An Optimization-Based Algorithm for Indoor Localization of JADE Agents

Stefania Monica, Federico Bergenti
Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Università degli Studi di Parma
Parco Area delle Scienze 53/A, 43124 Parma, Italy
Email: {stefania.monica,federico.bergenti}@unipr.it

***Abstract*—This paper describes and evaluates an optimization-based localization algorithm which has been recently implemented to enrich the possibilities of the localization add-on module for JADE. The described algorithm targets indoor scenarios and it enables localization of JADE agents running on smart devices in known environments, provided that a conventional WiFi network is present. The algorithm assumes that WiFi access points in fixed and known positions are available, and it estimates the position of the smart device where the agent is running using estimates of the distance between the smart device and each access point. Distance estimates are used to build an optimization problem, whose solution is an estimate of the position of the smart device. The described algorithm uses particle swarm optimization to solve the built optimization problem, but it is open to the adoption of other optimization techniques. The validity of the proposed approach is supported by experimental results shown in the last part of the paper.**

I. Introduction

Among the longstanding debate on the characteristics that should be ascribed to agents, the fact that agents should be considered *situated entities* immersed in a partially observable environment seems consolidated. Agents acquire knowledge from the environment, and they act on it to achieve their goals. Whether the environment is physical or not is irrelevant for a generic characterization, but agents executing in physical environments are traditionally called *robots*, while agents executing in nonphysical environments are traditionally called *software agents*. Such a distinction is quickly fading because onboard software of robots has now features that were typical of software agents (see, e.g., [1]), and *smart devices* [2] provide software agents with a direct link with the physical world that should be taken into serious account to effectively implement the envisioned ideas of agent-based ubiquitous and pervasive computing. Today, smart devices are ideal candidates to host JADE containers with minimal restrictions [3], and it is perfectly reasonable to assume that JADE should provide platform-level functionality to let agents access the bidirectional link with the physical world that smart devices offer. In order to fully comply with the metaphor of *agents as situated entities*, JADE is demanded to provide agents with platform-level functionality to let agents sense the physical environment where they execute and act on it to bring about their goals.

Smart devices already integrate sophisticated sensors and actuators that can be effectively used to provide agents with a bidirectional link with the physical environment, but they do not normally provide means to let agents know their position in indoor environments. While localization can be considered a solved problem in outdoor environments because consolidated technologies like the *Global Positioning System* (*GPS*) are commonly available in virtually all smart devices, only few smart devices offer localization sensors for indoor environments. Just to mention one of the most promising possibilities in this respect, which is already available in some smart devices, the *Ultra-Wide Band* (*UWB*) technology guarantees accurate and robust indoor localization [4], and it has been already used for accurate indoor localization in industrial environments [5]–[7] and to provide agents with indoor localization capabilities [8]. However, besides the need of a specific type of smart device, the strongest limitation that we currently see in the adoption of UWB or similar technologies is that they require a dedicated infrastructure, which needs to be implemented in indoor environments just to support localization. Notably, WiFi networks can be considered ubiquitous in indoor environments today, and the use of standard WiFi infrastructures to support localization, rather than dedicated infrastructures, can be considered a major advancement and an enabler for applications [9], [10].

In this paper we document recent developments of experiments intended to provide JADE agents running on common Android devices with localization capabilities in known indoor environments using only ordinary WiFi networks. In detail, the targeted scenario assumes that a JADE agent is interested in accurate and timely estimates of its dynamic position within an indoor environment where known and static WiFi *Access Points* (*APs*) are available. The proposed method is based on the possibility of acquiring estimates of the distance between the smart device where the agent is running and each AP using the received power of signals from responding APs during standard network discovery. Such distance estimates are then processed to estimate the position of the smart device in the indoor environment. Once computed, position estimates are immediately made available to the JADE agent. The processing of distance estimates can be performed using one of the localization algorithms available in the literature [11], but in this paper we describe and evaluate the features of an algorithm that was introduced in [12] and further improved in [13], [14] to overcome the inherent numerical instability

of classic localization algorithms. The described algorithm, which was originally designed to use UWB, turns the localization problem into a specific optimization problem, which is solved using *Particle Swarm Optimization* (*PSO*), even if other optimization techniques could be adopted.

