Spatial area determination problem: Definition and solution method based on Memetic Algorithm

Son Duy Dao, Antoine Mallégol, Patrick Meyer, Mehrdad Mohammadi, Sophie Loyer

Version of Record: https://www.sciencedirect.com/science/article/pii/S1568494622004598
Manuscript_94c5aca404f84a36c50785e3f649d8b3

Spatial Area Determination Problem: Definition and Solution Method Based on Memetic Algorithm

Son Duy Dao[1,*], Antoine Mallégol[1], Patrick Meyer[1,*], Mehrdad Mohammadi[1], Sophie Loyer[2]

[1] IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
son.dao@mymail.unisa.edu.au, antoine.mallegol@ imt-atlantique.fr,
patrick.meyer@ imt-atlantique.fr, mehrdad.mohammadi@ imt-atlantique.fr

[2] Shom, F-29200 Brest, France
sophie.loyer@shom.fr

[*] Corresponding authors

Spatial Area Determination Problem: Definition and Solution Method Based on Memetic Algorithm

Abstract: Spatial area determination problem is defined herein as an optimization problem in which it is required to determine both the location and shape of a spatial area, given some constraints on the area (e.g., size) while maximizing or minimizing an objective function defined on spatial data (e.g., risk, cost, safety, security, etc.). The spatial area determination problem can be found in various domains such as hydrographic survey planning, conservation planning, military planning, etc., and it recently attracts research attention. Currently, there is no formal definition of the problem and the related solution methods are very limited. In this paper, first, the spatial area determination problem is defined and formulated, and then a solution method based on a Memetic Algorithm is developed to solve the problem. To deal with the constraints of the problem and to enhance the robustness of the traditional Memetic Algorithm, several innovations in the proposed Memetic Algorithm are introduced. Unlike the traditional Memetic Algorithm, the proposed Memetic Algorithm employs three crossover operators, two mutation operators, and a local search operator. In addition, the proposed Memetic Algorithm has a mechanism to automatically restart its search process if it gets stuck in the local optima. Moreover, parameters of the proposed Memetic Algorithm are systematically tuned by the Taguchi experimental design method to maximize its performance. The outperformance of the proposed Memetic Algorithm is validated through 18 test instances, 24 T-tests, and a Friedman test against four popular optimization algorithms, namely Simulated Annealing, Particle Swarm Optimization, Genetic Algorithm, and traditional Memetic Algorithm. The results indicate that, on average, the proposed Memetic Algorithm provided 36.5%, 43.3%, 20.4%, and 22.4% better solution, compared to Simulated Annealing, Particle Swarm Optimization, Genetic Algorithm, and the traditional Memetic Algorithm, respectively.

Keywords: Spatial area determination problem, Spatial data, Memetic algorithm, Optimization.

1. Introduction

Spatial data, also known as geospatial or geographic data, is the information about the geographic location of features and boundaries on the earth surface, such as natural features, land areas, ocean surfaces, etc. Spatial data can be mapped and usually stored as coordinates and topologies [1]. Spatial data is a foundation of a number of decision problems such as land-use planning [2], biodiversity conservation planning [3], maritime spatial planning [4], or military planning [5].

Given some spatial data, determining the best zone(s)/area(s) for a particular purpose can be very challenging because of complex constraints. For example, in the conservation planning, how does one find the best area to protect the fauna/flora, given the environment, economics and government policy constraints? or in military planning, where is the best area for military exercise, given some constraints on security, cost, and effectiveness [6, 7].

These challenging decision making problems have recently attracted the attention of the industry, especially that of the French National Hydrographic and Oceanographic Service (Shom). The Shom is in charge of the national hydrography program which plans hydrographic surveys in maritime areas under French responsibility [8]. In hydrographic survey planning, given some spatial data on navigation (e.g., risk), determining the most reasonable ocean area, not just the riskiest area, to make a hydrographic survey with the given equipment, labor, and completion time is not a trivial task because of the complex constraints.

These decision making problems can be modeled as optimization problems, in which the goal is to determine the best location and/or shape of a spatial area for a certain purpose, given some constraints. There are two sub-optimization problems here, i.e., optimal location and optimal shape of a spatial area. Solving these two sub-optimization problems simultaneously is a relatively new class of optimization problems, and not much research on it has been done in the literature. The main objective of this research is to address this type of optimization problems, which is called the spatial area determination problem (SADP). In this paper, first, the SADP is defined and formulated, and then an optimization solution method based on a Memetic Algorithm (MA) is proposed. Due to the nature of the constraints in the SADP, a new version of MA, with innovations in the chromosome encoding, crossover and mutation operators, constraint handling strategies, and algorithm structure, is developed herein.

Thereby, this research has two major contributions to the literature: (1) Providing the first and formal definition and formulation of SADP; and (2) developing an improved version of MA to solve the SADP.

The rest of the paper is organized as follows. In Section 2, the latest research related to the SADP and its resolution is reviewed. Section 3 proposes the formulation of the SADP. Section 4 presents the proposed MA for finding the optimal/sub-optimal solution(s) to the problem. Finally, the numerical results and conclusions are provided in Sections 5 and 6, respectively.

2. Literature Review

Finding an optimal spatial area/zone for particular purposes such as hydrographic survey, fishing, plant conservation, emergency evacuation, military exercise, etc. is not a trivial task, especially for complex inter-connected systems, since the decision maker usually faces a large number of conflict criteria and do not know exactly which solution is better. In a research of Hsu & Peeta [9], a risk-based spatial zone determination method was developed for disaster evacuation operations. In this method, the goal is to determine risk-based evacuation sub-zones for stage-based evacuation operations in a disaster region so that information-based evacuation procedures could be done in real-time for the sub-zone(s) with the highest evacuation risk to maximize the evacuation system performance. As the name suggests, this method determines the optimal spatial zones based on the assessment of evacuation risk only.

To overcome the drawbacks of almost all of the existing theories for spatial conservation planning (i.e., focusing on identifying no-take reserves), Klein et al. [10] developed a multi-zone planning tool called “Marxan with Zones” for network design of four types of protected areas in the context of California's Marine Life Protection Act. This spatial marine zoning method can be used for both marine and terrestrial conservation planning. Data from this study shows that zoning configuration produced by the developed method is 9% better for every fishery while the conservation goal is not compromised.

Geneletti and Duren [11] combined spatial multi-criteria and multi-objective evaluation methods to solve the protected area zoning problems in the context of the Paneveggio-Pale di S. Martino Natural Park (Italy). There are four steps to implement in this protected area

zoning method. The first step is to partition the area into homogeneous land units which represent the basic spatial elements of the zoning mosaic. The second step is to perform three multi-criteria evaluations to map land suitability for different zone types with different protection levels. The third step is to assign each land unit to a protection level through a multi-objective land allocation procedure. The final step is to conduct the sensitivity analysis to check the robustness of the zoning scheme.

Kazemzadeh-Zow et al. [12] developed a spatial zoning approach simulating long-term expansion of Mashhad city in Iran. In this method, a mix of external and internal variables for predicting urban growth was considered. In addition, this spatial zoning method differentiates the local-scale urban dynamics in districts from the socio-economic characteristics. First, Thiessen polygons were used in this method to identify districts with different morphology and functional attributes. An urban growth was then simulated for each district using a multi-layer perceptron neural network and Markov chains analysis. Finally, the multi-layer perceptron and Markov chains algorithms were used to derive transition maps from non-urban to urban use of land and to determine spatial evolution of built-up areas at the metropolitan scale.

Regarding the maritime spatial zoning, there have been a number of studies. To manage southern bluefin tuna in the eastern Australia longline fishery, Hobday et al. [13] proposed a dynamic spatial zoning method, in which the habitat model conditioned with temperature preference data from satellite and the ocean model were combined to produce near real-time habitat predictions. In the fishing region, several habitat types based on occurrence probability of southern bluefin tuna are identified, and then the manager will determine and regulate the fishers to access these fishing zones. Obviously, beside the habitat prediction model, this spatial zoning method relies on human judgement to determine the best zones for fishing.

A research called zoning marine protected areas through spatial multiple-criteria analysis was conducted by Villa, Tunesi & Agardy [14] in which the case study of the Asinara island, national marine reserve of Italy, was used. This method used the spatial multiple-criteria analysis for determining the suitability of marine areas for different uses and protection levels, by using geographic information systems (GIS) for land assessment/evaluation coupled with a formal statement of the design priorities as seen from the different viewpoints of all stakeholders. The objective data with the contrasting priorities of different stakeholders was integrated by multiple-criteria analysis to determine the optimal spatial arrangement of

different protection levels. In addition, Habtemariam and Fang [15] developed an interdisciplinary zoning method for multiple-use marine protected area, which combines spatial multi-criteria analysis, geographic information system and stakeholder consultation, with a case study of the Sheik Seid Marine National Park in Eritrea.

Regarding spatial zoning optimization, Crossman et al. [16] developed a spatial optimization decision support system to determine optimal solutions to a maritime spatial problem of identifying comprehensive, adequate and representative locations for conservation planning in South Australia. In this method, a database of spatial layers (Geographic Information System environment - ESRI's ArcGIS) describing the biophysical features of the marine environment (i.e., bathymetry, sea surface temperature, chlorophyll 'a' concentration levels, benthic and coastal habitat types, and shoreline exposure and type) was used to identify surrogate ecological regions. An integer programming algorithm (ILOG's CPLEX) was then used to find the locations that most efficiently represent these surrogates of biodiversity

In the research of Wie and Chai[17], an intelligent GIS-based spatial zoning system with multi-objective hybrid metaheuristic algorithm was developed to draw territory lines for geographical or spatial zones for the purpose of space control. In this method, a geographic information system and a hybrid metaheuristic (i.e., Tabu search and Scatter search algorithms) were used to generate non-dominated alternatives. Li et al. [18] attempted to use the urban cellular automata coupled with Ant Colony Optimization to solve a zoning protected natural area problem under a changing landscape. Outperformance of this method against three traditional optimization algorithms, i.e., Simulated Annealing, Iterative Relaxation and Density Slicing, has been tested and confirmed in the metropolitan region of Guangzhou, China, by using Geographical Simulation and Optimization System (GeoSOS) software.

As can be seen from the literature review, the SADP has not been fully established yet - some important constraints of the problem such as the area size and the area shape were not simultaneously considered in the models. In addition, the existing optimization solution methods for the SADP are very limited. To fill these gaps, first, the SADP is formally defined and formulated, and then an optimization solution method based on MA for the problem is developed in this paper. Details of the problem definition and the developed MA will be presented in Sections 3 and 4, respectively.

3. Problem Definition

In this paper, the SADP is defined as follows. Given a spatial data and the required size of the interest area (e.g., 20 km2), the goal is to find the location and shape of an area that maximizes the coverage of the high-valued polygon(s), as illustrated in Fig. 1. The values of the spatial data in Fig. 1 are represented by their colors, e.g., black is for 0 and white is for 6. Obviously, without the area size constraint, the white polygon always contains the optimal solution(s). When considering the size constraint of the interest area, the optimal solution(s) is always located in the white polygon if the required size of the interest area is smaller than the size of the white polygon, as illustrated in the sub-figure (a). With the constraint of larger area size, the optimal solution(s) will be in the yellow polygons as shown in sub-figures (b and c); and with the constraint of very large area size, the optimal solution(s) can be located in the polygons as shown in sub-figure (d). As can be seen from Fig. 1, with the same spatial data and different area size constraints, the optimal solutions to SADP can be located in different polygons and can have different shapes. Therefore, SADP is a constrained optimization problem.

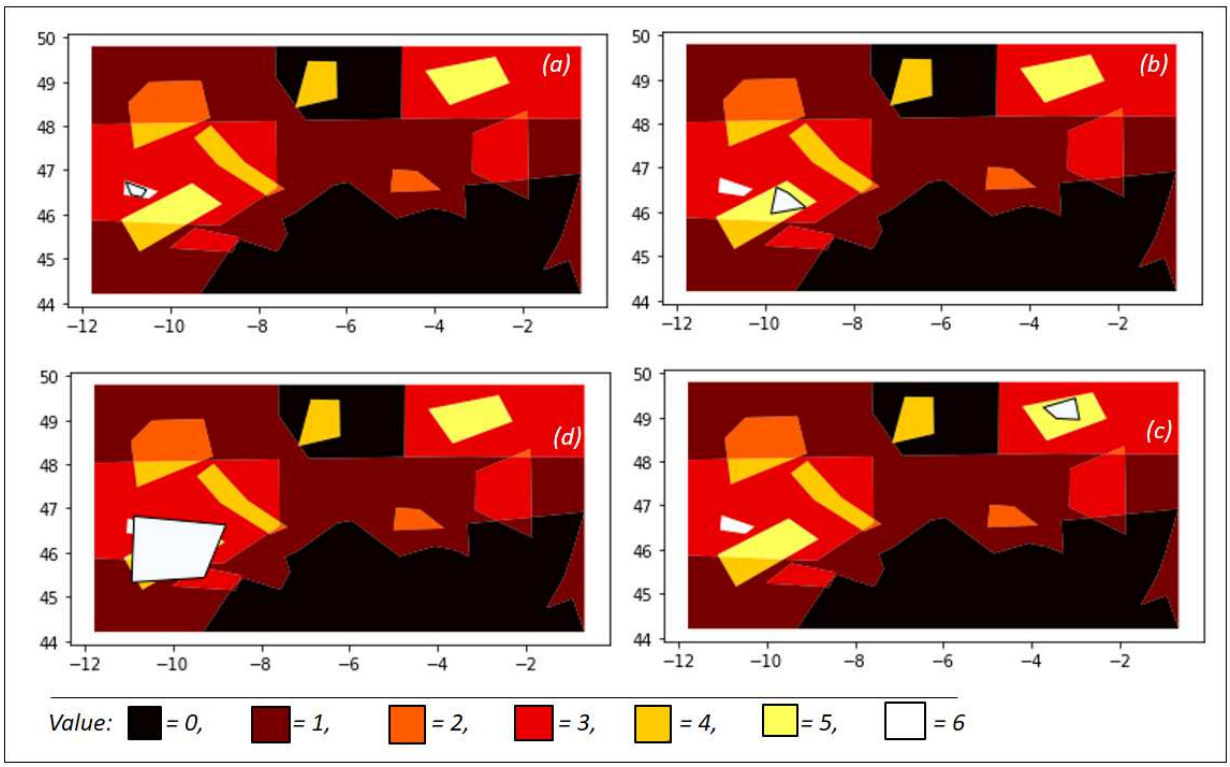


Fig. 1: Locations of proposed area (white quadrangles with black frames) with different area size constraints

SADP can have several applications, for example, in conservation planning (finding the best area to protect the fauna/flora, given the environment, economics and government policy constraints), in military planning (finding the best area for military exercise, given some constraints on security, cost, and effectiveness), or in hydrographic survey planning (finding the most reasonable ocean area, not just the riskiest area, to make a hydrographic survey with the given equipment, labor, and completion time, given some spatial data on navigation risk) [7].

The assumptions of the problem are listed as follows:

- The spatial data is vector data, i.e., made of points, lines and polygons.
- Each of these points, lines and polygons contains a value which can be an integer or a real number.

The next step is to define the decision variables. In this paper, it is considered that the proposed area should be a quadrangle area with a given size (surface). To represent a quadrangle area on the spatial data, two main decisions are to be made: the location and the shape of the area. The location of the area is represented by a (x, y) coordinate in a Cartesian coordinate system and the shape of the area is determined by positioning the four corners of the area. Each corner can be positioned by two characteristics: 1) its angle to one of the coordinate axis and 2) its distance to the center of the area. In order to take all these variables into account, the following solution representation is proposed for a quadrangle area with four corners (C1 to C4):

$$(x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4)$$

where (as illustrated in Fig. 2):

- x, y are the coordinates of the center O of the quadrangle area.
- $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are four angles used to determine the locations of the four corners C1 to C4, related to the coordinate axes as shown in Fig. 2.
- d_1, d_2, d_3, d_4 are the direct distances of the four corners C1 to C4 to the area center O, respectively.

