remote sensing

MDPI

Article

Multi-Static Radar System Deployment Within a Non-Connected Region Utilising Particle Swarm Optimization

Yi Han [1], Xueting Li [2,*], Tianxian Zhang [1] and Xiaobo Yang [1]

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; 201911012002@std.uestc.edu.cn (Y.H.); txzhang@uestc.edu.cn (T.Z.); yangxb@uestc.edu.cn (X.Y.)
[2] School of Aeronautics and Astronautics, Sichuan University, Chengdu 610065, China
* Correspondence: lixueting@scu.edu.cn

Abstract: This paper is mainly devoted to studying the deployment problem of a multi-static radar system (MSRS) within a non-connected deployment region using multi-objective particle swarm optimization (MOPSO). By modeling and reformulating the problem, it can be represented as a multi-objective mixed integer programming (MOMIP), which eliminates the need for additional constraints. To enhance the algorithm performance, integer variables and continuous ones are treated separately employing multiple velocity formulas. The velocity formulas for integer variables are modified using the sigmoid function and genetic operation, leading to the proposal of two MSRS deployment algorithms, namely MOPSO-Sigmoid and MOPSO-Gene. To evaluate the performance of the proposed algorithms, they are compared with two existing MOPSO-based algorithms. The first algorithm is the MSRS deployment algorithm for the non-connected deployment region that addresses the additional constraint problem model. The second algorithm is based on an existing conventional MOPSO algorithm and addresses the equivalent MOMIP problem model. A numerical study demonstrates that MOPSO-Sigmoid and MOPSO-Gene exhibit promising efficiency and effectiveness.

Keywords: multi-static radar system (MSRS) deployment; non-connected deployment region; multi-objective particle swarm optimization (MOPSO); multi-objective segmented decision variable problem (MOSDVP); multi-objective mixed integer programming (MOMIP); multiple velocity formula

check for updates

Citation: Han, Y.; Li, X.; Zhang, T.; Yang, X. Multi-Static Radar System Deployment Within a Non-Connected Region Utilising Particle Swarm Optimization. *Remote Sens.* **2024**, *16*, 4004. https://doi.org/10.3390/rs16214004

Academic Editor: Michael Obland

Received: 2 August 2024
Revised: 9 October 2024
Accepted: 23 October 2024
Published: 28 October 2024

1. Introduction

In recent years, there has been a growing interest among researchers in the field of multi-static radar system (MSRS) [1–3]. In comparison to conventional mono-static radar, MSRS has the potential to enhance performance, yet it also introduces a multitude of new challenges. Among these, the issue of node deployment has attracted the attention of many researchers [4–6]. The computational burden associated with the optimal node deployment criteria for MSRS is escalating, due to the introduction of new applications, such as advanced waveform design [7], novel tracking technology [8], advanced intelligent electronic countermeasure [9], and remote sensing [10,11]. In order to address the computational challenge, heuristic algorithms like particle swarm optimization (PSO) have been employed in many studies on radar or sensor deployment [12–17]. On the other hand, in practical applications, it is often necessary to consider multiple conflicting performance indicators simultaneously. Consequently, some researchers have constructed unconstrained multi-objective optimization problem models for the MSRS deployment problem and solved them with multi-objective particle swarm optimization (MOPSO) and its variants [18,19]. Nevertheless, the aforementioned works are predicated on the assumption that the MSRS deployment region is connected.

Due to geographical constraints, meteorological conditions, special requirements, or other reasons, it is not uncommon for the deployment region of MSRS to be complex and scattered. It can be said that the deployment region of MSRS is often divided into several non-connected subregions by the regions that are unsuitable for deployment in practice, such as rivers, steep hills, and so on. The non-connected deployment region of MSRS could be treated as a constrained multi-objective problem (MOP) and often solved using constraint handling mechanisms [20,21].

However, the introduction of constraint handling mechanism may result in an increase in algorithm complexity and a concomitant rise in computing cost when attempting to solve the MSRS deployment problem, particularly when MOPSO is employed. To the authors' best knowledge, at present, this problem has not been paid enough attention by researchers. Consequently, further research is required to investigate MSRS deployment within non-connected deployment region. This is indeed the main topic of this paper.

1.1. Overview

The following literature overview contains three main areas of research: **(1)** the MSRS deployment, **(2)** the constraint handling mechanism with PSO, and **(3)** the solving of mixed-integer programming (MIP) solution with PSO.

1.1.1. MSRS Deployment

The deployment of nodes in MSRS has been demonstrated to enhance its performance. The optimal location for each node is determined through traversal while adhering to strict criteria [4,5]. A algorithm for MSRS deployment is proposed, utilizing a sequential exhaustive enumeration approach. However, this method is susceptible to significant computational constraints [6]. The computational challenge has been addressed in numerous studies by employing PSO, which has demonstrated excellent performance in solving continuous variable problems [12–14,16]. Based on the aforementioned works, an MSRS deployment problem with multiple conflicting performance indicators is investigated [18,19]. Another work is devoted to the iteration convergence criterion [20]. Nevertheless, these works are all based on the assumption that the MSRS deployment region is connected. There is a paucity of relatively systematic studies on the case of non-connected deployment regions.

1.1.2. Constraint Handling Mechanisms with PSO

Many studies have focused on the constraints handling mechanism, which can be broadly classified into three categories. **(1)** Modifying the ordering of solutions [22], which changes the ordering rule of candidate solutions so that feasible solutions are identified. Although intuitive, this approach may reduce the efficiency of the algorithm when solving complex problems. **(2)** Modifying the constraints [23], which regards constraints as additional objectives. This mechanism could be inconvenient when the number of constraints considered is large. **(3)** Modifying the objective function, which is also known as the penalty function mechanism. Among these mechanisms, the penalty function is the most direct and commonly used, due to the universality of PSO towards optimization problems.

The fundamental principle underlying the penalty function mechanism is punishing the infeasible solutions during the calculation of objective functions. The penalty function mechanism can be subclassified into two categories, static [24] and adaptive [25,26]. The former approach is relatively simpler, which introduces a fixed penalty value to all infeasible solutions when calculating the objective functions. In contrast, the latter approach modifies the penalty function continuously throughout the iteration, based on the information gathered from the search process. Although the modified criteria vary, the most commonly used one is to determine the penalty function according to the constraint violation of each infeasible solution. Nevertheless, the aforementioned methodologies inevitably result in increased algorithm complexity when compared to unconstrained versions.

1.1.3. MIP Solution with PSO

To date, only a limited number of studies have focused on the application of multi-objective mixed integer programming (MOMIP) with PSO. A real-world application of MOMIP, multi-objective portfolio optimization is studied in [27], which is further transformed into a standard MOP by assuming dependence among integer variables and continuous ones on their investment problem. This method is ingenious, but it is not appropriate for general MOMIP like MSRS deployment.

Meanwhile, many methods have been proposed for single-objective mixed-integer programming (SOMIP). Considering that some methods for SOMIP could offer valuable insights for the solving MOMIP, we also investigated the heuristic algorithms based on the SOMIP solving methodologies, as follows. The most enlightening and most commonly used methodology for solving SOMIP is to treat the integer variables as continuous during the iterations, and to obtain the true value of each integer variable by rounding operation [28,29]. While these methods are capable of addressing the SOMIP, their algorithm performance is constrained due to the inability to fully leverage the characteristics of integer variables inherent in the problem. On the other hand, some researchers have attempted to integrate some specific mechanisms, such as the branch-and-bound method, into PSO to develop an SOMIP customized algorithm [30]. However, it results in the optimization objective being recalculated multiple times within each iteration, which is a challenging solution for MOMIP, particularly in cases where the computational objective functions are complex, such as the MSRS deployment problem. Therefore, the methods employed for SOMIP cannot be directly employed to the domain of MOMIP.

In summary, according to the available literature, the problem of solving MOMIP with PSO remains unresolved.

1.2. Motivation

The motivation of this paper mainly stems from engineering practice. In practical application, the MSRS deployment region is often divided into several non-connected subregions due to geographical limitations, meteorological conditions, or other reasons. For further details, please refer to Figure 1 and the context provided in Section 2.1. These deployment problems, which we refer to as MSRS deployment within a non-connected deployment region, can be properly abstracted and modeled as a typical multi-objective segmented decision variable problem (MOSDVP). As mentioned in our previous work [19,20], given the complexity of the problem at hand, we model the non-connected deployment region as additional constraints and attempt to address it, employing MOPSO with a constraint-handling mechanism to address it. However, MOPSO is demonstrably less efficient and effective than when solving the MSRS deployment problem with a connected deployment region. This phenomenon promptly attracted our attention.

To eliminate the additional constraints associated with the non-connected deployment region, we reformulate this problem into an MOMIP and employ the rounding operation for integer variables to solve this MOMIP. The results indicate that this method remains less efficient than conventional methods. To address this, we modify the velocity formula for integer variables while maintaining those for continuous variables. Finally, we solve the deployment problem in an efficient and effective manner.

1.3. Original Contributions

The original contributions of this paper are as follows.

(1) In order to address the MSRS deploying problem within a non-connected deployment region, we conducted further investigation and extended the problem model. Rather than introducing additional constraints, we reformulated the problem into an MOMIP by dividing the deployment region and introducing integer variables. This allows the full potential of the matching algorithm to be realized.

(2) Furthermore, in order to effectively address the attached integer variables, we propose two MSRS deployment algorithms for the non-connected deployment region. These algorithms employ two altering velocity formulas for integer variables based on the sigmoid function and genetic operation. The results of the comparative experiment results demonstrate that the proposed algorithms exhibit a statistically significant superiority

over traditional algorithms. Moreover, these proposed algorithms can also be applied in general MOMIP.