This paper is organized as follows. Section II fixes notation and describes the implemented algorithm. Section III shows experimental results obtained in a representative indoor scenarios. Finally, Section IV concludes the paper and outlines possible future developments.

II. Optimization-based Localization

The described algorithm is implemented in a localization add-on module for JADE, which has been recently implemented to provide a framework to host localization algorithms, and to offer agent developers the possibility of choosing among different algorithms to address the specific needs of applications [10]. A discussion of the architecture of this module, which is briefly summarized in [8], is not needed to describe the algorithm and its features.

A. Notation and Reference Scenarios

The smart device where the agent is running, which is denoted as *Target Node* (*TN*) in the rest of the paper, is situated in an indoor environment that contains M WiFi APs. Let

$$\underline{s}_i = (x_i, y_i)^T \qquad i \in \{1, \ldots, M\}. \tag{1}$$

be the coordinates of such APs. We assume that APs are static and we also assume that their coordinates $\underline{s}_i$ are known to the agent. Note that, in order to simplify notation, we focus on the description of the algorithm on bi-dimensional scenarios. The generalization of the described algorithm to three-dimensional scenarios is straightforward. Moreover, under the assumption that the smart device approximatively moves on a plane, [10] shows how to relate a localization problem in a three-dimensional scenario to a proper localization problem in a bi-dimensional scenario.

The ranging capabilities that the smart device is requested to offer in order to support the implemented algorithm concern the possibility of measuring estimates of the distances to the M APs. Such estimates of distances are obtained by processing the average received power of the WiFi signals traveling between the TN and each responding AP during standard network discovery. According to the Friis transmission equation [4], the average received power $\bar{P}(r)$ can be expressed as a function of the distance r between a transmitter and a receiver. By inverting the Friis transmission equation, the value of r as a function of $\bar{P}(r)$ can be expressed as

$$r = r_0 \cdot 10^{-\frac{\bar{P}(r) - P_0}{10\beta}} \tag{2}$$

where P_0 is the known power at reference distance r_0, and β depends on the characteristics of the transmission. Hence, in order to derive an estimate of the distance r between the TN and a generic AP, it is sufficient to measure the average received power of the signal traveling between them and to apply (2). Such a measure of the average received power is always provided by operating systems to applications during standard WiFi network discovery because it is normally used to let the user choose among available networks. Each range estimate can also be associated with the corresponding AP and, eventually, with its coordinates, because communications between the TN and an AP during network discovery include the *Basic Service Set IDentification* (*BSSID*) of the latter, which can be used to identify the responding AP. Hence, assuming that each known BSSID can be associated with the coordinates $\underline{s}_i$ of the corresponding AP, each distance estimate can be related to the coordinates of the corresponding AP. Notably, needed information to support the computation of distance estimates is always available to applications and neither low-level access to hardware, nor privileged operating system services are needed. Indeed, the TN is not even requested to be connected to the WiFi network, because only network discovery is used.

We have already discussed the performance of some localization algorithms in the context of agent-based indoor localization, such as the *circumference intersection* algorithm [10] and the *two-stage maximum-likelihood* algorithm [8]. In this paper, we focus on a localization approach based on optimization, where the localization problem is rewritten in terms of an optimization problem. In order to describe the algorithm, let us first introduce proper notation and makes relevant geometric considerations. Let

$$\underline{u} = (x, y)^T \tag{3}$$

be the true position of the TN, which is supposed to be unknown and which is what should be estimated. The true distance between the TN and the $i-$th AP can be denoted as

$$r_i \triangleq ||\underline{u} - \underline{s}_i|| \qquad i \in \{1, \ldots, M\}. \tag{4}$$

The knowledge of true distances $\{r_i\}_{i=1}^M$, together with the knowledge of coordinates $\{\underline{s}_i\}_{i=1}^M$ of the APs, would easily determine the position of the TN. In this case, the coordinates of the TN could be found by simply intersecting the circumferences centered in $\{\underline{s}_i\}_{i=1}^M$, whose radii are $\{r_i\}_{i=1}^M$. This translates into the following system of M quadratic equations