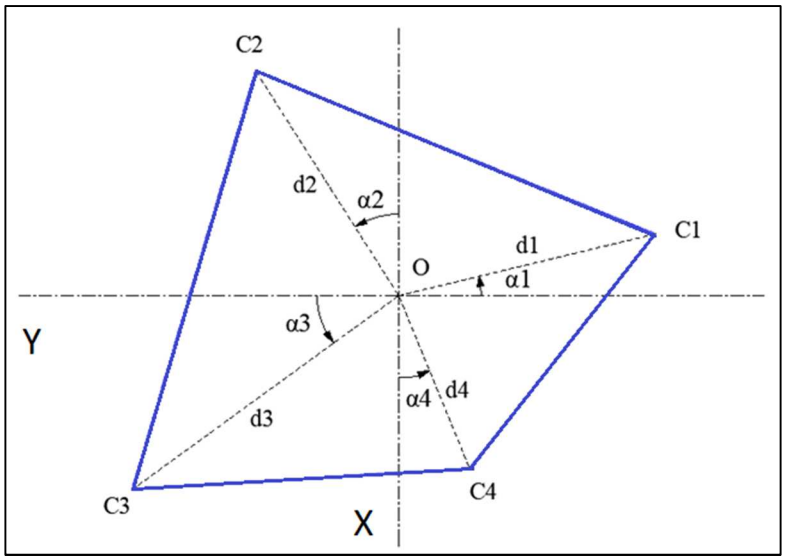


Fig. 2: An interest area with its decision variables

In the SADP, it is to maximize the coverage of the high-valued polygon(s) as the objective function, which is calculated by Eq. 1 and visually illustrated in Fig. 3.

$$F = \sum_{i=1}^{n} A_i * (V_i)^c \tag{1}$$

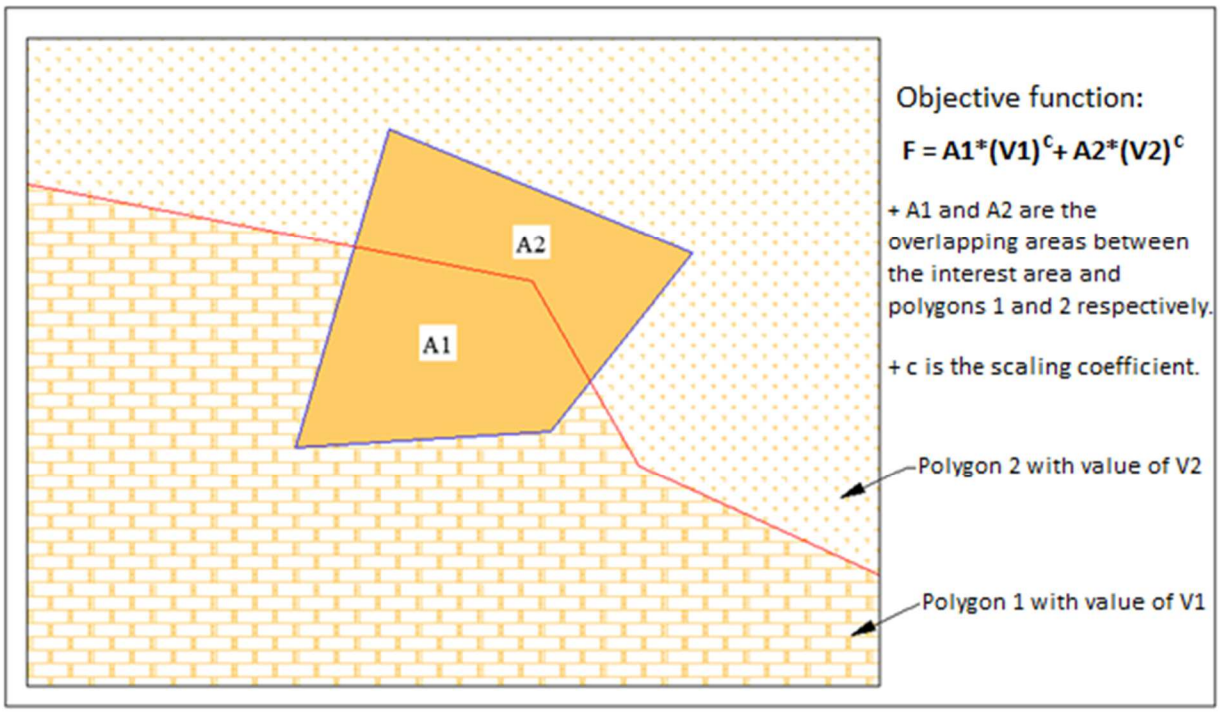


Fig. 3: A simplified example of how the objective function is calculated

where:

- A_i is the overlapping area between the interest area and the i^{th} polygon on the spatial data - for example, A_1 and A_2 as illustrated in Fig. 3. It should be noted that A_i is a function of the decision variables: $x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4$ ($A_i = f_i(x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4)$ and the spatial map. However, this function (i.e., an exact mathematical function expressing the direct relationship between the decisions variables $x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4$ and the overlapping area A_i) cannot be generally determined because it depends on the spatial data and it is a "black-box" problem. A general procedure to calculate the overlapping area A_i, given the decision variables: $x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4$ and the spatial data, is presented in Section 4.5.
- V_i is the value of the i^{th} polygon on the spatial data – it is like a value in a heat map (for example, V_1 and V_2 as illustrated in Fig. 3, or the values 0 to 6 represented by the colors in Fig. 1).
- c is the scaling coefficient, which is used to adjust the importance of particular polygons on the spatial data to the objective function. For one spatial data, the value of *c* should be the same. If *c* is great than 1, the importance of the polygons with the top values will be increased; and if it is less than 1, the importance of the polygons with the top values will be decreased.
- n is the number of polygons on the spatial data. For example, the spatial data in Fig. 3 has only two polygons. For complex spatial data, *n* can be a large number. The minimum value of *n* is 1, which would be found in a simplest spatial data.

The main input parameter of the problem is the required size of the interest area S ($S > 0$).

Finally, the list of the constraints are provided as follows:

- $\sum_{i=1}^{n} A_i = S$ (the area size constraint, for example, S =20 km^2).
- $x_{min} \leq x \leq x_{max}$ (the x coordinate of the center of the spatial area must be in within a certain polygon determined by x_{min} and x_{max}).
- $y_{min} \leq y \leq y_{max}$ (the y coordinate of the center of the spatial area must be in within a certain polygon determined by y_{min} and y_{max}).
- $\alpha_{min} \leq \alpha_1, \alpha_2, \alpha_3, \alpha_4 \leq \frac{\pi}{2} - \alpha_{min}$ (the range of each angle α_i to ensure the spatial area to be quadrangle, and not to be twisted).

- $d_{min} \leq d_1, d_2, d_3, d_4 \leq d_{max}$ (the range of each distance d_i to ensure the spatial area to be quadrangle, and not to be twisted).
- $x_{min}, x_{max}, y_{min}, y_{max}$ are the minimum and maximum coordinates of the spatial map.
- $\alpha_{min} > 0$ (the minimum angle to ensure the spatial area to be quadrangle, and not to be twisted).
- $d_{min}, d_{max} > 0$ (the minimum and maximum distances to ensure the spatial area to be quadrangle, and not to be twisted).

As can be seen from the objective function and constraints, the SADP is a "black-box" optimization problem (i.e., a problem in which analytical or derivative information is not available [19]) because there is no general mathematical function expressing the direct relationship between the decisions variables $x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4$ and the objective function F (more precisely, the overlapping area A_i).

In general, optimization solution methods can be classified into two main categories: exact methods and approximate methods [20]. Each approach has its own advantages and disadvantages. Although exact methods are capable of guaranteeing the optimal solutions, they may fail when solving the "black-box" and/or complex optimization problems. Although approximate methods can work with any type of optimization problems, the optimal solutions may not be guaranteed [21, 22]. Generally speaking, when dealing with complex optimization problems, approximate methods are more popular than exact methods [23].

Because the SADP is a "black-box" optimization problem, exact optimization method is not a good option to solve it; instead, meta-heuristic algorithms (approximate methods) are used to solve the problem. Details of the proposed optimization solution method for the SADP will be presented in the next Section.

4. Proposed Memetic Algorithm

There have been a number of approximate (stochastic) optimization solution methods such as Genetic Algorithm (GA) [24], Particle Swarm Optimization (PSO) [25], Harmony Search (HS) [26], Cuckoo Search (CS) [27], Ant Colony Optimization (ACO) [28], Tabu Search (TS) [29], Pattern Search (PS) [30], Greedy Algorithm [31], Simulated Annealing (SA) [32], Hill

Climbing (HC) [33], and Memetic Firefly Algorithm [34]. In addition, there have been various hybrid algorithms [35-38], in which two or more stochastic optimization algorithms are combined/integrated together to improve the solution quality.

MA is a combination of an evolutionary search based optimization algorithm with the problem-specific local search to balance the exploration and exploitation of the algorithm to enhance the solution quality [39]. The main benefit of a MA is to combine a global search algorithm such as GA, PSO, ACO, etc. with a local search algorithm to improve the optimal solution [40, 41]. Recently, MAs have been proven to be powerful and effective in solving complex optimization problems [42-45].

In the research of Lu et al. [46], MA was used to solve the multiple traveling repairman problem with profits (TRPP). In this MA, a randomized greedy construction method for initial solution generation, a variable neighborhood search for local refinement, and a dedicated route-based crossover operator for solution recombination were combined. The effectiveness of the developed MA was demonstrated in solving a large set of instances of TRPP. Eremeev and Kovalenko [47] developed another MA with optimal recombination for the asymmetric travelling salesman problem. In this MA, a crossover operator based on an exact algorithm was used to solve the optimal recombination problem on cubic digraphs, and a mutation operator was employed to make random jumps in 3-opt or 4-opt neighborhoods. In addition, a greedy constructive heuristic was used in this MA to generate the initial population. Huang et al. [48] proposed a so called niching MA for multi-solution traveling salesman problem. In this MA, a niche preservation technique to enable the parallel search, an adaptive neighborhood strategy to balance the exploration and exploitation, a critical edge-aware method to provide effective guidance to the reproduction, and a local search strategy to improve the search efficiency were proposed. Comprehensive experiments were conducted to confirm the effectiveness of the proposed MA.

Yadegari et al. [49] proposed a MA for closed-loop supply chain network design. In this MA, a priority-based encoding/decoding method based on a flexible combinatorial neighborhood search strategy was developed. Moreover, a technique to convert the discrete representation to a continuous one was proposed to avoid time-consuming repair process in discrete solution representation. Finally, a multi-start simulation annealing is integrated into the MA to enhance the search performance. The outperformance of the proposed MA, compared to

commercial solvers and GA, was validated in various test problems ranging from small size to large size.

For the hub location and routing problem with distinct collection and delivery tours, Yang et al. [50] proposed a MILP model and the related MA to solve it. For scheduling and planning problems with a single objective, Rahman et al. [51], Alsmady et al. [52], Yağmur and Kesen [53], Jin et al. [54] have developed various versions of MA to them. For multiple objective optimization, there have been a number of research works using MA such as Gong et al. [42], Decerle et al. [55], Abedi et al. [56], Zhang et al. [57], Spencer et al. [58], Pistolesi and Lazzerini [59], Sun et al. [60]. MA has been used to solve many other optimization problems such as gene selection problem in microarray data [61], training recurrent neural networks for the energy efficiency problem [62], preventing epidemic spreading in networks [63], medical data classification [64], feature selection for handwritten word recognition [65], composing distributed data-intensive Web services [66], the 3-D protein structure prediction problem [67], etc. Nevertheless, there is no research work using MA to solve the SADP yet. To fill this gap, an improved version of MA is developed to solve the SADP in this paper.

Fig. 4 shows our proposed MA for solving the SADP. To deal with the constraints of the spatial area determination problem and to enhance the robustness of the traditional MA, several innovations in the proposed MA are introduced. Unlike the traditional MA [40], the proposed MA employs three crossover operators, two mutation operators, and a local search operator during the search process. It is noted that the crossovers, mutations, and local search operators can be executed concurrently and/or consecutively – which depend on the PC processor and programming technique. In this article, consecutive computation was used for simplification. In addition, offspring chromosomes, which are generated by the three crossovers, two mutations, and the local search, can have a chance to compete with parent chromosomes for survival from one generation to the next. Moreover, the proposed MA has a mechanism to automatically restart its search process if it gets stuck in the local optima. In other words, the proposed MA is capable of restarting its search process whenever the quality of the best chromosome obtained so far is not improved within a prefixed number of generations (i.e., value P in Fig. 4).

It should be noted that four notations t, j, P and G in Fig. 4 represent (1) the current number of successive generations where in the quality of the best chromosome obtained so far has not

been improved, (2) the current number of generations of the proposed MA, (3) the prefixed number of successive generations where in the quality of the best chromosome obtained so far has not been improved, and (4) the maximum number of generations of the proposed MA, respectively. Initial population of the proposed MA is randomly generated, which contains a number of chromosomes (this number is equal to the population size). Every chromosome in the proposed MA must be feasible – which means that all the constraints of the problem under consideration in each chromosome must be satisfied. This initial population will be evolved along the iterations of the proposed MA, thanks to the crossover, mutation, local search, evaluation, and selection operators. In the final population, there is a solution(s) with the highest fitness value, which can be selected and considered as the optimal solution(s). Although, this selected solution may not be the global optimal solution, it is a good solution which is very near/close to the global optimal solution. The details of the proposed MA components such as chromosome encoding, crossovers, mutations, local search, fitness evaluation, selection mechanism, and parameter tuning will be presented in the subsequent Sections.

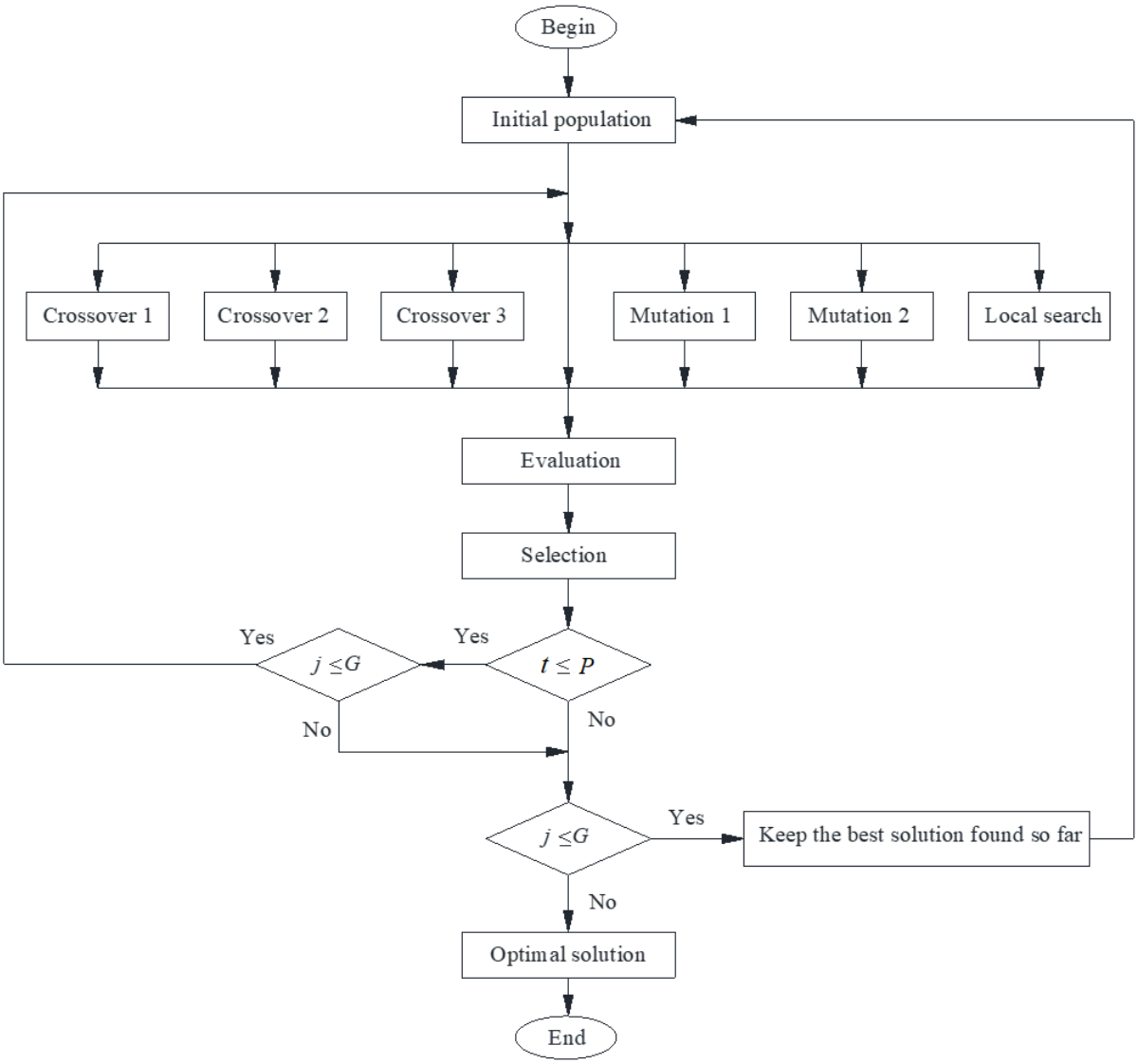


Fig. 4: Innovative structure of the proposed Memetic Algorithm

4.1. Chromosome Encoding

The chromosome of the proposed MA, which represents a solution to the SADP, is shown in Fig. 5. Each chromosome has 10 decision variables $(x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4)$ as shown in the highlighted cells in Fig. 5, and its encoded solution to the SADP is visualized in Fig. 2.

