Received: 7 February 2024 | Revised: 3 July 2024 | Accepted: 15 August 2024

IET Wireless Sensor Systems

DOI: 10.1049/wss2.12092

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

APOTSA: Anchor Placement Optimisation Using Discrete Tabu Search Algorithm for Area-Based Localisation

Sayyidshahab Nabavi[1,2] | Joachim Schauer[3] | Carlo Alberto Boano[2] | Kay Römer[2]

[1]Institute of Electronic Engineering, FH JOANNEUM, Graz, Austria

[2]Institute of Technical Informatics, Graz University of Technology, Graz, Austria

[3]Institute of Software Design and Security, FH JOANNEUM, Graz, Austria

Correspondence

Sayyidshahab Nabavi.
Email: sayyidshahab.nabavi@student.tugraz.at

Funding information

Austrian Science Fund, Grant/Award Number: DFH 5-N

Abstract

Recently, there has been an increasing interest in indoor localisation due to the demand for location-based services. Diverse techniques have been described in the literature to improve indoor localisation services, but their accuracy is significantly affected by the number and location of the anchors, which act as a reference point for localising tags in a given space. The authors focus on indoor area-based localisation. A set of anchors defines certain geographical areas, called residence areas, and the location of a tag is approximated by the residence area in which the tag is placed. Hence the position is not given by exact coordinates. In this approach, placing the anchors such that the resulting residence areas are small on average yields a high-quality localisation accuracy. The authors' main contribution is the introduction of a discretisation method to calculate the residence areas for a given anchor placement more efficiently. This method reduces the runtime compared to the algorithms from the literature dramatically and hence allows us to search the solution space more efficiently. The authors propose APOTSA, a novel approach for discovering a high-quality placement of anchors to improve the accuracy of area-based indoor localisation systems while requiring a shorter execution time than existing approaches. The proposed algorithm is based on Tabu search and optimises the localisation accuracy by minimising the expected residence area. APOTSA's localisation accuracy and time of execution are evaluated by different indoor-localisation scenarios involving up to five anchors. The results indicate that the expected residence area and the time of execution can be reduced by up to 9.5% and 99% compared to the state-of-the-art local search anchors placement (LSAP) algorithm, respectively.

KEYWORDS

indoor navigation, optimisation

1 | INTRODUCTION

Indoor localisation systems are becoming more popular due to their usage in indoor navigation and location-based services. A huge number of systems that rely on different technologies and provide varying levels of performance in terms of precision, latency, cost, dependability, and overall complexity can be found in the literature. The most prominent wireless communication technologies, such as Wi-Fi, Bluetooth, RFID, and Ultra-wideband (UWB), can also be used as effective localisation technologies with an accuracy ranging from a few centimetres up to a few metres [1]. Different techniques have been developed to improve the accuracy, reliability, and robustness of indoor localisation techniques. The Angle of Arrival (AOA) and Time of Flight (TOF) approaches [2] are important examples. AOA employs triangulation based on angles between anchors and a tag [3]. Conversely, TOF calculates the distance between the anchor and the tag by estimating the TOF of the received signal multiplied by the speed of the light and applying trilateration to determine the tag's position [4]. Another frequently-used technique is Received Signal Strength (RSS), which leverages the measurement of

signal power loss to calculate the distance. In this research work, we improve an existing approach in indoor localisation, which is known as *area-based localisation*: it is very cost-effective and offers a good trade-off between localisation accuracy and system complexity based on RSS. It should be noted that although there are technologies such as UWB or Bluetooth low energy, that typically localise a tag with an accuracy of a few centimetres, the advantage of using RSS is that this method works for any wireless technique that measures the signal strength. Essentially this could be useful in cases where neither large bandwidth nor multi-antenna arrays are available, or if only very simple receivers are at hand. Hence, it becomes a perfect option in such situations, especially when estimating the rough position is sufficient [5, 6]. Instead of determining a single-point position, area-based localisation seeks to find small areas where the node is located. Due to the different phenomena that heavily influence radio signals (e.g. multi-path propagation, non-line-of-sight conditions, and environmental changes), localisation errors are inevitable, and depending on the technology used, some of them are more serious than others. The comparison [7] between area-based localisation and multilateration under different settings reveals that area-based localisation is more robust than multilateration in minimising location estimation errors, since the probability of having an erroneous estimate of being near or far is negligible compared to having an erroneous estimate of distance.

Area estimation is critical. In area-based localisation, the location uncertainty is naturally linked to a geometric region known as the residence area. As the number of anchors increases, more complex and irregular shapes are constructed (see Figure 1), making it more challenging to compute residence areas. Based on the geometry of the residence area, existing area-based localisation methods may be classified as circle-based [8], triangle-based [5], or symmetric/half-symmetric lens-based (HSL) area localisation [9]. Lasla et al. [10] have observed that the circle- and triangle-based approaches suffer from two main problems compared to the HSL approach: '(i) In some cases, the decision about the presence of a sensor inside a given area is misleading, which affects the correctness of the obtained residence areas, or (ii) they require high anchor density to achieve a low location estimation error [10]'. For this reason, HSL is considered superior to those approaches.

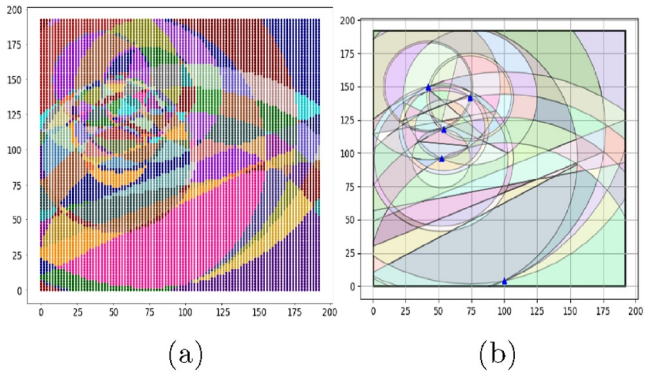

FIGURE 1 Comparison of our discrete approach and Shapely method for the same solution. (a) Our approach, (b) shapely.

Number and placement of anchors influence the localisation accuracy. In area-based localisation methods, the precision of localisation is notably influenced by both the number and placement of anchors. Therefore, it is important to optimally place a sufficient number of anchors to achieve the desired level of accuracy. Cheriet et al. [11] presented a heuristic-based method for determining the optimum anchor locations for the symmetric/half-symmetric lens-based localisation. A central idea of their heuristic is using the Shapely library [12] to get the residence areas induced by anchors at certain spots. It should be mentioned that although on the one hand the LSAP approach yields impressive results, on the other hand, it is a time-consuming approach that requires a significant amount of time to find the optimum solutions, particularly for higher resolutions and higher number of anchors.

Contributions. Based on the mentioned principles, we introduce the APOTSA approach utilising a simple discretisation for computing the residence areas. This discretisation yields a good approximation for the residence areas and significantly improves the computation time. Only in the very last step, we utilise the Shapely library to get the exact size of the residence areas. This methodology aims to find the anchors' best placement, leading to better position accuracy in the target area. Also, for comparison, we use three methods presented in ref. [11], namely genetic algorithm anchors placement (GAAP), which is based on a genetic algorithm, Bruteforce, which is an algorithm that enumerates all possible solutions to find the correct one, and Local Search-Based Anchor Placement (LSAP), which commences by seeking optimum solutions within a low-resolution grid of the discretised search domain, then it continues to explore higher-resolution search domains in the vicinity of the anchor to identify more optimal solutions. Note that the GAAP approach is faster than LSAP; however, it can only be applied to small-scale problems. Therefore, we aim to develop an algorithm that is comparable to GAAP with respect to computation time but gives better results than LSAP.