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = r_1^2 \\ \ldots \\ (x - x_M)^2 + (y - y_M)^2 = r_M^2. \end{cases} \tag{5}$$

Unfortunately, since true distances $\{r_i\}_{i=1}^M$ between APs and the TN are unknown, localization can only be performed using the following system of quadratic equations

$$\begin{cases} (\hat{x} - x_1)^2 + (\hat{y} - y_1)^2 = \hat{r}_1^2 \\ \ldots \\ (\hat{x} - x_M)^2 + (\hat{y} - y_M)^2 = \hat{r}_M^2 \end{cases} \tag{6}$$

which is obtained from (5) by replacing the values of true distances $\{r_i\}_{i=1}^M$, with their estimates, denoted as $\{\hat{r}_i\}_{i=1}^M$. Due to errors on distance estimates, the M circumferences corresponding to the equations in (6) often do not intersect in a single point and, for this reason, a proper localization

algorithm needs to be considered in order to find the proper estimates of the position of the TN, which is denoted as

$$\underline{\hat{u}} = (\hat{x}, \hat{y})^T. \tag{7}$$

In order to derive a proper localization algorithm, let us first observe that (6) can be re-written in matrix notation as

$$\underline{1}\, \underline{\hat{u}}^T \underline{\hat{u}} + \underline{\underline{A}}\, \underline{\hat{u}} = \underline{\hat{k}} \tag{8}$$

where $\underline{1}$ is vector with M elements equal to 1, $\underline{\hat{k}}$ is a vector whose $i-$th element is $\hat{r}_i^2 - (x_i^2 + y_i^2)$, and $\underline{\underline{A}}$ is the following $M \times 2$ matrix

$$\underline{\underline{A}} \triangleq -2 \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_M & y_M \end{pmatrix}. \tag{9}$$

Possible algorithms to solve (8), based on least square techniques, on Taylor series expansion, or on maximum-likelihood methods, can be found, for instance, in [15]. The rest of this section describes an alternative algorithm, which was introduced to overcome numerical instability problems of mentioned approaches.

B. The PSO-Based Localization Algorithm

Various algorithms to solve localization problems expressed as systems of equations like (8) can be found in the significant body of literature on the subject (see, e.g., [15] and referenced literature). All such algorithms typically suffer from numerical instability in correspondence of peculiar configurations of network nodes in space, for example if APs are aligned as discussed in [14]. In order to derive a more robust algorithm, in [14] it is proposed to reformulate (8) as an optimization problem, which is solved using *Particle Swarm Optimization* (*PSO*) [16], as outlined in the rest of this section. Note that the proposed PSO-based approach was originally designed to use UWB signaling, and the description of its evolution to WiFi signaling is the major contribution of this paper.

Observe that (8) can be written as a minimization problem whose solution is an estimate of the position of the TN

$$\underline{\hat{u}} = \arg \min_{\underline{u}} F(\underline{u}) \tag{10}$$

where $F(\underline{u})$ represents the *fitness function* associated with the problem, which is defined as

$$F(\underline{u}) = ||\hat{k} - (\underline{1}\, \underline{\hat{u}}^T \underline{\hat{u}} + \underline{\underline{A}}\, \underline{\hat{u}})||. \tag{11}$$

In order to solve the minimization problem (10), among the wide range of possibilities, we propose to use the PSO algorithm for its proved effectiveness and robustness. According to such an algorithm, the set of potential solutions of a minimization problem can be considered as a swarm of particles whose positions and velocities are iteratively updated according to proper rules. Such rules are inspired by biological phenomena like the movements of birds in swarms. In the context of optimization problems, such rules are meant to move all the particles towards the position corresponding to the optimal solution of the considered minimization problem.

