Discrete Particle Swarm Optimization Algorithm for Solving Optimal Sensor Deployment Problem

Milan R. Rapačić, Željko Kanović, Zoran D. Jeličić

Abstract — This paper addresses the Optimal Sensor Deployment Problem (OSDP). The goal is to maximize the probability of target detection, with simultaneous cost minimization. The problem is solved by the Discrete PSO (DPSO) algorithm, a novel modification of the PSO algorithm, originally presented in the current paper. DPSO is general-purpose optimizer well suited for conducting search within a discrete search space. Its applicability is not limited to OSDP, it can be used to solve any combinatorial and integer programming problem. The effectiveness of the DPSO in solving OSDP was demonstrated on several examples.

Index Terms — Optimal sensor deployment, Particle Swarm Optimization, Discrete optimization

I. INTRODUCTION

A wide variety of problems in modern engineering can be seen as problems of optimal deployment of a group of agents within a predefined search space. The nature of the agents, off course, as well as the nature of the search space and the deployment goal may vary significantly from case to case. This paper addresses some of the problems of this type, Optimal Sensor Deployment Problem (OSDP).

OSDP is widely studied in literature. Distributed wireless sensor networks are being increasingly pervasive in many practical applications for either military or civil purposes [1]. In general, the task is to maximize the target detection probability where maximal deployment cost has been specified in advance, or, as it was done in this paper, to simultaneously maximize the target detection probability and minimize the deployment cost. A review of recent developments in the field of distributed sensor networks can be found in [2]. Recently, OSDP of moving sensors was addressed in [3] and [4]. Deployment problem involving underwater acoustic sensors was analyzed in [5]. Variations of OSDP were also discussed in [6] and [7].

It is important to realize that, although the deterministic detection model is considered frequently in literature, the detection process is stochastic. In nature, Target i is more likely to be present at certain points of the surveillance region than in others. Moreover, this probability may even change in time. Sensors are also non-deterministic. The probability that a sensor actually detects a target is, in general, a function of its state between time t and time t+h. This problem is not so, it is commonly more convenient to consider only a discrete set of possible locations for sensor deployment. The stochastic nature of the authors of the current paper, the stochastic nature of sensors themselves was addressed for the first time in [1], were various stochastic sensor models were also discussed.

In most practical applications, the surveillance region is continuous. Sometimes, however, sensors can be placed at only certain, discrete points within this region. Even if it is not so, it is commonly more convenient to consider only a discrete set of possible locations for sensor deployment. The PSO algorithm, a novel modification of the PSO algorithm, originally presented in [8] has been confirmed in [2] and [7] that combinatorial OSDP is in general NP-complete, or in another words, that there is no polynomial-time algorithm for its exact solution. It is therefore of great importance to investigate heuristic approaches that would pr onude near-optimal solutions in reasonable amount of time. Genetic algorithm (GA) with custom-made genetic operators and encoding is utilized in [1], while tabu-search (TS) based method was proposed in [7].

The solution presented in this paper is based on the Particle Swarm Optimization (PSO) algorithm. Contrary to the solution presented in [1], the optimization is not conducted with respect to the number of sensors (we consider the number of sensors to be fixed, known in advance). Nevertheless, the presented solution is flexible in its other aspects; it is not restrictive with respect to the size and the dimensionality of the detection region, the type and the number of sensors to be deployed, a s with respect to the previously known target probability, which is, in general, considered to be variable within the detection area – target is more likely to be active within certain areas of the detection region than within others. However, due to fixed number of sensors, the presented solution is computationally cheaper than the one presented in [1]. Since the original PSO algorithm assumes that the search space is continuous, a new modification of the PSO, the Discrete PSO (DPSO), proposed. DP SO 

change in time. Sensors are also non-deterministic. The probability that a sensor actually detects a target is, in general, a function of its state between time t and time t+h, but it also depends on other factors, such are environmental conditions and measurement noise. To the best knowledge of the authors of the current paper, the stochastic nature of sensors themselves was addressed for the first time in [1], were various stochastic sensor models were also discussed.