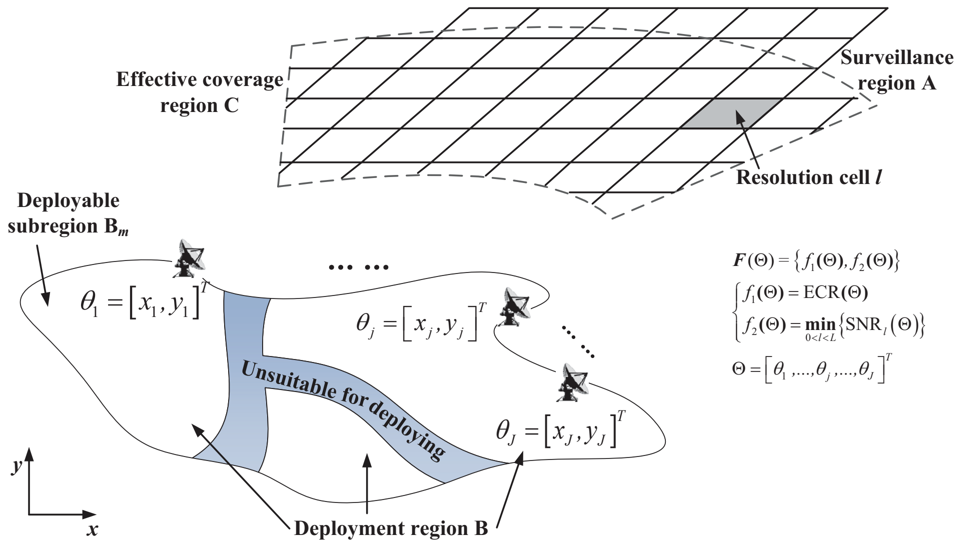
Effective coverage region C
Surveillance region A
Resolution cell *l*
Deployable subregion $\mathbf{B}_m$
$\theta_1 = [x_1, y_1]^T$
$\theta_j = [x_j, y_j]^T$
$\theta_J = [x_J, y_J]^T$
Unsuitable for deploying
Deployment region B
$\boldsymbol{F}(\Theta) = \{f_1(\Theta), f_2(\Theta)\}$
$\begin{cases} f_1(\Theta) = \mathrm{ECR}(\Theta) \\ f_2(\Theta) = \min\limits_{0<l<L}\{\mathrm{SNR}_l(\Theta)\} \end{cases}$
$\Theta = [\theta_1, \ldots, \theta_j, \ldots, \theta_J]^T$
y
x

Figure 1. Sketch of the MSRS deployment problem within a non-connected deployment region.

1.4. Organization

The rest of this paper is organized as follows. Section 2 presents the problem formulation of MSRS deployment within a non-connected deployment region. This formulation includes two variations: one with additional constraints and another with additional integer variables. In Section 3, the customized algorithms for MSRS deployment within a non-connected region problem are presented. Section 4 presents the results of the proposed algorithms on a benchmark test set and on MSRS deployment within a non-connected deployment region. Section 5 presents a conclusion.

2. Problem Formulation

The following section outlines the methodology employed to establish and extend the mathematical model of an MSRS deployment problem within a non-connected region, to align with the mathematical solving tool adopted.

To fully consider this problem and to avoid overcomplicating it, we focus on the radar detection mission in our study. For the sake of convenience in expression, this problem will be studied in two-dimensional space. It is also worth noting that the model and methodology proposed in this paper are readily extended to other radar application areas by replacing the optimization objectives.

2.1. Mathematical Modeling

In light of the fact that MSRS is designed to monitor a surveillance region with potential targets, we divide the surveillance region **A** into L resolution cells. The optimal deployment scheme can then be defined as the scheme with the overall best detection probability P_d performance of MSRS among all resolution cells.

The non-connected deployment region **B** can be defined as a combination of non-connected deployable subregions, i.e., $\mathbf{B} = \bigcup \mathbf{B}_m$, where J widely separated nodes are deployed within.

Moreover, the effective coverage region **C** is defined as the region where the detection probability of the lth resolution cell $P_d(l)$ is greater than the detection probability threshold of the MSRS P_{dt}. The sketch of this optimization problem is shown as Figure 1.

In light of the aforementioned considerations, the overall optimal P_d performance scheme for the MSRS should simultaneously satisfy both of the following criteria: **(1)** the first objective is to ensure the most reliable monitoring of all resolution cells, which means maximizing the area of the effective coverage region, **C**; **(2)** the second objective is to ensure the most even distribution of P_d among all resolution cells.

In practice, the optimal performance objective can be represented by effective cover rate (ECR), which is the coverage ratio of the area of effective coverage region to the surveillance region. The computational formulas for ECR can be expressed as:

$$\mathrm{ECR}(\Theta) = \frac{area(\mathbf{C}(\Theta) \bigcap \mathbf{A})}{area(\mathbf{A})}, \tag{1}$$

where $area(\bullet)$ represents the area of this region and Θ represents the deployment scheme.

To ascertain the specific mathematical expressions of the evenly distributed performance objective, it was necessary to conduct a study of the arbitrary resolution cell and the value of P_d. By using a monopulse square rate detector, the detection probability of the lth resolution cell $P_d(l)$ can be expressed as [31,32]:

$$P_d(l) = Q_{J\times J}\left(\sqrt{2\mathrm{SNR}_l}, \sqrt{2\gamma_T}\right), \tag{2}$$

where $Q_u(\alpha, t)$ represents the Marcum Q function (see [33] for detail definition), SNR_l represents the ratio of total signal energy to noise power (SNR) of the lth resolution cell, and γ_T represents the detection threshold of MSRS.

Given that the threshold γ_T is set according to a predetermined false alarm probability, $P_d(l)$ is solely dependent on SNR_l, which is to say, the objective is to identify the scheme that results the most evenly distributed SNR_l. To facilitate the calculation, a uniform distribution can be translated into a focus on the minimum. Hence, the objective of distributing performance evenly can be proposed as a minimum SNR (minSNR). The computational formula can be given as:

$$\mathrm{minSNR}(\Theta) = \min_{0<l<L}\{\mathrm{SNR}_l(\Theta)\}. \tag{3}$$

It is evident that, whilst the total energy remains constant, an increase in the minSNR results in a more uniform energy distribution. Therefore, the uniform distribution of energy can be converted to maximize the minSNR.

Furthermore, by assuming that the transceiver channels of each node of MSRS are orthogonal, SNR_l can be expressed as the sum of the signal energy divided by the noise power of each transceiver channel. By jointly processing the received signals from all $J \times J$ transceiver channels, SNR_l can be expressed as:

$$\mathrm{SNR}_l = \sum_{i=1}^{J}\sum_{j=1}^{J}\mathrm{SNR}_l(i,j), \tag{4}$$

where J is the number of nodes of MSRS. By assuming that all J nodes of MSRS have the same parameters, for a typical targets whose radar cross section is σ, the SNR of the transceiver channel for transmitting from ith node and receiving at jth node $\mathrm{SNR}_l(i,j)$ can be expressed as [19,34]:

$$\mathrm{SNR}_l(i,j) = D_0\frac{\sigma^{ij}R_{\max}^4}{\sigma\left(R_{i,l}R_{j,l}\right)^2}, \tag{5}$$

where D_0 is the detectability factor, $R_{\max}$ is the maximum detection range when radar node works independently, σ^{ij} is the bistatic radar cross section of transceiver channel for transmitting from ith node and receiving at jth node, and $R_{i,l}$ is the distance between the ith node and the lth resolution cell.

Given that D_0 is defined in accordance with the typical characteristics of the target and that $R_{\max}$ can be regarded as a constant in the MSRS deployment problem, the performance of MSRS is determined by $\sigma^{ij}/_{\sigma}$, $R_{i,l}$ and $R_{j,l}$. In other words, the optimal solution to this problem depends on the position of each node of MSRS.

Consequently, the decision vector Θ of this problem is defined by the location of each node, represented by the coordinates (x, y). Meanwhile, it is evident that the objective of MSRS deployment is to identify the MSRS deployment scheme that maximizes the aforementioned objectives. Hence, the form of MSRS deployment within a non-connected deployment region problem is shown as:

$$\begin{aligned} \text{maximize } \mathbf{F}(\Theta) &= \{f_1(\Theta), f_2(\Theta)\}, \\ &= \{\mathrm{ECR}(\Theta), \mathrm{minSNR}(\Theta)\}, \\ \Theta &= \left(x_1, y_1, \ldots, x_J, y_J\right)^T \\ s.t.\ \ \left(x_j, y_j\right) &\in \mathrm{B} = \bigcup \mathrm{B}_m,\ j = 1, \cdots, J. \end{aligned} \tag{6}$$

where $(\bullet)^T$ is the transpose operation.

Problem (6) is evidently a constrained multi-objective optimization problem, comprising a constraint and two conflicting optimization objective functions. It is therefore impossible to identify a single solution that can simultaneously optimize all of the objective functions.

Our previous work have demonstrated that the computational complexity of the optimization objectives for this problem exceeds that of traditional exact solution algorithms [19,20]. Therefore, we adopt the MOPSO, a well-tested heuristic algorithm architecture, to address this deployment problem.

However, our previous works did not study the case with a non-connected deployment region in sufficient depth, merely offering an initial conceptual solution. The numerical simulations have demonstrated that the problem (6) constrains the performance of MOPSO, resulting in challenges in calculation accuracy and computational cost. Therefore, it is necessary to abstract the problem into a mathematical model and extend it into a more appropriate form.

2.2. Model Extending

It is evident that the defining characteristic of problem (6) lies in its non-connected solution space, i.e., the deployment region **B** is non-connected. Furthermore, it can be considered a problem with several additional constraints compared to a deployment problem within a connected deployment region. So we abbreviate this model as the additional constraint problem model (AC model).

AC model can be abstracted as:

$$\begin{aligned} &\text{maximize } \mathbf{F}(\Theta) = [f_1(\Theta), f_2(\Theta)]^T \\ &\text{subject to : } \ \Theta \in \Omega, \end{aligned} \tag{7}$$

where Θ represents the N-dimensional decision vector, where N is the number of decision variables (in our case $N = 2 \times J$). $\Omega \subset R^N$ is non-empty and non-connected, and it represents the solution space, i.e., deployment region. In mathematics, a problem with a non-connected solution space Ω, can be collectively referred to as MOSDVP, where some or all of the decision variables exhibit a segmented value range.