The PSO-based algorithm that we adopted to solve the minimization problem (10) works as follows. First, the positions of the particles are randomly initialized in the search space, which, in our context, corresponds to the physical indoor environment where the APs and the TN are situated. The initial positions are denoted as

$$\underline{x}^{(i)}(0) \qquad i \in \{1, \ldots, S\} \tag{12}$$

where i is the index of the generic particle, and S is the number of particles. Analogously, the velocity of the $i-$th particle is initialized with the value

$$\underline{v}^{(i)}(0) \qquad i \in \{1, \ldots, S\}. \tag{13}$$

After the initialization phase, positions and velocities of all the particles are updated at each iteration $t \in \mathbb{N}$ to simulate interactions among individuals. More precisely, at the $t-$th iteration, the velocity of the $i-$th particle whose position is $\underline{x}^{(i)}(t)$ is updated according to the following rule [17]

$$\begin{aligned} \underline{v}^{(i)}(t+1) \;=\; & \omega(t)\underline{v}^{(i)}(t) \\ & + c_1 R_1(t)(\underline{y}^{(i)}(t) - \underline{x}^{(i)}(t)) \\ & + c_2 R_2(t)(\underline{y}(t) - \underline{x}^{(i)}(t)) \end{aligned} \tag{14}$$

where, as discussed in [18],

1) $\underline{y}^{(i)}(t)$ is the best position reached so far;
2) $\underline{y}(t)$ is the best position globally reached so far;
3) $\omega(t)$ is the so called *inertial factor*;
4) c_1 is a positive parameter called *cognition* parameter;
5) c_2 is a positive parameter called *social* parameters; and
6) $R_1(t)$ and $R_2(t)$ are independent random variables uniformly distributed in $(0, 1)$.

From (14) it can be easily observed that the velocity of a particle at the $(t+1)-$th iteration is obtained as the sum of three addends. The first addend is related to the velocity of the particle at the previous iteration, which is weighed according to the inertial factor $\omega(t)$. The second addend is meant to move each particle towards the best position it reached so far. Note that such a best position is the one which corresponds to the lowest value of the fitness function and, therefore, $\underline{y}^{(i)}(t)$ can be expressed as

$$\underline{y}^{(i)}(t) = \arg \min_{\underline{z} \in X^{(i)}(t)} F(\underline{z}) \tag{15}$$

where

$$X^{(i)}(t) = \{\underline{x}^{(i)}(0), \ldots, \underline{x}^{(i)}(t)\}. \tag{16}$$

Finally, the third addend aims at moving each particle towards the global best position, which is the position that corresponds to the smallest value of the fitness function among all those reached by any particle in the swarm [19]. Hence, $y(t)$ is expressed as

$$\underline{y}(t) = \arg \min_{\underline{z} \in Y(t)} F(\underline{z}) \tag{17}$$

where

$$Y(t) = \{\underline{y}^{(1)}(t), \ldots, \underline{y}^{(S)}(t)\}. \tag{18}$$

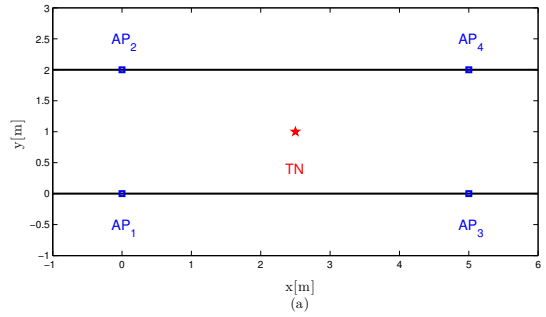

Fig. 1. The positions of the four APs (blue squares) and of the TN (red star) of the first scenario are shown in considered room. The walls of the room are shown in black.

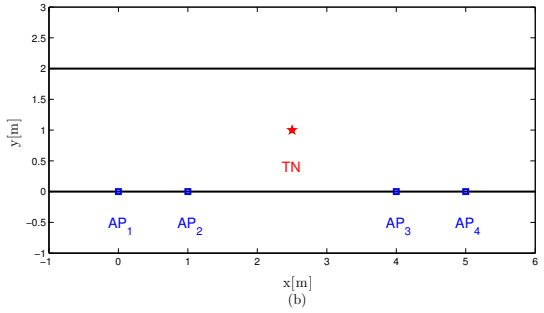

Fig. 2. The positions of the four APs (blue squares) and of the TN (red star) of the second scenario are shown in considered room. The walls of the room are shown in black.