In most practical applications, the surveillance region is continuous. Sometimes, however, sensors can be placed at only certain, discrete points within this region. Even if it is not so, it is commonly more convenient to consider only a discrete set of possible locations for sensor deployment. The PSO algorithm, a novel modification of the PSO algorithm, originally presented in [8] has been confirmed in [2] and [7] that combinatorial OSDP is in general NP-complete, or in another words, that there is no polynomial-time algorithm for its exact solution. It is therefore of great importance to investigate heuristic approaches that would pr onude near-optimal solutions in reasonable amount of time. Genetic algorithm (GA) with custom-made genetic operators and encoding is utilized in [1], while tabu-search (TS) based method was proposed in [7].

The solution presented in this paper is based on the Particle Swarm Optimization (PSO) algorithm. Contrary to the solution presented in [1], the optimization is not conducted with respect to the number of sensors (we consider the number of sensors to be fixed, known in advance). Nevertheless, the presented solution is flexible in its other aspects; it is not restrictive with respect to the size and the dimensionality of the detection region, the type and the number of sensors to be deployed, as well as with respect to the previously known target probability, which is, in general, considered to be variable within the detection area – target is more likely to be active within certain areas of the detection region than within others. However, due to fixed number of sensors, the presented solution is computationally cheaper than the one presented in [1]. Since the original PSO algorithm assumes that the search space is continuous, a new modification of the PSO, the Discrete PSO (DPSO), was proposed. DP SO 

change in time. Sensors are also non-deterministic. The probability that a sensor actually detects a target is, in general, a function of its state between time t and time t+h, but it also depends on other factors, such are environmental conditions and measurement noise. To the best knowledge of the authors of the current paper, the stochastic nature of sensors themselves was addressed for the first time in [1], were various stochastic sensor models were also discussed.

In most practical applications, the surveillance region is continuous. Sometimes, however, sensors can be placed at only certain, discrete points within this region. Even if it is not so, it is commonly more convenient to consider only a discrete set of possible locations for sensor deployment. The PSO algorithm, a novel modification of the PSO algorithm, originally presented in [8] has been confirmed in [2] and [7] that combinatorial OSDP is in general NP-complete, or in another words, that there is no polynomial-time algorithm for its exact solution. It is therefore of great importance to investigate heuristic approaches that would pr onude near-optimal solutions in reasonable amount of time. Genetic algorithm (GA) with custom-made genetic operators and encoding is utilized in [1], while tabu-search (TS) based method was proposed in [7].
original P SO algorithm. Thus, a practical implementation, however, takes into account the specific nature of combinatorial search space. DPSO is inspired by the binary PSO described in [9], and is not specifically designed for the OSDP. In fact, it can be used, just the same, with any combinatorial or integer programming problem.

The rest of the paper is organized as follows. The OSDP can be formulated in numerous ways. One can, for example, determine the efficiency and cost into a single objective function which is

Formal statement of the OSDP is of crucial importance. It establishes a connection between the OSDP and a variety of other, both theoretical and practical problems of seemingly quite different nature. An interesting overview of such problems, that has a mong other includes signal compression, numerical integration and clustering, is given in [10]. Different formulations are also considered in [1].

It will be assumed that at the detection probability of each sensor depends solely on its distance from the target. Denote the sensor location by \( r_s \), the target location by \( r_t \). Let \( \| \cdot \| \) be suitably chosen distance measure (norm). In that case, the detection probability model for a sensor will be assumed as

\[
sdp(s,r) = \begin{cases} 
\exp(-\frac{\| r - r_s \|^2}{2\sigma^2}), & \| r - r_s \| \leq d_{\text{max}}, \\
0, & \| r - r_s \| > d_{\text{max}}. 
\end{cases}
\]

where \( \sigma \) and \( d_{\text{max}} \) are parameters specific to each particular sensor. Thus, the model for an arbitrary new sensor \( s \) (known as a Gaussian sensor model) is obtained from \( d_{\text{max}} \) in the detection range, the width (standard deviation) of the distribution, while the detection range \( d_{\text{max}} \) determines the maximal distance at which a sensor is capable of detecting a target. Other sensor models are considered in [1]. It should be mentioned that the solution proposed in the current paper does not depend on the particular sensor model, nor does it, in general, assume that all sensors obey the same detection probability model.

It is also assumed that, statistically, sensors do not affect one another. In other words, if \( d \) is any deployment, and \( d + s \) is a new deployment, obtained from \( d \) by insertion of an arbitrary new sensor \( s \), than at each point \( r \)

\[
ddp(d+s,r) = ddp(d;r) + sd(s;r) - ddp(d;r) \cdot sd(s;r). \tag{5}
\]

Since the detection probability of an empty deployment is zero (no sensors), the equation (5) can be used to calculate the target detection probability, or the entire deployment recursively.