Accordingly, the decision variables of MOSDVP can be further divided into two categories according to whether value range is segmented or connected. This can be expressed as $\Theta = [\Theta_S; \Theta_C]^T$, where Θ_S is the vector of segmented variables and Θ_C is the vector of connected variables. They represent non-connected dimensions and connected dimensions, respectively, in the deployment region.

For the sake of brevity, the subsequent content is all based on the unconstrained MOSDVPs, i.e., problems discussed below contain no constraints other than those caused by the segmented variables. The conclusions are also applicable to constrained MOSDVPs by introducing a proper constraint handling mechanism.

It can be argued that an unconstrained MOSDVP can be treated as a generic constrained MOP, which contains only linear inequality constraints. This implies that an MOSDVP can be refined to

$$\begin{aligned} &\text{maximize } \mathbf{F}(\boldsymbol{\Theta}_{\boldsymbol{S}}, \boldsymbol{\Theta}_{\boldsymbol{C}}) \\ &s.t. \begin{cases} \boldsymbol{\Theta} \in [lb_1, ub_1] \times \cdots \times [lb_N, ub_N] \subseteq R^N \\ \theta_s \in [lb_s^1, ub_s^1] \cup \cdots \cup \left[lb_s^{K_s}, ub_s^{K_s}\right], \forall \theta_s \in \boldsymbol{\Theta}_{\boldsymbol{S}} \end{cases} \end{aligned} \tag{8}$$

where for any segmented variable $\theta_s \in \boldsymbol{\Theta}_{\boldsymbol{S}}$, its value range is divided into K_s subintervals, and lb is the interval lower bound and ub is the interval upper bound.

To circumvent these additional constraints, we propose an alternative approach to determine each segmented variable θ_s by employing a combination of a connected variable $\hat{\theta}_s$ and an additional integer variable z_s. This approach can be expressed as:

$$\begin{aligned} &\theta_s = lb_{z_s} + \hat{\theta}_s \times \|ub_{z_s} - lb_{z_s}\|_2 \\ &\begin{cases} \hat{\theta}_s \in [0,1] \\ z_s \in \{1, \cdots, K_s\} \end{cases} \end{aligned} \tag{9}$$

where $\|\bullet\|_2$ represents the the length of the corresponding interval. This approach can be interpreted as first determining which segment θ_s is within, using z_s, and then determining the specific value in this segment using $\hat{\theta}_s$.

Further, substituting (9) into problem (8), we can obtain the equivalent model for MOSDVP as:

$$\begin{aligned} &\text{maximize } \mathbf{F}\left(\hat{\Theta}_S, \Theta_C, Z\right) \\ &s.t. \begin{cases} \Theta_C \in [lb_1, ub_1] \times \cdots \times \left[lb_{N_C}, ub_{N_C}\right] \subseteq R^{N_C} \\ \hat{\Theta}_S \in [0,1] \times \cdots \times [0,1] \subseteq R^{N_S} \\ Z_S \in \{1, \cdots, K_1\} \times \cdots \times \left\{1, \cdots, K_{N_S}\right\} \subseteq Z^{N_S} \end{cases} \end{aligned} \tag{10}$$

where $\hat{\Theta}_S$ is the vector that consists of θ_ss, Z_S is the vector that consists of z_ss, while N_C and N_S represent the number of continuous variables and segmented variables, respectively. Obviously, this problem model (10) contains only connected variables (i.e., $\theta \in [\hat{\Theta}_S; \Theta_C]^T$) and integer variables (i.e., $z \in Z$), which is an MOMIP. Hence, we abbreviate this extended model as the MOMIP model.

3. MOPSO-Based Deployment Algorithm

This section mainly concerns the customized algorithm for MSRS deployment within a non-connected region problem. Without loss of generality, we utilize the crowding distance based MOPSO (MOPSO-CD) [35], a well-established and widely utilized MOPSO variant, as the fundamental algorithm framework for the subsequent customized algorithms. To the authors' best knowledge, the methodologies proposed in this paper can also be applied to any MOPSO-based algorithms.

Firstly, the basic concepts related to the MOPSO-CD algorithm are defined as follows:

MOPSO adopts the Pareto optimality criterion, as outlined in [36], which involves identifying a set of optimal trade-off solutions rather than a single optimal solution. The set of all Pareto optimal tradeoff solutions is designated as the Pareto set PS. The set of optimization objective function vectors obtained by all the solutions from PS is referred to as the Pareto front PF.

MOPSO, as a quintessential population-based heuristic algorithm, explores the solution space by changing the position of candidate solutions (also known as particles) through iterations. In particular, P particles are randomly distributed throughout the solution space as the initial solution. Subsequently, the location of each particle is gradually adjusted by combining its current position (inertia component), its own experience (cognitive component), and the optimal solution found by other particles (social component). For original continuous variables, the velocity formula for particle p at iteration t is as:

$$\begin{aligned} v_n^p(t+1) = w \times v_n^p(t) + c_1 \times r_1 \times \left(pb_n^p(t) - \theta_n^p(t)\right) \\ + c_2 \times r_2 \times \left(gb_n(t) - \theta_n^p(t)\right), \end{aligned} \tag{11}$$

$$\theta_n^p(t+1) = \theta_n^p(t) + v_n^p(t+1), \tag{12}$$

where $x_n^p(t)$ represents the value of the nth decision variable of the particle p at the iteration t. w is the inertia weight and c is the acceleration constant. The random real value r is uniformly distributed in the interval $[0,1]$. pb^p is the individual best solution of the particle p (also termed as personal best or personal leader), and gb is the best solution found by the entire swarm (also known as global best or global leader).

At each iteration, the trajectory of each particle is recorded, and the superior solutions found up to that point are used to create and update the *archive*. In other words, the superior solutions identified by the entire swarm are stored in the *archive*. Hence, upon the fulfillment of the termination condition, the optimal trade-off solutions discovered by the algorithm are preserved within the *archive*.

The definition of superior solutions, the selection rules for gb and pb, and the release method for inferior solutions vary across different MOPSO variants. In general, the fundamental difference lies in the definition of the superior solution and the ranking of candidate solutions. In this paper, the standard for evaluating the quality of solutions is the combination of Pareto dominance and crowding distance(CD).

Which is to say the solutions are sorted into two steps.

First, the non-dominated solutions that outperform all remaining solutions in at least one optimization objective are identified and ranked above other solutions. Second, among these non-dominated solutions, they are arranged in descending order of CD.

This mechanism has the potential to reduce the clustering density of solutions, thereby reducing the probability of falling into the local PS.

The overall algorithm is presented in Algorithm 1. The two major components, namely **(1)** the problem transformation for MSRS deployment within a non-connected region and **(2)** the modification of velocity formulas for integer variables, are described in Section 3.1 and Section 3.2, respectively. Subsequently, the extant MSRS deployment algorithms for comparing are described in Section 3.3.

Algorithm 1: MSRS Deployment algorithm within non-connected region.

- w is the inertia weight
- c is the acceleration constant
- T_{max} is the total iteration number
- N_P is the number of particles
- N_C is the number of continuous variables
- N_S is the number of segmented variables
- N_{bin} is the number of binary variables
- $\Theta^p(t)$ is the position of the pth particle at iteration t
- Θ_S is the vector of segmented variables
- Θ_C is the vector of connected variables

```
Step 0: Problem transformation
for $s = 1, 2, \ldots, N_S$ do
    Map segmented variable $\theta_s$ to a combination of integer variable $z_s$ and a
    connected variable $\hat{\theta}_s \in [0,1]$ using (9);
    Encode $z_s$ into a linear combination of binary variables $z_{\mathrm{bin}}$ by means of binary
    encoding;
end
Step 1: Initialization
for $p = 1, 2, \ldots, N_P$ do
    for $n = 1, 2, \ldots, N_C + N_S + N_{\mathrm{bin}}$ do
        Randomly initialize particle position $\Theta^p(0)$ and the individual best
        solution $pb^p = \Theta^p(0)$;
        Initialize $v_n^p(0) = 0$;
    end
    Calculate optimization objectives $\mathbf{F}(\Theta^p(0))$;
    Initialize archive as the non-dominated solutions in initialized solutions and
    global best solution $gb$ as randomly selected from archive;
end
for $t = 1, 2, \ldots, T_{max}$ do
    Step 2: Sort the non-dominated solutions in archive in descending CD;
    Step 3: Particle position update with modified velocity formula
    For MOPSO-Sigmoid, follow the Algorithm 2 and for MOPSO-Gene, follow
    the Algorithm 3;
    Step 4: Perform mutation operator proposed in [37];
    Step 5: Boundaries check for each dimension;
    for $p = 1, 2, \ldots, N_P$ do
        Step 6: Calculate $\boldsymbol{F}(\Theta^p(t))$ ;
        Step 7: Replace $pb^p$ with the non-dominated solutions of $(pb^p \bigcup \Theta^p(t))$;
        Step 8: Replace archive with the non-dominated solutions of
        $(archive \bigcup \Theta^p(t))$;
    end
    Step 9: Sort the solutions in archive in descending CD and randomly select the
    $gb$ for each particle from a specified top portion of the sorted archive;
end
Step 10: Solution output;
For each solution in archive, calculate $\theta_s$ according to $z_{\mathrm{bin}}$ and $\hat{\theta}_s$ using (9);
Step 11: Output the final solutions $\Theta = [\Theta_S; \Theta_C]^T$ in archive;
```

3.1. Problem Transformation for MOMIP Model

As mentioned in Section 2, the deployment of MSRS within a non-connected region can be transformed into an MOMIP like (10). In this context, the integer variables distinguish this problem from a standard MOP. Hence, the most critical issue to be addressed is the handling of integer variables.