Typically, the inertial factor $\omega(t)$ is chosen as a decreasing function of t to guarantee low dependence of the solution on the initial population, and to reduce the exploration ability of the swarm as the number of iterations increases, which makes the method similar to a local search in last iterations [17].

The velocities computed with (14) are used to update the positions of particles at each iteration according to the following rule

$$\underline{x}^{(i)}(t+1) = \underline{x}^{(i)}(t) + \underline{v}^{(i)}(t) \qquad i \in \{1, \ldots, S\}. \tag{19}$$

From (19) it can be observed that the position of the $i-$th particle at the $(t+1)-$th iteration is simply obtained by adding $\underline{v}^{(i)}(t)$ to its previous position.

The execution of the PSO algorithm terminates when a proper termination condition is met. Normally, the termination condition includes that a sufficient number of iterations was performed, or that the fitness function reached a satisfactory value. Once the execution of the algorithm terminates, the solution of the minimization problem is the position of the particle with the lowest value of the fitness function.

The PSO algorithm outlined above is used to solve the localization problem (10). In order to obtain the experimental results shown in the next section, we set the value of the inertial factor to 0.5. The values of c_1 and c_2 are equal and they are set to 2, so that the average values of $c_1 R_1(t)$ and of $c_2 R_2(t)$ correspond to 1. The size of the population S is set to 40, and the algorithm terminates after 50 iterations. These values proved to be effective for localization purposes as documented in the experimental results presented in [12].

III. Experimental Results

This section shows experimental results obtained using the described optimization-based algorithm, as implemented in the localization add-on module for JADE. Presented results are obtained in a representative scenario which consists in a section of a corridor whose width is 2 m. Note that we consider a bi-dimensional scenario, however, the algorithm can be generalized to three-dimensional scenarios as shown in [20]. Moreover, under the assumption that the smart device moves on a plane, [10] shows how to relate a localization problem in a three-dimensional scenario to a proper localization problem in a bi-dimensional scenario.

We assume that $M = 4$ APs are used to estimate the position of the TN and, in the same environment, we consider two different configurations of such APs. The two configurations are shown in Fig. 1 and in Fig. 2, respectively. In this figures, the positions of the APs are marked with blue squares, the position of the TN is marked with a red star, and the black lines show the walls of the corridor. Note that, even if the algorithm is evaluated for $M = 4$ APs, it can be executed with an arbitrary number of APs. The minimum number of APs to localize a TN in a bi-dimensional environment is three, but the availability of more APs contributes to improve the accuracy of localization.

For each configuration of APs, the optimization-based localization algorithm outlined in Section II is applied 100 times, thus leading to 100 independent position estimates for the TN. In the following, the position estimates of the TN at the $j-$th iteration are denoted as

$$\hat{\underline{u}}^{(j)} = (\hat{x}^{(j)}, \hat{y}^{(j)}) \qquad j \in \{1, \ldots, 100\}. \tag{20}$$

The performance of the proposed localization algorithm is analyzed in terms of a localization error computed as the distance between each position estimate $\hat{\underline{u}}^{(j)}$ and the true position of the TN, denoted as $\underline{u}$, which is known. Such a localization error is

$$d^{(j)} \triangleq ||\hat{\underline{u}}^{(j)} - \underline{u}|| \qquad j \in \{1, \ldots, 100\}. \tag{21}$$

The values of $\{d^{(j)}\}_{j=1}^{100}$ can be used to compute the average localization error, denoted as d_{avg}, which is expressed as

$$d_{\text{avg}} = \frac{1}{100} \sum_{j=1}^{100} d^{(j)}. \tag{22}$$

We start by considering experimental results obtained in the scenario shown in Fig. 1. In this case, the coordinates of the APs, expressed in meters, are

$$\begin{aligned} \underline{s}_1 &= (0,0)^T \qquad \underline{s}_2 = (0,2)^T \\ \underline{s}_3 &= (5,0)^T \qquad \underline{s}_4 = (5,2)^T. \end{aligned} \tag{23}$$