Figure 1. Gaussian detection probability as a function of mutual distance between a sensor and a target, calculated using equation (4), with \( \sigma=1 \) and \( d_{\text{max}}=2 \).
B. Discrete Reformulation of the OSDP

In the sequel, it is assumed that the surveillance region is planar and rectangular. The procedure is directly applicable to arbitrarily shaped regions. Evaluation of (2) implies calculation of a surface (two-fold) integral. In order to calculate this integral the surveillance region is divided by a grid of imaginary lines to a net of equally sized cells. The procedure is depicted in Fig. 2. The “width” and “height” of each cell will be denoted by \( l_x \) and \( l_y \), while the number of nodes in each direction will be denoted by \( N_x \) and \( N_y \), respectively.

\[
eddp(i,j) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} ddp(d; r(i,j)) \cdot tp(r(i,j)) \cdot l_x \cdot l_y ,
\]

with \( r(i,j) \) being the coordinate of the node \( n \) the intersection of the \( i \)-the vertical and \( j \)-th horizontal grid-line. It is assumed that \( ddp \) and \( tp \) change relatively slow, meaning that their value does not change significantly within the single cell of the grid, then the double sum (6) is a good approximation of the surface integral (2). Finally, the optimality criteria (3) can be replaced by

\[
I_d(d) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} ddp(d; r(i,j)) \cdot tp(r(i,j)) - \beta \cdot dc(d) ,
\]

where \( \beta = \alpha \cdot (l_x \cdot l_y)^{-1} \). The criterion (7) is approximately equivalent to the original criteria (3).

III. PSO Algorithm

Particle Swarm Optimization (PSO) algorithm is a modern optimization technique, inspired by social behavior of animals moving in large groups – insects and birds in particular. It was originally proposed by Kennedy and Eberhart in 1995, and it has developed since trough the work of many authors [11, 12]. It has been successfully applied in a variety of engineering problems [13], [14]. An overview of the algorithm development, improvements and applications can be found in [9].

The basic idea of the PSO algorithm is to steer each particle to personal best and global best position. More formally, the velocity is calculated as

\[
\dot{v}_i[n+1] = \underbrace{w[n] \cdot v_i[n]}_{\text{inertia}} + 
\underbrace{cp[n] \cdot r_i[n] \cdot (p_i[n] - x_i[n]) +}_{\text{personal best}} 
\underbrace{cg[n] \cdot r_g[n] \cdot (g[n] - x_i[n])}_{\text{global best}}
\]

while the next position is obtained as

\[
x_i[n+1] = x_i[n] + v_i[n+1].
\]

Parameters \( w, cp \) and \( cg \) figuring in (8) are inertia, cognitive and social factor (or coefficient), respectively. Factors \( cp \) and \( cg \) are commonly known as acceleration factors. The inertia factor controls the stability of the algorithm, and it was not present in the original paper by Kennedy and Eberhart [8]. It was introduced (as a quantity different than 1) by Shi and Eberhart in [11]. It is known from various studies, including [11] and [15], that inertia value should not be greater than 1, and that it should decrease as the optimization process develops. It is common to decrease the inertia linearly from 0.9 to 0.4. Values of the acceleration coefficients determine the relative impact of local to global knowledge on the behavior of the particles. Larger cognitive factors move more autonomously, while less regard to the results obtained by other particles. Such behavior is crucial in later, exploitation oriented, stages of the optimization process. Larger social factor (or) means that the particles move in strong relation to each other, that they all tend to explore good solutions found by a swarm as a whole. Such behavior is crucial in later, exploration oriented, stages of the optimization process. In these later stages, the algorithm is expected to fine-tune the good solutions it already found. In the original paper [8] both acceleration factors were set to 2, but it was later demonstrated by Ratnaweera et al in [12] that linear decrease of the cognitive component from 2.5 to 0.5, while the next position is obtained as

\[
x_i[n+1] = x_i[n] + v_i[n+1].
\]

