

Article

Toward a Distributed Potential Game Optimization to Sensor Area Coverage Problem

Jun Huang, Jie Chen , Rongcheng Dong *, Xinli Xiong and Simao Xu

The College of Electronic Engineering, National University of Defense Technology, No. 460, Huangshan Road, Shushan District, Hefei 230037, China; huangjun0111@nudt.edu.cn (J.H.); jchen202209@163.com (J.C.); xiongxinli_@nudt.edu.cn (X.X.); xusimao@sina.com (S.X.)

* Correspondence: dongrch1995@outlook.com

Abstract

The sensor coverage problem is a well-known combinatorial optimization problem that continues to attract the attention of many researchers. The existing game-based algorithms mainly pursue a feasible solution when solving this problem. This problem is described as a potential game, and a memory-based greedy learning (MGL) algorithm is proposed, which can ensure convergence to Nash equilibrium. Compared with existing representative algorithms, our proposed algorithm performs the best in terms of average coverage, best value, and standard deviation within a suitable time. In addition, increasing memory length helps to generate a better Nash equilibrium.

Keywords: sensor area coverage; potential game; distributed optimization; greedy learning; memory length

MSC: 68W15



Academic Editor: Dun Han

Received: 20 June 2025

Revised: 16 August 2025

Accepted: 17 August 2025

Published: 22 August 2025

Citation: Huang, J.; Chen, J.; Dong, R.; Xiong, X.; Xu, S. Toward a Distributed Potential Game Optimization to Sensor Area Coverage Problem. *Mathematics* **2025**, *13*, 2709. <https://doi.org/10.3390/math13172709>

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The sensor area coverage problem is a fundamental optimization problem in wireless sensor systems, which has been proven to be NP-hard, with the goal of efficiently utilizing limited sensors to cover the largest area with minimal cost [1]. This problem is also a typical engineering application optimization problem, such as military reconnaissance [2,3], unmanned autonomous intelligent systems [4,5], Internet of Things [6,7], disaster relief [8,9], etc.

Due to the wide application of sensor area coverage problem, it has attracted the attention of many researchers who are committed to developing various optimization approaches for solving this problem. Early well-known approaches include genetic algorithm [10,11], particle swarm algorithm [12,13], simulated annealing algorithm [14,15], ant colony algorithm [16,17], etc. Those algorithms have long been proven to solve optimization problems and ensure that at least one feasible solution can be obtained. However, with the rapid development of science and technology, sensors have been mass-produced, especially when hundreds or thousands of sensors cooperate to perform this problem, the above algorithms are not applicable [18]. The reason is that those algorithms are essentially centralized algorithms, requiring a central manager to make decisions for all sensors [19]. This requirement not only causes a heavy computational burden on the sensor system, but also results in high communication requirements, and the robustness of the system will also be poor [20].

Therefore, how to develop distributed approaches to solve the sensor area coverage problem is the current research trend, and the most common one currently available is local search algorithm [21]. This algorithm starts from an initial solution, generates neighboring solutions through neighborhood actions, determines the quality of neighboring solutions, and repeats the above process until the termination condition is reached [22]. Although the local search algorithm is simple and easy to implement, it also has a drawback that it is prone to getting stuck in local optima, and the quality of the final solution is closely related to the choice of the initial solution. Note that a local optimal solution is usually not a global optimal solution and may not even reach an approximate optimal solution [23]. Therefore, in recent years, researchers have turned their attention to game theory.

Game theory is a theory that studies how individuals make reasonable strategies in complex interactions [24,25]. By establishing a game model for optimization problems and designing corresponding game algorithms, an equilibrium solution can be obtained [26]. As a special type of game, potential game has been widely applied. In a potential game, there exists a potential function, and the individual utility changes caused by changing strategies can be mapped to this potential function. By optimizing this potential function, the equilibrium state (i.e., Nash equilibrium) of the game can be achieved [27]. For optimization problems that can be modeled as potential games, various game algorithms have emerged, such as log-linear learning algorithm [28], binary log-linear learning algorithm [29], fictitious play algorithm [30], etc. The above algorithms all have a common point, that is, at each time step, each player only uses the information from the most recent past time step (i.e., local information) to make decisions. The goal of those algorithms is to obtain a feasible solution, but this feasible solution is usually far from the approximate optimal solution.

For this, for the sensor area coverage problem, we present a potential game model, where the potential function reflects the global objective, and a player's utility function is the wonderful life utility. Then, we propose a random memory algorithm (RMA), in which each player has a memory length that can store past step strategies, and make decisions based on local information. Subsequently, we verify that our proposed RMA can converge to a Nash equilibrium under any memory length. Numerical simulation will further demonstrate the value of our proposed RMA.

The main contributions of this paper are summarized as follows.

- (1) We present a potential game for the sensor area coverage problem, where player's utility function is the wonderful life utility.
- (2) We propose a game-based distributed algorithm, i.e., the RMA, which can converge to a Nash equilibrium under any memory length.
- (3) Numerical simulations demonstrate the effectiveness and superiority of our proposed RMA by comparing with the existing representative optimization algorithms. In addition, we found through numerical simulations that increasing the memory length can achieve better sensor area coverage.