To handle integer variables, each integer variable z is encoded into a linear combination of binary variables z_{bin}s by means of binary encoding and concatenated together by order. For further information, please refer to the following citations: [38–41]. Hence, the problem model (10) is transformed into a multi-objective mixed binary programming model (MOMBP model) as:

$$\begin{aligned}&\text{maximize } \mathbf{F}\left(\hat{\Theta}_{\text{S}}, \Theta_{\text{C}}, Z_{\text{bin}}\right)\\&s.t.\begin{cases}\Theta_{\text{C}} \in [lb_1, ub_1] \times \cdots \times \left[lb_{N_C}, ub_{N_C}\right] \subseteq R^{N_C}\\\hat{\Theta}_{\text{S}} \in [0,1] \times \cdots \times [0,1] \subseteq R^{N_S}\\Z_{\text{bin}} \in \{0,1\} \times \cdots \times \{0,1\} \subseteq Z^{N_{\text{bin}}}\end{cases}\end{aligned} \tag{13}$$

where z_{bin} represents a binary-encoded integer variable, which is henceforth referred to as binary variable. Meanwhile, Z_{bin} is the vector of z_{bin}, $N_{\text{bin}} = \sum_s \lceil \log_2(K_s) \rceil$ is the number of binary variables and $\lceil \bullet \rceil$ represents the round up operator.

In comparison to the standard MOP, the MOMBP model exhibits a significant distinction in its binary variables. Given the inability of generic MOPSO to solve binary variables, we proceeded to modify the algorithm.

3.2. Velocity Formulas for Integer Variables

The primary challenge in solving the MOMBP model is that the generic velocity formulas for particle motion, i.e., (11) and (12), are designed for continuous variables and therefore not directly applicable to discrete variables.

However, for MOPSO, it was observed that the motion of each variable is independent, which implies that the projections of particle trajectories in different dimensions are independent of one another. In other words, the velocity formulas for binary variables can be modified to enhance variable adaptation, while the velocity formulas for continuous variables adhere to the generic formulas. This methodology can be designated as the multiple velocity formulas. The following two multiple velocity formula-based MSRS deployment algorithms are proposed.

3.2.1. MSRS Deployment Algorithm Based on Sigmoid Function (MOPSO-Sigmoid)

The natural number operations utilized in the generic position iteration Formula (12) are unsuitable for binary variables. However, the Boolean operations are specifically designed for binary variables. Consequently, the inversion operator of a Boolean operation is employed to calculate the position iteration of the binary variable z_{bin} in the MOMBP model (13). While the velocity v in (11) is normalized and taken as a probability, it is used to control whether the value of the binary variable z_{bin} changes. The detailed methodology is as follows:

Firstly, in order to maintain the group behavior methodology of PSO, the velocity iteration formula of z_{bin} essentially remains the same as that given in (11), with the addition of an additional normalization operator. Among the existing velocity normalization methods in PSO, the sigmoid function is the most widely used because it does not require the introduction of additional parameters. Hence, this function is employed in this paper. The specific method is to normalize the velocity with the sigmoid function, which can be defined as:

$$\bar{v} = Sigmoid(v) = \frac{1}{1+e^{-v}}, \tag{14}$$

where $\bar{v}$ is the normalized velocity. Obviously, the normalized velocity $\bar{v} \in (0,1)$.

Further, the position iteration Formula (12) of binary variable motion can be replaced by:

$$z_n^p(t+1) = \begin{cases} \bar{z}_n^p(t), & if \;\; rand < \bar{v}_n^p(t+1) \\ z_n^p(t), & else \end{cases} \tag{15}$$

where $\bar{z}$ is the inversion operator and *rand* is a random real number uniformly distributed in $(0,1)$. The detail modified velocity formula is presented in Algorithm 2.

Algorithm 2: The modified dynamics based on the sigmoid function.

```
• $N_C$ is the number of continuous variables
• $N_S$ is the number of segmented variables
• $N_{bin}$ is the number of binary variables
• $t$ is the serial number of this iteration
for $p = 1, 2, \ldots, N_P$ do
    Step 3 (a) For continuous variables $\theta_n \in \hat{\Theta}_S \cup \Theta_C$, calculate the position in the next iteration:
    for $n = 1, 2, \ldots, N_C + N_S$ do
        Update the $v_n^p(t+1)$ and $x_n^p(t+1)$ using (11) and (12), respectively;
    end
    Step 3 (b) For binary variables $z_n \in Z_{bin}$, calculate the position in the next iteration:
    for $n = 1, 2, \ldots, N_{bin}$ do
        Calculate the velocity of the next iteration $v_n^p(t+1)$ using (11);
        Calculate the normalized velocity $\bar{v}_n^p(t+1)$ using (14);
        if $rand < \bar{v}_n^p(t+1)$ then
            $z_n^p(t+1) = \bar{z}_n^p(t)$;
        else
            $z_n^p(t+1) = z_n^p(t)$;
        end
    end
end
```

In summary, this customized algorithm is based on the sigmoid function and normalized velocity, hence it can be named MOPSO-Sigmoid.

3.2.2. MSRS Deployment Algorithm Based on Genetic Operation (MOPSO-Gene)

Given the promising results achieved by genetic algorithms in optimizing binary variables [37], we propose a novel approach combining the genetic operation (including mutation and crossover) with the group behavior methodology of PSO (i.e., (11) and (12)). This integration leads to the development of a modified velocity formula for binary variables motion. The specific details of the mutation operator and crossover operator are provided below.

The mutation operator is primarily implemented through the use of binary mutation, which entails the execution of an inversion operator within a Boolean operation, with the mutation probability designated as p_{mutate}. In this paper, the mutation probability p_{mutate} is proportional to the inertia weight w in (11). This proportionality reflects the inertia component observed in swarm behavior. The mutation operator can be expressed as:

$$z_n^p(t+1) = \bar{z}_n^p(t), \;\; if \; rand < p_{mutate} \tag{16}$$
$$p_{mutate} = \frac{1}{N_P} \times w$$

where N_P represents the number of particles, while *rand* is a random real number uniformly distributed in the interval $(0,1)$.

The crossover operator is to modify the value of z_n^p, which represents the nth binary variable of particle p, to that of the personal leader pb_n^p or the global leader gb_n as specified in (11). The specific crossover object between pb and gb is determined by the ratio of the two acceleration factors c_1 and c_2 in (11). This ratio corresponds to the relative emphasis on individual experience or group experience in swarm behavior. The crossover probability p_{cross}, is set to a relatively large constant. The crossover operator can be expressed as:

$$z_n^p = \begin{cases} z_n^p, & if\ rand_1 \geq p_{cross} \\ pb_n^p, & if\ rand_1 < p_{cross}, rand_2 < \frac{c_1}{c_1+c_2} \\ gb_n, & if\ rand_1 < p_{cross}, rand_2 \geq \frac{c_1}{c_1+c_2} \end{cases} \tag{17}$$

where $rand_1$ and $rand_2$ are random real numbers uniformly distributed in the interval $(0,1)$.

Similar to MOPSO-Sigmoid, this algorithm based on the aforementioned genetic operation is named MOPSO-Gene. The detail modified velocity formula of MOPSO-Gene is shown in Algorithm 3.

Algorithm 3: The modified dynamics based on genetic operation.

- N_C is the number of continuous variables
- N_S is the number of segmented variables
- N_bin is the number of binary variables
- N_P is the number of particles
- w is the crossover probability
- t is the serial number of this iteration
- p_{cross} is the crossover probability

```
for $p = 1, 2, \ldots, N_P$ do
    Step 3 (a) For continuous variables $\theta_n \in \hat{\Theta}_\text{S} \cup \Theta_\text{C}$, calculate the position in the next iteration:
    for $n = 1, 2, \ldots, N_\text{C} + N_\text{S}$ do
        Update the $v_n^p(t+1)$ and $x_n^p(t+1)$ using (11) and (12), respectively;
    end
    Step 3 (b) For binary variables $z_n \in \text{Z}_\text{int}$, calculate the position in the next iteration:
    for $n = 1, 2, \ldots, N_\text{bin}$ do
        Perform mutation operation using (16);
        if $rand < p_{cross}$ then
            Perform crossover operation using (17);
        end
    end
end
```

3.3. *Comparing Algorithms for MOSDVP*

In order to demonstrate the efficacy of the proposed algorithms, two existing MOPSO-CD-based comparing algorithms for the MSRS deployment problem are introduced.

3.3.1. MSRS Deployment Algorithm for AC Model (MOPSO-Penalty)

The first comparing algorithm is to solve the MSRS deployment problem as an AC model (i.e., problem model (8)) directly. The primary challenge lies in the effective handling of the constraint that arises from the segmented decision variables. We utilize a additive adaptive penalty factor[42], which is comprised of two components. **(1)** A constant penalty factor $\tilde{r}_C$ is applied to any infeasible solution. **(2)** An additional adaptive penalty factor $\tilde{r}_S$ is applied according to the distance between the solution and the nearest feasible subregion boundary under the Euclidean norm. Consequently, the modified optimization problem can be expressed as:

$$\begin{aligned}
&\text{maximize } \hat{\mathbf{F}}(\boldsymbol{\Theta}_S, \boldsymbol{\Theta}_C) = \mathbf{F}(\boldsymbol{\Theta}_S, \boldsymbol{\Theta}_C) + \tilde{r} \\
&\tilde{r} = \begin{cases} \tilde{r}_C + \tilde{r}_S, & \mathrm{X} \notin \Omega \\ 0, & \mathrm{X} \in \Omega \end{cases} \\
&s.t. \;\; \boldsymbol{\Theta} \in [lb_1, ub_1] \times \cdots \times [lb_N, ub_N] \subseteq R^N
\end{aligned} \tag{18}$$

where $\hat{\mathbf{F}}$ represents the modified optimization objectives and $\tilde{r}$ represents the penalty factor. The expressions of constant penalty factor $\tilde{r}_C$ and adaptive penalty factor $\tilde{r}_S$ can be given as:

$$\tilde{r}_C = A \tag{19}$$

$$\tilde{r}_S = A \times \sum_{s=1}^{N_S} r_s \tag{20}$$

where A represents the magnitude of the penalty factor chosen according to every particular problem and r_s represents the degree to which θ_s (i.e., the sth element within $\boldsymbol{\Theta}_S$) violates the constraint measured by Euclidean distance. We call this comparing algorithm MOPSO-Penalty for short.