With APOTSA, we make the following contributions:

i) In the design of APOTSA, we adapt the Tabu search algorithm to a localisation approach called HSL, which tries to localise a node by minimising the expected area of its region, in order to find the optimal combination of anchor positions in a given area.
ii) We develop APOTSA as an area-based approach that discretises the given space, aiming to reduce the computation time while maintaining an acceptable accuracy.
iii) We evaluate the performance of APOTSA, and we execute the same scenarios as in ref. [11] for different numbers of anchors and resolutions. Results indicate that APOTSA is capable of having better performance in terms of execution time and expected area compared to the local search anchors placement (LSAP) algorithm by up to 9.5% and 99%, respectively. The rest of this article is structured as follows: Section 2 briefly overviews the state-of-the-art in

indoor localisation. Section 3 formally introduces the problem and the HSL technique. Section 4 presents our approach. Section 5 compares our approach to existing approaches from the literature.

2 | RELATED WORK

In this section, we conduct a brief overview of previous work relevant to anchors' placement optimisation to try to increase the precision of positioning and navigation.

In ref. [13], Ren et al. proposed RSSI quantisation using a genetic algorithm. Initially, they utilised a genetic algorithm to establish the optimum threshold for quantising RSSI and segmenting the sensing disks of nodes into distinct rings. Each ring is then assigned a specific binary code. This binary code sequence can be mapped to an overlapping area, which represents the potential location of the target, given that the area corresponding to a binary code sequence might not be unique. Also, in scenarios where users require real-time, accurate target locations, the paper also proposes a two-step centroid localisation algorithm. In ref. [14], Liu et al. proposed Landmarc, a LocAtion Detection based on dynamic active Rfid Calibration, which employed fixed location references known as reference tags to calibrate. They utilised RFID tags at a reasonable price. The location of tracking tags is estimated by comparing the received signal from reference tags and tracking tags at reader tags. Anzum et al. [15] propose a zone-based solution to the indoor localisation problem using the Counter Propagation neural Network (CPN) approach. In their research, they have subdivided the interest zone into subzones covered by a specific number of access points (APs). They have collected the signal powers of the APs as a vector to represent each zone using a vector of signal powers received from different APs. The aim is to predict a zone based on the signal sample vector. Lee et al. [16] investigate area-based positioning for indoor environments. They use the combination of KNN and Differential Evolution (DE) algorithm and RSSI channel model to improve the accuracy of indoor localisation. Pan et al. [17] propose a heuristic DE approach for optimising anchor placement to minimise the Cramer-Rao lower limit (CRLB) and estimate range error. They use area division and region combination methods to reduce the search space and parallelise optimisation in different areas by maximising the line-of-sight (LOS) coverage regions. Miao et al. [18] examine anchor placement optimisation challenges; they created a distance-dependent noise model to account for inaccurate distance measurements. They employed a statistical method known as the Fisher information matrix to evaluate the anchor placement. Zhou et al. [19] investigated the placement of anchors for unmanned-aerial-vehicle (UAV)-enabled localisation using the RSS approach. To tackle the problem, the user is considered to be located in a circular area, which is determined through UAV anchors. This research aims to minimise the average (localisation) mean-square error (MSE). Eventually, it is proven that placing the UAV anchors considering symmetrical horizontal angles and the same distance to the area centre can decrease the MSE lower bound. Tian et al. [20] proposed an innovative Wi-Fi and BLE localisation system APs approach. This approach aims to maximise positioning accuracy while ensuring a predefined level of coverage modelled as a certain degree of coverage. The Cramer-Rao lower bound (CRLB) is applied to evaluate the positioning accuracy. The Motley-Keenan model simulates the path loss based on the direct ray between the anchor and tags to testify the algorithm more practically. Wang et al. [21] introduced two anchor placement algorithms consisting of two parts. The first part is a greedy approach that employs sub-modular function properties for anchor placement. The second algorithm is a random sampling technique for placing the anchors that successfully localise all targets. They evaluated real-world and randomly-generated floor plans to assess these algorithms' performance. Based on their results, they achieved a reduction in the percentage of the number of required anchors for randomly generated floor plans compared to the existing methods. Younis et al. [22] proposed a theoretical framework for area-based indoor localisation. Their suggested approach involves considering a circle centred at the anchor's position with a radius equal to the anchor's range to model the coverage of each anchor. Their main goal is to increase the accuracy of indoor localisation by reducing the average of residence areas. Cheriet et al. [11] investigated the anchor placement in an area-based indoor localisation approach, which aims to increase the accuracy. They investigated an area-based localisation approach, namely HSL, which can lead to complex and irregular shaped residence areas. Their simulation framework is implemented in Python and is uses the Shapely library to estimate the area of these irregular shapes. They implemented heuristic algorithms, such as local search anchor placement Algorithm (LSAP) and GAAP, Brute force, and RND, which is a random walk-inspired algorithm, to find an optimal solution. While the results of their algorithms are very promising with respect to the solution quality (especially LSAP), these methods suffer from huge computational times induced by using the Shapely library. In fact, LSAP does not even find a feasible placement for five anchors and a fine-grained discretisation in reasonable time. Our approach now presents a different way of calculating the residence areas and does this by getting even better quality solutions and dramatically reducing the computational time needed.

3 | OVERVIEW OF HSL AREA-BASED LOCALISATION

In this section, we discuss the HSL area-based localisation technique introduced by Lasla et al. [9] as our approach is based on HSL. HSL is a range-free localisation method that estimates proximity information between a node and a set of k anchor nodes using RSS values. This information is employed to estimate the residence area of the node s with respect to the set of nearby anchors (Figure 2 shows a simple example using two anchors). HSL draws, for every pair of anchors a_i and a_j, two circles: one of them called $C(a_i)$ has its centre in a_i, and

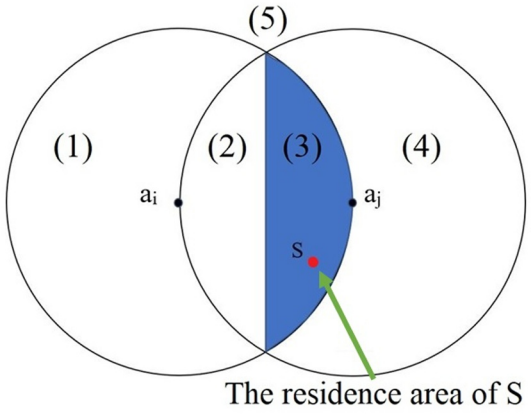

FIGURE 2 HSL approach.

the other one called $C(a_j)$ in a_j, and the radius of both of them corresponds to the distance between a_i and a_j. The geometric shape produced by the intersection of the two circles $C(a_i)$ and $C(a_j)$ is a symmetric lens. The perpendicular bisector of the section connecting the two anchors splits the symmetrical lens into two half-symmetric lenses as shown in Figure 2, which can be represented as $HSL(a_i, a_j)$ and $HSL(a_j, a_i)$. Moreover, we define: $HSL_{out}(a_i, a_j)$ as $C(a_i)$ - $HSL(a_i, a_j)$ and $HSL_{out}(a_j, a_i)$ as $C(a_j)$ - $HSL(a_j, a_i)$. The following cases introduce these four relevant areas for a node s that needs to be localised (see Figure 2):

1. If the distance from s to a_i is less than the distance from a_i to a_j and also the distance from s to a_j is larger than the distance from a_i and a_j, then s is considered to be in $HSL_{out}(a_i, a_j)$.
2. If s is closer to a_i than to a_j and also a_j is closer to s than to a_i, then s is considered to be in $HSL(a_i, a_j)$.
3. If s is closer to a_j than to a_i and also a_i is closer to s than to a_j, then s is considered to be in $HSL(a_j, a_i)$.
4. If the distance from s to a_j is less than distance from a_i to a_j and also the distance from s to a_i is larger than the distance from a_i and a_j, then s is considered to be in $HSL_{out}(a_j, a_i)$.
5. If the distance from s to both a_i and a_j is larger than the radius of both circles, then s is considered to be outside of both circles.

In the case of three or more anchors, we use all possible intersections resulting from them for the localisation of a tag node (see Figure 1).