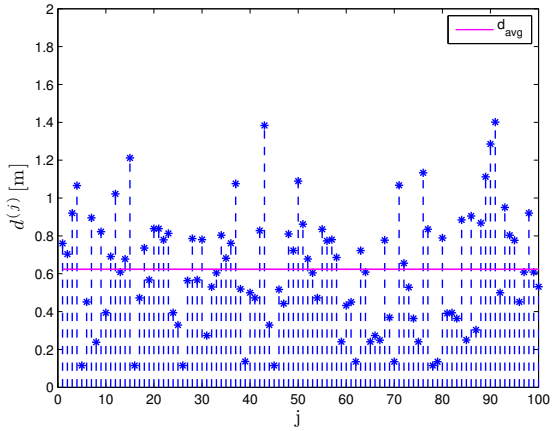

Fig. 3. The values of the localization error $d^{(j)}$ is shown for 100 position estimates for the configuration of APs in Fig. 1. The value of the average localization error is shown with a magenta line across the diagram.

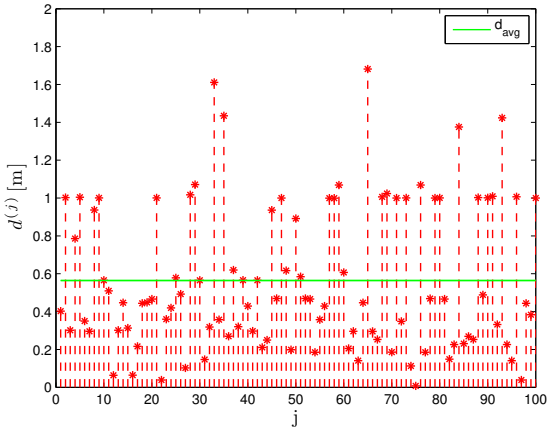

Fig. 4. The values of the localization error $d^{(j)}$ is shown for 100 position estimates for the configuration of APs in Fig. 2. The value of the average distance error is shown with a green line across the diagram.

The position of the TN, instead, has the following coordinates expressed in meters

$$\underline{u} = (2.5, 1)^T. \tag{24}$$

According to the algorithm outlined in Section II, range estimates for each one of the four APs are acquired and used to estimate the position of the TN. This procedure is applied 100 times, thus obtaining 100 position estimates $\{\hat{\underline{u}}^{(j)}\}_{j=1}^{100}$ for the TN. Obtained results are then used to evaluate distances $\{d^{(j)}\}_{j=1}^{100}$ between the j-th position estimate and the true position of the TN, according to (21). Such distances are used as a performance metrics to investigate the validity of the proposed localization algorithm. Fig. 3 shows the values of $\{d^{(j)}\}_{j=1}^{100}$ and also the value of the average localization error d_{avg}, which corresponds to 62 cm. Fig. 3 also shows that the maximum value of $\{d^{(j)}\}_{j=1}^{100}$ is approximately 1.4 m. This means that the distance between the true position of the TN and its estimates is at most 1.4 m, and it is 0.62 m on average, which makes the localization sufficiently accurate for many indoor localization purposes.

Let us now consider the results obtained with the configuration of APs shown in Fig. 2, where all APs are located on the same side of the corridor. In this case, the coordinates of the APs expressed in meters are

$$\begin{aligned} \underline{s}_1 &= (0,0)^T \qquad \underline{s}_2 = (1,0)^T \\ \underline{s}_3 &= (4,0)^T \qquad \underline{s}_4 = (5,0)^T. \end{aligned} \tag{25}$$

The TN is located in the same point as in the first scenario, and its coordinates are shown in (24). Note that the configuration of APs shown in Fig. 2 is less favorable for localization than that shown in Fig. 1, and many classic localization algorithm would fail in estimating the position of the TN. Actually, the large majority of classic localization algorithm rely on specific geometric considerations concerning the positions of the APs and of the TN, and they are based on the solutions of non-linear systems of equations. If all APs lay on the same line, as in the scenario shown in Fig. 2, then the matrices which appear in such classic algorithms become ill-conditioned, thus leading to position estimates which are typically far distant from the true position of the TN [12]. At the opposite, the PSO algorithm, being a refinement-based algorithm, does not suffer from ill-conditioning related to the geometric relations among the positions of the APs.