3.3.2. MSRS Deployment Algorithm for MOMIP Model (MOPSO-Round)

Another comparing algorithm is to solve the MSRS deployment problem as an MOMIP model. In contrast to the algorithms proposed in this paper, this algorithm addresses integer variables by employing a rounding operation, as described in [28,43]. This operation substitutes continuous variables for discrete ones, thereby enabling the algorithm to handle integer variables.

In this way, all decision variables can be treated uniformly throughout the iterations, resulting in the conversion of the MOMIP to a standard MOP. When solving a problem (10), the problem model is further reformulated as:

$$\begin{aligned}
&\text{maximize } \mathbf{F}\left(\hat{\Theta}_{\mathrm{S}}, \Theta_{\mathrm{C}}, [\hat{\mathrm{Z}}]\right) \\
&s.t. \begin{cases} \Theta_{\mathrm{C}} \in [lb_1, ub_1] \times \cdots \times \left[lb_{N_C}, ub_{N_C}\right] \subseteq R^{N_{\mathrm{C}}} \\ \hat{\Theta}_{\mathrm{S}} \in [0,1] \times \cdots \times [0,1] \subseteq R^{N_{\mathrm{S}}} \\ \hat{\mathrm{Z}} \in [1, K_1] \times \cdots \times \left[1, K_{N_S}\right] \subseteq R^{N_{\mathrm{S}}} \end{cases}
\end{aligned} \tag{21}$$

where $\hat{\mathrm{Z}}$ is the vector for substituting continuous variables for discrete ones (i.e., Z) and $[\bullet]$ is the rounding operation. Therefore, the core modification of this algorithm is to round substitution variables when calculating the optimization objectives. For this reason, we name it MOPSO-Round for short.

4. Numerical Study

This section presents a comparative analysis of the performance of the proposed customized algorithms for MSRS deployment within a non-connected region. For the sake of clarity, we first present a summary of the four customized algorithms proposed in the previous section, along with a brief overview of their corresponding problem models in Table 1. It is worth reiterating that all three problem models are, in fact, equivalent models, despite their respective scales of variables and complexities of objective functions differing to some extent.

Table 1. Summary of four customized algorithms and main features of their corresponding problem models.

Algorithm Name	**Problem Model**	**Decision Variables (Variable Number)**
MOPSO -Penalty[42]	AC model (18)	$X_S \cup X_C$ $(N_C + N_S)$
MOPSO -Round[28]	MOMIP model (21)	$\hat{X}_S \cup X_C \cup \hat{Z}$ $(N_C + 2 \times N_S)$
MOPSO -Sigmoid	MOMBP model (13)	$\hat{X}_S \cup X_C \cup Z_{\text{int}}$ $\left(N_C + N_S + \sum_s \lceil \log_2(K_s) \rceil\right)$
MOPSO -Gene		

The algorithms proposed in Section 3 are applied to both benchmark test functions and the MSRS deployment problem. A quantitative comparison is conducted among these algorithms.

4.1. Numerical Study on Benchmark Testing Set

Due to the pursuit of problem-independence by PSO, many benchmark test sets are formed to measure the performance of the algorithm. Each benchmark test set comprises a number of test problems exhibiting varying characteristics. The performance of an algorithm is gauged through statistical analysis of the outcomes observed across the test problems within the benchmark test set.

Specific to our work, since the benchmark test set specially designed for MOSDVP is not seen yet, we employ ZDT [44], a widely used benchmark test set, cut off the value range of part of decision variables for each test problem, and obtain the modified ZDT (MZDT) test set.

For convenience, for MZDT1, MZDT2, and MZDT3, the value range of the last 10 decision variables are cut into 2 segments (i.e., $x_j \in [0, 0.3] \cup [0.7, 1], j = 21, \cdots, 30$) and other components remain the same as shown in [38], (Equations (7)–(9)). For MZDT4 and MZDT6, to maintain a consistent number of binary variables, 10, we cut the last 5 variables into 3 segments (i.e., for MZDT4, $x_j \in [-5, -3] \cup [-2, 2] \cup [3, 5], j = 6, \cdots, 10$, and for MZDT6, $x_j \in [0, 0.2] \cup [0.3, 0.7] \cup [0.8, 1], j = 6, \cdots, 10$), and other components remain the same as shown in [38], (Equations (10) and (12)).

It is evident that the PS and PF of MZDT test problems remain unaltered in comparison to those of the ZDT test set.

4.1.1. Performance Metrics

In order to quantitatively assess the performance of these algorithms in different testing problems quantitatively, two performance metrics have been implemented in our research: the Hypervolume (HV) [45] and the Additive Unary Epsilon Indicator (I_ϵ^+) [46]. These performance metrics indicate the performance of an algorithm by evaluating the quality of the final output approximation Pareto front P^t. The following paragraphs provide further details on these performance metrics.

The quality of the approximation Pareto front P^t is determined by the size of the space that dominates the reference point r while being dominated by P^t. It can be defined as:

$$HV \triangleq \text{vol}\left[\left(\bigcup_{\forall u \in P^t} D(u)\right) \cap D^{(-1)}(r)\right] \tag{22}$$

where $D(a)$ represents the space that a dominates, $D^{(-1)}(a)$ represents the space that dominates a, and vol($\bullet$) represents the Lebesgue measure. A larger HV indicates a better approximation Pareto front.

I_ε^+ can be defined as:

$$I_\varepsilon^+ \triangleq \min_\varepsilon \{\exists u \in P^t | u \leq_{\varepsilon+} r\}, \tag{23}$$

where $v \leq_{\varepsilon+} r$ means $f_i(v) \leq \varepsilon + f_i(r), i = 1, \ldots, M$. A smaller I_ε^+ indicates a better convergent P^t. Therefore, I_ε^+ can only measures the convergence of the results.

To be fair, only feasible solutions obtained by MOPSO-Penalty are included in the calculation of performance metrics while infeasible ones are not included.

4.1.2. Parameter Settings

The common parameters of the four algorithms are all set to be the same. Parameters in the basic PSO velocity formula, acceleration constants $c_1 = c_2 = 2$, and inertia weight w decrease linearly from 0.8 to 0.4 with the number of iterations and all four algorithms select the global best gb from the top 10% sorted *archive*, as suggested in [35]. The magnitude of penalty factor in MOPSO-Penalty $A = 10$ and for MOPSO-Gene, the Crossover probability p_{cross} is set to 0.9, as suggested in [37].

In terms of computational cost, the number of particles is $N_P = 50$ while the maximum number of iterations is $T_{max} = 1000$. In terms of parameters of performance metrics, while the reference points of HV and I_ε^+ are (11,11) and (0,0), respectively.

4.1.3. Experimental Results and Comparative Analysis

Considering the random exploration nature of heuristic algorithms, each algorithm was executed 100 times independently for each test problem and all the performance metrics were subsequently measured and recorded statistically. In statistical language, hypothesis testing is a method of statistical inference used to determine whether differences between samples (i.e., the results obtained by each algorithm) are due to sampling error (i.e., accidentalia) or to substantive differences. To ascertain the relative superiority of the four algorithms, hypothesis testing was employed. Unfortunately, some of the samples do not conform to the normality assumption. Hence, the Wilcoxon signed ranks test is employed to assess the performance of the algorithms, as proposed in Section 3 of [47].

The four algorithms are compared in pairs. The null hypothesis, H_0, is set as the performance metrics obtained by one algorithm were not better than those obtained by another based on overall results. The alternative assumption, H_1, is that the performance metrics obtained by one algorithm were better than those obtained. At the significance level of $\alpha = 5\%$, statistical conclusions are drawn in Table 2.

In statistics, a small p-value means that the probability of observation under the null hypothesis H_0 is small. That is to say, a smaller p-value indicates a stronger evidence against H_0, and it is statistically obvious to draw such a conclusion. Therefore, the p-values of the Wilcoxon signed ranks test are shown in Table 2. The order of superiority of the four algorithms is as follows: MOPSO-Gene, MOPSO-Sigmoid, MOPSO-Round, and MOPSO-Penalty. This indicates that two velocity formula modification methodologies for integer variables are more effective than the others.

4.2. Numerical Study on MSRS Deployment Problem

In order to further substantiate the efficacy of the proposed algorithm in addressing the actual MSRS deploying problem within the non-connected deployment region, a series of simulation experiments was conducted, the results of which are presented below.

Table 2. The results of Wilcoxon Signed Ranks Test on MZDTs.

Null Hypothesis	***p*-Value**		**Statistical Conclusion**
MOPSO-Gene No better than **MOPSO-Sigmoid**	HV	2.863×10^{-14}	Reject
	I_ϵ^+	6.247×10^{-26}	
MOPSO-Gene No better than **MOPSO-Round**	HV	$< 1 \times 10^{-50}$	Reject
	I_ϵ^+	$< 1 \times 10^{-50}$	
MOPSO-Gene No better than **MOPSO-Penalty**	HV	$< 1 \times 10^{-50}$	Reject
	I_ϵ^+	$< 1 \times 10^{-50}$	
MOPSO-Sigmoid No better than **MOPSO-Penalty**	HV	1.295×10^{-29}	Reject
	I_ϵ^+	2.480×10^{-14}	
MOPSO-Sigmoid No better than **MOPSO-Round**	HV	5.483×10^{-19}	Reject
	I_ϵ^+	$< 1 \times 10^{-50}$	
MOPSO-Round No better than **MOPSO-Penalty**	HV	4.779×10^{-3}	Reject
	I_ϵ^+	4.616×10^{-5}	

4.2.1. Simulation Model

For simplicity, we assume that the nodes of MSRS are deployed within a non-connected deployment region in this simulation. This region **B** is composed of two non-connected rectangular subregions each of size 100 km × 100 km. The treatment of deployment region in a special shape other than rectangular can be found in our previous work [20].