4 | APOTSA

In area-based localisation, the precision of a node's estimated position relies on the size of its residence area, which is influenced by the deployment of the anchors. This study focuses on the optimal placement of a specific number of anchors to optimise localisation accuracy. For this, the algorithm focuses on minimising the size of the residence areas. Assume that in the service area S (which is equal to the area which the anchors should cover), K anchors $\{a_1, a_2, \ldots, a_k\}$ are deployed; due to this deployment and based on the HSL method, the area S is divided into m non-overlapping areas $\{s_1, s_2, \ldots, s_m\}$ as depicted in Figure 1. The sum of these residence areas is equal to the size of the area S, and the nodes that need to be localised can be at arbitrary positions in S. Hence, intuitively, an algorithm that gives small areas can be considered as good. As already observed by Cheriet et al. [11], this can be formalised in the following way: for a given node v that is uniformly placed at a random place in S, let X be a random variable that yields the residence area where v is located in. Hence $X \in \{s_1, s_2, \ldots, s_m\}$ and the probability that node v is in the area s_i can be calculated as (cf. [11]):

$$\Pr[X = s_i] = \frac{s_i}{S} \tag{1}$$

Hence, minimising the size of the expected residence area $E[X]$ gives the following:

$$minimize\ E[X] = \sum_{i=1}^{m} s_i \Pr[X = s_i] = \frac{1}{S} \sum_{i=1}^{m} s_i^2 \tag{2}$$

Subject to:

$$\sum_{i=1}^{m} s_i = S \tag{3}$$

For computational reasons, Cheriet et al. [11] did not allow the anchors to be placed anywhere in S but only at some predefined positions at an $n \times n$ grid covered by S. As illustrated in Figure 1, they used the Shapely library to compute the residence areas induced by the placement of the anchors and the corresponding circles and half lenses of the HSL approach. In contrast to the usage of the Shapely library and depicted in Figure 1, in our approach, we employed a discretisation approach for area calculation, which is described in the next subsection.

4.1 | Discretisation

Our approach uses two grids, called *a_grid*, for the anchor placement and *g_grid* for the expected residence area calculation as depicted in Figure 3. Both grids of nodes are covered by S and the number of points in the first and second grid is $n_a \times n_a$ and $n_g \times n_g$, respectively. Under the assumption that all of the points of *g_grid* cover an equal-sized area of S and none of them intersects with any of the other areas, we define the area that a point of *g_grid* covers as Point_Area:

$$Point\_Area = \frac{S}{n_g \times n_g} \tag{4}$$

Then, instead of exactly calculating the areas s_i by using the Shapely library, we simply approximate these areas by the number of points of *g_grid* that are in s_i times the Point_Area. This discretisation induces small errors but results in a very performant computation compared to using the Shapely library. For checking in which residence area s_i a point lies, we simply have to check its position with respect to all the circles and half lenses of the HSL approach. For getting a good approximation, *g_grid* is much more fine-grained compared to the *a_grid*. Note that the points of *g_grid* are positioned in such a way that none of them has the same position as any of the points of *a_grid*. Algorithmically, this is done by shifting the coordinates of *g_grid* vertically and horizontally by a small value which is indicated by ϵ. This shifting reduces the chance that a point of *g_grid* used for the expected residence area calculation is on the boundary of a residence area—in which case the error resulting from the discretisation is quite high as illustrated in Figure 3.

Formally let g_i denote the number of points of the fine-grained grid that are in area s_i and G the total number of points of *g_grid*. Then in our approach, Equations (1) and (2) can be approximated as follows:

$$\Pr[X = s_i] \approx \frac{g_i}{G} \tag{5}$$

$$\min E[X] \approx \sum_{i=1}^{m} g_i Pr[X = s_i] = \frac{1}{G} \sum_{i=1}^{m} g_i^2 \tag{6}$$

4.2 | Tabu search approach

In our approach, we use a Tabu search algorithm [23] to find the optimal placement for the anchors. This method is a metaheuristic approach to address combinatorial optimisation problems. This algorithm is able to escape local optima by avoiding returning to already examined solutions, and it can be applied to both discrete and continuous problems. Frequently in Tabu search, a set of neighbouring solutions of a feasible solution is explored in order to get a better objective value. Note that Tabu search is easy to implement since, on a high level, it guides a local search algorithm in such a way that it is not trapped in local optima. As LSAP is based on local search and already gave promising results, we decided to use Tabu search instead of other prominent meta-heuristic algorithms. We define the neighbourhood that we explore as *N_SOL*, and this neighbourhood is composed of the list of anchors and their respective movements. There are many conceivable ways to define a neighbourhood by altering the anchors' placement, and a larger neighbourhood usually leads to a longer computation time. In order to bound the computational resources needed, we control the size of the neighbourhood by selecting a subset of anchors of size γ whose position is possibly changed in each step (called *A_set*), and for each anchor only four possible moves are allowed (represented by *D_set*). Figure 4 shows this list and Figure 5 the possible positional changes of an anchor. We allow an anchor to stay unchanged, move up, down, left, or right (encoded by the numbers 0, 1, 2, 3, 4 respectively). A parameter *jump* additionally indicates how many grid points in the respective direction an anchor should move. Putting this together, we get that *N_SOL* is of size $2 \cdot \gamma$: position $k \leq \gamma$ represents some anchor j and position $k + \gamma$ the corresponding move of anchor j. Formally, we can define *N_SOL* as follows:

$$\begin{aligned} &N\_SOL = (A\_set, D\_set) \\ &A\_set = A_1, A_2, \ldots, A_\gamma \\ &D\_set = (\zeta_1, \zeta_2, \ldots, \zeta_\gamma) \\ &\zeta\_i \in \{0, 1, 2, 3, 4\} \\ &j \in \{0, \ldots, m\} \end{aligned}$$

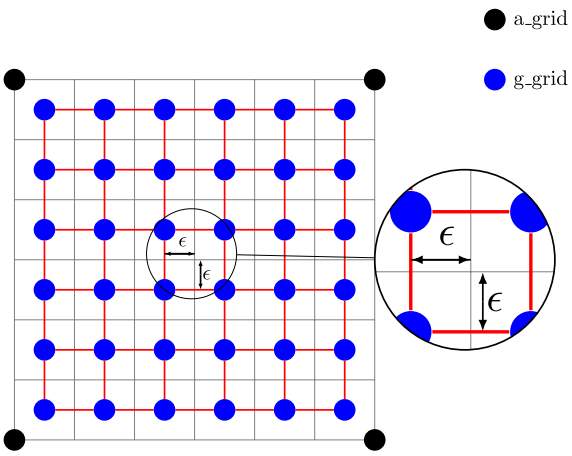

FIGURE 3 Griding method.

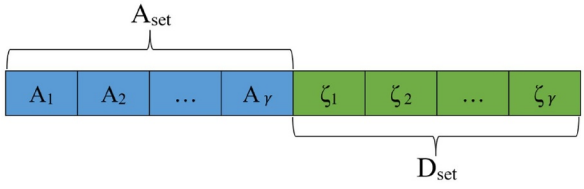

FIGURE 4 Solution representation by N_SOL.

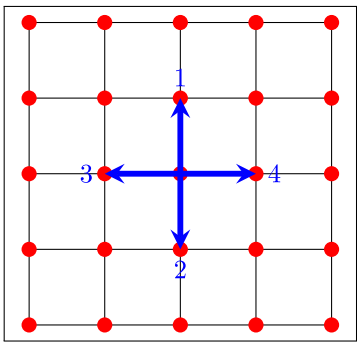

FIGURE 5 Neighbourhood for jump = 1.

Note that $A_1, A_2, \ldots, A_\gamma$ corresponds to a set of distinct anchors of size γ. For example, a γ of two in the case of three anchors indicates that a neighbourhood involves all subsets of size two of the set of anchors, that is, $\{\{0,1\},\{0,2\},\{1,2\}\}$. For a given parameter setting, all feasible moves N_SOL are represented by a list called *Act_list*. Algorithm 1 describes the creation of this list. In this algorithm, the combination is a function that takes a list and an input γ as an input and returns a list containing all possible combinations of length γ of the input list.