O	C1	C2	C3	C4
x	α1	α2	α3	α4
y	d1	d2	d3	d4

Fig. 5: Chromosome encoding

It should be noted that chromosomes in MA are randomly generated. When solving constrained optimization problems, one has to handle some constraints in order to randomly generate feasible chromosomes. For the SADP, there are a set of constraints as explained in Section 3, and the hardest constraint is the required size (surface) of the interest area, called size constraint hereafter. To avoid solving an inverse problem (i.e., randomly generate the center O and 3 corners C1 to C3 of the interest area, and then solve the size constraint equation to find the last corner C4 - that would be computationally expensive), a simple but effective procedure is proposed, as illustrated in Figs. 6-7, to handle the size constraint of the SADP. The basic idea here is to repair infeasible chromosomes to meet the size constraint by incrementally making it bigger or smaller, by incrementally adding or removing small amount of Δd to the all variables d_1, d_2, d_3, d_4 as illustrated in Figs. 6-7. The pseudo-code of generating feasible chromosomes is shown in Algorithm 1. The inputs of Algorithm 1 are 9 parameters including: $x_{min}, x_{max}, y_{min}, y_{max}, \alpha_{min}, d_{min}, d_{max}, \Delta d, S$; and the outputs are 10 decision variables including: $x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4$. For more detailed meanings of the above parameters and decision variables, it is advised to refer to Section 3 and Fig. 6. It is noted that Fig. 8 shows 50 typical random chromosomes generated by the proposed procedure in Algorithm 1. As can be seen from Fig. 8, the chromosomes have various shapes but have approximately the same sizes.

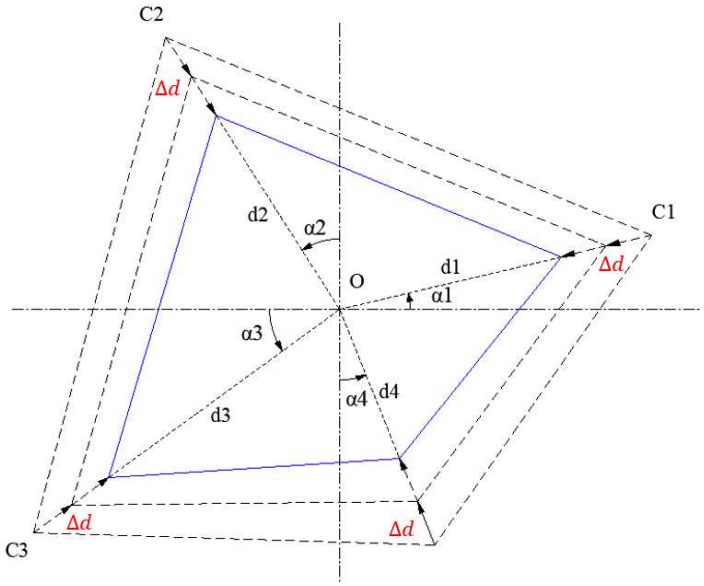


Fig. 6: Incrementally make infeasible chromosome smaller to meet the size constraint

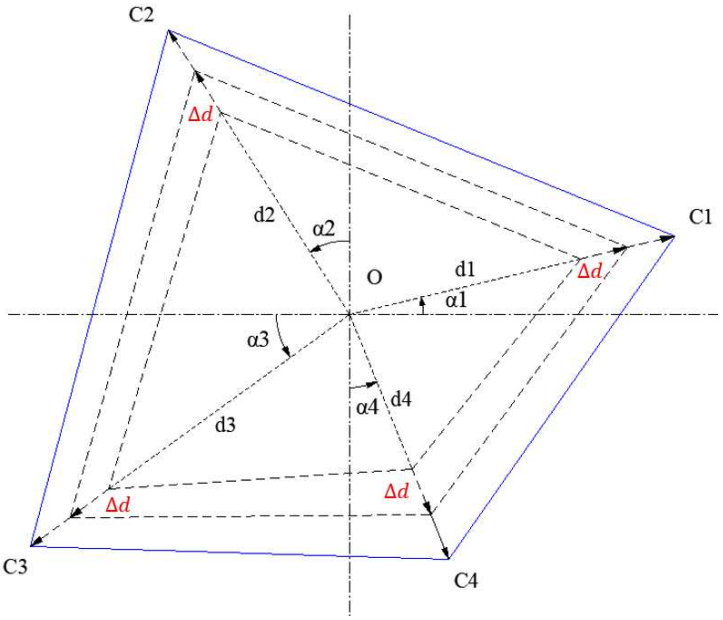


Fig. 7: Incrementally make infeasible chromosome bigger to meet the size constraint

Algorithm 1: Pseudo-code to generate a feasible chromosome

```
Inputs: x_min, x_max, y_min, y_max, α_min, d_min, d_max, Δd, S
Randomly generate x, y, α_1, d_1, α_2, d_2, α_3, d_3, α_4, d_4 within the upper and lower bounds
Calculate AS (area size of the chromosome)
if AS < S:
    while AS < S:
        d_1 = d_1 + Δd
        d_2 = d_2 + Δd
        d_3 = d_3 + Δd
        d_4 = d_4 + Δd
        Check and repair d_1, d_2, d_3, d_4 according to their bounds
        Calculate AS
    end
else:
    while AS > S:
        d_1 = d_1 − Δd
        d_2 = d_2 − Δd
        d_3 = d_3 − Δd
        d_4 = d_4 − Δd
        Check and repair d_1, d_2, d_3, d_4 according to their bounds
        Calculate AS
    end
end
Output: x, y, α_1, d_1, α_2, d_2, α_3, d_3, α_4, d_4 (feasible chromosome)
```

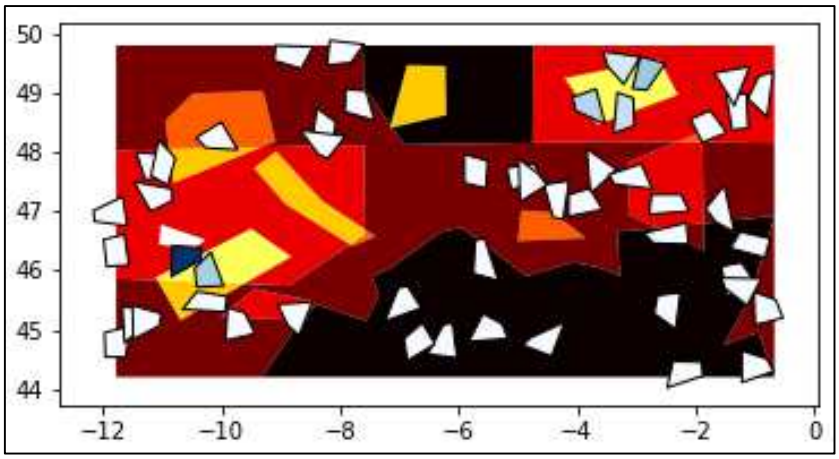


Fig. 8: 50 typical random chromosomes

4.2. Crossover Operators

To thoroughly explore the search space of the SADP, the proposed MA employs three crossovers, namely crossover 1, crossover 2 and crossover 3, applied to three different parts of the chromosomes, as shown in Figs. 9-11. For each crossover, one needs to exchange the values in the highlighted cells. The inputs of each crossover operator are two parents (one parent is randomly selected from the population, and another one is selected from the population using the roulette wheel selection rule [68]). The reason to select such two parents for the crossover operator is to balance the exploitation and exploration in the proposed MA. The outputs of each crossover are two offsprings (children). Except crossover 1, after doing the crossovers as shown in Figs. 9-11, some offsprings may become infeasible due to the violation from certain constraints of the SADP. Therefore, it is needed to check and repair all offspring chromosomes to ensure their feasibilities.

It is noted that three crossovers 1-3 are not related to each other, and they usually have different inputs and outputs. Crossover 1 is quite straightforward to implement and its offsprings are always feasible (no offspring repair process is required). To do crossover 1, one just needs to select two parent chromosomes and then exchange the values in the highlighted cells as illustrated in Fig. 9 to get feasible offsprings. However, doing crossovers 2 and 3 are more complicated because they involve offspring repair processes. The pseudo-codes for implementing crossovers 2 and 3 are proposed in Algorithms 2 and 3, respectively. It should be noted that the inputs of each algorithm (Algorithm 2 or 3) are two parent chromosomes (one parent is randomly selected from the population, and another one is selected from the

population using the roulette wheel selection rule [68]); and the outputs are two feasible offspring chromosomes.

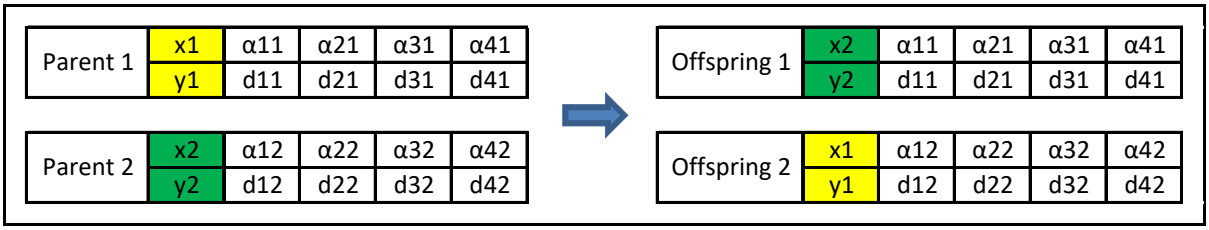


Fig. 9: Crossover 1

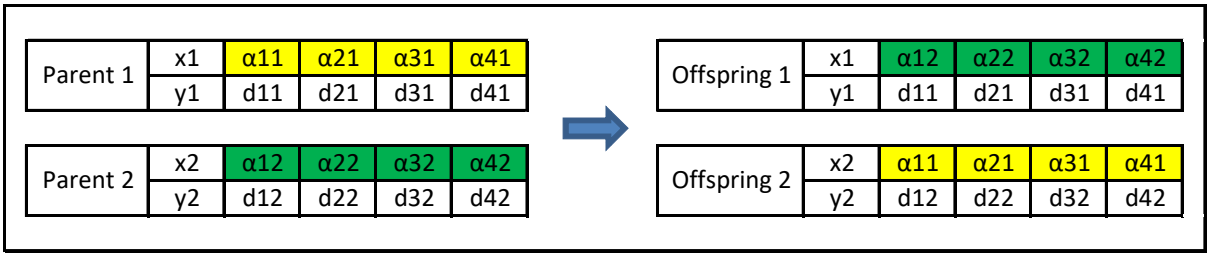


Fig. 10: Crossover 2

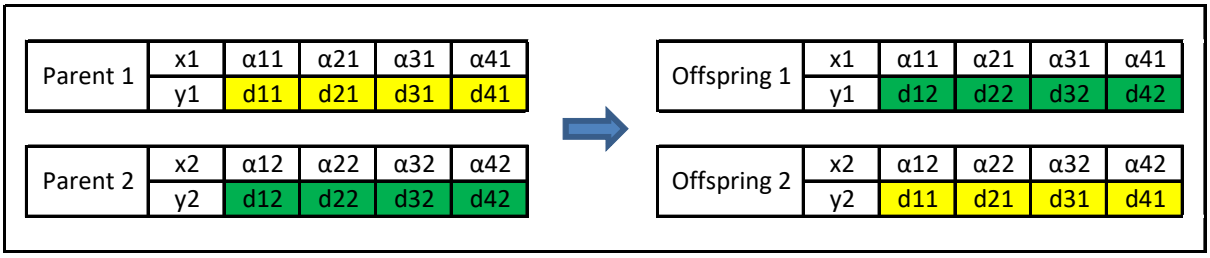


Fig. 11: Crossover 3

Algorithm 2: Pseudo-code to employ crossover 2

```
Inputs: P1,P2 (two parent chromosomes)
Exchange the α values in the highlighted cells as shown in Table 5
for i = 1:2
      Calculate ASi (area size of the offspring chromosome i)
      if ASi < S:
               while ASi < S:
                    d1 = d1 + Δd
                    d2 = d2 + Δd
                    d3 = d3 + Δd
                    d4 = d4 + Δd
                    Check and repair d1,d2,d3,d4 according to their bounds
                    Calculate ASi
               end
      else:
               while ASi > S:
                    d1 = d1 − Δd
                    d2 = d2 − Δd
                    d3 = d3 − Δd
                    d4 = d4 − Δd
                    Check and repair d1,d2,d3,d4 according to their bounds
                    Calculate ASi
               end
      end
end
Outputs: O1,O2 (two offspring chromosomes)
```

Algorithm 3: Pseudo-code to do crossover 3

```
Inputs: P1,P2 (two parent chromosomes)
Exchange the d values in the highlighted cells as shown in Table 6
for i = 1:2
      Calculate ASi (area size of the offspring chromosome i)
      if ASi < S:
               while ASi < S:
                    d1 = d1 + Δd
                    d2 = d2 + Δd
                    d3 = d3 + Δd
                    d4 = d4 + Δd
                    Check and repair d1,d2,d3,d4 according to their bounds
                    Calculate ASi
               end
      else:
               while ASi > S:
                    d1 = d1 − Δd
                    d2 = d2 − Δd
                    d3 = d3 − Δd
                    d4 = d4 − Δd
                    Check and repair d1,d2,d3,d4 according to their bounds
                    Calculate ASi
               end
      end
end
Outputs: O1,O2 (two offspring chromosomes)
```

4.3. Mutation Operators

To enhance the diversification capabilities of mutation operators to better explore the search space of the problem, the proposed MA employs two mutation operators, namely mutation 1 (2 scenarios) and mutation 2 (8 scenarios), applied to different parts of the chromosomes, as shown in Figs. 12-21 . For each mutation, 2 values in the highlighted cells are exchanged as illustrated in Figs. 12-21. Like the crossover operators in Section 4.2, the inputs of each mutation are two parent chromosomes (one parent is randomly selected from the population, and another one is selected from the population using the roulette wheel selection rule [68]). The outputs of each mutation are two offspring chromosomes.