The rest of this paper is organized as follows. Section 2 describes the sensor area coverage problem and presents a potential game model for this problem. Section 3 presents our proposed RMA in detail and provides theoretical analysis on its convergence. Section 4 provides numerical simulations to verify the performance of our proposed RMA. Section 5 summarizes this article and provides prospects for future research directions. The main notations of this article are listed in Abbreviations.

2. Preliminaries and Game Model

2.1. Sensor Area Coverage Problem

Consider a set of sensors, forming a set $V = \{1, 2, \dots, n\}$, which are randomly deployed to monitor (cover) a area Θ . Each sensor $i \in V$ can cover the effective area $\Theta_i(r_i)$ of a circle under a coverage radius r_i when it is in working condition. The location of the i th sensor can be denoted by loc_i . Without loss of generality, assume that the communication radius c of all sensors is the same. The set Γ_i composed of neighbors of sensor i can be defined as

$$\Gamma_i = \{j \in V : \|loc_i - loc_j\| \leq c\}. \quad (1)$$

Then, the sensor area coverage problem is whether each sensor chooses a coverage decision that collectively maximizes the global objective function

$$f(x) = \bigcup_{i \in V} \Theta_i(r_i) - \sum_{i \in V} C_i(r_i), \quad (2)$$

where $\bigcup_{i \in V} \Theta_i(r_i)$ is the union of all areas covered by the sensors, $C_i(r_i)$ is the cost function of sensor i under the coverage radius r_i , and $C_i(r_i)$ is expressed as

$$C_i(r_i) = k S_{\max}(r_i), \quad (3)$$

where $S_{\max}(r_i)$ is the maximum area covered by sensor i , $k \in (0, 1)$ is the penalty value, and $S_{\max}(r_i)$ is expressed as

$$S_{\max}(r_i) = \pi r_i^2, \quad (4)$$

2.2. Potential Game

In this subsection, we present the sensor area coverage problem as a game and prove that this game is a potential game.

Definition 1. The sensor area coverage problem can be defined as a game $G = \{V, X, F\}$, where

- (i) $V = \{1, 2, \dots, n\}$ is the set of all players (sensors);
- (ii) $X = \prod_{i=1}^n X_i$ is the strategy space, X_i is the strategy set of player i , $X_i = \{r^0, r^1, \dots, r^m\}$, and $r^0 = 0$, $r^z > 0$, $z \in \{1, \dots, m\}$.
- (iii) $F = \{f_1, f_2, \dots, f_n\}$ is the set of utility (i.e., payoff) functions, where $f_i \in F$ is the utility function of player i , $i \in V$.

Based the Definition 1, $x = (x_1, x_2, \dots, x_n)$ is the strategy profile, x_i is the strategy of player i , and $x_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$. Thus, we have $x = (x_i, x_{-i})$. Then, we will provide the definition of potential game.

Definition 2 ([31]). Consider a game $G = (V, X, F)$ with individual utility function f_i , if there exists a function φ satisfies

$$f_i(x'_i, x_{-i}) - f_i(x''_i, x_{-i}) = \varphi(x'_i, x_{-i}) - \varphi(x''_i, x_{-i}), \quad (5)$$

then the game $G = (V, X, F)$ is called potential game, and φ is the corresponding potential function.

Considering that when a pair of sensors are covered, once there is coverage overlap, they are within the communication range of each other. Thus, the communication radius $c = 2r^m$. The utility function of player i is designed as

$$f_i(x_i, x_{-i}) = [\Theta_i(x_i) \setminus \bigcup_{j \in \Gamma_i} \Theta_j(x_j) - C_i(x_i)] / S_{\max}(r^m). \quad (6)$$

Theorem 1. A game $G = (V, X, F)$ with individual utility function f_i in (6) is potential game, and the corresponding potential function is expressed as

$$\varphi(x) = f(x) / \Delta. \quad (7)$$

Proof. By (2), we have

$$f(x_i, x_{-i}) = \Theta_i(x_i) \setminus \bigcup_{j \in \Gamma_i} \Theta_j(x_j) + \bigcup_{j \in V \setminus i} \Theta_j(x_j) - C_i(x_i) - \sum_{j \in V \setminus i} C_j(x_j).$$

Further, for $x'_i, x''_i \in X_i$, we have

$$\begin{aligned} \varphi(x'_i, x_{-i}) - \varphi(x''_i, x_{-i}) &= \left[\Theta_i(x'_i) \setminus \bigcup_{j \in \Gamma_i} \Theta_j(x_j) - C_i(x'_i) \right] / \Delta - \left[\Theta_i(x''_i) \setminus \bigcup_{j \in \Gamma_i} \Theta_j(x_j) - C_i(x''_i) \right] / \Delta \\ &= f_i(x'_i, x_{-i}) - f_i(x''_i, x_{-i}). \end{aligned}$$

Thus, the game $G = (V, X, F)$ with individual utility function f_i in (6) is potential game, and the corresponding potential function is (7). Then, the proof of Theorem 1 is completed. $\square$

Remark 1. The sensor coverage problem is an important application problem in sensor system, (2) transforms this problem into a combinatorial optimization problem, whose optimal solution is equivalent to an optimal sensor area coverage solution. In addition, (2) serves as the bridge that casts this problem into a potential game, which can be solved within a distributed optimization framework. Note that for any optimization problem that can be modeled as a potential game, there must exist at least one Nash equilibrium.