Algorithm 1 Create_Act_list

```
1: function Create_Act_list (AN, γ)
2:     A_set ← combination([0, AN − 1], γ)
3:     D_set ← combination([0, 4], γ)
4:     Act_list ← [ ]
5:     for i ← 1 to length(A_set) do
6:         for j ← 1 to length(D_set) do
7:             Act_list.append((A_set[i],
               D_set[j]))
8:         end for
9:     end for
10:    return Act_list
11: end function
```

The pseudocode of our approach is depicted in Algorithm 4. For clarity, also Figure 6 presents the flowchart of APOSTA The inputs of this algorithm include the following parameters: *Tabu_Length* defines the length of the Tabu list; *AN* defines the number of anchors within the given scenario; *Max_Iteration* corresponds to the upper limit on iterations for the Tabu search algorithm; *MAX_X* and *MAX_Y* denote the dimensions of the search space; the parameter γ has been previously described; *P_sol* represents the best solution found on the smaller grid size and is used to initialise a first solution on the larger grid; *resolution* is the grid resolution which is used for the algorithm. Line two of the pseudocode outlines the generation of two-dimensional grids, the *a_grids* and *g_grids*, which consist of Cartesian coordinates for placing anchors and calculating areas, respectively. Also, the number of grids on each side of *a_grids* and *g_grids* is equal to n_a and n_g, respectively, as discussed in Section 4.1. In our implementation, we set n_a and n_g as follows:

$$\begin{aligned} n_a &= \text{resolution} \\ n_g &= \min(\max(81, \text{resolution} \times 3 + 1), 200) \end{aligned} \tag{7}$$

To ensure a fair comparison with other algorithms, we initialised n_a equal to resolution. Therefore, we used identical grids for anchor placement and for n_g. Since the resolution starts from very low values, we set it to 200 for cases where the resolution is low, aiming to maintain an acceptable level of accuracy. Subsequently, in line 3, the *Act_list* is created. The Tabu search algorithm employs the *Create_Act_list* to create new solutions (see Algorithm 1). Gen_Sol is the algorithm

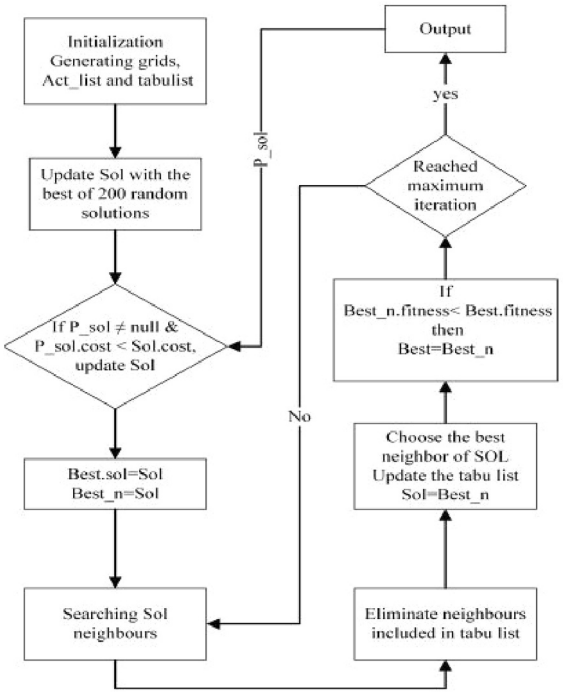

FIGURE 6 APOTSA flowchart.

which applies a specific action to generate a new solution by moving the place of anchors towards, up, down, left, or right by *jump* steps as depicted in Algorithm 2. The main part of the Tabu search algorithm is depicted from line 13 on. The algorithm stores the best solution found in the current iteration as *Best_N*, the best solution found so far in *Best.sol*, and updates the Tabu list as illustrated in Algorithm 3. This involves setting the current solution in a state of tabu for the duration of *Tabu_Length* iterations. The algorithm is precluded from revisiting this solution within the subsequent *Tabu_Length* steps. Subsequently, the algorithm tries to search for the best neighbour of the contemporary solution. In each iteration, the most suitable solution is substituted with the best ever-found solution of the algorithm in case of having a lower cost.

Algorithm 2 Gen_SOL

```
1: Input: sol, Action, jump
2: n_sol ← [ ]
3: for j ← γ to 2 × γ do
4:     if Action==1 then
5:         n_sol[j − γ] ← Up(sol[j − γ], jump)
6:     else if Action==2 then
7:         n_sol[j − γ] ← Down(sol[j − γ], jump)
8:     else if Action==3 then
9:         n_sol[j − γ] ← Left(sol[j − γ], jump)
10:     else if Action==4 then
11:         n_sol[j − γ] ← Right(sol[j − γ],
            jump)
12:    end if
13: end for
14: return n_sol
```

Algorithm 3 Updating the Tabu List

```
1:function Update_Tabu_List (SOL,Tabu_list)
2:  Tabu_list.append(SOL)
3:  if length(Tabu_list) > Tabu_Length then
4:     /*Delete oldest solution in Tabu
      list.*/
5:     Tabu_list.pop(0)
6:  end if
7: end function
```

Algorithm 4 APOTSA

```
   Input: Tabu_Length, AN, Max_Iteration
   MAX_X, MAX_Y, γ, P_sol, resolution
   Output: Optimal Placement
1: function TabuSearch (Input)
2:     Generating a_grids & g_grids
3:     Act_list ← Create_Act_list(AN, γ)
4:     nAction ← len(Act_list)
5:     Sol ← Best(200 random solutions)
6:     if P_sol exists and has less cost Sol
       then
7:         Sol ← P_sol
8:     end if
9:     Best_N & Best.sol ← [ ]
10:    Best_N.fitness, Best.fitness ← ∞
11:    jump ← random(2, ⌈resolution/10 + 2⌉)
12:    Best_N ← Sol
13:    for j ← 0 to Length(Act_list) do
14:        N.sol←Gen_Sol(Sol, Act_list[j],
             jump)
15:        N.fitness ← Cost(N.sol)
16:        if N.fitness < Best_N.fitness then
17:            Best_N ← N
18:        end if
19:        Update_Tabu_List(Sol, Tabu_list)
20:        Sol ← Best_N
21:    end for
22:   if Best_N.fitness < Best.fitness then
23:        Best ← Best_N
24:   end if
25:   return Best.sol
26: end function
```

5 | EVALUATION

In this section, we assess the performance of APOTSA in comparison to other methods. This evaluation is structured into several subsections, each presenting insights into specific aspects of APOTSA and the compared algorithms.

5.1 | Performance metrics

In our analysis, we compare the performance of APOTSA with other approaches using two main metrics: Expected residence area (in terms of EX) and execution time. The EX parameter for APOTSA is computed by the discrete version using Equations (5) and (6). For the other approaches, EX is computed through the Shapely library, using Equations (1)–(3). For a fair comparison, generated solutions by APOTSA are evaluated using the Shapely library, similar to other approaches. It should be noted that we corrected a minor bug for our computational experiments in the source code of the LSAP and the other approaches used for comparisons [24]. The error was due to calculating the expected area using Shapely in *our_library.py* which is provided by Lasla et al. [24]. The reason was that while the algorithm tries to calculate the intersection area caused by the anchors, generated intersections have intersections in each circles, which is ignored in the calculation. It should be mentioned that, in terms of the time of execution, the corrected version takes much more time. This increment is attributed to the necessity of verifying the presence of intersections in each iteration. Furthermore, these intersections may result in the generation of inaccurate expected areas. Time of execution refers to the duration an algorithm requires to discover a solution for a given scenario in seconds. This metric is computed using the time function from the *process_time*() in Python [25]. Considering the extended time spans for some of the scenarios in our simulation, the logarithmic scale is used to represent the graphs. Note that lower values for both the expected area and time of execution are desired.