Parameters \( w, cp \) and \( cg \) figuring in (8) are inertia, cognitive and social factor (or coefficient), respectively. Factors \( cp \) and \( cg \) are commonly known as acceleration factors. The inertia factor controls the stability of the algorithm, and it was not present in the original paper by Kennedy and Eberhart [8]. It was introduced (as a quantity different than 1) by Shi and Eberhart in [11]. It is known from various studies, including [11] and [15], that inertia value should not be greater than 1, and that it should decrease as the optimization process develops. It is common to decrease the inertia linearly from 0.9 to 0.4. Values of the acceleration factors determine the relative impact of local to global knowledge on the behavior of the particles. Larger cognitive factors move more autonomously, while less regard to the results obtained by other particles. Such behavior is crucial in later, exploitation oriented, stages of the optimization process. Larger social factor (or) means that the particles move in strong relation to each other, that they all tend to explore good solutions found by a swarm as a whole. Such behavior is crucial in later, exploration oriented, stages of the optimization process. In these later stages, the algorithm is expected to fine-tune the good solutions it already found. In the original paper [8] both acceleration factors were set to 2, but it was later demonstrated by Ratnaweera et al in [12] that linear decrease of the cognitive component from 2.5 to 0.5, while the next position is obtained as

\[
x_i[n+1] = x_i[n] + v_i[n+1].
\]
simultaneous linear increase of social component from 0.5 to 2.5 is more appropriate. The reasons for this are clear from the above discussion. These conclusions have also been verified by the authors of this paper in their recent work [16], [17] and [18].

The optimization stops when predefined number of iterations has been achieved. Other stopping criteria can be introduced, but they are not used within this paper.

IV. DPSO

The classical PSO algorithm described in previous section is not applicable if the search space is discrete, that is if the position of each particle is bound to a discrete set of values. A modification of the classical algorithm is proposed in the current paper suitable for optimization within such a search space. The proposed solution is inspired by binary PSO algorithm, explained in [9].

It will be asumed that the search space is mapped into some rectangular subspace of \( Z^m \), where \( Z \) is the set of integers, and \( m \) is a arbitrary natural number denoting the dimension of the search space. The velocity vector \( v \) is calculated using the same formula as in the classical PSO, formula (8), but is saturated afterward using the hyperbolic tangent function to obtain a new quantity, that is referred to as the saturated velocity:

\[
\nu_{[n+1]} = \frac{1 - \exp(-v_{[n+1]})}{1 + \exp(-v_{[n+1]})} = \tanh(v_{[n+1]}/2).
\]

(10)

The position of each particle is no longer calculated using (9), but as

\[
x_{[n+1]} = x_{[n]} + \text{round}(\nu_{[n+1]} \times \Delta x_{\max}).
\]

(11)

Maximal displacement \( \Delta x_{\max} \) is a new parameter of the algorithm, and should be specified in advance. The \( \times \) symbol denotes element-wise prduct of two vectors, meaning that if \( a = [a_i] \) and \( b = [b_i] \) are vectors, their element-wise product is \( a \times b = [a_i \cdot b_i] \). The idea behind equations (10) and (11) is simple. If one would calculate the next position of a particle by equation (9), only by chance would the result be an integer number. One possibility is to round the resulting position. This would be satisfactory only if the number of possible positions is very large, that is if the problem is quasi-continuous. By using equations (10) and (11), the saturated velocity vector is in fact the amount of maximal displacement that will be applied in the current step. By having a maximal displacement parameter, the algorithm can accommodate various problem scales, ranging from binary to continuous problems. In bounded domains it is possible to replace or binary sum in (11) by a modulo-sum operator. That way, the search spaces becomes effectively limitless, because its bound points become neighbors.

It is convenient to limit the velocity to some predefined maximal value prior to applying the saturation function (10). This maximal velocity (\( v_{\max} \)) is another parameter of the algorithm. In context of the binary PSO it is usually chosen to be between 4 and 6. In the current paper, \( v_{\max} = 6 \) was used. The reason for velocity clamping is clear from the Fig.

3 depicting the saturation function, which maps all very large positive or negative velocities to almost the same value, 1 and -1, respectively. If the velocity is allowed to grow without restrictions, the algorithm would soon become inert and insensitive to the local properties of the objective function.

Notice that none of the features of the DPSO is specifically designed to accommodate O SPD. The modifications made to the classical PSO are only designed to adopt the algorithm to a combinatorial search space. In its other aspects, the nature of the original PSO is preserved within the proposed algorithm.