3. Distributed Algorithm Designed and Analysis

3.1. Algorithm Design

For an optimization problem that can be modeled as a potential game, it is crucial to design a suitable distributed algorithm, which requires a reasonable balance between the solution quality of the algorithm's convergence and the runtime. For this, we propose a dynamic fictitious play (DFP) distributed algorithm, in which each player can dynamically adjust their own strategy in parallel based on local information. The procedure of our proposed DFP distributed algorithm is described as follows.

Step 1: At each time step t , each player randomly chooses its current strategy for the next time step or chooses to update its strategy.

Step 2: Each player i involved in updating strategy calculates its utility $f_i(x_i^t, x_{-i}^t)$ and best response BR_i^t , then records strategy x_i^t and BR_i^t into a set Λ_i^t .

Step 3: Each player i involved in updating strategy choose a strategy $\hat{x}_i^t$ from the set Λ_i^t for the next time step-based dynamic selection rule, i.e., $x_i^{t+1} = \hat{x}_i^t$.

In Step 2, BR_i^t is expressed as

$$BR_i^t = \operatorname{argmax}_{x_i^t \in X_i} f_i(x_i^t, x_{-i}^t). \quad (8)$$

In Step 3, the dynamic selection rule

$$\hat{x}_i^t = \begin{cases} BR_i^t, & w.p. \delta_i^t \\ x_i^t, & w.p. \text{ else} \end{cases}, \quad (9)$$

where δ_i^t is expressed as

$$\delta_i^t = \frac{e^{f_i(BR_i^t, x_{-i}^t)}}{\sum_{x_i \in \Lambda_i^t} e^{f_i(x_i, x_{-i}^t)}}. \quad (10)$$

Remark 2. (1) There exists multiple distributed algorithm used to solve potential game optimization problems, such as log-linear learning algorithm [28], binary log-linear learning algorithm [29], etc. Those algorithms have one point in common, which is that at each time step, only one player has the opportunity to update its strategy, while the remaining players keep their strategies unchanged. This sequential update strategy mechanism can cause those algorithms needing a long time to obtain a satisfactory solution in large-scale systems.

(2) Although there exist algorithms that allow each player to update their strategies at every time step, such as fictitious play algorithm [30]. In this algorithm, δ_i^t is set to a fixed value, which can cause this algorithm to easily fall into inefficient local optimal solution. In contrast, in our proposed DFP algorithm, δ_i^t is not limited to a fixed value, but dynamically changes based on local information.

(3) Our proposed DFP distributed algorithm is simple and easy to understand, and each player can independently update their strategy based on local information synchronization at each time step. This algorithm is suitable for multiple application fields. Especially in hostile environments, multiple drones collaborate in combat, and the sensors on each drone can autonomously monitor the target, as the sensors consume energy during operation. Therefore, in order to maximize efficiency and quickly execute monitoring tasks, we hope that each sensor can make independent decisions, and the DFP algorithm is suitable.

3.2. Convergence Analysis

For an algorithm, its convergence is crucial. Without convergence, the algorithm is difficult to apply in practice. Thus, in this subsection, we further analysis the convergence of this algorithm. First, we present the definition of Nash equilibrium.

Definition 3 ([32]). A Nash equilibrium $x^* = (x_1^*, \dots, x_n^*)$ is such such a strategy profile where no player can unilaterally change its strategy to increase its utility, i.e.,

$$f_i(x_i^*, x_{-i}) \geq f_i(x_i', x_{-i}), \quad (11)$$

where $\forall x_i' \in X_i$.

Theorem 2. Consider a potential game $G = (V, X, F)$ with individual utility function f_i in (4), our proposed DFP distributed algorithm can converge to a Nash equilibrium with probability 1.

Proof. Since the number of sensors is finite and each player i have finite strategy set, i.e., $|X_i| < +\infty$. Thus, the number of Nash equilibrium is finite, and they form a set $X^{\text{ne}} = \{x^{\text{ne},1}, \dots, x^{\text{ne},g}\}$, g is the number of all Nash equilibriums, i.e., $g < |X|$. We will divide into two steps to prove the convergence of our proposed DFP algorithm. Firstly, we will prove that this algorithm can ensure that each strategy profile will touch a Nash equilibrium, and then prove that once touched, it locks onto this Nash equilibrium.