As done in the first scenario, 100 range estimates from each one of the four APs are acquired, but in this case the configuration of APs is shown in Fig. 2. The values of distances $\{d^{(j)}\}_{j=1}^{100}$ corresponding to the 100 position estimates are then evaluated according to (21) and they are shown in Fig. 4. Fig. 4 also shows the value of the average localization error d_{avg}, which corresponds to 56 cm (green line). Finally, Fig. 4 shows that the distance between the true position of the TN and its estimates is always smaller than 1.7 m, since the maximum value of $\{d^{(j)}\}_{j=1}^{100}$ is 1.68 m.

A comparison between the results shown in Fig. 3 and in Fig. 4 emphasizes that the values of the average localization errors d_{avg} are close in the two scenarios. The value of d_{avg} with the configuration of APs shown in Fig. 2, which is the less favorable, is 6 cm lower than that obtained with the configuration of APs shown in Fig. 1, even if the maximum value of $\{d^{(j)}\}_{j=1}^{100}$ is smaller in the first scenario.

IV. Conclusions

This paper described and evaluated empirically an optimization-based localization algorithm which is used to provide agents with accurate and timely estimates of their position in indoor environments using only ordinary WiFi infrastructures. The discussed algorithm is implemented in the localization add-on module for JADE which provides a framework to host different localization algorithms and to let agent developers choose among them to address the specific needs of applications. In detail, the proposed approach uses the communication between the smart device where the agent

is running and the APs of the network, which are assumed to be static and in known positions, to acquire estimates of the distance between the smart device and each AP. Such estimates are used to build a specific optimization problem, which is then solved using PSO. The solution of such an optimization problem is an estimate of the position of the smart device which hosts the agent. No dedicated infrastructure is needed to enable localization, and smart devices are not even requested to be connected to one of the available WiFi networks because only standard network discovery messages are used.

Possible envisaged applications of the described algorithm include *location-aware social games* [21], which can be adopted, for instance, inside museums and exhibitions to attract the interest of children, or to create personalized itineraries with treasure hunts. Such games can be effectively implemented using the localization add-on module for JADE together with the *Agent-based Multi-User Social Environment* (*AMUSE*) [22], [23], a recent evolution of JADE that offers platform-level functionality to help developers in the implementation of social games. Illustrative results presented in the last part of the paper show that the performance of the proposed algorithms can be considered sufficiently good for these types of applications. In addition, location-aware smart emergency applications [24], which were typically intended for outdoor environments because they need accurate localization, could be retargeted to indoor environments with the help of the described algorithm.

Future work on the research discussed in this paper involves further investigation on the performance of the described algorithm in different indoor environments and with different configurations of APs. Moreover, it is of interest to study the effects of proper pre-processing of the range estimates in the first phase of the algorithm. Actually, as also suggested by experimental results shown in previous section, the performance of the algorithm could benefit from proper averaging of range estimates or of position estimates.

REFERENCES

[1] L. Fichera, F. Messina, G. Pappalardo, and C. Santoro, "A Python framework for programming autonomous robots using a declarative approach," *Science of Computer Programming*, vol. 139, pp. 36–55, 2017.

[2] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Wiley, 2009.

[3] F. Bergenti, G. Caire, and D. Gotta, "Agents on the move: JADE for Android devices," in *Proceedings of the* 15^{th} *Workshop Dagli Oggetti agli Agenti* (WOA 2014), ser. CEUR Workshop Proceedings, vol. 1260. RWTH Aachen, 2014.

[4] S. Gezici and H. V. Poor, "Position estimation via Ultra-Wide-Band signals," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 386–403, February 2009.

[5] S. Monica and G. Ferrari, "Impact of the number of beacons in PSO-based auto-localization in UWB networks," in *Proceedings of the International Conference on the Applications of Evolutionary Computation* (EvoApplications 2013)*, track on Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems* (EvoCOMNET 2013), Vienna, Austria, April 2013, pp. 42–51.

[6] S. Monica and G. Ferrari, "Optimized anchors placement: An analytical approach in UWB-based TDOA localization," in *Proceedings of the* 9^{th} *International Wireless Communications & Mobile Computing Conference* (IWCMC 2013), Cagliari, Italy, July 2013, pp. 982–987.