Mutation 1 has two scenarios, one applied to x coordinate and another one applied to y coordinate, as shown in Figs. 12-13. Whenever the mutation 1 is called during the search process, the proposed MA will randomly select only one scenario to implement. To do mutation 1, the following steps are needed: (1) select two parent chromosomes using the rules mentioned above; (2) randomly select a mutation scenario; (3) exchange 2 values in the highlighted cells as shown in Figs. 12 or 13 to get the offsprings. As the offsprings of mutation 1 are always feasible, no repair process for offspring chromosomes is needed.

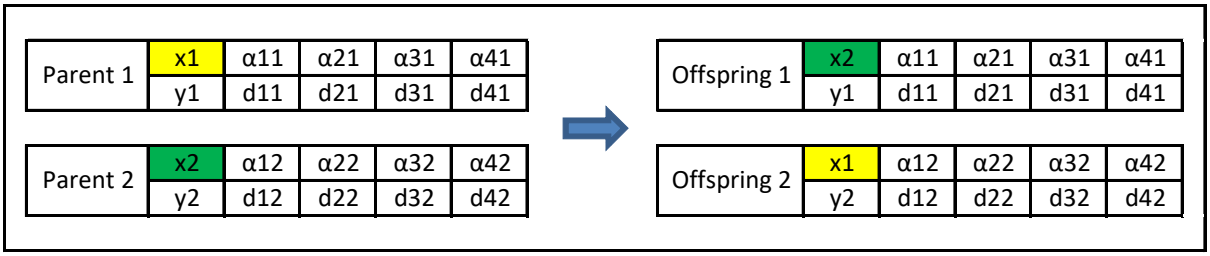


Fig. 12: Mutation 1 – Scenario 1

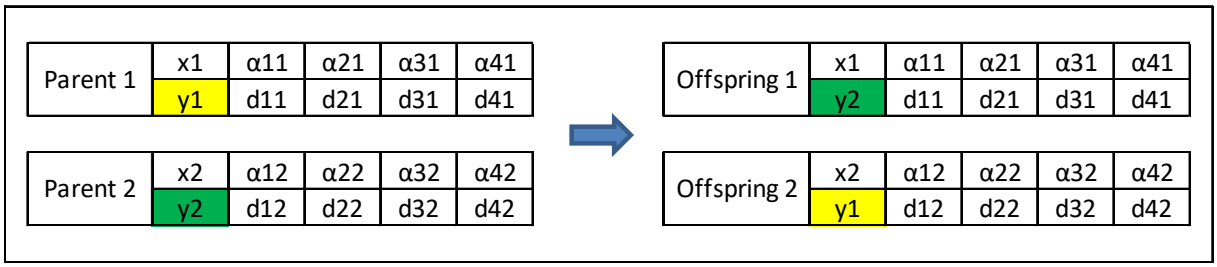


Fig. 13: Mutation 1 – Scenario 2

Mutation 2 has 8 scenarios as shown in Figs. 14-21. Whenever mutation 2 is called during the search process, the proposed MA will randomly select only one scenario to implement. Mutation 2 is more complicated than mutation 1 because it involves an offspring repair process. Indeed, the offsprings of mutation 2 may be not feasible because the size constraint may be violated after the values as illustrated in Figs. 14-21. To handle the constraints involved, a pseudo-code as shown in Algorithm 4 is proposed for implementing mutation 2. It is noted that, similar to Algorithms 2 and 3, the inputs of Algorithm 4 are two parent chromosomes (one parent is randomly selected from the population, and another one is selected from the population using the roulette wheel selection rule [68]); and the outputs are two feasible offspring chromosomes.

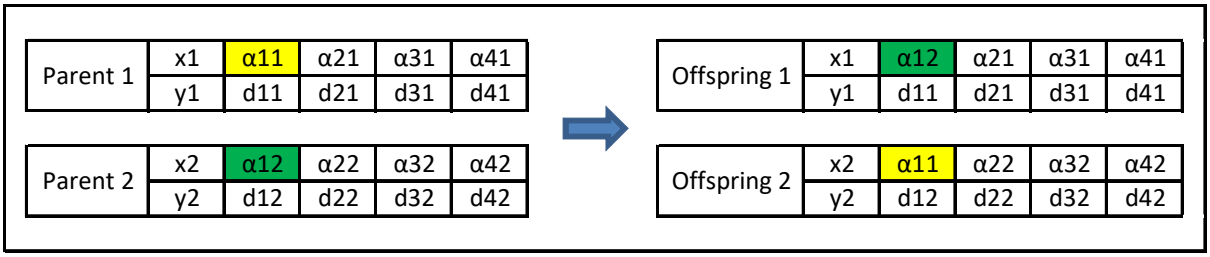


Fig. 14: Mutation 2 - Scenario 1

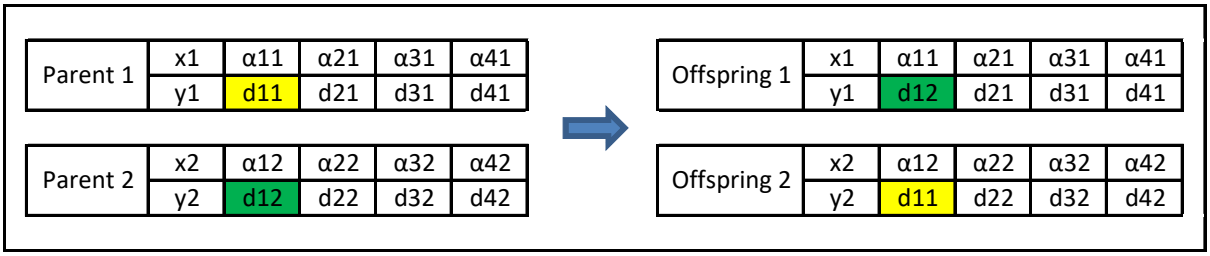


Fig. 15: Mutation 2 - Scenario 2

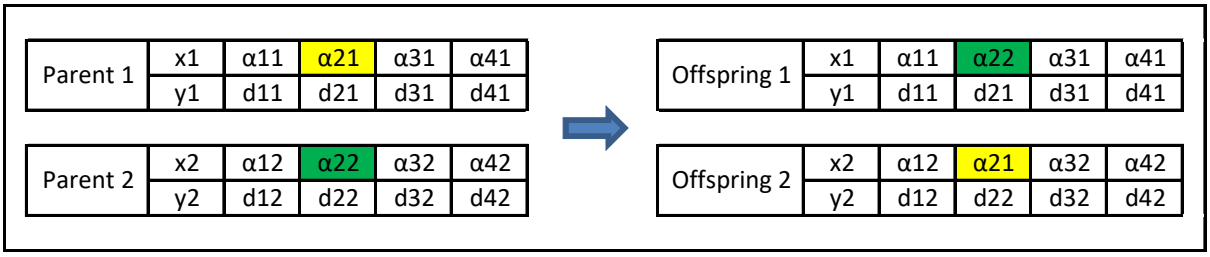


Fig. 16: Mutation 2 - Scenario 3

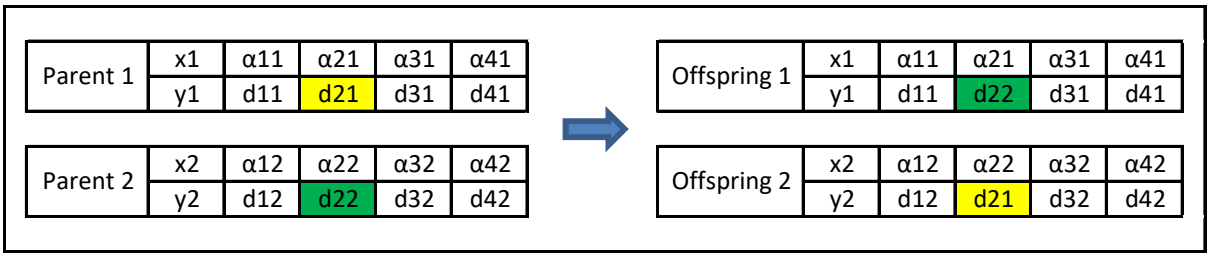


Fig. 17: Mutation 2 - Scenario 4

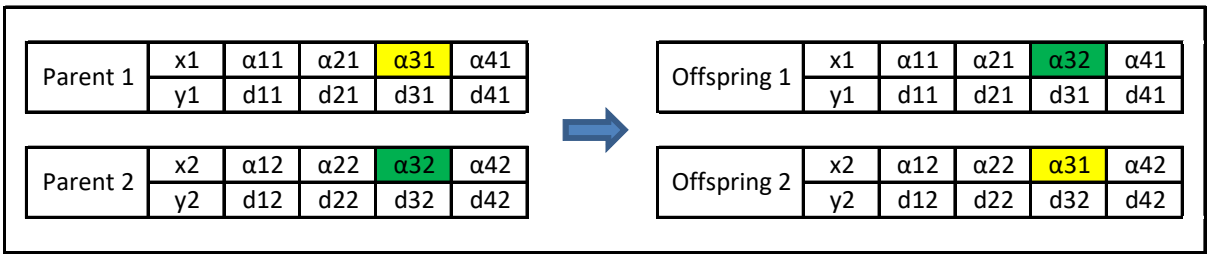


Fig. 18: Mutation 2 - Scenario 5

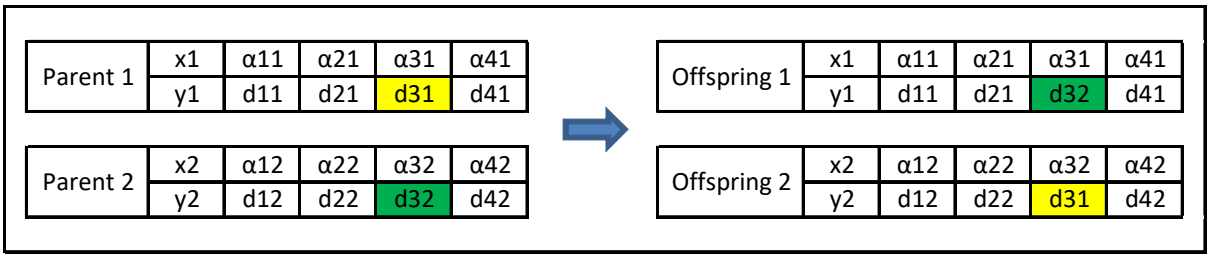


Fig. 19: Mutation 2 - Scenario 6

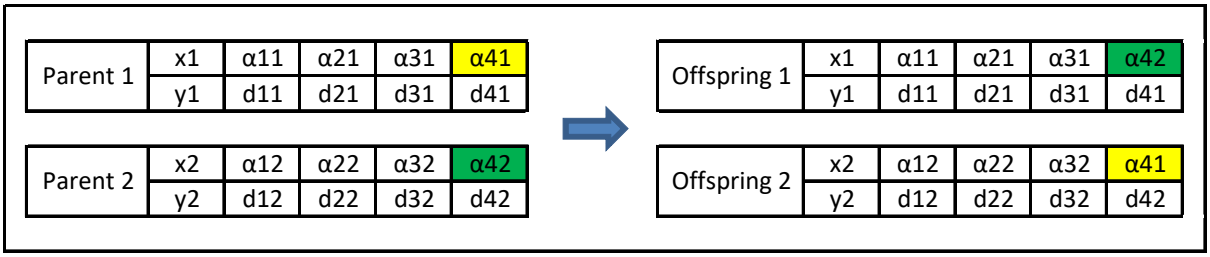


Fig. 20: Mutation 2 - Scenario 7

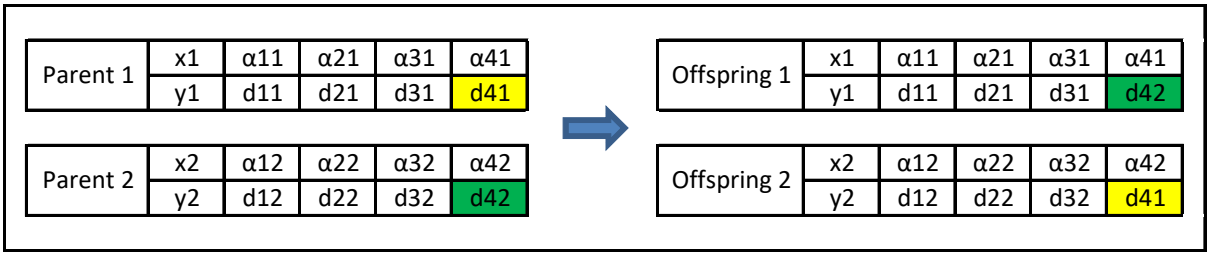


Fig. 21: Mutation 2 - Scenario 8

Algorithm 4: Pseudo-code to do mutation 2

```
Inputs: P1,P2 (two parent chromosomes)
Randomly select one mutation scenario in Tables 11-18
Exchange two values in the highlighted cells as shown in the selected scenario
for i = 1:2
      Calculate ASi (area size of the offspring chromosome i)
      if ASi < S:
              while ASi < S:
                  d1 = d1 + Δd
                  d2 = d2 + Δd
                  d3 = d3 + Δd
                  d4 = d4 + Δd
                  Check and repair d1,d2,d3,d4 according to their bounds
                  Calculate ASi
              end
      else:
              while ASi > S:
                  d1 = d1 − Δd
                  d2 = d2 − Δd
                  d3 = d3 − Δd
                  d4 = d4 − Δd
                  Check and repair d1,d2,d3,d4 according to their bounds
                  Calculate ASi
              end
      end
end
Outputs: O1,O2 (two offspring chromosomes)
```

4.4. Local Search Operator

Local search algorithm has been integrated with other algorithms to solve various complex optimization problems in the literature [69]. In this paper, to better explore the search space of the SADP, the proposed MA employs a local search operator (shown in Fig. 22) that applies a small change on a given solution to search the neighborhood of that solution thoroughly. As can be seen from Fig. 22, the local search has only one parent chromosome (initial solution for the local search) and one offspring chromosome. The parent chromosome of the local search is selected from the population using the roulette wheel selection rule [68]. The local search has 10 scenarios applied to 10 decision variables as shown in Fig. 22. Every time that the local search is called during the search process, the proposed MA will randomly select only one scenario to implement. It is noted that $\delta x, \delta y, \delta \alpha, \delta d$ in Fig. 22 are random small values that can be positive or negative. The offspring chromosome of the local search may be infeasible because some constraints can be violated after employing the local search operator. Therefore, the local search requires an offspring repair process. The pseudo code of the proposed local search operator is presented in Algorithm 5.

