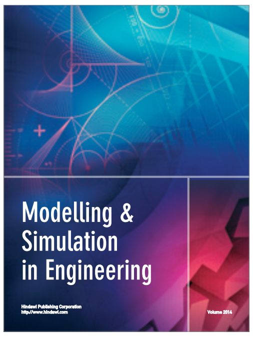
The Scientific World Journal

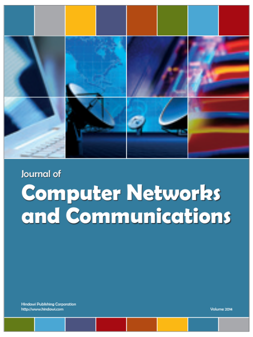

International Journal of
Distributed Sensor Networks

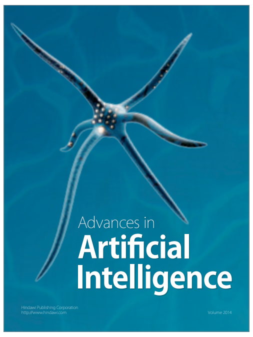

Journal of
Industrial Engineering

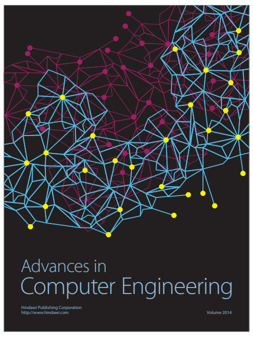

Applied Computational Intelligence and Soft Computing


Advances in
Fuzzy Systems

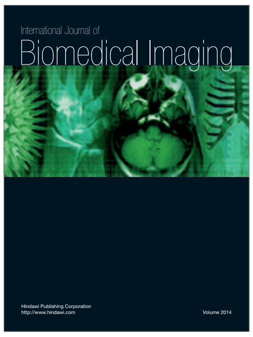

Modelling & Simulation in Engineering

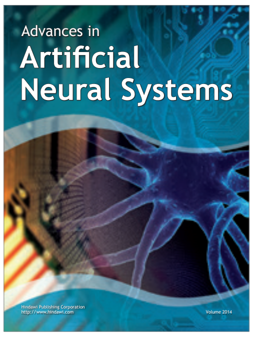

Journal of
Computer Networks and Communications

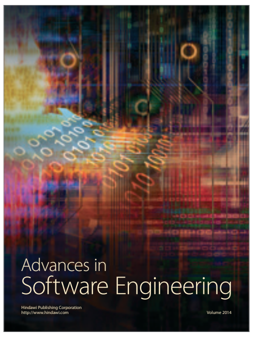

Advances in
Artificial Intelligence


Advances in
Computer Engineering

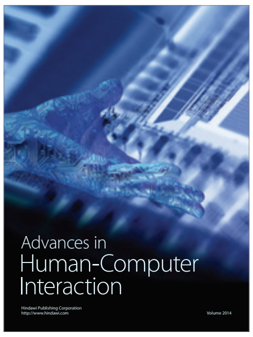
International Journal of
Computer Games Technology

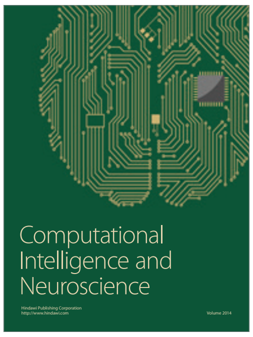

International Journal of
Biomedical Imaging


Advances in
Artificial Neural Systems


Advances in
Software Engineering

Journal of
Robotics


Advances in
Human-Computer Interaction


Computational Intelligence and Neuroscience

International Journal of
Reconfigurable Computing

Journal of
Electrical and Computer Engineering