Part I: At each time step t , strategy profile $x = (x_1, \dots, x_n)$ is not a Nash equilibrium. Thus, there must exists at least a player i that satisfies $x_i^t \neq BR_i^t$. Then, according to Step 2 and Step 3 of our proposed DFP algorithm, there exists a positive probability for player i updating its strategy x_i^t to BR_i^t , and all its neighbors will keep their strategies, $x_j^{t+1} = x_j^t, \forall j \in \Gamma_i$, i.e.,

$$p(x_i^t \rightarrow BR_i^t, x_j^t \rightarrow x_j^t) = (1 - \delta_i^t) \prod_{j \in \Gamma_i} \delta_j^t > 0. \quad (12)$$

Similarly, there exists a positive probability for the strategy profile x^t transform into the strategy profile x^{t+1} , i.e.,

$$p(x^t \rightarrow x^{t+1}) = \prod_{i \in C^t} (1 - \delta_i^t) \prod_{j \in V \setminus C^t} \delta_j^t > 0, \quad (13)$$

where $C^t = \{i | x_i^t \neq BR_i^t, \forall i \in V\}$ is the set of player involved in updating strategies, and their neighbors keep their strategies.

Further, we have

$$\varphi(x^{t+1}) - \varphi(x^t) = \sum_{i \in C^t} (f_i(x_i^{t+1}, x_{-i}^t) - f_i(x_i^t, x_{-i}^t)) > 0, \quad (14)$$

where $x_{-i}^{t+1} = x_{-i}^t$.

Thus, the value of the potential function φ strictly increases with a positive probability over time step t . Since the limited number of players, the value of the potential function is also finite. Thus, there must exist a time step t^{ne} such that $\varphi(x^{t'}) = \varphi(x^{t^{\text{ne}}})$, where $t' \geq t^{\text{ne}}$, and $x^{t^{\text{ne}}} \in X^{\text{ne}}$. This means that our proposed DFP algorithm can ensure that each strategy profile will touch a Nash equilibrium with a positive probability.

If the probability of an event E occurring with a positive probability at any time step t , i.e., $p^t(E) > 0$. Then, the event E will eventually occur due to $p(E) = 1 - \prod_{z=1}^{+\infty} (1 - p^z(E)) = 1$. This means that our proposed DFP algorithm can ensure that each strategy profile must touch a Nash equilibrium in infinite time.

Part II: At time step t , if strategy profile $x = (x_1, \dots, x_n)$ is a Nash equilibrium, then we have $x_i^t = BR_i^t$, and $\hat{x}_i^t = x_i^t, \forall i \in V$. Thus, we have $x_i^{t+1} = x_i^t$. Similarly, we have $x_i^{t+\tau} = x_i^t, \forall \tau \geq 0$. This means that any Nash equilibrium must touch this Nash equilibrium and lock it in.

Based on the analysis of part I and II, our proposed DFP distributed algorithm can converge to an Nash equilibrium with probability 1. Then, the proof of Theorem 2 is completed. $\square$

Remark 3. (1) Theorem 2 shows that our proposed DFP distributed algorithm can ensure that a Nash equilibrium can be obtained. A Nash equilibrium can be obtained through other ways. For example, any non-Nash equilibrium transforms into a Nash equilibrium within a time step.

(2) For common distributed game algorithms used to solve sensor coverage problems, such as log-linear learning algorithm [28], binary log-linear learning algorithm [29], etc. Although these algorithms can usually obtain a feasible solution, they theoretically cannot converge. For the FP algorithm, δ_i^t is a fixed value, and this algorithm is not affected by the strategy during the decision-making process.

4. Numerical Simulation

To evaluate the performance of our proposed DFP distributed algorithm, we compared it numerically with representative algorithms that can be modeled as potential games. Let us consider such a scenario: in a wireless sensor network composed of sensors from multiple unmanned aerial vehicles, the main requirement is to allow some sensors to monitor (i.e., sense) and collect effective information about the target area. Since sensors consume energy when in working, and the area the monitored area, the more energy is consumed. Therefore, how the sensor decides (to work or to turn off) to maximize the global objective value can be regarded as the sensor coverage problem of this paper. For fairness, each algorithm runs independently 1000 times to calculate the average value under each setting. All numerical simulations are realized on the MATLAB R2023a and the same computer with 3.20 GHz CPU and 16.0 G RAM.

4.1. Simulation Set

In this subsection, the area $\Theta_i(r_i)$ can be discretized as a two-dimensional grid represented by the Cartesian product $\{0, 1, \dots, d\} \times \{0, 1, \dots, d\}$. The area length d is set from 20 to 200, and the number n of agents is set to 20, 50, 100, 200, respectively. The strategy set X_i of player i is set to $\{0, 5\}$, i.e., $X_i = \{0, 5\}, \forall i \in V$. the communication radius $c = 2r^m = 10$. In other words, if a sensor i does not work, its sensing radius x_i and the communication radius c are both 0. If a sensor i is working, its sensing radius x_i is 5, and the communication radius c is 10. A visual diagram as shown in Figure 1, where n is 15 and d is 20. In Figure 1, each solid circle represents a sensor, and the black dots indicate sensors in the turn off state, meaning its strategy is 0; the orange-red solid dots represent sensors in working state, the orange-red solid dots represent sensors in 5. We can clearly see that within a given area, active (working) sensors near the center can cover a complete circle area, while active sensors near the periphery do not cover a complete circle area. Due to the possible overlap in coverage area between active sensors, it can be seen from (6) that the benefits brought by each active sensor coverage are its own unique coverage area, and its cost is only related to the coverage radius. Therefore, our proposed MGL algorithm allows each active sensor to access the strategies of neighboring sensors for decision-making, with the expectation of achieving global goal maximization. The k in (4) is set to 0.2. For each 1000 independent runs of the algorithm, we calculate its the corresponding average value $\bar{f}$, best value f_{best} , standard deviation δ , and average runtime $\bar{T}$.