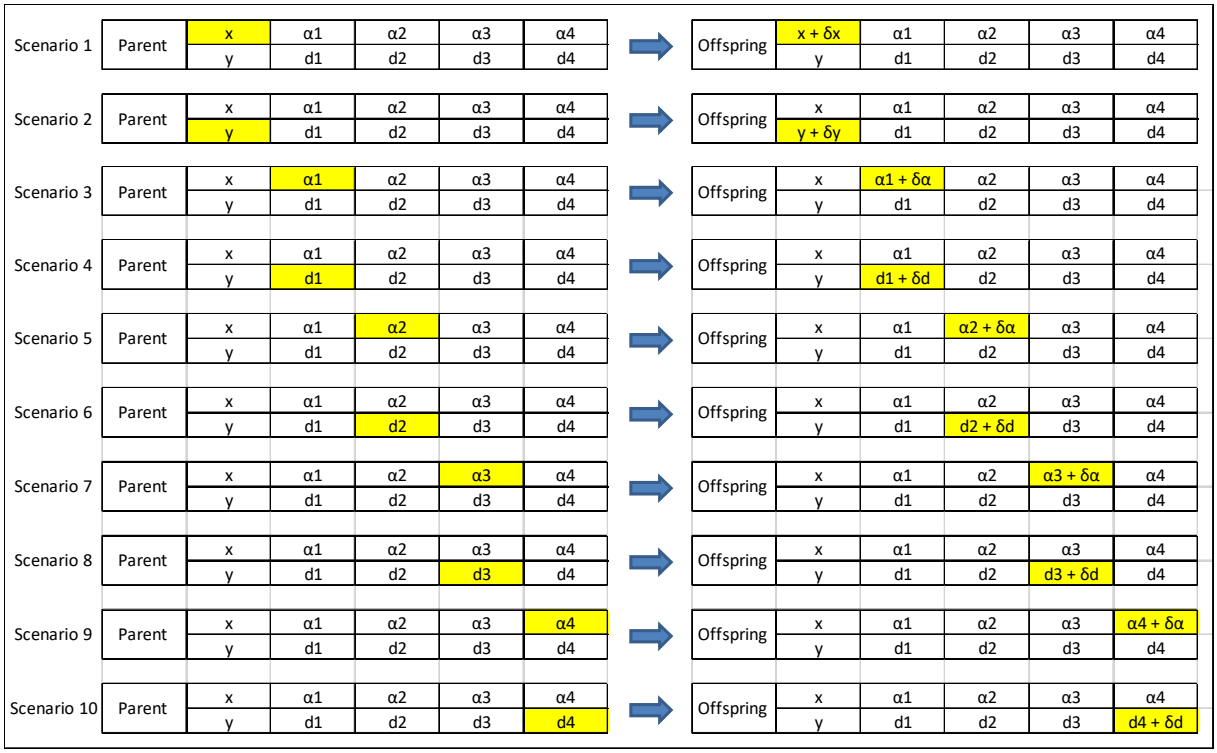


Fig. 22: Local search operator

Algorithm 5: Pseudo-code to do the local search

Input: P *(one parent chromosome)*
Randomly select one local search scenario in Table 20
Randomly generate the related value: $\delta x, \delta y, \delta \alpha$ *or* δd
Update the related value as shown in Table 20
Calculate AS *(area size of the offspring chromosome)*
if $AS < S$*:*
    while $AS < S$*:*
        $d_1 = d_1 + \Delta d$
        $d_2 = d_2 + \Delta d$
        $d_3 = d_3 + \Delta d$
        $d_4 = d_4 + \Delta d$
        Check and repair d_1, d_2, d_3, d_4 *according to their bounds*
        Calculate AS
    end
else:
    while $AS > S$*:*
        $d_1 = d_1 - \Delta d$
        $d_2 = d_2 - \Delta d$
        $d_3 = d_3 - \Delta d$
        $d_4 = d_4 - \Delta d$
        Check and repair d_1, d_2, d_3, d_4 *according to their bounds*
        Calculate AS
    end
end
Output: O *(one offspring chromosome)*

4.5. Evaluation and Selection Operators

Fitness value of a chromosome (quality of a solution) is calculated using Eq. 1, and is visually illustrated in Fig. 23, wherein an interest area (i.e., the chromosome) overlaps with two polygons with data values of V1 and V2 on the spatial data. The most complex part in calculating the fitness value of a chromosome is the calculation of the overlapping area A_i; and therefore, the following three-step procedure is proposed:

Step 1: Build a quadrangle (i.e., a chromosome) using the decision variables $(x, y, \alpha_1, d_1, \alpha_2, d_2, \alpha_3, d_3, \alpha_4, d_4)$ as illustrated in Fig. 23.

Step 2: Find all of the geometries in the spatial data which intersect with the built quadrangle in Step 1.

Step 3: Determine the area of each intersection in Step 2 (i.e., the overlapping areas: A_1 and A_2).

For more complex spatial data, the chromosome may overlap with multiple polygons, and more computing time will be required to calculate the fitness value.

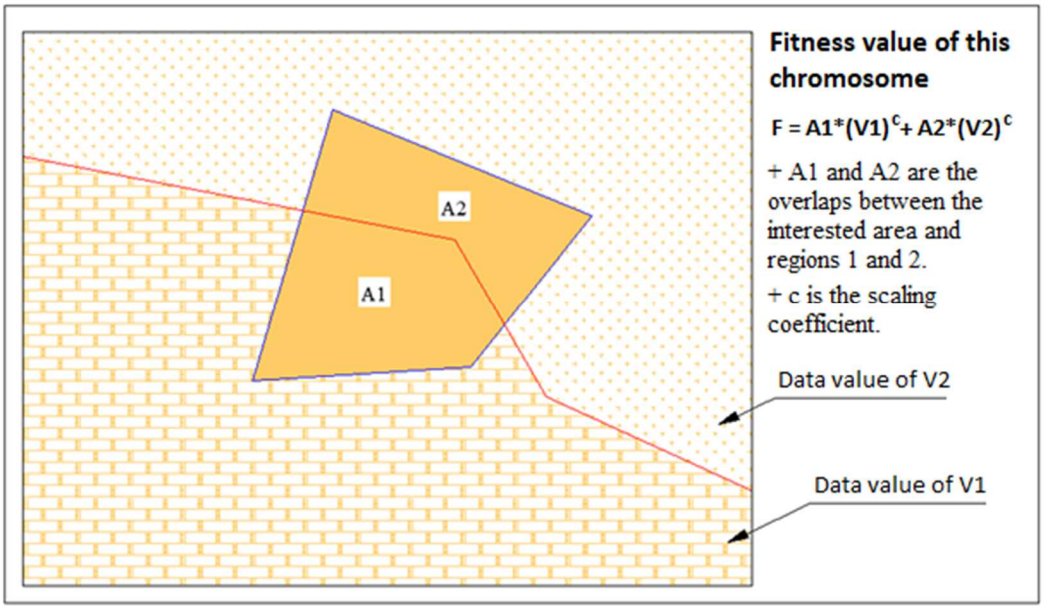


Fig. 23: Fitness value calculation – A simple example

In the proposed MA, the traditional roulette wheel selection [68] (also known as the fitness proportionate selection) is used to select chromosomes for the evolutionary process.

Parameters of the proposed MA such as population size, crossover rate, mutation rate, and local search rate are tuned by the Taguchi experimental design method [70]. More details about the parameter tuning will be presented in Section 5. The proposed MA was coded in Python and its effectiveness will be tested in the next Section.

5. Test Cases

Performance of the proposed MA is evaluated through 18 test cases with various configurations as shown in Table 1. These test cases are formed based on different area sizes (i.e., 20, 40, 60, 1000, 1500, and 2000 km^2), three computing times (i.e., 30, 60, and 90 seconds), and two spatial data shown in Figs. 23-24. The spatial data are in different scales and the values (risk scores) are represented by their colors. With each area size, our goal is to find the location and shape of an area that has a maximum coverage of the high-risk polygon(s). It should be noted that the optimal location of the solution to the SADP is not always in the polygons with the highest risk score, as illustrated in Fig. 1. In fact, the optimal location of the solution depends not only on the spatial data but also on the area size and the area shape. With the same spatial data, the location of the optimal solution can be changed if one changes the size and/or the shape of the area. That is why the SADP becomes an optimization problem and requires advanced algorithms to be solved.

Table 1: Test cases

Test case	Area size (km2)	Computing time (s)	Spatial map
1	20	30	1
2	20	60	1
3	20	90	1
4	40	30	1
5	40	60	1
6	40	90	1
7	60	30	1
8	60	60	1
9	60	90	1
10	1000	30	2
11	1000	60	2
12	1000	90	2
13	1500	30	2
14	1500	60	2
15	1500	90	2
16	2000	30	2
17	2000	60	2
18	2000	90	2

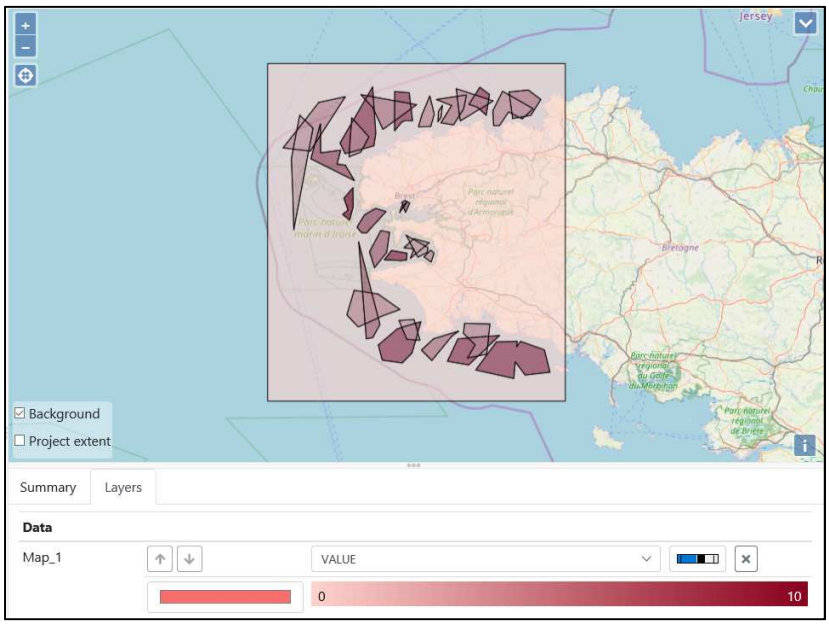


Fig. 24: Spatial data 1

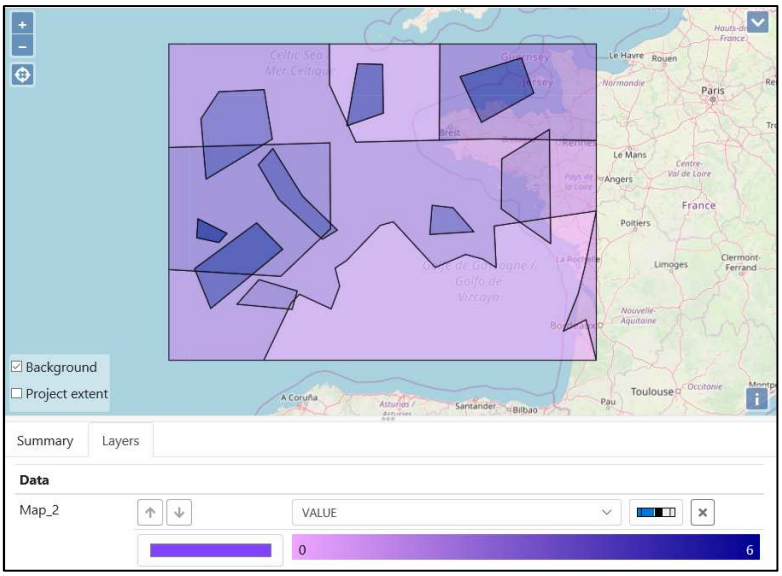


Fig. 25: Spatial data 2

The outperformance of the proposed MA is validated against four popular optimization algorithms, namely SA, PSO, GA, and traditional MA. All five optimization algorithms were coded in Python and ran on the same laptop computer with a processor: Intel(R) Core (TM) i5-6200U CPU @ 2.3 GHz, and RAM: 8.00 GB. It is noted that the traditional MA is exactly the same as the proposed MA except that the traditional MA does not have a mechanism to automatically restart its search process; and the GA is exactly the same as the traditional MA except that the GA does not have the local search operator.

To make a fair comparison, the parameters of all the optimization algorithms, i.e., SA, PSO, GA, the traditional MA, and the proposed MA, were systematically tuned by Taguchi experimental design method [70]. The parameters and three experimental levels of the optimization algorithms are shown in Tables 3-5.

Table 2: Parameters and experimental levels of the GA, traditional MA and proposed MA

No.	Parameter	Code	Experimental level		
			1	2	3
1	Population size	ps	50	100	150
2	Rate of crossover 1	c1	0.1	0.3	0.5
3	Rate of crossover 2	c2	0.1	0.3	0.5
4	Rate of crossover 3	c3	0.1	0.3	0.5
5	Rate of mutation 1	m1	0.1	0.3	0.5
6	Rate of mutation 2	m2	0.1	0.3	0.5
7	Rate of local search	ls	0.1	0.3	0.5

Note: The GA does not have the parameter ls (rate of local search).

Table 3: Parameters and experimental levels of PSO

No.	Parameter	Code	Experimental level		
			1	2	3
1	Number of particles	np	50	100	150
2	Inertia constant	ic	0.5	0.7	0.9
3	Cognitive constant	cc	1	2	3
4	Social constant	sc	1	2	3

Table 4: Parameters and experimental levels of SA

No.	Parameter	Code	Experimental level		
			1	2	3
1	Initial temperature	it	50	100	150
2	Cooling coefficient	cc	0.75	0.85	0.95
3	Number of trials at each temperature level	nt	25	50	75

Taguchi experimental layout and related experimental data related to the proposed MA, traditional MA, GA, PSO and SA are given in Tables 6 to 10, respectively. It is noted that throughout this paper, the scaling coefficient c in the objective function (Eq. 1) is set to 5; and the Taguchi experiments were conducted on the spatial data 2 with the area size of 1000 km^2. As can be seen from Tables 6 to 10, the number of Taguchi experiments related to each algorithm is not the same because it depends on the number of parameters to be tuned. For example, 27 experiments were required for the GA, traditional MA or proposed MA, but only

9 experiments were required for the PSO and the SA algorithms. In addition, to make a fair comparison, the same computing time equal to 30 seconds is set for each experiment, and each experiment is repeated 5 times (5 independent runs), as shown in Tables 6-10.

Table 5: Taguchi experimental layout and related experimental data (Proposed MA)

Experiment	Parameter							Computing time (s)	Experimental result				
	ps	c1	c2	c3	m1	m2	ls		Run 1	Run 2	Run 3	Run 4	Run 5
1	50	0.1	0.1	0.1	0.1	0.1	0.1	30	718.5	681.1	722.0	691.3	692.8
2	50	0.1	0.1	0.1	0.3	0.3	0.3	30	653.7	729.2	722.7	736.0	738.6
3	50	0.1	0.1	0.1	0.5	0.5	0.5	30	730.3	723.5	727.5	732.1	728.8
4	50	0.3	0.3	0.3	0.1	0.1	0.1	30	672.1	719.2	726.1	723.9	719.6
5	50	0.3	0.3	0.3	0.3	0.3	0.3	30	723.4	700.7	713.2	690.5	735.8
6	50	0.3	0.3	0.3	0.5	0.5	0.5	30	726.2	724.9	720.9	741.7	711.7
7	50	0.5	0.5	0.5	0.1	0.1	0.1	30	687.0	731.6	737.9	730.4	728.6
8	50	0.5	0.5	0.5	0.3	0.3	0.3	30	732.9	722.9	730.5	681.9	700.5
9	50	0.5	0.5	0.5	0.5	0.5	0.5	30	701.4	725.1	722.0	728.7	695.1
10	100	0.1	0.3	0.5	0.1	0.3	0.5	30	728.6	724.3	733.0	701.5	717.5
11	100	0.1	0.3	0.5	0.3	0.5	0.1	30	729.9	745.0	734.3	722.9	728.7
12	100	0.1	0.3	0.5	0.5	0.1	0.3	30	732.2	652.9	706.5	717.9	720.3
13	100	0.3	0.5	0.1	0.1	0.3	0.5	30	732.0	714.5	714.7	695.7	729.7
14	100	0.3	0.5	0.1	0.3	0.5	0.1	30	724.9	720.5	726.3	731.6	681.1
15	100	0.3	0.5	0.1	0.5	0.1	0.3	30	691.6	720.8	723.6	716.3	727.6
16	100	0.5	0.1	0.3	0.1	0.3	0.5	30	722.4	685.0	699.5	721.2	728.7
17	100	0.5	0.1	0.3	0.3	0.5	0.1	30	702.5	678.3	639.2	714.9	646.5
18	100	0.5	0.1	0.3	0.5	0.1	0.3	30	738.2	630.5	667.1	719.2	714.9
19	150	0.1	0.5	0.3	0.1	0.5	0.3	30	718.4	621.0	717.8	571.7	722.7
20	150	0.1	0.5	0.3	0.3	0.1	0.5	30	724.5	683.8	730.2	736.2	717.1
21	150	0.1	0.5	0.3	0.5	0.3	0.1	30	717.7	663.3	738.6	726.3	716.6
22	150	0.3	0.1	0.5	0.1	0.5	0.3	30	739.8	700.6	722.8	735.0	738.7
23	150	0.3	0.1	0.5	0.3	0.1	0.5	30	728.2	722.4	683.6	721.2	734.2
24	150	0.3	0.1	0.5	0.5	0.3	0.1	30	715.2	732.7	709.2	700.1	737.1
25	150	0.5	0.3	0.1	0.1	0.5	0.3	30	723.5	722.7	703.3	653.4	664.4
26	150	0.5	0.3	0.1	0.3	0.1	0.5	30	712.2	704.3	724.2	737.8	727.3
27	150	0.5	0.3	0.1	0.5	0.3	0.1	30	718.0	726.8	643.6	693.6	719.6