5.2 | Experimental setup

APOTSA implementation. We implemented our Tabu search approach in Python to evaluate its effectiveness and compared it with the existing approaches described in ref. [11]. Note that it would be possible to implement our algorithm in a faster programming language (such as C, C++, or Java) since we do not use any special non-standard libraries (except for one call to the Shapely library at the very end of our algorithm). However, the implementation in Python allows for a fair comparison to the existing approaches, which were also implemented in Python.

In APOTSA, the input parameters affect the quality of the solution and also influence the time of execution. The following Table 1 lists all the parameters that we used:

Simulation settings. We used the same instances that were already presented in ref. [11]: the area S in which nodes need to be localised corresponds to a square, and this square covers multiple grids (a_grids) used for potential anchor placement, namely 3×3, 5×5, 9×9, 17×17, 33×33, and 97×97. Moreover, the number of anchors that are placed ranges from three to five. Figure 7 shows the increase in the number of the *a_grids* (blue grids) from 5×5 to 9×9: this is done by adding one grid

TABLE 1 Simulation parameters.

Parameter	Value
Tabu_Length	10
Anchors_Number	3, 4, 5
Resolution	3, 5, 9, 17, 33, 97
Max_iteration	20, 30
MAX_X	192
MAX_Y	192
γ	2, 3, AN (number of anchors)

point (red dot) between each grid point of the previous step (blue dot). Intuitively, a good solution for the 5×5 case can be used as a starting point for a refined solution for the more fine-grained 9×9 grid. This observation was already algorithmically exploited by Cheriet et al. [11] successfully, and therefore, we also implemented this idea in our Tabu search by searching for an anchor placement on a 3×3 grid and passing the corresponding solution that we found to the next more refined grid (in Algorithm 4 this is done with *P_Sol*).

5.3 | Baselines for comparison

We compared our approach with the LSAP (local search anchor placement) approach, a Brute Force approach, and a genetic algorithm (GAAP) introduced by Cheriet et al. [11], and the corresponding implementations can be found in the GitHub repository [24]. Our evaluations focus on reducing both the execution times of the algorithms and the expected residence areas they found. We executed all the algorithms 10 times for any of the different instances on the same servers (Intel(R) Xeon(R) Gold 6230 CPU @ 2.10 GHz, 64 cores, 600 GB RAM) to have a fair comparison, and the results presented in this paper are the averages of the corresponding running times and expected residence areas.

Local Search Anchor Placement (LSAP). Cheriet et al. [11] developed a local search algorithm. The algorithm first searches for high-quality solutions on an a_grids of small dimension and then (step by step) increases the resolution of the grid in order to get better solutions. Each step passes the previous solution to the problem with a higher resolution. Refining the resolution increases the number of locations where the anchors can be placed. The higher resolutions are obtained by adding new points between the points of the lower resolution (as depicted in Figure 7).

Brute Force algorithm (BF). The BF algorithm tests all feasible anchor placement solutions for a specific scenario. As discussed in ref. [11], for m anchors and a grid of size $n \times n$, the number of feasible anchor placements is computed as follows:

$$\binom{m}{n^2} = \frac{n^2}{m!(n^2 - m)!} \tag{8}$$

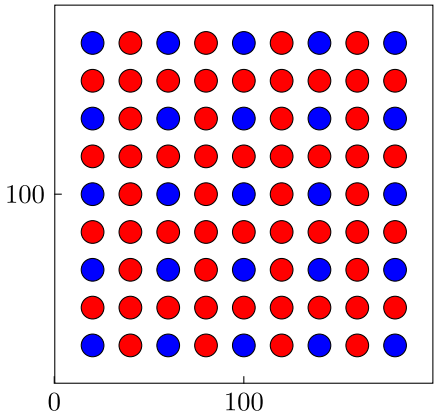

FIGURE 7 Expanding grid from 5 to 9.

Obviously, for higher resolutions, the number of feasible solutions grows dramatically and hence makes this approach impracticable. Therefore, the BF can only be applied to low-resolution scenarios. For this reason, the results shown in the subsection 5.4 are not available for this algorithm.

Genetic Algorithm (GAAP). In the last years, metaheuristic algorithms have obtained great attention in solving numerous optimisation problems [26]. The genetic algorithm is one of the optimisation algorithms which is inspired by natural evolution and has two major stages: selection and reproduction. The algorithm attempts to pick the best solution from the population during the selection phase. In the reproduction phase, the algorithm attempts to develop a new solution utilising the current generation using tactics like mutation and crossover. Lasla et al. [11] employed the genetic algorithm anchor placement approach as a comparative technique. They adopt a genetic algorithm for this problem, and it is initialised with randomly-generated solutions in each stage; the best solutions are passed to the next level.

5.4 | Results

In this subsection, we discuss the most important results of our computational experiments. Generally, it can be seen that by increasing the number of anchors or grid size, the EX value decreases. However, this reduction is associated with a noticeable increase in execution time.

Figure 8 displays the expected residence areas achieved for three anchors. The figure shows that increasing the resolution leads to a decrease in the expected residence area for all algorithms. Among the approaches, the genetic algorithm found the worst result, however, in a short amount of time compared to the other approaches (as shown in Figure 9 and on larger grid sizes). The Brute Force algorithm obviously needed the longest execution time. Our approach consistently outperformed the LSAP in terms of execution time. Since the number of anchors here equals three to avoid repetitive results, the scenario where γ equals the number of anchors is omitted in Figures 8 and 9. Figures 10 and 11 illustrate the expected

residence areas and execution time for four anchors, respectively. It can be seen that our methods outperform the comparative approaches, with a considerable decrease in execution time as well. Additionally, it can be seen that in the case of 3 and 4 anchors, GAAP is faster than other algorithms, though it yields poorer results regarding the size of the expected area compared to others. Conversely, In the case of 5 anchors for higher resolutions, it can be seen that our algorithm performs better in terms of the execution time and the size of the expected area. Figures 12 and 13 represent the five anchors case. Only our approach and the genetic algorithm can be executed for higher-resolution grids within a runtime limit of 100,000 s. Our approaches can reduce the expected residence area by more than 30% compared to the genetic algorithm. In the 97×97 resolution, our algorithm with $\gamma = 2$ consumes even less time than the genetic algorithm. In any case, our approach achieved a reduction from 88 to 98 units of EX. Note that in all cases, the tradeoff between the solution quality and the consumed time is evident in our approach for the different values of γ.

Figure 14 shows the sum of the difference between areas calculated by our discretisation and the Shapely library. It can be seen that the difference decreases when the resolution of the grids is increased. For fine-grained grids, the areas are almost equal. The violin plots presented in Figure 15 showcase a comparative analysis of the distribution of the sizes of the residence areas obtained by the different algorithms for 10 runs. Since the algorithms perform approximately the same in lower resolutions, the plots are shown for the 17×17, 33×33, and 97×97 cases. For three anchors, the algorithms show a similar performance, except for the GAAP approach. In the case of 4 anchors, the Tabu search with $\gamma = AN$ produces fewer areas of size greater than 500 compared to the other approaches, which indicated that not only the expected size of the areas is superior compared to the other approaches. In the case of 5 anchors, due to

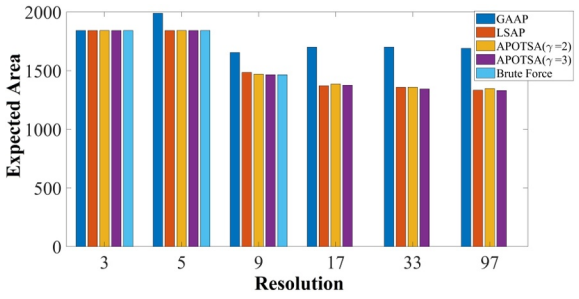

FIGURE 8 Expected residence area with three anchors.