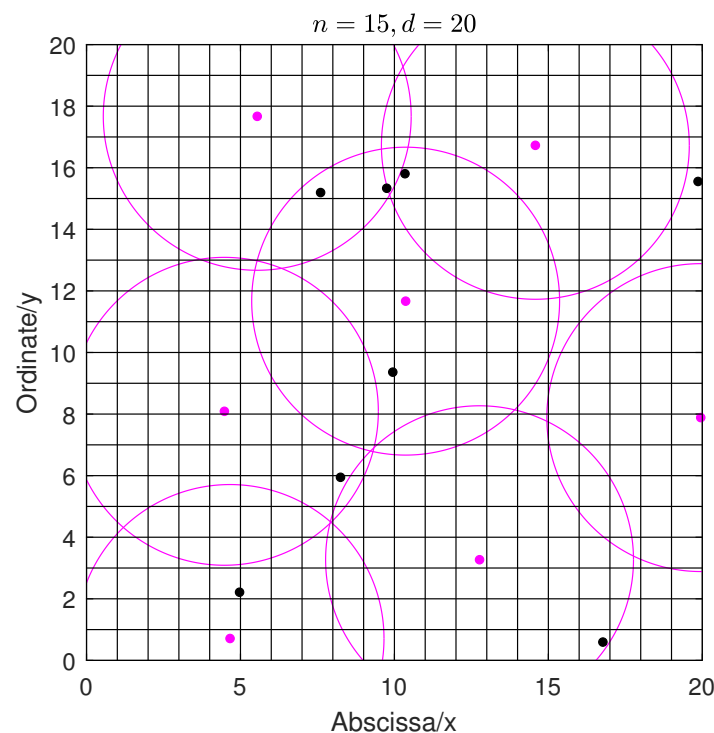


Figure 1. Schematic diagram of sensor coverage. The coordinates of the fifteen sensors are (13.43,24.27), (16.63,53.01), (14.91,6.64), (43.73,50.18), (31.02,47.42), (50.31,1.78), (13.98,2.13), (38.30,9.81), (59.58,46.66), (29.84,28.08), (31.09,35.00), (59.85,23.66), (22.81, 45.58), (29.25,46.00), and (24.75,17.84), respectively.

4.2. Compared with the Existing Representative Algorithms

Typical representative algorithms that participate in numerical simulation comparisons include log-linear learning (LLL) algorithm [28], fictitious play (FP) algorithm [30], best response (BR) algorithm [33], and genetic algorithm (GA) [34]. The simple process of those algorithms are as follows:

LLL algorithm: This is a typical distributed algorithm that can be used to solve problems that can be modeled as potential games. The main idea of this algorithm is that, at each time step t , a player i is randomly selected to choose a strategy x_i from its strategy profile X_i with the probability p_{x_i} for the next time step, and the probability p_{x_i} is expressed as

$$p_{x_i}(\alpha) = \frac{e^{\alpha f_i(x_i, x_{-i}^t)}}{\sum_{\tilde{x}_i \in X_i} e^{\alpha f_i(\tilde{x}_i, x_{-i}^t)}}, \quad (15)$$

where $\alpha \geq 0$. When $\alpha = 0$, player i is randomly selects a strategy from its strategy set X_i for the next time step. When α shows a tendency towards positive infinity, player i selects the best response BR_i^t for the next time step. Note that regardless of the given task value of α , the LLL algorithm cannot converge. Therefore, for the LLL algorithm, we will sample at the $100n$ time step, and α is set 1000.

FP algorithm: This is a commonly distributed algorithm that can be used to solve problems that can be modeled as potential games. This algorithm has the characteristics of simplicity and fast convergence. The main idea of this algorithm is that, at each time step t , each player i chooses strategy x_i^t with probability $\omega \in (0, 1)$ or selects the best response BR_i^t with probability $1 - \omega$ for the next time step. Since a small ω induce a better solution, ω is thus set to 0.1. This algorithm can converge to a Nash equilibrium.

BR algorithm: This is also a distributed algorithm that can be used to solve problems that can be modeled as potential games. The main idea of this algorithm is that, at each time step t , a player i is randomly selected to choose the best response BR_i^t . This algorithm can converge to a Nash equilibrium.

GA: This is a classic metaheuristic algorithm, a centralized optimization algorithm that simulates natural selection and genetic mechanisms and belongs to the branch of evolutionary computation. This algorithm has been applied to various fields, with main operations including selection, crossover, and mutation. In this section, the fitness function adopts (2).

The numerical simulation results are shown in Table 1. From this Table, we can see the following:

(1) When we observe the solution quality, i.e., average value $\bar{f}$ and standard deviation σ , our proposed DFP algorithm performs the best result by comparing with other algorithms, followed by the GA, LLL algorithm, BR algorithm and FP algorithm. Specifically, compared to the FP algorithm, the reason why our proposed DF algorithm performs better in this regard is that our algorithm has dynamic probability in strategy selection, while in the FP algorithm, each player tends to choose the best response.