Table 6: Taguchi experimental layout and related experimental data (Traditional MA)

Experiment	Parameter							Computing time (s)	Experimental result				
	ps	c1	c2	c3	m1	m2	ls		Run 1	Run 2	Run 3	Run 4	Run 5
1	50	0.1	0.1	0.1	0.1	0.1	0.1	30	295.6	295.5	296.9	296.9	295.9
2	50	0.1	0.1	0.1	0.3	0.3	0.3	30	311.9	304.5	725.4	307.8	297.5
3	50	0.1	0.1	0.1	0.5	0.5	0.5	30	300.5	722.9	743.8	296.5	738.3
4	50	0.3	0.3	0.3	0.1	0.1	0.1	30	298.7	295.0	656.4	728.3	689.1
5	50	0.3	0.3	0.3	0.3	0.3	0.3	30	301.8	96.4	730.2	305.9	732.2
6	50	0.3	0.3	0.3	0.5	0.5	0.5	30	298.9	300.0	298.7	296.3	289.1
7	50	0.5	0.5	0.5	0.1	0.1	0.1	30	659.4	697.5	303.1	300.0	693.5
8	50	0.5	0.5	0.5	0.3	0.3	0.3	30	298.1	696.4	729.7	312.1	724.6
9	50	0.5	0.5	0.5	0.5	0.5	0.5	30	743.5	297.7	296.0	298.9	299.7
10	100	0.1	0.3	0.5	0.1	0.3	0.5	30	700.0	298.0	731.6	749.0	300.8
11	100	0.1	0.3	0.5	0.3	0.5	0.1	30	652.4	543.8	737.2	480.1	719.7
12	100	0.1	0.3	0.5	0.5	0.1	0.3	30	311.6	299.3	726.7	718.3	303.0
13	100	0.3	0.5	0.1	0.1	0.3	0.5	30	727.5	310.7	735.8	732.0	311.2
14	100	0.3	0.5	0.1	0.3	0.5	0.1	30	307.8	306.6	310.8	309.2	300.2
15	100	0.3	0.5	0.1	0.5	0.1	0.3	30	300.2	725.1	723.0	732.4	658.6
16	100	0.5	0.1	0.3	0.1	0.3	0.5	30	311.0	712.6	722.3	303.0	311.3
17	100	0.5	0.1	0.3	0.3	0.5	0.1	30	466.3	304.9	301.5	568.3	310.4
18	100	0.5	0.1	0.3	0.5	0.1	0.3	30	676.6	720.8	304.0	720.5	300.3
19	150	0.1	0.5	0.3	0.1	0.5	0.3	30	731.8	311.3	729.9	704.0	740.0
20	150	0.1	0.5	0.3	0.3	0.1	0.5	30	722.5	718.8	694.5	720.4	299.7
21	150	0.1	0.5	0.3	0.5	0.3	0.1	30	704.7	648.6	735.3	301.7	632.7
22	150	0.3	0.1	0.5	0.1	0.5	0.3	30	305.8	650.8	578.0	715.8	687.5
23	150	0.3	0.1	0.5	0.3	0.1	0.5	30	301.9	609.3	723.9	301.8	681.3
24	150	0.3	0.1	0.5	0.5	0.3	0.1	30	696.6	723.8	698.4	682.9	727.1
25	150	0.5	0.3	0.1	0.1	0.5	0.3	30	718.9	733.5	621.8	719.3	728.8
26	150	0.5	0.3	0.1	0.3	0.1	0.5	30	724.9	726.1	310.5	738.6	723.8
27	150	0.5	0.3	0.1	0.5	0.3	0.1	30	703.6	670.1	675.9	661.1	682.0

Table 7: Taguchi experimental layout and related experimental data (GA)

Experiment	Parameter						Computing time (s)	Experimental result				
	ps	c1	c2	c3	m1	m2		Run 1	Run 2	Run 3	Run 4	Run 5
1	50	0.1	0.1	0.1	0.1	0.1	30	304.1	295.9	296.5	293.5	296.7
2	50	0.1	0.1	0.1	0.3	0.3	30	301.4	620.5	611.6	697.2	311.0
3	50	0.1	0.1	0.1	0.5	0.5	30	698.5	296.8	276.4	491.8	705.3
4	50	0.3	0.3	0.3	0.1	0.1	30	295.6	275.2	585.8	294.2	295.3
5	50	0.3	0.3	0.3	0.3	0.3	30	300.7	558.9	297.7	296.6	735.7
6	50	0.3	0.3	0.3	0.5	0.5	30	291.8	678.1	308.0	724.7	690.0
7	50	0.5	0.5	0.5	0.1	0.1	30	641.1	306.8	304.6	305.8	304.5
8	50	0.5	0.5	0.5	0.3	0.3	30	298.7	295.4	303.3	721.8	306.6
9	50	0.5	0.5	0.5	0.5	0.5	30	305.6	290.3	400.2	301.1	309.4
10	100	0.1	0.3	0.5	0.1	0.3	30	741.9	711.1	294.2	308.4	723.5
11	100	0.1	0.3	0.5	0.3	0.5	30	301.5	701.7	560.3	677.2	730.2
12	100	0.1	0.3	0.5	0.5	0.1	30	300.1	303.5	677.2	310.8	723.8
13	100	0.3	0.5	0.1	0.1	0.3	30	596.8	496.8	680.7	296.9	301.0
14	100	0.3	0.5	0.1	0.3	0.5	30	308.5	723.0	305.0	302.2	297.8
15	100	0.3	0.5	0.1	0.5	0.1	30	296.8	302.1	660.4	730.5	719.0
16	100	0.5	0.1	0.3	0.1	0.3	30	307.9	690.6	505.8	485.5	300.8
17	100	0.5	0.1	0.3	0.3	0.5	30	727.9	310.8	747.9	301.9	681.0
18	100	0.5	0.1	0.3	0.5	0.1	30	709.1	295.4	705.1	710.8	700.5
19	150	0.1	0.5	0.3	0.1	0.5	30	700.5	727.2	662.3	310.0	735.3
20	150	0.1	0.5	0.3	0.3	0.1	30	630.5	688.1	640.1	382.7	296.8
21	150	0.1	0.5	0.3	0.5	0.3	30	303.7	663.4	579.9	309.3	711.2
22	150	0.3	0.1	0.5	0.1	0.5	30	309.3	301.6	731.9	671.9	310.4
23	150	0.3	0.1	0.5	0.3	0.1	30	620.6	301.8	680.7	706.1	671.9
24	150	0.3	0.1	0.5	0.5	0.3	30	300.9	298.1	713.3	714.7	302.6
25	150	0.5	0.3	0.1	0.1	0.5	30	733.8	639.4	690.6	623.6	310.6
26	150	0.5	0.3	0.1	0.3	0.1	30	437.3	686.7	689.1	305.2	653.8
27	150	0.5	0.3	0.1	0.5	0.3	30	735.5	714.9	733.0	657.7	663.3

Table 8: Taguchi experimental layout and related experimental data (PSO)

Experiment	Parameter				Computing time (s)	Experimental result				
	np	ic	cc	sc		Run 1	Run 2	Run 3	Run 4	Run 5
1	50	0.5	1	1	30	668.9	707.4	502.3	308.5	603.7
2	50	0.7	2	2	30	290.8	657.3	639.4	663.6	655.3
3	50	0.9	3	3	30	495.2	292.0	286.2	283.2	290.6
4	100	0.5	2	3	30	711.6	700.0	293.1	300.9	297.2
5	100	0.7	3	1	30	672.0	700.5	680.0	621.3	415.8
6	100	0.9	1	2	30	291.6	657.3	618.0	297.7	659.3
7	150	0.5	3	2	30	718.6	290.5	718.7	291.9	715.2
8	150	0.7	1	3	30	707.7	682.9	303.2	604.3	635.8
9	150	0.9	2	1	30	678.4	302.1	306.3	304.7	303.9

Table 9: Taguchi experimental layout and related experimental data (SA)

Experiment	Parameter			Computing time (s)	Experimental result				
	it	cc	nt		Run 1	Run 2	Run 3	Run 4	Run 5
1	50	0.75	25	30	669.5	629.0	291.8	267.6	294.0
2	50	0.85	50	30	294.4	278.9	634.3	297.0	575.1
3	50	0.95	75	30	299.2	297.5	299.2	354.1	22.6
4	100	0.75	50	30	542.2	643.6	473.4	299.5	295.2
5	100	0.85	75	30	655.8	674.7	595.5	608.3	295.2
6	100	0.95	25	30	298.1	296.7	294.0	295.3	302.7
7	150	0.75	75	30	300.5	299.4	260.5	603.9	296.7
8	150	0.85	25	30	541.5	172.5	602.5	523.8	682.2
9	150	0.95	50	30	317.2	647.4	643.0	288.8	520.3

With the data from Taguchi experiments (Tables 6-10), the main effects of the parameters as shown in Figs. 26-30 are plotted, which show the effects of the parameters on the objective function value or the fitness value. From these five main effect plots, the parameters of the algorithms are tuned as Table 10. These parameters stay fixed for all 18 test cases of Table 1, and the performance of the algorithms is compared in the next Section.