As shown in Figure 2, the surveillance region **A** is also defined as a rectangular area with size of 300 km × 300 km. In order to reduce the computational burden, the surveillance region **A** was divided into 900 square resolution cells of 10 km × 10 km resolution. The deployment region is comprised of two unconnected square deployable subregions. Hence, the specific MSRS deployment problem of this simulation can be expressed as:

$$\begin{aligned} \text{maximize } \mathbf{F}(\Theta) &= \{f_1(\Theta), f_2(\Theta)\}, \\ &= \{\text{ECR}(\Theta), \text{minSNR}(\Theta)\}, \\ \Theta &= (\theta_1, \dots, \theta_J)^T \\ &= (x_1, y_1, \dots, x_J, y_J)^T \\ s.t. &\begin{cases} x_j \in [0,100] \cup [200,300] \\ y_j \in [50,150] \end{cases}, \ j = 1, \cdots, J. \end{aligned} \tag{24}$$

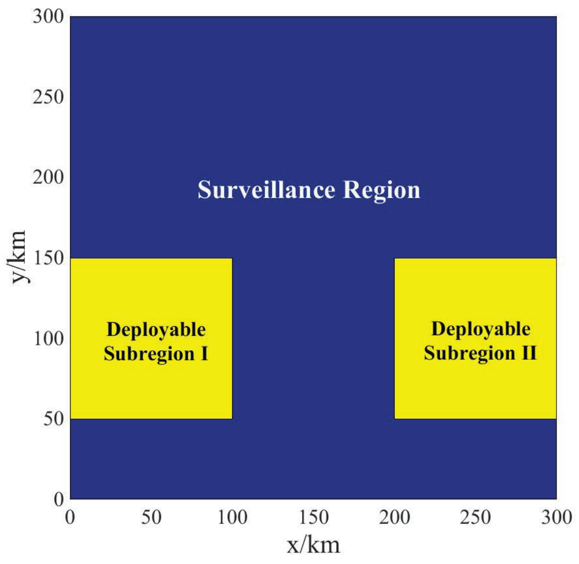

Figure 2. Sketch of surveillance region and deployable subregions.

4.2.2. Parameter Settings

Firstly, the parameters of the MSRS deployment problem are set in accordance with the actual designed performance as follows.

The detection probability threshold value of MSRS is set at $P_{dt} = 0.8$ for each resolution cell, the detectability factor $D_0 = 12.5$dB, and the constant false alarm rate $P_{fa} = 10^{-6}$.

For the typical target of MSRS with a radar cross section (RCS) of 2 m^2, the power range of each single node is set to $R_{max} = 30$km. To assess the resilience of the proposed algorithm to changes in problem size, four test cases were conducted, with the number of nodes, J, set to 4, 6, 8, and 10, respectively.

In regard to the parameters of the algorithms, the magnitude of the penalty factor in the MOPSO-Penalty is set to $A = -100$, which is sufficient to accommodate the maximization problem. The maximum number of iterations is set to $T_{max} = 500$, which is limited by the calculation requirements of the practical application. The remaining parameters of algorithms are consistent with the settings given in Section 4.1 and are not separately stated.

In terms of parameters of performance metrics, the reference points of HV and I_{ϵ}^{+} are (0,0) and (10,10), respectively.

4.2.3. Result Analysis

Similar to the experiment on MZDTs, each algorithm was run 100 times independently for each case. For the sake of brevity, MOPSO-Gene is abbreviated as "Gen.", MOPSO-Sigmoid is abbreviated as "Sig.", MOPSO-Round is abbreviated as "Rou.", and MOPSO-Penalty is abbreviated as "Pen.".

Firstly, we present the solutions obtained by the four algorithms. Due to the large number of solutions, we only plot the best and worst solutions obtained by each algorithm as shown in Figure 3. In the figure, the best solutions obtained by MOPSO-Gene are abbreviated as "Gen.-Bes.", and the worst solutions obtained by it are abbreviated as "Gen.-Wor.". The same abbreviation was applied to the other three groups. From this figure, the following phenomena can be observed:

(1) There is a trade-off between the ECR and the minSNR, which means that no solution can have the best of both performances.

(2) In the majority of cases, with the exception of the MSRS with four nodes, the red and blue curves are situated further to the upper left. Given that this is a maximization problem, the closer the curve is to the top left, the better the set of solutions. This evidence indicates that MOPSO-Gene and MOPSO-Sigmoid yield superior outcomes.

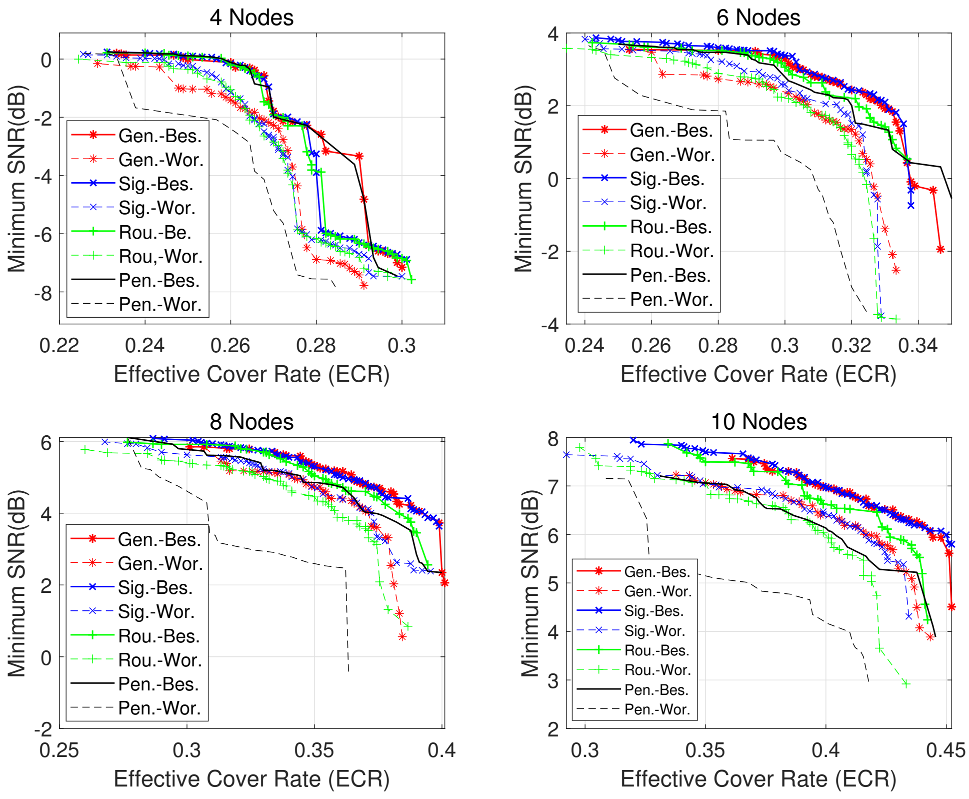

Figure 3. The best and worst of solutions obtained by four algorithms when solving MSRS with 4, 6, 8, and 10 nodes, respectively. The best solutions are drawn with a solid line, and the worst solutions are drawn with a dashed line.

(3) As the MSRS scale increases, the performance of the solutions obtained by each algorithm also improves, although the improvement amplitude varies. In cases of a larger scale, it is evident that the black curve is at a disadvantage, as it is situated in the lower right of the figure. This evidence indicates that MOPSO-Penalty, as a classical algorithm, is deficient in its ability to address complex cases.

(4) In each case, the distance between the two black curves (i.e., Pen.-Bes. and Pen.-Wor.) is significantly larger than the other three. This demonstrates that the disparity in the quality of solutions yielded by MOPSO-Penalty is greater than that of the other three. Consequently, MOPSO-Penalty is deemed to be unreliable.

In terms of computational cost, the box plots of the computational time of the four algorithms for solving the MSRS deployment problem are shown in Figure 4. All computations are performed in an AMD Ryzen 9 3950X operating at 3.5 GHz and 32 GB of RAM operating at 3600 MHz. The figure shows that MOPSO-Penalty takes longer time to converge, while the other three, although slightly different, have roughly the same computational cost.

The observations presented in this paper allow for the preliminary qualitative conclusion that the two algorithms proposed have advantages. However, to provide a more comprehensive quantitative conclusion, we introduce performance metrics in multi-objective optimization and statistical methods. The corresponding performance metrics statistics for each case are presented below. In order to provide a comprehensive evaluation of the performance of each algorithm, the relationship between the average values of HV and I_ϵ^+ on each case and the number of iterations is presented in Figure 5. The performance metrics of the final solutions obtained by the four algorithms are presented in Table 3. The **best values** of each statistical indicator are marked in highlight, the **second values** are in bold and the **worst values** are in red. The statistic analysis results of the Wilcoxon signed ranks test of HV and I_ϵ^+ statistics on four cases are presented in Table 4.

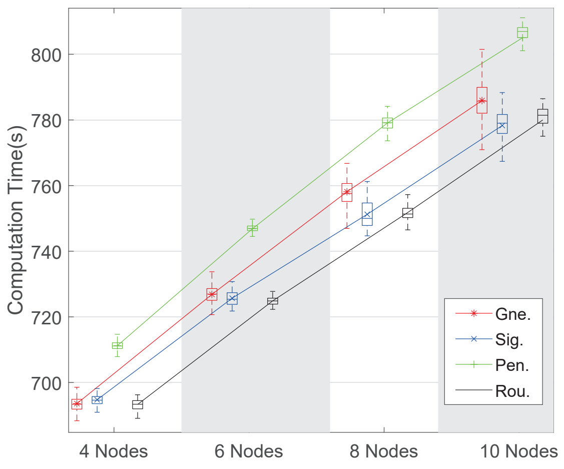

Figure 4. The box plots of computation time when solving the MSRS deployment problem. All computations are performed in an AMD Ryzen 9 3950X operating at 3.5 GHz and 32 GB of RAM operating at 3600 MHz.

Table 3. Performance metrics statistics of the final approximations on the MSRS deployment problem.