(2) When we further observe the best value f_{best} , under all experimental settings (that is n and l), our proposed DFP algorithm can achieve the best values by comparing with other algorithms. In some cases, other algorithms can also achieve optimal values, such as in small-scale systems, i.e., $n = 20, 50$, the GA can also achieve the best values. It is worth noting that under various settings, we have found through extensive simulation testing that for a given setting of n/l , our proposed algorithm reaches its optimal value every 1000 runs.

(3) When we observe the average runtime $\bar{T}$, FP algorithm performs the best result by comparing with other algorithms, followed by the GA, DFP algorithm, BR algorithm, and LLL algorithm. The reason why the LLL algorithm performs poorly in terms of average runtime is that its update strategy mechanism is essentially a serial update strategy mechanism, and its average running time is related to the system size (i.e., the number of sensors), while other algorithms are essentially parallel update strategy mechanism. Note that the average runtime of our proposed DFP algorithm is very close to that of the FP algorithm.

Table 1. Comparison results among different algorithms: Average value $\bar{f}$ /Best value f_{best} /Standard deviation σ /Average runtime $\bar{T}$ (Unit: Seconds).

n/l	Algorithm Comparison																			
	$\bar{f}$					f_{best}					σ					$\bar{T}$				
	DFP	GA	LLL	BR	FP	DFP	GA	LLL	BR	FP	DFP	GA	LLL	BR	FP	DFP	GA	LLL	BR	FP
20/20	279.7	278.8	278.1	276.9	276.8	282.2	282.2	281.8	281.4	281.4	0.80	0.96	1.35	0.88	0.85	0.006	0.591	0.469	0.010	0.004
20/30	555.2	553.9	553.0	552.7	551.8	558.0	558.0	557.4	557.0	557.0	1.24	1.51	1.88	1.34	1.30	0.004	0.590	0.469	0.009	0.003
20/40	785.2	783.7	782.6	781.6	781.5	789.6	789.6	788.6	788.0	788.0	1.88	2.33	2.70	1.99	1.95	0.003	0.591	0.468	0.009	0.002
50/40	1018.1	1016.9	1016.0	1014.5	1014.4	1023.0	1023.0	1023.0	1022.4	1022.4	1.62	1.90	2.31	1.72	1.68	0.006	1.476	1.171	0.024	0.004
50/60	1824.2	1822.4	1821.1	1819.6	1819.5	1829.2	1829.2	1828.4	1827.6	1827.6	1.88	2.27	2.73	1.98	1.93	0.005	1.477	1.171	0.024	0.003
50/80	2271.5	2269.7	2268.2	2267.0	2266.8	2276.6	2276.6	2276.0	2275.4	2275.4	2.21	2.75	3.15	2.32	2.26	0.004	1.475	1.170	0.023	0.002
100/80	3220.2	3218.8	3217.8	3216.1	3215.9	3226.6	3225.8	3225.8	3225.0	3225.0	1.70	2.18	2.65	1.81	1.75	0.006	2.951	2.341	0.047	0.004
100/100	4008.8	4006.0	4005.4	4003.0	4002.9	4016.6	4015.6	4015.6	4014.8	4014.8	1.94	2.49	3.04	2.06	2.00	0.006	2.950	2.340	0.047	0.004
100/150	5070.2	5069.9	5066.1	5063.9	5063.8	5078.0	5078.0	5077.2	5076.4	5076.4	2.23	2.84	3.40	2.38	2.30	0.005	2.950	2.340	0.047	0.003
200/150	8433.7	8430.4	8428.0	8425.3	8425.2	8441.2	8438.6	8437.0	8436.2	8436.2	2.23	2.78	3.30	2.35	2.29	0.007	5.896	4.678	0.094	0.005
200/200	9850.3	9848.3	9846.8	9844.1	9843.91	9860.2	9859.0	9858.2	9857.4	9857.4	2.52	4.10	3.75	2.62	2.57	0.006	5.896	4.677	0.094	0.004
200/300	11,429.5	11,427.0	11,424.8	11,422.2	11,422.0	11,440.8	11,438.8	11,437.4	11,436.6	11,436.6	2.87	3.51	4.24	2.98	2.92	0.005	5.895	4.675	0.093	0.003

Bold values indicate the best value after comparison.

In sum, our proposed DFP algorithm can obtain a satisfactory solution within a suitable time.

5. Conclusions and Future Research Directions

This paper investigates the issue of the multi-agent area coverage problem and addresses it within a distributed framework. We have treated each sensor as a player and use the game theory approach to solve it. We have established a potential game for this problem and designed a novel dynamic fictitious play distributed algorithm, which theoretically demonstrates that this algorithm can ensure obtaining a Nash equilibrium. Simulation results have shown that compared with typical optimization algorithms, our proposed algorithm can obtain the best results in terms of average coverage, best value, and standard deviation in a suitable time.

For future research directions, we will consider the following three aspects: (i) developing a novel distributed optimization method to achieve the optimal solution of sensor area coverage problem; (ii) researching a sensor area coverage problem in more complex environments, such as sensors being able to move, exit, or join; (iii) explore applying the developed optimization approaches to real-world scenarios.