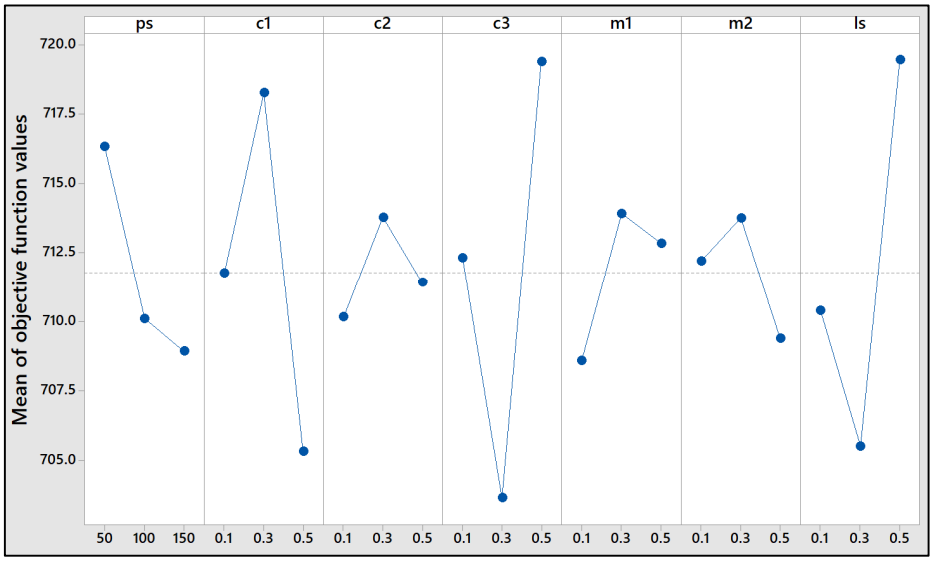


Fig. 26: Main effect plot of the proposed MA

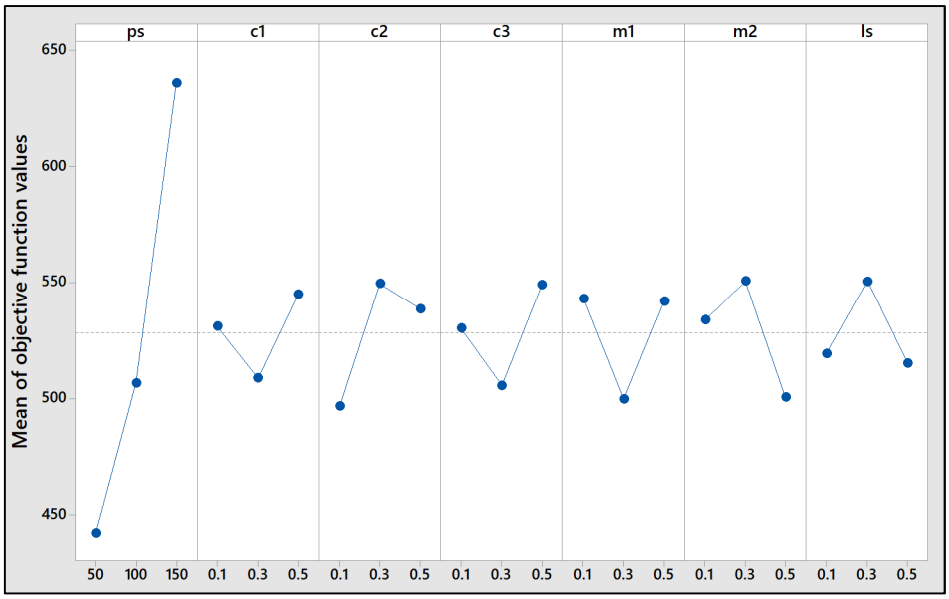


Fig. 27: Main effect plot of the traditional MA

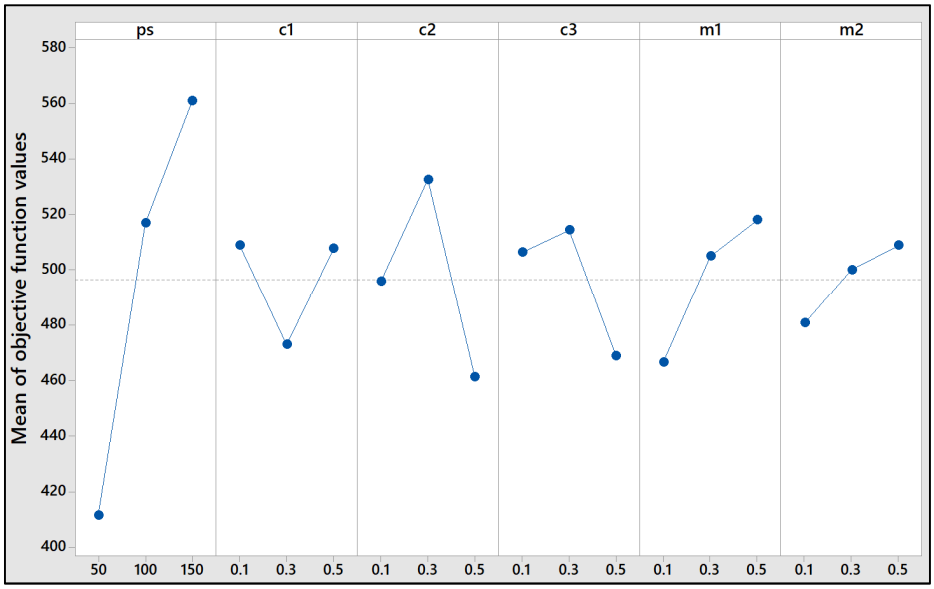


Fig. 28: Main effect plot of GA

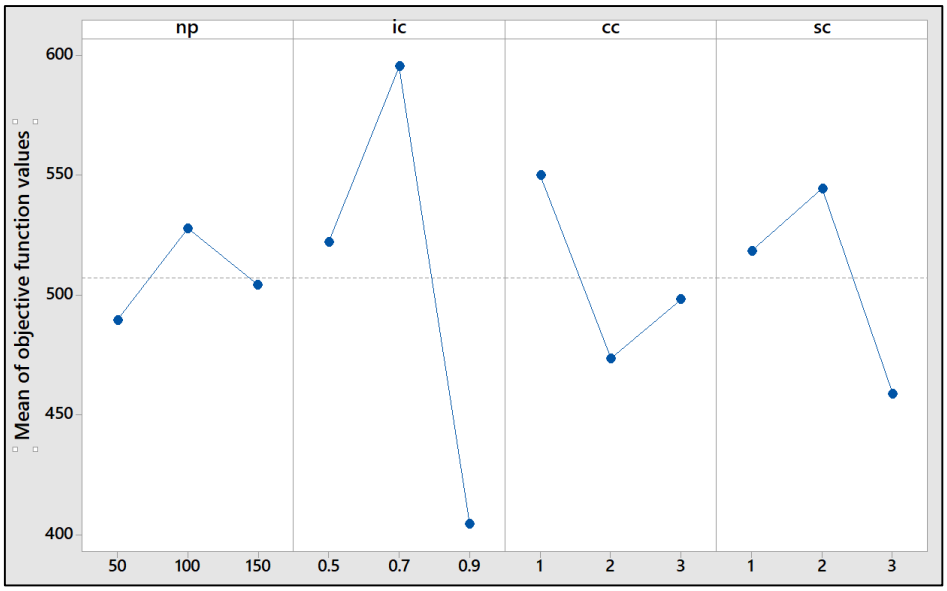


Fig. 29: Main effect plot of PSO

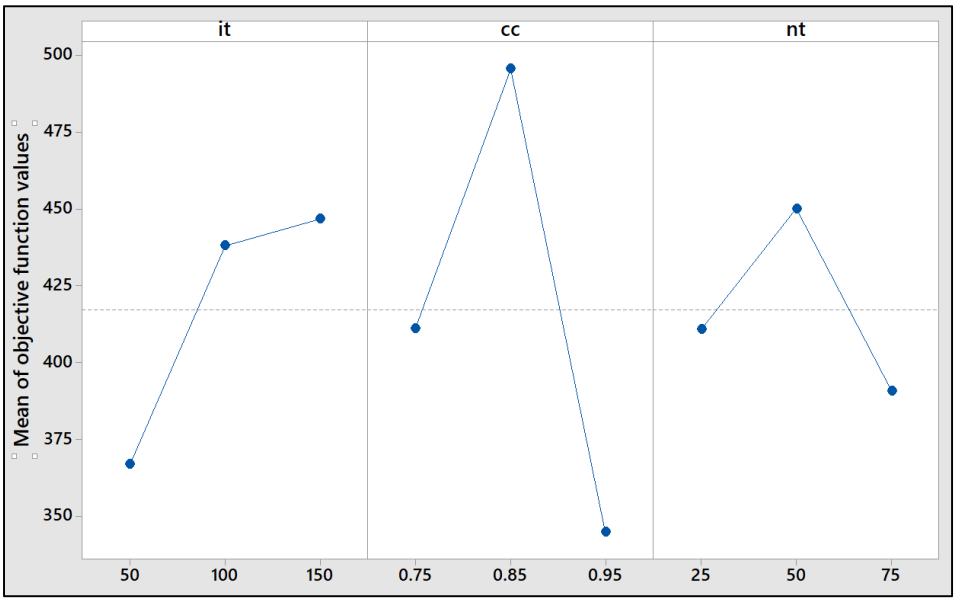


Fig. 30: Main effect plot of SA

Table 10: Tuned parameters of the algorithms

No.	Parameter	Code	SA	PSO	GA	Traditional MA	Proposed MA
1	Initial temperature	it	150	-	-	-	-
2	Cooling coefficient	cc	0.85	-	-	-	-
3	Number of trials at each temperature level	nt	50	-	-	-	-
4	Number of particles	np	-	100	-	-	-
5	Inertia constant	ic	-	0.7	-	-	-
6	Cognative constant	cc	-	1	-	-	-
7	Social constant	sc	-	2	-	-	-
8	Population size	ps	-	-	150	150	50
9	Rate of crossover 1	c1	-	-	0.1	0.5	0.3
10	Rate of crossover 2	c2	-	-	0.3	0.3	0.3
11	Rate of crossover 3	c3	-	-	0.3	0.5	0.5
12	Rate of mutation 1	m1	-	-	0.5	0.1	0.3
13	Rate of mutation 2	m2	-	-	0.5	0.3	0.3
14	Rate of local search	ls	-	-	-	0.3	0.5

6. Results and Discussions

To validate the effectiveness of the proposed MA, a comprehensive comparative study has been conducted, and various insights on the behaviour of the proposed algorithm as well as its potential applications in other domains are provided herein. The effectiveness of the proposed MA in 18 test cases is compared with those of 4 well-known optimization algorithms in the literature, i.e., SA, PSO, GA, and the traditional MA.

Performances of SA, PSO, GA, the traditional MA, and the proposed MA in 18 test cases are reported in Table 11. For each test case, each optimization algorithm was independently run 100 times to get a consistent result; and a set of performance measures including min, max, average, and standard deviation (std) of the fitness values (the values of the objective function found by different optimization algorithms) are reported. It is noted that the SADP is a maximization problem (as mentioned in Section 3) – therefore, the larger the fitness values, the better.

Table 11: Performance comparison of the algorithms

Test case	Number of independent runs	Simulated Annealing				Particle Swarm Optimization				Genetic Algorithm				Traditional MA				Proposed MA			
		Min	Max	Average	Std	Min	Max	Average	Std	Min	Max	Average	Std	Min	Max	Average	Std	Min	Max	Average	Std
1	100	39.3	165.4	100.8	44.1	54.0	166.7	115.7	54.1	54.5	166.2	111.8	42.9	54.4	166.2	105.1	46.3	91.1	166.8	160.1	16.7
2	100	50.7	165.7	126.1	40.7	54.1	166.4	102.5	53.9	54.5	166.5	112.5	45.8	54.2	167.2	115.1	48.6	118.7	167.3	165.0	4.9
3	100	54.4	165.4	143.0	33.8	53.6	166.7	106.9	54.0	54.5	166.6	111.5	42.7	54.4	166.4	106.0	47.9	161.1	167.5	165.8	0.9
4	100	58.3	311.8	180.2	74.6	109.1	334.4	193.2	98.8	109.7	328.2	201.7	86.2	109.8	328.0	217.6	80.7	107.0	332.1	279.0	58.8
5	100	109.5	311.3	237.9	61.8	108.8	334.9	197.0	100.2	109.8	334.4	210.6	81.0	109.8	333.0	225.0	90.9	268.4	334.5	321.0	10.6
6	100	109.5	321.2	255.9	53.8	107.8	333.3	186.8	98.4	109.9	332.6	218.1	82.8	110.0	333.5	225.8	89.2	294.6	334.7	322.3	8.7
7	100	97.6	434.9	246.3	87.4	162.3	464.1	247.9	116.7	165.6	449.3	264.2	104.2	164.9	445.6	286.3	109.3	210.7	464.4	397.2	60.6
8	100	88.5	413.9	296.3	85.8	164.1	466.0	284.4	127.6	166.0	455.5	320.8	99.5	165.8	454.6	300.3	115.5	349.1	457.2	427.3	20.8
9	100	165.5	424.4	326.7	73.8	162.1	453.9	254.7	117.7	166.0	453.9	322.4	108.0	165.8	460.2	319.3	121.7	392.4	462.1	437.2	16.7
10	100	84.6	714.6	474.5	184.4	285.3	726.1	460.3	185.3	297.9	742.0	611.8	164.6	300.5	746.1	608.0	165.2	561.0	750.2	711.8	30.7
11	100	176.3	727.6	545.4	166.2	287.0	737.9	544.2	176.4	299.3	748.5	627.3	151.8	298.4	747.2	597.3	186.6	685.5	748.3	727.2	13.0
12	100	293.3	742.9	617.8	113.2	280.3	727.1	528.5	182.6	298.3	747.3	641.3	138.0	295.2	749.2	620.8	181.9	692.1	751.3	730.3	11.2
13	100	142.3	1022.5	649.7	232.4	410.2	1098.4	767.7	243.0	460.8	1096.1	865.0	217.3	458.4	1089.7	793.6	243.6	903.3	1112.3	1017.0	48.2
14	100	148.6	1059.0	752.9	211.5	436.5	1073.5	776.7	231.2	460.3	1120.1	911.0	218.4	464.3	1111.0	880.2	247.0	932.8	1126.0	1049.2	38.6
15	100	453.7	1038.5	860.7	147.5	418.0	1070.9	742.5	242.3	466.6	1118.1	893.1	217.2	465.0	1126.7	881.7	274.8	952.7	1120.1	1064.5	33.1
16	100	168.7	1287.3	839.3	243.7	539.8	1290.4	875.6	256.7	629.6	1384.1	1095.5	225.1	619.6	1378.0	1045.6	254.5	990.6	1373.3	1231.6	75.6
17	100	602.6	1294.2	971.1	197.1	575.7	1273.4	896.8	255.8	630.3	1378.4	1138.7	230.0	631.0	1423.2	1131.4	238.6	1045.3	1404.1	1258.8	61.3
18	100	474.5	1268.9	979.3	207.4	527.7	1316.6	915.7	249.0	632.1	1378.5	1099.0	251.1	632.8	1425.3	1136.7	281.5	1111.8	1419.8	1277.2	55.8

The comparison results as shown in Table 11 reveal that the proposed MA outperforms SA, PSO, GA and the traditional MA in all 18 test cases considering all four performance measures (min, max, average, and standard deviation (std) of the fitness values). In terms of the solution quality measured by the "Average" value, on average, in all 18 test cases, the proposed MA provided 36.5%, 43.3%, 20.4%, and 22.4% better solution compared to SA, PSO, GA, and the traditional MA, respectively. In addition, in terms of the solution quality measured by the "Max" value, on average, in all 18 test cases, the proposed MA was 6.9, 4.0, 1.0, and 0.3% better than SA, PSO, GA, and the traditional MA, respectively. Furthermore, in terms of the solution quality measured by the "Min" value, on average, in all 18 test cases, the proposed MA performed 197.4, 108.3, 91.0, and 91.4% better than SA, PSO, GA, and the traditional MA, respectively. The consistency (robustness) of the algorithms in finding the same final solution is measured through the std value. In this regard, the proposed MA outperformed SA, PSO, GA, and the traditional MA, with 74.9%, 80.1%, 77.4%, and 80.0% better, respectively. The overall performance comparison in terms of the objective function values (the fitness values) found by 5 algorithms is shown in Fig. 31. Clearly, the proposed MA is better than the 4 benchmark algorithms.

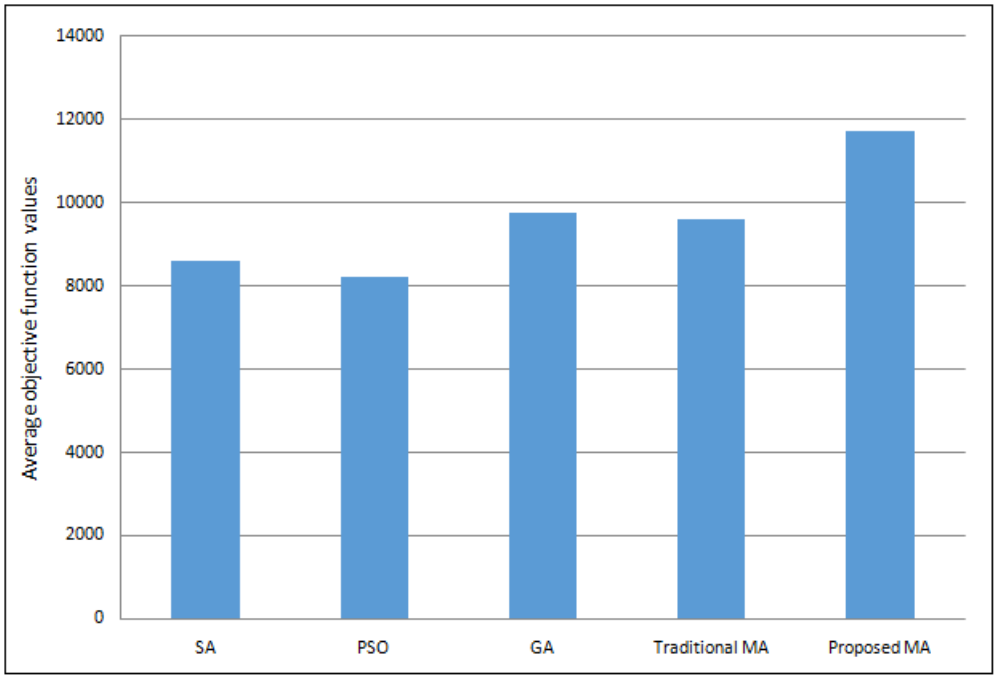


Fig. 31: Visualization of the overall performance comparison

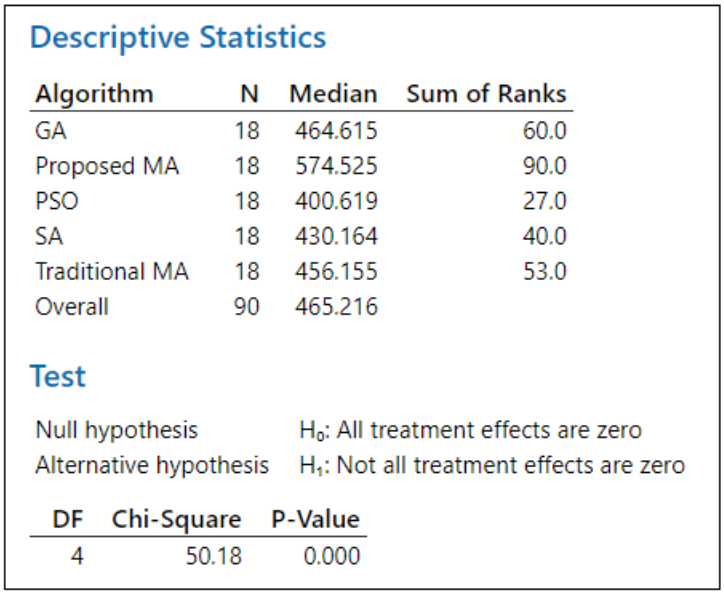
It is noted that like other stochastic optimization algorithms, SA, PSO, GA, the traditional MA and the proposed MA could not provide exactly the same solutions in different runs - that is why in this research, each optimization algorithm was independently run 100 times to get a consistent result, as mentioned before. In addition, to deal with the stochastic nature of the stochastic optimization algorithms, well-known statistical tests (i.e., T-test and the Friedman test) were used here to compare and validate the performances of the optimization algorithms.

A set of 24 T-tests were conducted to confirm the outperformance of the proposed MA, against SA, PSO, GA, and the traditional MA. The results of these tests have been provided in Table 12. Based on the solution scales, 18 test cases were divided into 6 groups for more accurate comparison. In each group, four T-tests for four hypotheses (i.e., the proposed MA is significantly better than SA, the proposed MA is significantly better than PSO, the proposed MA is significantly better than GA, and the proposed MA is significantly better than the traditional MA) were conducted. Statistically speaking, the results of Table 12 confirms that the proposed MA is significantly better than SA, PSO, GA, and the traditional MA, with 95% confidence intervals.