V. RESULTS

In the sequel, a problem of optimal deployment involving 4 sensors on a planar grid with 100x100 nodes is considered. All of the cells in the grid are of equal size, with equal vertices of length \( l_x = l_y = l \).

Figure 3. The saturation function \( \tanh(v/2) \). Notice that for \( v > 6 \) the value of saturation function is almost exactly 1. Similar situation is when \( v < -6 \). In that case, the saturation function is almost -1.

Figure 4. Graphical representation of target probability. Higher probability is depicted with lighter shades.

The solution is implemented using the programming language Python 2.5 [19] and its extension modules for numerical data processing and data visualization SciPy [20] and Matplotlib [21]. Regarding target probability, two separate cases were considered, both depicted in Fig. 4. Lighter areas are those with higher target probability. In the first case (Fig. 4a) it was assumed that the target probability obeys Gaussian probability distribution, centered at the very middle of the surveillance region, at node (50, 50), with standard deviation equal to 20l. In the second case (Fig.
it is assumed that the target distribution is the superposition of two separate Gaussian distributions with equal standard deviations of $\sigma = 10l$, centered at nodes $(25, 25)$ and $(75, 75)$.

### A. OSDP involving sensors of the same type

A problem involving $4$ identical sensors is considered first. All sensors have equal Gaussian probabilities (4), with standard deviation $\sigma = 10l$ and maximal range $d_{\text{max}} = 30l$. Since all sensors are the same, it is assumed that they are of equal price. Therefore, all possible deployment strategies have the same cost, and the cost term of the optimality criteria (7) can be neglected ($\beta = 0$). PSO parameters were chosen to be $w = 0.8$ and $cp = cg = 2$; maximal displacement was chosen to be $10$; the number of particles was chosen to be $30$, the number of iterations was set to $100$.

Figures 5 and 6 depict detection probability for the deployment obtained by DPSO algorithm for the target probabilities shown in Fig 4a and 4b, respectively.

### B. OSDP involving sensors of different type

Let us consider the problem of optimal deployment of $4$ different sensors. The assumption is that types of sensors are known in advance, and also that the number of sensors of each type is not bounded (optimal deployment may involve all four sensors of the same type, but it also possible that all selected sensors be of different type). The target probability will be assumed to be the one presented in Fig 4a. Table 1 shows the characteristics of different sensor types, including standard deviation of their Gaussian distribution, maximal detection range, and relative price.

<table>
<thead>
<tr>
<th>Sensor type</th>
<th>std. dev.</th>
<th>max. range</th>
<th>rel. price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>$5l$</td>
<td>$10l$</td>
<td>$0.3$</td>
</tr>
<tr>
<td>Type 2</td>
<td>$5l$</td>
<td>$20l$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>Type 3</td>
<td>$10l$</td>
<td>$25l$</td>
<td>$0.5$</td>
</tr>
<tr>
<td>Type 4</td>
<td>$15l$</td>
<td>$40l$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

Parameters of the PSO were the same as above. The solution is depicted in Fig 7. All selected sensors are of type 4. It is interesting, and of course expected, that if the price of a single sensor type would decrease dramatically, it would be selected even if it is not the most efficient one. Indeed, if the price of the sensor type 3 would decrease to $0.01$ relative units ($98\%$ decrease) all selected sensors are of type 3. The detection probability of the optimal deployment in this case is depicted in Fig. 8.

### VI. CONCLUSION

This paper addressed the OSDP problem involving stationary sensors within a surveillance region with target probability that vary from one point of the region to another, but that is stationary in time. A novel combinatorial optimizer, DPSO, is presented, and the obtained results testify that it is effective and promising technique. Several
issues remain open, and deserve further research. First, it would be interesting to apply DPSO to several other variations of OSDP, but also to other related problems discussed in [10]. Also, it would be interesting to investigate its behavior when applied parallel to the conventional PSO in solving hybrid optimization problems, where some of the variables are chosen from a combinatorial search space. Finally, a large number of variations of the original PSO have been proposed in literature. Most of these variations are applicable to the DPSO, and their efficacy in the combinatorial optimization context should be investigated.

Figure 8. Optimal surveillance region coverage, with 4 different sensors and target probability as depicted in figure 4a. The price of the third sensor type was reduced to only 2% of its original price.

REFERENCES