Author Contributions: Software, X.X.; Validation, J.C.; Formal analysis, R.D.; Investigation, J.H.; Resources, J.H. and S.X.; Data curation, X.X. and S.X.; Writing—original draft, J.H.; Writing—review & editing, J.C. and R.D.; Project administration, R.D.; Funding acquisition, S.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the high-level talent fund No. 22-TDRCJH-02-013.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Notation	Definitions
$\Theta_i(r_i)$	the area covered by sensor i under coverage radius r_i
$C_i(r_i)$	the cost function of sensor i under coverage radius r_i
V	the set of players (sensors)
n	the number of players
F	the set of cost functions
X	the strategy space

X_i	the strategy set of player i
x	the strategy profile (solution)
x^*	the Nash equilibrium
X^{ne}	the set of all Nash equilibriums
x_i	the strategy of player i
x_{-i}	the strategy profile of all players expect player i
BR_i^t	the best response of player i at time step t
r_i	the coverage radius of player i
c	the communication radius of all sensors
f_i	the utility of player i
f_{best}	the best value in a set of data
φ	the potential function
Γ_i	the neighbor set of player i

References

1. Zhou, E.; Liu, Z.; Zhou, W.; Lan, P.; Dong, Z. Position and orientation planning of the uav with rectangle coverage area. *IEEE Trans. Veh. Technol.* **2025**, *74*, 1719–1724. [\[CrossRef\]](#)
2. Li, X.; Lu, X.; Chen, W.; Ge, D.; Zhu, J. Research on uavs reconnaissance task allocation method based on communication preservation. *IEEE Trans. Consum. Electron.* **2024**, *70*, 684–695. [\[CrossRef\]](#)
3. Shaofei, M.; Jiansheng, S.; Qi, Y.; Wei, X. Analysis of detection capabilities of leo reconnaissance satellite constellation based on coverage performance. *J. Syst. Eng. Electron.* **2018**, *29*, 98–104. [\[CrossRef\]](#)
4. Chen, J.; Ling, F.; Zhang, Y.; You, T.; Liu, Y.; Du, X. Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system. *Swarm Evol. Comput.* **2022**, *69*, 101005. [\[CrossRef\]](#)
5. Hu, B.-B.; Zhang, H.-T.; Liu, B.; Ding, J.; Xu, Y.; Luo, C.; Cao, H. Coordinated navigation control of cross-domain unmanned systems via guiding vector fields. *IEEE Trans. Control Syst. Technol.* **2024**, *32*, 550–563. [\[CrossRef\]](#)
6. Zhu, C.; Fan, X.; Deng, X.; Liu, S.; Yin, D.; Gao, H.; Yang, L.T. Area coverage reliability evaluation for collaborative intelligence and meta-computing of decentralized industrial internet of things. *IEEE Internet Things J.* **2025**, *12*, 13734–13745. [\[CrossRef\]](#)
7. Yang, F.; Shu, L.; Duan, N.; Yang, X.; Hancke, G.P. Complete area c-probability coverage in solar insecticidal lamps internet of things. *IEEE Internet Things J.* **2023**, *10*, 22764–22774. [\[CrossRef\]](#)
8. Gui, J.; Cai, F. Coverage probability and throughput optimization in integrated mmwave and sub-6 ghz multi-uav-assisted disaster relief networks. *IEEE Trans. Mob. Comput.* **2024**, *23*, 10918–10937. [\[CrossRef\]](#)
9. Shi, K.; Peng, X.; Lu, H.; Zhu, Y.; Niu, Z. Application of social sensors in natural disasters emergency management: A review. *IEEE Trans. Comput. Soc. Syst.* **2023**, *10*, 3143–3158. [\[CrossRef\]](#)
10. Cao, Y.; Feng, W.; Quan, Y.; Bao, W.; Dauphin, G.; Song, Y.; Ren, A.; Xing, M. A two-step ensemble-based genetic algorithm for land cover classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 409–418. [\[CrossRef\]](#)
11. Hanh, N.T.; Binh, H.T.T.; Hoai, N.X.; Palaniswami, M.S. An efficient genetic algorithm for maximizing area coverage in wireless sensor networks. *Inf. Sci.* **2019**, *488*, 58–75. [\[CrossRef\]](#)
12. Wang, S.; Zhou, A. Leader prediction for multiobjective particle swarm optimization. *IEEE Trans. Evol. Comput.* **2024**, *29*, 1356–1370. [\[CrossRef\]](#)
13. Bonnah, E.; Ju, S.; Cai, W. Coverage maximization in wireless sensor networks using minimal exposure path and particle swarm optimization. *Sens. Imaging* **2020**, *21*, 4. [\[CrossRef\]](#)
14. He, Y.; Huang, J.; Li, W.; Zhang, L.; Wong, S.-W.; Chen, Z.N. Hybrid method of artificial neural network and simulated annealing algorithm for optimizing wideband patch antennas. *IEEE Trans. Antennas Propag.* **2024**, *72*, 944–949. [\[CrossRef\]](#)
15. Wu, X.; Yang, Y.; Xie, Y.; Ma, Q.; Zhang, Z. Multiregion mission planning by satellite swarm using simulated annealing and neighborhood search. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 1416–1439. [\[CrossRef\]](#)
16. Dai, L.-L.; Pan, Q.-K.; Miao, Z.-H.; Suganthan, P.N.; Gao, K.-Z. Multi-objective multi-picking-robot task allocation: Mathematical model and discrete artificial bee colony algorithm. *IEEE Trans. OnIntelligent Transp. Syst.* **2024**, *25*, 6061–6073. [\[CrossRef\]](#)
17. Xie, X.; Yan, Z.; Zhang, Z.; Qin, Y.; Jin, H.; Xu, M. Hybrid genetic ant colony optimization algorithm for full-coverage path planning of gardening pruning robots. *Intell. Serv. Robot.* **2024**, *17*, 661–683. [\[CrossRef\]](#)
18. Aga, R.S.; Duncan, L.; Davidson, L.; Ouchen, F.; Aga, R.; Heckman, E.M.; Bartsch, C.M. Design and fabrication of a metal resistance strain sensor with enhanced sensitivity. *IEEE Sens. Lett.* **2024**, *8*, 2504004. [\[CrossRef\]](#)
19. Zhou, X.; Rao, W.; Liu, Y.; Sun, S. A decentralized optimization algorithm for multi-agent job shop scheduling with private information. *Mathematics* **2024**, *12*, 971. [\[CrossRef\]](#)