			MOPSO -Penalty	**MOPSO -Round**	**MOPSO -Sigmoid**	**MOPSO -Gene**
4 nodes	HV	Bes.	**50.54**	50.37	50.24	**50.72**
		Ave.	48.23	47.91	**48.60**	**49.48**
		Wor.	7.595	42.11	**45.28**	**44.78**
		Var.	59.19	4.869	**1.147**	**0.7427**
	I_{ϵ}^{+}	Bes.	6.053	6.068	**6.004**	**5.966**
		Ave.	6.306	6.259	**6.167**	**6.080**
		Wor.	10.29	6.853	**6.495**	**6.546**
		Var.	0.5777	4.733×10^{-2}	1.123×10^{-2}	7.333×10^{-3}
6 nodes	HV	Bes.	84.64	84.85	**84.97**	**85.15**
		Ave.	78.49	82.49	**83.01**	**83.69**
		Wor.	57.64	76.86	**81.28**	**81.92**
		Var.	57.16	2.905	**0.4514**	**0.3894**
	I_{ϵ}^{+}	Bes.	2.694	2.670	**2.665**	**2.650**
		Ave.	3.296	2.906	**2.856**	**2.792**
		Wor.	5.351	3.462	**3.027**	**2.968**
		Var.	57.16	2.905	**0.4514**	**0.3894**
		Var.	0.5526	2.813×10^{-2}	4.408×10^{-3}	3.848×10^{-3}

Table 3. *Cont.*

			MOPSO -Penalty	**MOPSO -Round**	**MOPSO -Sigmoid**	**MOPSO -Gene**
8 nodes	HV	Bes.	95.99	96.39	**96.87**	**96.87**
		Ave.	90.15	94.42	**95.29**	**96.08**
		Wor.	78.67	92.71	**93.76**	**94.80**
		Var.	18.23	0.4259	**0.3010**	**0.1557**
	I_ϵ^+	Bes.	1.640	1.593	**1.554**	**1.545**
		Ave.	2.201	1.793	**1.703**	**1.634**
		Wor.	3.326	1.961	**1.852**	**1.759**
		Var.	0.1748	4.001×10^{-2}	2.957×10^{-2}	1.557×10^{-2}
10 nodes	HV	Bes.	100.8	102.0	**102.6**	**103.0**
		Ave.	95.75	101.0	**101.6**	**102.4**
		Wor.	91.41	100.1	**100.6**	**101.2**
		Var.	6.234	0.2132	**0.1455**	9.571×10^{-2}
	I_ϵ^+	Bes.	1.105	1.030	**1.021**	**1.013**
		Ave.	1.698	1.202	**1.131**	**1.093**
		Wor.	2.116	1.296	**1.231**	**1.197**
		Var.	5.944×10^{-2}	2.002×10^{-3}	1.390×10^{-3}	9.483×10^{-4}

In light of the aforementioned evidence, the following phenomena are observed:

(1) The comparative results on two performance metrics are essentially consistent, indicating that each algorithm exhibits consistent performance in terms of the two performance metrics across all test problems. Combined with the definition of these performance metrics, this phenomenon implies that each algorithm proposed in this paper is consistent in terms of convergence and uniformity.

(2) With regard to the efficiency of MSRS deployment, it can be observed that MOPSO-Penalty requires a greater number of iterations to converge than the other three algorithms. Meanwhile, there is no significant difference among other three algorithms. As shown in Figure 5, the black line, which represents MOPSO-Penalty, requires the greatest number of iterations to achieve convergence.

This phenomenon demonstrates that MOPSO-Penalty exhibits inferior convergence rates in comparison to the other three algorithms. Moreover, it can be concluded that MOPSO-Penalty, as a typical constraint-handling method, necessitates a greater computational complexity when attempting to resolve the MSRS deployment problem.

(3) In regard to the efficacy of the MSRS deployment, the relevant statistical indicators of the ultimate outcomes are delineated in Table 3. Table 3 has 32 rows, each of which represents a statistical indicator, and it has four columns, each representing an algorithm. For the MOPSO-Gene column, 30 are highlighted. This means that MOPSO-Gene obtains the best results of 93.75% (30 out of 32) statistical indicators on four cases. For MOPSO-

Sigmoid, 22 of which are in bold or in highlight. This indicates that MOPSO-Sigmoid is a top-two algorithm in 96.875% (31 out of 32) of statistical indicators, outperforming other algorithms, besides MOPSO-Gene.

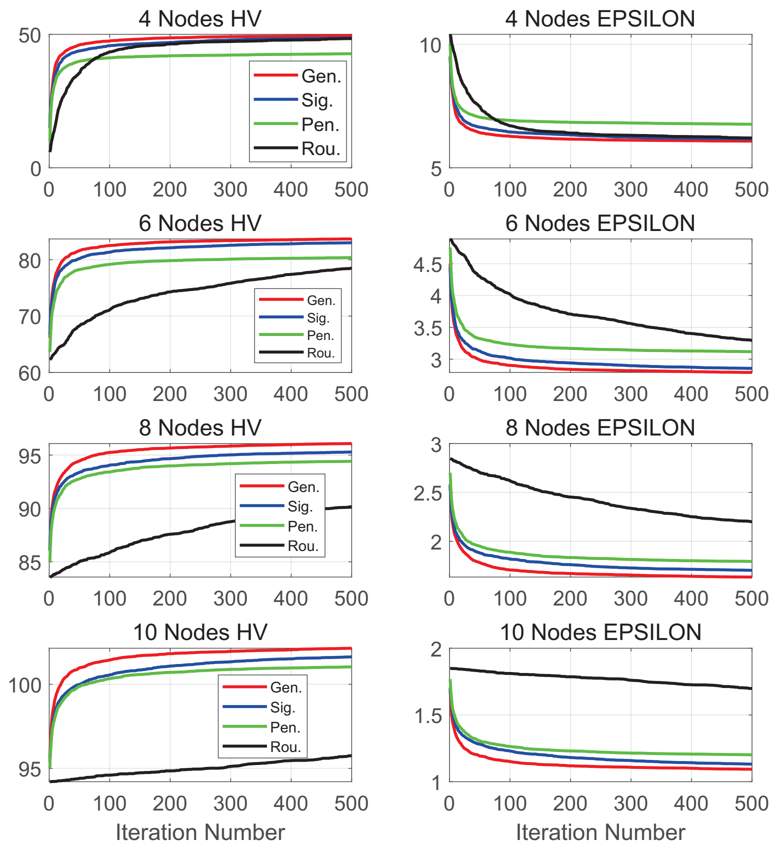

Figure 5. The relationship between average values of each performance metric and number of iterations when solving the MSRS deployment problem. Each row is for a different case (from top to bottom are MSRS with 4, 6, 8, and 10 nodes) and each column is for a different performance metric (from left to right are HV and I_{ϵ}^{+}).

And as indicated in Table 4, the superiority of MOPSO-Gene over MOPSO-Sigmoid is not statistically significant at the 5% level. This suggests that there is no significant difference between the two modifications in terms of their effectiveness in solving the MSRS deployment problem.

With regard to MOPSO-Round and MOPSO-Penalty, MOPSO-Round exhibited smaller variances in HV and I_{ϵ}^{+} and the average level is better than those data obtained by MOPSO-Penalty in Table 3. The results demonstrate that MOPSO-Round outperforms MOPSO-Penalty in addressing the MSRS deployment problem. And according to Table 4, this superiority is statistically significant under the significance level at 5%. This phenomenon indicates that the constraint handling method (i.e., MOPSO-Penalty) is inadequate in comparison to other algorithms when addressing the MSRS deployment problem.

(4) In the terms of robustness, MOPSO-Penalty exhibits a lack of performance, as evidenced by a decline in efficiency and effectiveness as the system scale increases. In the larger scale case (MSRS with 8 or 10 nodes), MOPSO-Penalty is unable to complete convergence within the given computational cost.

In conclusion, the algorithms based on the multiply velocity formula method, (i.e., MOPSO-Gene and MOPSO-Sigmoid) have been demonstrated to have a statistically significant advantage in terms of efficiency and effectiveness.

In contrast, the algorithm based on the constraint handling method (MOPSO-Penalty) has been found to be unsatisfactory when solving the MSRS deployment problem, particularly when the scale of MSRS is large.

Table 4. The Results of Wilcoxon Signed Ranks Test on the MSRS deployment problem.

Null Hypothesis	*p*-Value		Statistical Conclusion
MOPSO-Gene no better than **MOPSO-Sigmoid**	HV	0.2785	**Accept**
	I_{ϵ}^{+}	0.1576	
MOPSO-Gene no better than **MOPSO-Round**	HV	6.557×10^{-5}	Reject
	I_{ϵ}^{+}	6.787×10^{-6}	
MOPSO-Gene no better than **MOPSO- Penalty**	HV	7.431×10^{-5}	Reject
	I_{ϵ}^{+}	3.922×10^{-6}	
MOPSO-Sigmoid no better than **MOPSO-Round**	HV	1.419×10^{-2}	Reject
	I_{ϵ}^{+}	1.803×10^{-2}	
MOPSO-Sigmoid no better than **MOPSO-Penalty**	HV	9.575×10^{-3}	Reject
	I_{ϵ}^{+}	9.649×10^{-3}	
MOPSO-Round no better than **MOPSO-Penalty**	HV	3.922×10^{-2}	Reject
	I_{ϵ}^{+}	1.712×10^{-3}	

5. Conclusions

We devoted this paper to deploying MSRS within a non-connected deployment region utilising MOPSO. To eliminate the necessity for additional constraints, this MSRS deployment problem was reformulated into a MOMIP by dividing the solution space and introducing integer variables. Moreover, two MSRS deployment algorithms, MOPSO-Sigmoid and MOPSO-Gene, are proposed, respectively, based on two altering velocity formulas for integer variables. These formulas are based on the sigmoid function and the genetic operation. A numerical study has demonstrated that the proposed MSRS deployment algorithms exhibit a statistically significant advantage in effectiveness with the same computational cost.

Author Contributions: conceptualization X.L. and T.Z.; methodology, Y.H. and T.Z.; software and validation, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, X.L. and T.Z.; supervision, X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grants 61971109, in part by the GF Science and Technology Special Innovation Zone Project, and in part by the Fundamental Research Funds of Central Universities under Grant ZYGX2020ZB031.

Data Availability Statement: Data are contained within the article. All data in this paper are generated by simulation and the details have been presented in Section 4.