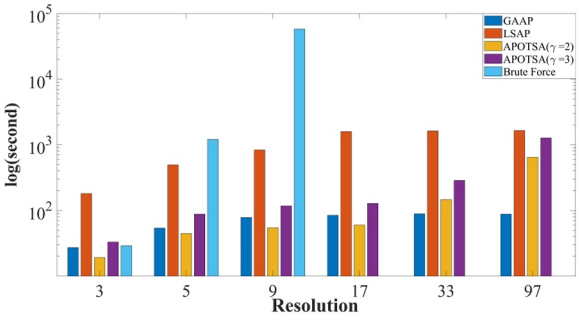

FIGURE 9 Time of execution with three anchors.

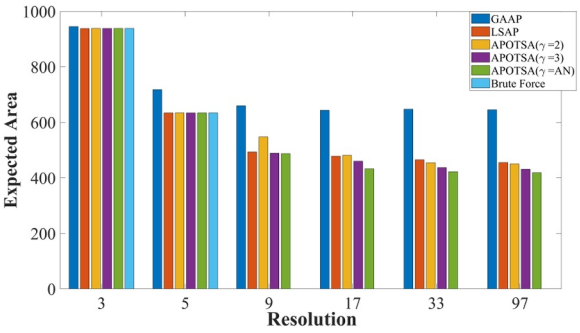

FIGURE 10 Expected residence area with four anchors.

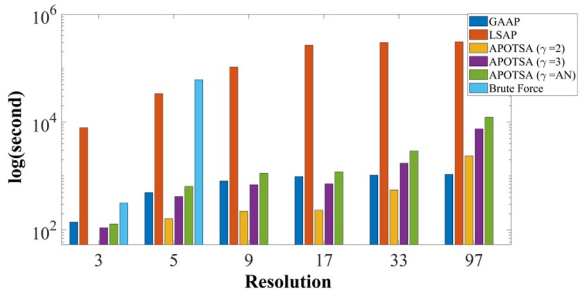

FIGURE 11 Time of execution with four anchors.

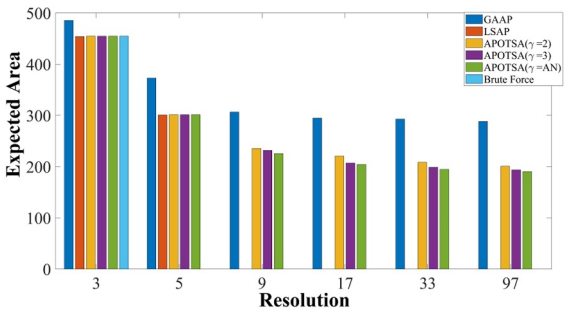

FIGURE 12 Expected residence area with five anchors.

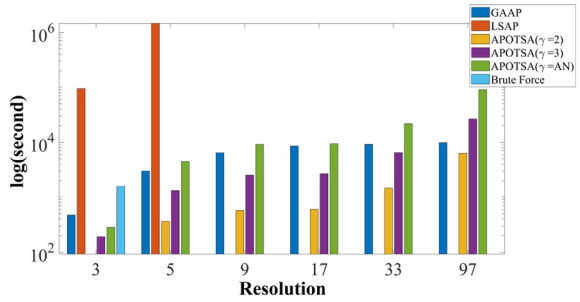

FIGURE 13 Time of execution with five anchors.

extensive computation time, results for Bruteforce and LSAP are unavailable; among the illustrated approaches, $\gamma = AN$ yields the lowest maximum area compared to the other approaches. It is worth noting that the GAAP approach often gives a very good median area; however, due to the variance in the observed areas, its EX is worse compared to the other approaches. Also, to have a more precise comparison, the results for expected residence area and time of execution for each algorithm are depicted in Tables 2 and 3 respectively.

5.5 | Scalibility

In our study, to prove the scalability of our method, we assessed it across numerous room dimensions, including 150 × 235, 170 × 216, and 300 × 123. These dimensions are selected to be comparable and have the same area as the area investigated in ref. [11] and to showcase the adaptability of our approach to different room layouts. We employed the APOTSA algorithm for these layouts with γ values of 2 and 3, as detailed in Tables 4–6. We also tested configurations with 3,

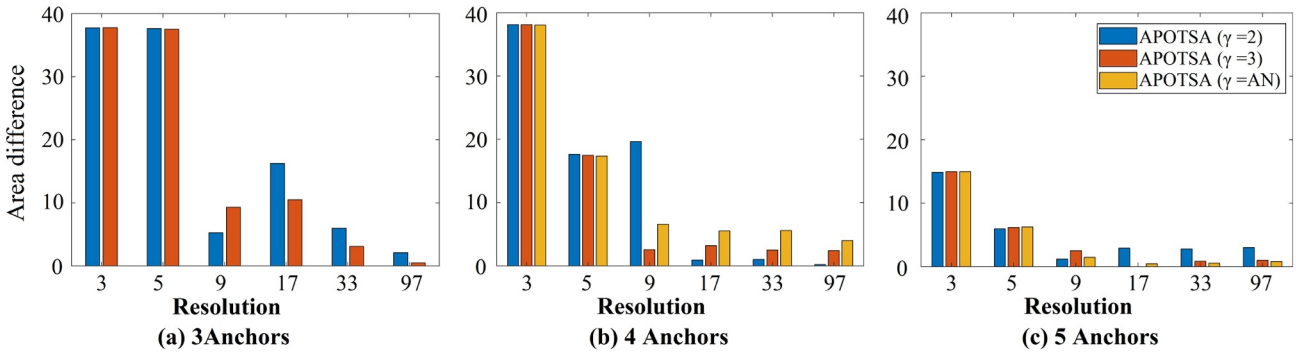

FIGURE 14 Difference in area calculation.