[7] S. Monica and G. Ferrari, "UWB-based localization in large indoor scenarios: Optimized placement of anchor nodes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 987–999, April 2015.

[8] S. Monica and F. Bergenti, "Location-aware JADE agents in indoor scenarios," in *Proceedings of* 16^{th} *Workshop Dagli Oggetti agli Agenti* (WOA 2015), ser. CEUR Workshop Proceedings, vol. 1382. RWTH Aachen, 2015, pp. 103–108.

[9] S. Monica and F. Bergenti, "A comparison of accurate indoor localization of static targets via WiFi and UWB ranging," in *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*, 2016, pp. 111–123.

[10] S. Monica and F. Bergenti, "Experimental evaluation of agent-based localization of smart appliances," in *Proceedings of the European Conference on Multi-Agent Systems (EUMAS)*, Sevilla, Spain, 2016.

[11] Z. Farid, R. Nordin, and M. Ismail, "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*, vol. 2013, 2013.

[12] S. Monica and G. Ferrari, "Particle swarm optimization for auto-localization of nodes in wireless sensor networks," in *Proceedings of the* 11^{th} *International Conference on Adaptive and Natural Computing Algorithms* (ICANNGA 2013), Lausanne, Switzerland, April 2013, pp. 456–465.

[13] S. Monica and G. Ferrari, "Swarm intelligent approaches to auto-localization of nodes in static UWB networks," *Applied Soft Computing*, vol. 25, pp. 426–434, December 2014.

[14] S. Monica and G. Ferrari, "A swarm-based approach to real-time 3D indoor localization: Experimental performance analysis," *Applied Soft Computing*, vol. 43, pp. 489–497, June 2016.

[15] G. Shen, R. Zetik, and R. S. Thomä, "Performance comparison of TOA and TDOA based location estimation algorithms in LOS environment," in *Proceedings of the* 5^{th} *Workshop on Positioning, Navigation and Communication* (WPNC 2008), Hannover, Germany, March 2008, pp. 71–78.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks* (ICNN), Perth, Australia, November 1995, pp. 1942–1948.

[17] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation* (ICEC), Washington, DC, July 1999, pp. 69–73.

[18] R. Eberhart and J. Kennedy, "A new optimizer using particles swarm theory," in *Proceedings of the* 6^{th} *International Symposium on Micro Machine and Human Science* (MHS), Nagoya, Japan, October 1995, pp. 39–43.

[19] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence Journal*, vol. 1, no. 1, pp. 33–57, June 2007.

[20] S. Monica and G. Ferrari, "A swarm intelligence approach to 3D distance-based indoor UWB localization," in *Proceedings of the International Conference on the Applications of Evolutionary Computation* (EvoApplications 2015)*, track on Nature-inspired Techniques for Communication Networks and other Parallel and Distributed Systems* (EvoCOMNET 2015), Copenaghen, Denmark, April 2015.

[21] F. Bergenti and S. Monica, "Location-aware social gaming with AMUSE," in *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection:* 14^{th} *International Conference, PAAMS 2016*, Y. Demazeau, T. Ito, J. Bajo, and M. J. Escalona, Eds. Springer International Publishing, 2016, pp. 36–47.

[22] F. Bergenti, G. Caire, and D. Gotta, "An overview of the AMUSE social gaming platform," in *Proceedings of the* 14^{th} *Workshop Dagli Oggetti agli Agenti* (WOA 2013), ser. CEUR Workshop Proceedings, vol. 1099. RWTH Aachen, 2013.

[23] F. Bergenti, G. Caire, and D. Gotta, "Agent-based social gaming with AMUSE," in *Procs.* 5^{th} *Int'l Conf. Ambient Systems, Networks and Technologies (ANT 2014) and* 4^{th} *Int'l Conf. Sustainable Energy Information Technology (SEIT 2014)*, ser. Procedia Computer Science, vol. 32, 2014, pp. 914–919.

[24] A. Poggi and F. Bergenti, "Developing smart emergency applications with multi-agent systems," *International Journal of E-Health and Medical Communications*, vol. 1, no. 4, pp. 1–13, 2010.