Acknowledgments: We want to appreciate the academic editor and reviewers for their insightful comments which greatly helped us improve the quality of this manuscript. We would also like to extend our gratitude to the production editors and other contributors who helped to facilitate the publication of this article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Baker, C.; Hume, A. Netted radar sensing. *IEEE Aerosp. Electron. Syst. Mag.* **2003**, *18*, 3–6. [CrossRef]
2. Godrich, H.; Haimovich, A.M.; Blum, R.S. Target localization accuracy gain in MIMO radar-based systems. *IEEE Trans. Inf. Theory* **2010**, *56*, 2783–2803. [CrossRef]
3. Javadi, S.H.; Farina, A. Radar networks: A review of features and challenges. *Inf. Fusion* **2020**, *61*, 48–55. [CrossRef]
4. He, Q.; Lehmann, N.H.; Blum, R.S.; Haimovich, A.M. MIMO radar moving target detection in homogeneous clutter. *IEEE Trans. Aerosp. Electron. Syst.* **2010**, *46*, 1290–1301. [CrossRef]
5. Chen, W.J.; Narayanan, R.M. Antenna placement for minimizing target localization error in UWB MIMO noise radar. *IEEE Antennas Wirel. Propag. Lett.* **2011**, *10*, 135–138. [CrossRef]
6. Radmard, M.; Chitgarha, M.M.; Majd, M.N.; Nayebi, M.M. Antenna placement and power allocation optimization in MIMO detection. *IEEE Trans. Aerosp. Electron. Syst.* **2014**, *50*, 1468–1478. [CrossRef]
7. Zhang, T.; Cui, G.; Kong, L.; Yi, W.; Yang, X. Phase-modulated waveform evaluation and selection strategy in compound-Gaussian clutter. *IEEE Trans. Signal Process.* **2012**, *61*, 1143–1148. [CrossRef]
8. Yi, W.; Morelande, M.R.; Kong, L.; Yang, J. A computationally efficient particle filter for multitarget tracking using an independence approximation. *IEEE Trans. Signal Process.* **2012**, *61*, 843–856. [CrossRef]
9. Wang, Y.; Zhang, T.; Kong, L.; Ma, Z. A Stochastic Simulation Optimization based Range Gate Pull-off Jamming Method. *IEEE Trans. Evol. Comput.* **2022**, *27*, 580–594. [CrossRef]
10. Zhu, R.; Yu, D.; Ji, S.; Lu, M. Matching RGB and Infrared Remote Sensing Images with Densely-Connected Convolutional Neural Networks. *Remote Sens.* **2019**, *11*, 2836. [CrossRef]
11. Wu, Z.; Zhao, F.; Zhang, L.; Cao, Y.; Qian, J.; Xu, J.; Yang, L. Fast Frequency-Diverse Radar Imaging Based on Adaptive Sampling Iterative Soft-Thresholding Deep Unfolding Network. *Remote Sens.* **2023**, *15*, 3284. [CrossRef]
12. Yang, Y.; Yi, W.; Zhang, T.; Cui, G.; Kong, L.; Yang, X.; Yang, J. Fast optimal antenna placement for distributed MIMO radar with surveillance performance. *IEEE Signal Process. Lett.* **2015**, *22*, 1955–1959. [CrossRef]
13. Pizzuti, C. Evolutionary computation for community detection in networks: A review. *IEEE Trans. Evol. Comput.* **2017**, *22*, 464–483. [CrossRef]
14. Li, H.; Zhao, G.; Qin, L.; Yang, Y. Design and optimization of a hybrid sensor network for traffic information acquisition. *IEEE Sens. J.* **2019**, *20*, 2132–2144. [CrossRef]
15. Banerjee, B.P.; Raval, S. A Particle Swarm Optimization Based Approach to Pre-tune Programmable Hyperspectral Sensors. *Remote Sens.* **2021**, *13*, 3295. [CrossRef]
16. Houssein, E.H.; Gad, A.G.; Hussain, K.; Suganthan, P.N. Major advances in particle swarm optimization: Theory, analysis, and application. *Swarm Evol. Comput.* **2021**, *63*, 100868. [CrossRef]
17. Chen, T.; Qi, J.; Xu, M.; Zhang, L.; Guo, Y.; Wang, S. Deployment of Remote Sensing Technologies for Effective Traffic Monitoring. *Remote Sens.* **2023**, *15*, 4674. [CrossRef]
18. Pradhan, P.M.; Panda, G. Connectivity constrained wireless sensor deployment using multiobjective evolutionary algorithms and fuzzy decision making. *Ad Hoc Netw.* **2012**, *10*, 1134–1145. [CrossRef]
19. Yang, Y.; Zhang, T.; Yi, W.; Kong, L.; Li, X.; Wang, B.; Yang, X. Deployment of multistatic radar system using multi-objective particle swarm optimisation. *Iet Radar Sonar Navig.* **2018**, *12*, 485–493. [CrossRef]
20. Zhang, T.; Liang, J.; Yang, Y.; Cui, G.; Kong, L.; Yang, X. Antenna deployment method for multistatic radar under the situation of multiple regions for interference. *Signal Process.* **2018**, *143*, 292–297. [CrossRef]
21. Liang, J.; Ban, X.; Yu, K.; Qu, B.; Qiao, K.; Yue, C.; Chen, K.; Tan, K.C. A Survey on Evolutionary Constrained Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2023**, *27*, 201–221. [CrossRef]
22. Runarsson, T.P.; Yao, X. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evol. Comput.* **2000**, *4*, 284–294. [CrossRef]
23. Ray, T.; Singh, H.K.; Isaacs, A.; Smith, W. Infeasibility driven evolutionary algorithm for constrained optimization. In *Constraint-Handling in Evolutionary Optimization*; Springer: Berlin/Heidelberg, Germnay, 2009; pp. 145–165.
24. Homaifar, A.; Qi, C.X.; Lai, S.H. Constrained optimization via genetic algorithms. *Simulation* **1994**, *62*, 242–253. [CrossRef]
25. Farmani, R.; Wright, J.A. Self-adaptive fitness formulation for constrained optimization. *IEEE Trans. Evol. Comput.* **2003**, *7*, 445–455. [CrossRef]
26. Xia, Z.; Liu, Y.; Lu, J.; Cao, J.; Rutkowski, L. Penalty method for constrained distributed quaternion-variable optimization. *IEEE Trans. Cybern.* **2020**, *51*, 5631–5636. [CrossRef]

27. Chen, Y.; Zhou, A.; Das, S. Utilizing dependence among variables in evolutionary algorithms for mixed-integer programming: A case study on multi-objective constrained portfolio optimization. *Swarm Evol. Comput.* **2021**, *66*, 100928. [CrossRef]
28. Juang, C.F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Trans. Syst. Man Cybern. B* **2004**, *34*, 997–1006. [CrossRef]
29. Liao, T.; Socha, K.; de Oca, M.A.M.; Stützle, T.; Dorigo, M. Ant colony optimization for mixed-variable optimization problems. *IEEE Trans. Evol. Comput.* **2013**, *18*, 503–518. [CrossRef]
30. Nema, S.; Goulermas, J.; Sparrow, G.; Cook, P. A hybrid particle swarm branch-and-bound (HPB) optimizer for mixed discrete nonlinear programming. *IEEE Trans. Syst. Man Cybern. A* **2008**, *38*, 1411–1424. [CrossRef]
31. De Maio, A.; Lops, M. Design principles of MIMO radar detectors. *IEEE Trans. Aerosp. Electron. Syst.* **2007**, *43*, 886–898. [CrossRef]
32. Cui, G.; De Maio, A.; Piezzo, M. Performance prediction of the incoherent radar detector for correlated generalized Swerling-chi fluctuating targets. *IEEE Trans. Aerosp. Electron. Syst.* **2013**, *49*, 356–368. [CrossRef]
33. Richards, M.A. *Fundamentals of Radar Signal Processing*; McGraw-Hill Education: New York, NY, USA, 2014.
34. Skolnik, M.I. *Radar Handbook*; McGraw-Hill Education: New York, NY, USA, 2008.
35. Raquel, C.R.; Naval, P.C., Jr. An effective use of crowding distance in multiobjective particle swarm optimization. In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Washington, DC, USA, 25–29 June 2005; pp. 257–264.
36. Miettinen, K. *Nonlinear Multiobjective Optimization*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 12.
37. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
38. Kennedy, J.; Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; IEEE: New York, NY, USA, 1997; Volume 5, pp. 4104–4108.
39. Yang, S.; Wang, M.; Jiao, L. A quantum particle swarm optimization. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), Portland, OR, USA, 19–23 June 2004; IEEE: New York City, NY, USA, 2004; Volume 1, pp. 320–324.
40. Afshinmanesh, F.; Marandi, A.; Rahimi-Kian, A. A novel binary particle swarm optimization method using artificial immune system. In Proceedings of the EUROCON 2005—The International Conference on "Computer as a Tool", Belgrade, Serbia, 21–24 November 2005; IEEE: New York, NY, USA, 2005; Volume 1, pp. 217–220.
41. Pampara, G.; Franken, N.; Engelbrecht, A.P. Combining particle swarm optimisation with angle modulation to solve binary problems. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; IEEE: New York, NY, USA, 2005; Volume 1, pp. 89–96.
42. Gao, W.F.; Yen, G.G.; Liu, S.Y. A dual-population differential evolution with coevolution for constrained optimization. *IEEE Trans. Cybern.* **2014**, *45*, 1108–1121. [CrossRef] [PubMed]
43. dos Santos Coelho, L. An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications. *Reliab. Eng. Sys. Safe.* **2009**, *94*, 830–837. [CrossRef]
44. Zitzler, E.; Deb, K.; Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef]
45. Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.M.; Da Fonseca, V.G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evol. Comput.* **2003**, *7*, 117–132. [CrossRef]
46. Knowles, J.D.; Thiele, L.; Zitzler, E. A tutorial on the performance assessment of stochastic multiobjective optimizers. *Comput. Eng. Netw. Lab. (TIK)* **2006**, *214*. [CrossRef]
47. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]