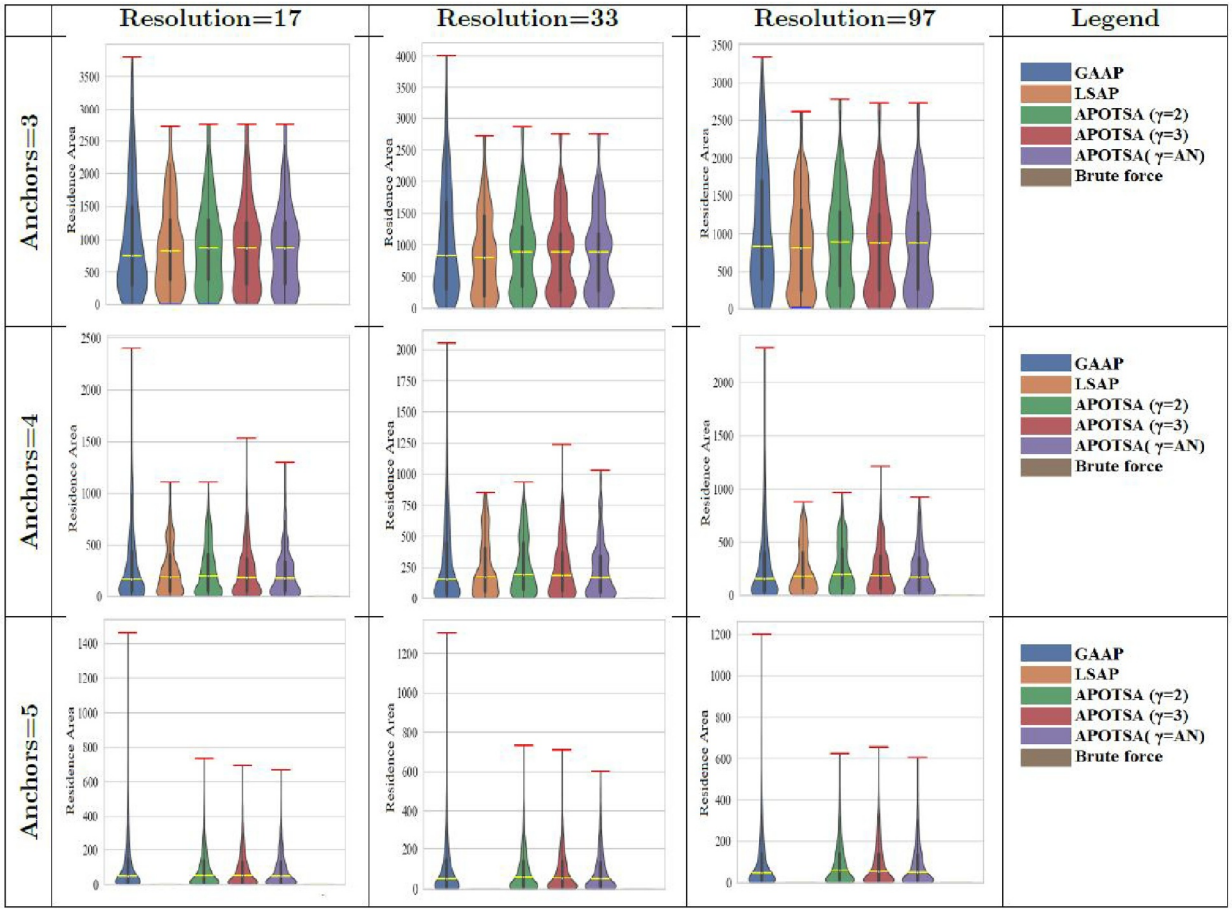

FIGURE 15 Distribution of residence areas of 10 times execution.

TABLE 2 Expected residence area comparison.

Anchor	Resolution	Genetic	Brute force	LSAP	APOTSA $\gamma = 2$	APOTSA $\gamma = 3$	APOTSA $\gamma = AN$
3	3	1842	1842	1842	1842	1842	–
3	5	1989	1842	1842	1842	1842	–
3	9	1654	1464	1485	1470	1464	–
3	17	1700	–	1371	1385	1375	–
3	33	1701	–	1358	1358	1344	–
3	97	1690	–	1334	1348	1331	–
4	3	944	938	938	938	938	938
4	5	717	634	634	634	634	634
4	9	659	–	493	547	489	487
4	17	643	–	478	481	460	432
4	33	647	–	465	454	437	421
4	97	645	–	455	450	431	418
5	3	485	454	454	454	454	454
5	5	372	–	301	301	301	301
5	9	306	–	–	235	231	225
5	17	294	–	–	220	207	204
5	33	292	–	–	208	198	194
5	97	288	–	–	200	193	190

4, and 5 anchors, similar to our comparative analyses. The results indicate that, despite minor variations due to different dimensions, our method consistently achieves comparable outcomes across different numbers of anchors and resolutions. Since the room layouts are not square in these cases, there might be slight adjustments or shifts in the anchor positions while increasing the resolution. For example, in 150×246, increasing the resolution from 17 to 33 for 3 anchors alters the coordinates of the anchor positions. As a result, the solution found for a lower grid might appear at different coordinates in a higher-resolution grid. It should be noted that the resolution size is optional; therefore, it is also possible to use different grid sizes.

5.6 | Discussion

In this subsection, we discuss the benefits of APOTSA.

5.6.1 | Flexibility

As discussed in the previous sections, the γ parameter adjusts the equilibrium between the quality of the expected area and the time of execution. This parameter lies within the interval $(1, AN)$; as this parameter is set higher, the algorithm tries to search for more solutions to find the optimal one.

5.6.2 | Speed

In our approach, to avoid complicated calculations, we use a discretisation method to calculate the expected residence area. This method introduces approximates the true residence areas in a reasonable way (as indicated in Figure 14). For higher resolutions, these approximations are very good, and since we use the Shapely library in our very last call, it is guaranteed that the areas found by the algorithm are correct.

5.6.3 | Flexibility in room layout

The HSL approach requires that the anchor is in the line of sight of any tag. Therefore our approach can be applied to any convex layout. Note that we do not consider the presence of obstacles and multipath propagation, which are phenomenons that are critical to any indoor localisation method.

5.6.4 | Flexibility in griding

Our approach can use grids of any possible size, and more fine-grained grids will lead to better results; however, at the price of a higher execution time. Hence by choosing a certain grid size we can control the trade-off between solution quality and runtime in a very convenient way.

TABLE 3 Time of execution comparison.

Anchor	Resolution	Genetic	Brute force	LSAP	APOTSA $\gamma = 2$	APOTSA $\gamma = 3$	APOTSA $\gamma = AN$
3	3	27	29	184	19	33	–
3	5	54	1207	495	45	89	–
3	9	78	57,865	838	55	118	–
3	17	84	–	1585	60	128	–
3	33	89	–	1613	146	287	–
3	97	88	–	1633	646	1263	–
4	3	141	316	7829	54	110	129
4	5	496	61,164	33,778	163	414	643
4	9	814	–	105,280	224	696	1130
4	17	977	–	268,145	233	714	1199
4	33	1040	–	302,497	554	1721	2914
4	97	1078	–	309,012	2364	7448	12,410
5	3	474	–	94,654	92	192	284
5	5	3000	1568	1,456,860	365	1327	4468
5	9	6430	–	–	575	2512	9122
5	17	8566	–	–	604	2669	9383
5	33	9197	–	–	1458	6438	21,895
5	97	9853	–	–	6311	26,576	90,507

TABLE 4 APOTSA $\gamma = 2$.

		150 × 246		170 × 216		300 × 123	
Anchor	Resolution	EX	Time	EX	Time	EX	Time
3	3	2495	31	1587	32	2150	39
3	5	1551	89	1587	82	1885	74
3	9	1289	106	1433	97	1618	87
3	17	1267	101	1327	100	1436	91
3	33	1322	167	1307	163	1410	178
3	97	1313	812	1302	616	1404	794
4	3	862	84	715	76	1041	90
4	5	593	334	531	272	693	216
4	9	525	383	470	359	505	339
4	17	467	430	442	387	458	382
4	33	468	628	407	578	471	617
4	97	468	3031	412	2616	459	2842
5	3	495	156	392	128	630	153
5	5	294	709	265	670	377	489
5	9	247	989	236	903	282	792
5	17	230	1108	210	837	270	1010
5	33	222	1868	205	1429	261	1470
5	97	220	7882	202	6071	256	6756

6 | CONCLUSIONS

In this paper, we propose an approach for optimised anchor placement in area-based indoor localisation, especially in the HSL setting (Table 7). Due to the complexity of the calculation of the areas, which are induced by the placement of the anchors, we developed a discretisation method to calculate these areas approximately. This method helped to significantly reduce the computational resources needed. Furthermore, our method is based on Tabu search, and we are using representations of the solutions, including the anchors and their movements, in such a way that, again, the running time is kept low. This version of the Tabu search algorithm leads to efficiently exploring the search space by changing the placement of subsets of anchors and results in high-quality solutions. We compared our approach to existing approaches from the literature, and in order to get a fair comparison, we used Python for all algorithms. Our approach (for 4 and 5 anchors) was as fast as or slightly slower than the fastest approach from the literature (GAAP) but yielded much better results. The LSAP approach from the literature, which is able to produce solutions of comparable quality as our approach in the three anchor cases and only slightly worse solutions in the four anchor cases, did not find solutions for five anchors and larger grids within 100,000 s. Also, in the three and four anchor cases, its runtime was dramatically longer than that of APOTSA.

TABLE 5 APOTSA $\gamma = 3$.

		150 × 246		170 × 216		300 × 123	
Anchor	**Resolution**	**EX**	**Time**	**EX**	**Time**	**EX**	**Time**
3	3	2495	53	1587	51	2150	63
3	5	1551	157	1587	147	1885	129
3	9	1289	208	1433	179	1618	161
3	17	1267	206	1380	210	1430	155
3	33	1281	327	1363	241	1413	305
3	97	1245	1719	1359	1452	1403	1401
4	3	862	162	748	142	1041	205
4	5	593	936	532	869	693	557
4	9	508	1222	477	1194	505	1030
4	17	491	1470	447	1236	458	1190
4	33	493	2292	430	1906	475	2063
4	97	430	9093	441	8489	467	8951
5	3	495	321	392	310	630	348
5	5	294	2613	266	2445	377	1666
5	9	244	3977	236	4021	282	3077
5	17	219	4504	214	3478	268	3635
5	33	217	7712	208	5615	271	6950
5	97	216	33,209	203	26,731	267	32,301

TABLE 6 APOTSA $\gamma = AN$.