20. Gao, Y.; Yang, S.; Li, F.; Trajanovski, S.; Zhou, P.; Hui, P.; Fu, X. Video content placement at the network edge: Centralized and distributed algorithms. *IEEE Trans. Mob. Comput.* **2023**, *22*, 6843–6859. [[CrossRef](#)]
21. Luo, C.; Xing, W.; Cai, S.; Hu, C. Nusc: An effective local search algorithm for solving the set covering problem. *IEEE Trans. On Cybernetics* **2024**, *54*, 1403–1416. [[CrossRef](#)] [[PubMed](#)]
22. Seyedkolaei, A.A.; Nasser, S.H. Facilities location in the supply chain network using an iterated local search algorithm. *Fuzzy Inf. Eng.* **2023**, *15*, 14–25. [[CrossRef](#)]
23. Zeng, L.; Chiang, H.-D.; Liang, D.; Xia, M.; Dong, N. Trust-tech source-point method for systematically computing multiple local optimal solutions: Theory and method. *IEEE Trans. Cybern.* **2022**, *52*, 11686–11697. [[CrossRef](#)]
24. Guo, C.; Song, Y. Multi-subject decision-making analysis in the public opinion of emergencies: From an evolutionary game perspective. *Mathematics* **2025**, *13*, 1547. [[CrossRef](#)]
25. Liu, S.; Li, L.; Zhang, L.; Shen, W. Game theory based dynamic event-driven service scheduling in cloud manufacturing. *IEEE Trans. Autom. Sci. Eng.* **2024**, *21*, 618–629. [[CrossRef](#)]
26. Zhang, Y.; Xiang, Z. Nash equilibrium solutions for switched nonlinear systems: A fuzzy-based dynamic game method. *IEEE Trans. Fuzzy Syst.* **2025**, *33*, 2006–2015. [[CrossRef](#)]
27. Varga, B.; Inga, J.; Hohmann, S. Limited information shared control: A potential game approach. *IEEE Trans. Hum.-Mach. Syst.* **2023**, *53*, 282–292. [[CrossRef](#)]
28. Li, Z.; Liu, C.; Tan, S.; Liu, Y. A timestamp-based log-linear algorithm for solving locally-informed multi-agent finite games. *Expert Syst. Appl.* **2024**, *249*, 123677. [[CrossRef](#)]
29. Yan, K.; Xiang, L.; Yang, K. Cooperative target search algorithm for uav swarms with limited communication and energy capacity. *IEEE Commun. Lett.* **2024**, *28*, 1102–1106. [[CrossRef](#)]
30. Dumitrescu, R.; Leutscher, M.; Tankov, P. Linear programming fictitious play algorithm for mean field games with optimal stopping and absorption. *ESAIM: Math. Model. Numer. Anal.* **2023**, *57*, 953–990. [[CrossRef](#)]
31. Monderer, D.; Shapley, L.S. Potential games. *Games Econ.* **1996**, *14*, 124–143. [[CrossRef](#)]
32. Nash, J.F., Jr. Equilibrium points in n-person games. *Proc. Natl. Acad. Sci. USA* **1950**, *36*, 48–49. [[CrossRef](#)]
33. Thiran, G.; Stupia, I.; Vandendorpe, L. Best response dynamics convergence for generalized nash equilibrium problems: An opportunity for autonomous multiple access design in federated learning. *IEEE Internet Things J.* **2024**, *11*, 18463–18482. [[CrossRef](#)]
34. Kim, Y.; Khir, R.; Lee, S. Enhancing genetic algorithm with explainable artificial intelligence for last-mile routing. *IEEE Trans. Evolutionary Comput.* **2025**, 1–17. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.