		150 × 246		170 × 216		300 × 123	
Anchor	**Resolution**	**EX**	**Time**	**EX**	**Time**	**EX**	**Time**
3	3	2495	57	1587	50	2150	65
3	5	1551	164	1587	149	1885	127
3	9	1289	206	1432	182	1618	162
3	17	1267	219	1327	215	1430	167
3	33	1322	359	1330	277	1413	347
3	97	1313	1694	1367	1493	1403	1354
4	3	862	204	714	186	1041	268
4	5	593	1403	531	1224	693	869
4	9	499	2027	477	2134	505	1655
4	17	466	2347	447	2183	458	1926
4	33	467	3073	409	2916	468	3033
4	97	463	15,640	399	14,733	468	14,887
5	3	495	466	392	432	630	519
5	5	294	6694	265	7736	377	4373
5	9	224	12,762	235	13,461	269	10,609
5	17	221	15,830	203	16,051	266	11,996
5	33	213	24,489	196	22,965	261	19,936
5	97	212	122,716	195	114,225	259	97,138

TABLE 7 List of abbreviations.

Abbreviation	**Meaning**
LSAP	Local search anchors placement
RSS	Received signal strength
AOA	Angle of arrival
TOF	Time of flight
UWB	Ultra-wideband
RFID	Radio frequency identification
HSL	Half-symmetric lens
CPN	Counter propagation neural network
DE	Differential evolution
RSSI	Received signal strength indicator
CRLB	Cramer-rao lower bound
MSE	Mean square error
GAAP	Genetic algorithm anchors placement
NLOS	Non-line of sight
BLE	Bluetooth low energy
APs	Access points

7 | FUTURE WORK

The proposed method is based on an idealistic setting, since well-known real-world phenomena, such as NLOS conditions and noise, are not incorporated. Therefore applying APOTSA in a real-world setting needs further extensions for dealing with these phenomena. This could be done by adding obstacles to the service area and including ranging noise in a laboratory setting and evaluate the performance of APOTSA there and then modifying it. Moreover, as APOTSA is an area-based localisation method, the most important metric used is the expected residence area. However, this metric does not take the shape of an area into consideration. In practice, a perfect circle is better for the localisation of a tag than a small crescent. Therefore incorporating the shapes of the areas could also be an interesting research direction for area-based localisation in the future.

AUTHOR CONTRIBUTIONS

Sayyidshahab Nabavi: Investigation, methodology, writing—original draft, writing—review & editing. **Joachim Schauer**: Formal analysis, methodology, writing–original draft, writing—review & editing. **Carlo Alberto Boano**: Writing—review & editing. **Kay Römer**: Writing—review & editing.

ACKNOWLEDGEMENTS

This research was funded by the Austrian Science Fund (FWF) [10.55776/DFH5] within the DENISE project. For the purpose of open access, the authors have applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission

CONFLICT OF INTEREST STATEMENT

No Conflict of interest statement.

DATA AVAILABILITY STATEMENT

Data will be made available on request.

ORCID

Sayyidshahab Nabavi https://orcid.org/0000-0003-4772-9651

REFERENCES

1. Flueratoru, L., et al.: HTC vive as a ground-truth system for anchor-based indoor localization. In: Proceeding of the 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE (2020)
2. Zhang, H., Tan, S.Y., Seow, C.K.: TOA-based indoor localization and tracking with inaccurate floor plan map via MRMSC-PHD filter. In: IEEE Sensors Journal, vol. 19, pp. 9869–9882 (2019)
3. Toasa, F.A., et al.: Experimental demonstration for indoor localization based on AoA of bluetooth 5.1 using software defined radio. In: Proceeding of the IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), pp. 1–4. IEEE (2021)
4. Zafari, F., Gkelias, A., Leung, K.K.: A survey of indoor localization systems and technologies. In: IEEE Communications Surveys & Tutorials, vol. 21, pp. 2568–2599 (2019)
5. He, T., et al.: Range-free localization schemes for large scale sensor networks. In: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (2003)
6. Liu, C., et al.: Range-free sensor localisation with ring overlapping based on comparison of received signal strength indicator. Int. J. Sens. Netw. 2, 5–6 (2007)
7. Lasla, N., Bachir, A., Mohamed, Y.: Area-based vs. multilateration localization: a comparative study of estimated position error. In: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1138–1143. IEEE (2017)
8. Sheu, J.-P., Chen, P.-C., Hsu, C.-S.: A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement. In: IEEE Transactions on Mobile Computing, vol. 7.9 (2008)
9. Lasla, N., et al.: An effective area-based localization algorithm for wireless networks. IEEE Trans. Comput. 64, 8 (2014)
10. Lasla, N., et al.: Half-Symmetric Lens based localization algorithm for wireless sensor networks. In: Proceeding of the 37th Annual IEEE Conference on Local Computer Networks, pp. 320–323. IEEE (2012)
11. Cheriet, A., et al.: On optimal anchor placement for area-based localisation in wireless sensor networks. In: IET Wireless Sensor Systems, vol. 11.2 (2021)
12. Retrieved from: https://pypi.org/project/shapely/
13. Ren, Q., et al.: RSSI quantization and genetic algorithm based localization in wireless sensor networks. In: Ad Hoc Networks, vol. 107 (2020).102255
14. Flueratoru, L., et al.: HTC vive as a ground-truth system for anchor-based indoor localization. In: Proceeding of the 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE (2020)
15. Anzum, N., Afroze, S.F., Rahman, A.: Zone-based indoor localization using neural networks: a view from a real testbed. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE (2018)
16. Lee, S.-H., et al.: Performance of differential evolution algorithms for indoor area positioning in wireless sensor networks. Electronics 13.4, 705 (2024)
17. Pan, H., et al.: Indoor scenario-based UWB anchor placement optimization method for indoor localization. In: Expert Systems with Applications, vol. 205 (2022).117723
18. Miao, Q., Huang, B.: On the optimal anchor placement in single-hop sensor localization. Wireless Network 24, 1609–1620 (2018)
19. Zhou, F., et al.: Placement and concise MSE lower-bound for UAV-enabled localization via RSS. In: IEEE Transactions on Vehicular Technology, vol. 71, pp. 2209–2213 (2021)
20. Tian, Yu, et al.: Optimizing AP and beacon placement in WiFi and BLE hybrid localization. In: Journal of Network and Computer Applications, vol. 164 (2020).102673
21. Wang, H., et al.: Efficient beacon placement algorithms for time-of-flight indoor localization. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 119–128 (2019)
22. Lasla, N., et al.: On optimal anchor placement for efficient area-based localization in wireless networks. In: Proceeding of the IEEE International Conference on Communications (ICC), pp. 3257–3262. IEEE (2015)
23. Glover, F., Laguna, M.: Tabu search. In: Handbook of Combinatorial Optimization. Springer (1998)
24. url: https://github.com/noureddinel/anchor-placement-area-based-localization
25. Python 3 Documentation: time.Process_time. Accessed: [12.12.2023]. url: https://docs.python.org/3/library/time.html#time.process_time
26. Agrawal, P., et al.: Metaheuristic algorithms on feature selection: a survey of one decade of research (2009-2019). IEEE Access 9, 26766–26791 (2021)

How to cite this article: Nabavi, S., et al.: APOTSA: Anchor Placement Optimisation Using Discrete Tabu Search Algorithm for Area-Based Localisation. IET Wirel. Sens. Syst. 1–14 (2024). https://doi.org/10.1049/wss2.12092