Table 12: Set of T-test for confirming the outperformance of the proposed MA

Group	Test case	Hypothesis	Mean difference	P value
1	1 - 3	The proposed MA is significantly better than SA	40.3	0.042
		The proposed MA is significantly better than PSO	55.3	0.003
		The proposed MA is significantly better than GA	51.7	0.001
		The proposed MA is significantly better than the traditional MA	54.9	0.001
2	4 - 6	The proposed MA is significantly better than SA	82.8	0.027
		The proposed MA is significantly better than PSO	115.1	0.008
		The proposed MA is significantly better than GA	97.3	0.011
		The proposed MA is significantly better than the traditional MA	84.6	0.014
3	7 - 9	The proposed MA is significantly better than SA	130.8	0.019
		The proposed MA is significantly better than PSO	158.2	0.001
		The proposed MA is significantly better than GA	118.1	0.007
		The proposed MA is significantly better than the traditional MA	118.6	0.002
4	10 - 12	The proposed MA is significantly better than SA	177.2	0.026
		The proposed MA is significantly better than PSO	212.1	0.008
		The proposed MA is significantly better than GA	96.3	0.001
		The proposed MA is significantly better than the traditional MA	114.1	0.001
5	13 - 15	The proposed MA is significantly better than SA	289.2	0.022
		The proposed MA is significantly better than PSO	281.3	0.001
		The proposed MA is significantly better than GA	153.9	0.002
		The proposed MA is significantly better than the traditional MA	191.8	0.014
6	16 - 18	The proposed MA is significantly better than SA	326.0	0.010
		The proposed MA is significantly better than PSO	359.9	0.001
		The proposed MA is significantly better than GA	144.8	0.002
		The proposed MA is significantly better than the traditional MA	151.3	0.021

Furthermore, in order to evaluate the overall performance of the five algorithms in all 18 test cases, the Friedman test was conducted using Minitab and the result is shown in Fig. 32. As it can be seen from Fig. 32, the proposed MA outperforms the traditional MA, GA, PSO, as well as SA. More specifically, the proposed MA has the median of 574.525 while the traditional MA, GA, PSO, and SA have 456.155, 464.615, 400.619, and 430.164, respectively. In addition, the sum of ranks of the proposed MA is 90, while those of the traditional MA, GA, PSO, and SA are 53, 60, 27, and 40, respectively. Moreover, the very small P-value (0.000) indicates that the performance differences between the proposed MA and the other algorithms are statistically significant.



Descriptive Statistics

Algorithm	N	Median	Sum of Ranks
GA	18	464.615	60.0
Proposed MA	18	574.525	90.0
PSO	18	400.619	27.0
SA	18	430.164	40.0
Traditional MA	18	456.155	53.0
Overall	90	465.216	

Test

Null hypothesis	H_0: All treatment effects are zero
Alternative hypothesis	H_1: Not all treatment effects are zero

DF	Chi-Square	P-Value
4	50.18	0.000

Fig. 32: The result of the Friedman test

The results from T-test and Friedman test as shown above confirm that the proposed MA outperforms the traditional MA, GA, PSO, and SA. There are two main reasons behind the success of the proposed MA. First, the proposed MA has multiple crossover operators, multiple mutation operators and a local search; therefore, it can explore the complex search space of the problem better than the traditional MA, GA, PSO, and SA. Second, the proposed MA has a mechanism to automatically restart its search process if it gets stuck in the local optima. With these two innovations, performance of the proposed MA is significantly enhanced, and it is better than the existing optimization algorithms in the literature. These innovations in the proposed MA can be applied to other stochastic search-based optimization algorithms such as GA and PSO to enhance their performances. In addition, parallel computation techniques can be used to execute the crossover, mutation, and local search operators to shorten the computing time. Furthermore, the proposed MA can be customized to solve various real-world optimization problems such as product planning and scheduling optimization problems, vehicle routing problems, timetable scheduling problems, work-flow scheduling problems in hotels and hospitals, etc. For more details about the innovative structure of the proposed MA, it is advised to refer to Fig. 4.

Regarding the scalability and potential applications in other research fields, the proposed MA can be extended to solve other optimization problems with the same and/or larger number of decision variables. To do so, one needs to update the chromosome encoding, crossover,

mutation, objective function, and constraints; the rest of the MA can be kept the same. In addition, to maximize the algorithm performance when solving new problems, its parameter set (i.e., population size, crossover rate, mutation rate, and local search rate) needs to be tuned again using Taguchi experimental design method as shown in Section 5.

Although the proposed MA works very well in solving the problem under investigation (i.e., spatial area determination problem), it might have the following limits when being applied to solve highly constrained large-scale optimization problems in other research fields. First, there is no general method or rule for selecting the effective chromosome encoding for the proposed MA to solve a given optimization problem – which depends on the problem to be solved. Second, designing the effective crossovers and mutations requires a deep knowledge of the problem (especially the constraints) and the user experience in stochastic optimization algorithms. Finally, finding the optimal parameters of the proposed MA, using Taguchi experimental design method, might be complex and time-consuming. Each optimal parameter set should be applied to one problem only; for a new problem, one should do the parameter tuning again. In addition, a large number of experiments and ANOVA analysis are required in each parameter tuning as shown in Tables 5-9 and Figs. 26-30 in Section 5.

7. Conclusions and Future Work

In this paper, the spatial area determination problem, which can be found in various domains such as hydrographic survey planning, conservation planning, military planning, etc., has been defined and formulated. To solve the problem, an innovative solution method based on MA has been developed. For this aim, 18 test cases were used to demonstrate the outperformance of the proposed MA. The results indicate that in terms of solution quality, on average, the proposed MA provided 36.5%, 43.3%, 20.4%, and 22.4% better solution; and in terms of consistency, the proposed MA has 74.9, 80.1, 77.4, and 80.0% better consistency, compared to four popular optimization algorithms, namely SA, PSO, GA, and the traditional MA, respectively. Finally, a set of 24 T-tests and the Friedman test confirmed that the proposed MA is significantly better than the four benchmarking optimization algorithms.

As the future research directions, the applicability of the SADP in reality will be investigated, and the robustness of the proposed MA on more complex spatial data will be tested. In addition, finding the Big O notation, for example O(n!) or O(2^x), to express the

computational complexity of the proposed MA (a stochastic optimization problem) will be done in the future. Furthermore, developing a multi-objective mathematical model for the SADP with conflicting objectives could be an interesting research direction. Moreover, other meta-heuristic algorithms can be developed that may outperform the proposed MA in terms of solution quality and computational effort. In this regard, data-driven meta-heuristics can be developed that use machine learning techniques to improve the performance of classical meta-heuristics [71, 72]. In fact, data-driven meta-heuristics have shown superior performance when solving a wide variety of optimization problems [37, 38, 73]. Finally, parallel computation techniques can be used to execute the crossover, mutation, and local search operators of the proposed MA as shown in Fig. 4 to shorten the computing time.

Declaration of Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Beal, V. *Spatial data*. Access date: 06 June 2020; Available from: https://www.webopedia.com/TERM/S/spatial_data.html.

[2] Gwaleba, M.J. and U.E. Chigbu, *Participation in property formation: Insights from land-use planning in an informal urban settlement in Tanzania.* Land Use Policy, 2020. 92: p. 104482.

[3] Yang, F., et al., *Efficiency of unlocking or locking existing protected areas for identifying complementary areas for biodiversity conservation.* Science of The Total Environment, 2019. 694: p. 133771.

[4] Pınarbaşı, K., et al., *Key issues for a transboundary and ecosystem-based maritime spatial planning in the Bay of Biscay.* Marine Policy, 2020. 120: p. 104131.

[5] Liao, S.H., B.L. Sun, and R.Y. Wang, *A knowledge-based architecture for planning military intelligence, surveillance, and reconnaissance.* Space Policy, 2003. 19(3): p. 191-202.

[6] Dao, S.D., et al., *A Hybrid Iterated Greedy Algorithm for Hydrographic Survey Routing Problem.* Marine Geodesy, 2021: p. 1-26.

[7] Dao, S.D., et al., *An Innovative Genetic Algorithm for Spatial Zoning Optimization*, in *The 21st annual congress of the French society of operational research and decision support, 19-21 February, Montpellier, France*. 2020.

[8] Shom. *French National Hydrographic and Oceanographic Service*. 2021; Available from: https://www.shom.fr/.

[9] Hsu, Y.T. and S. Peeta, *Risk-based spatial zone determination problem for stage-based evacuation operations.* Transportation Research Part C: Emerging Technologies, 2014. 41: p. 73-89.

[10] Klein, C.J., et al., *Spatial marine zoning for fisheries and conservation.* Frontiers in Ecology and the Environment, 2010. 8(7): p. 349-353.

[11] Geneletti, D. and I. van Duren, *Protected area zoning for conservation and use: A combination of spatial multicriteria and multiobjective evaluation.* Landscape and Urban Planning, 2008. 85(2): p. 97-110.

[12] Kazemzadeh-Zow, A., et al., *A spatial zoning approach to calibrate and validate urban growth models.* International Journal of Geographical Information Science, 2017. 31(4): p. 763-782.

[13] Hobday, A.J., et al., *Dynamic spatial zoning to manage southern bluefin tuna (Thunnus maccoyii) capture in a multi-species longline fishery.* Fisheries Oceanography, 2010. 19(3): p. 243-253.

[14] Villa, F., L. Tunesi, and T. Agardy, *Zoning Marine Protected Areas through Spatial Multiple-Criteria Analysis: the Case of the Asinara Island National Marine Reserve of Italy.* Conservation Biology, 2002. 16(2): p. 515-526.

[15] Habtemariam, B.T. and Q. Fang, *Zoning for a multiple-use marine protected area using spatial multi-criteria analysis: The case of the Sheik Seid Marine National Park in Eritrea.* Marine Policy, 2016. 63: p. 135-143.

[16] Crossman, N., et al., *OSS: A spatial decision support system for optimal zoning of marine protected areas*, in *Proceedings of the international congress on modelling and simulation*, Z. Andre and M.A. Robert, Editors. 2005: 12-15 December, Melbourne, Australia, pp.1525-1531.

[17] Wie, B.C. and W.Y. Chai, *An Intelligent GIS-Based Spatial Zoning System with Multiobjective Hybrid Metaheuristic Method*, in *Innovations in Applied Artificial*

Intelligence, B. Orchard, C. Yang, and M. Ali, Editors. 2004, Springer: Berlin. p. 769-778.

[18] Li, X., et al., *Coupling urban cellular automata with ant colony optimization for zoning protected natural areas under a changing landscape.* International Journal of Geographical Information Science, 2011. 25(4): p. 575-593.

[19] Vu, K.K., et al., *Surrogate-based methods for black-box optimization.* International Transactions in Operational Research, 2017. 24(3): p. 393-424.

[20] Hanagandi, V. and M. Nikolaou, *A hybrid approach to global optimization using a clustering algorithm in a genetic search framework.* Computers & Chemical Engineering, 1998. 22(12): p. 1913-1925.

[21] Liberti, L. and S. Kucherenko, *Comparison of deterministic and stochastic approaches to global optimization.* International Transactions in Operational Research, 2005. 12(3): p. 263-285.

[22] Moles, C.G., P. Mendes, and J.R. Banga, *Parameter estimation in biochemical pathways: a comparison of global optimization methods.* Genome Research, 2003. 13(11): p. 2467–2474.

[23] Dao, S.D., K. Abhary, and R. Marian, *An improved structure of genetic algorithms for global optimisation.* Progress in Artificial Intelligence, 2016. 5(3): p. 155-163.

[24] Dao, S.D., K. Abhary, and R. Marian, *An innovative framework for designing genetic algorithm structures.* Expert Systems with Applications, 2017. 90: p. 196-208.

[25] Imran Hossain, S., et al., *Optimization of University Course Scheduling Problem using Particle Swarm Optimization with Selective Search.* Expert Systems with Applications, 2019. 127: p. 9-24.

[26] Papa, J.P., W. Scheirer, and D.D. Cox, *Fine-tuning Deep Belief Networks using Harmony Search.* Applied Soft Computing, 2016. 46: p. 875-885.

[27] Rakhshani, H. and A. Rahati, *Snap-drift cuckoo search: A novel cuckoo search optimization algorithm.* Applied Soft Computing, 2017. 52: p. 771-794.

[28] Olivas, F., et al., *Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems.* Applied Soft Computing, 2017. 53: p. 74-87.

[29] Zamli, K.Z., B.Y. Alkazemi, and G. Kendall, *A Tabu Search hyper-heuristic strategy for t-way test suite generation.* Applied Soft Computing, 2016. 44: p. 57-74.

[30] Beigi, A.M. and A. Maroosi, *Parameter identification for solar cells and module using a Hybrid Firefly and Pattern Search Algorithms.* Solar Energy, 2018. 171: p. 435-446.

[31] Sánchez-Oro, J. and A. Duarte, *Iterated Greedy algorithm for performing community detection in social networks.* Future Generation Computer Systems, 2018. 88: p. 785-791.

[32] Leite, N., F. Melício, and A.C. Rosa, *A fast simulated annealing algorithm for the examination timetabling problem.* Expert Systems with Applications, 2019. 122: p. 137-151.

[33] Al-Betar, M.A., et al., *Adaptive β−hill climbing for optimization.* Soft Computing, 2019. 23(24): p. 13489-13512.

[34] Nayak, J., et al., *Hyper-parameter tuned light gradient boosting machine using memetic firefly algorithm for hand gesture recognition.* Applied Soft Computing, 2021. 107: p. 107478.

[35] Khalilpourazari, S. and S. Khalilpourazary, *An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems.* Soft Computing, 2019. 23(5): p. 1699-1722.

[36] Chaudhary, D. and B. Kumar, *Cost optimized Hybrid Genetic-Gravitational Search Algorithm for load scheduling in Cloud Computing.* Applied Soft Computing, 2019. 83: p. 105627.

[37] Karimi-Mamaghan, M., et al., *A learning-based metaheuristic for a multi-objective agile inspection planning model under uncertainty.* European Journal of Operational Research, 2020. 285(2): p. 513-537.

[38] Karimi-Mamaghan, M., et al., *Hub-and-spoke network design under congestion: A learning based metaheuristic.* Transportation Research Part E: Logistics and Transportation Review, 2020. 142: p. 102069.

[39] Deng, J. and L. Wang, *A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem.* Swarm and Evolutionary Computation, 2017. 32: p. 121-131.

[40] Decerle, J., et al., *A memetic algorithm for a home health care routing and scheduling problem.* Operations Research for Health Care, 2018. 16: p. 59-71.

[41] Kurdi, M., *A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem.* Applied Soft Computing, 2020. 94: p. 106458.

[42] Gong, G., et al., *An effective memetic algorithm for multi-objective job-shop scheduling.* Knowledge-Based Systems, 2019. 182: p. 104840.

[43] Pereira, J., M. Ritt, and Ó.C. Vásquez, *A memetic algorithm for the cost-oriented robotic assembly line balancing problem.* Computers & Operations Research, 2018. 99: p. 249-261.

[44] Zhou, Q., U. Benlic, and Q. Wu, *An opposition-based memetic algorithm for the maximum quasi-clique problem.* European Journal of Operational Research, 2020. 286(1): p. 63-83.

[45] Nagy, B. and E.V. Moisi, *Memetic algorithms for reconstruction of binary images on triangular grids with 3 and 6 projections.* Applied Soft Computing, 2017. 52: p. 549-565.

[46] Lu, Y., et al., *Memetic algorithm for the multiple traveling repairman problem with profits.* Engineering Applications of Artificial Intelligence, 2019. 80: p. 35-47.

[47] Eremeev, A.V. and Y.V. Kovalenko, *A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem.* Memetic Computing, 2020. 12(1): p. 23-36.

[48] Huang, T., et al., *A Niching Memetic Algorithm for Multi-Solution Traveling Salesman Problem.* IEEE Transactions on Evolutionary Computation, 2020. 24(3): p. 508-522.

[49] Yadegari, E., A. Alem-Tabriz, and M. Zandieh, *A memetic algorithm with a novel neighborhood search and modified solution representation for closed-loop supply chain network design.* Computers & Industrial Engineering, 2019. 128: p. 418-436.

[50] Yang, X., N. Bostel, and P. Dejax, *A MILP model and memetic algorithm for the Hub Location and Routing problem with distinct collection and delivery tours.* Computers & Industrial Engineering, 2019. 135: p. 105-119.

[51] Rahman, H.F., R.K. Chakrabortty, and M.J. Ryan, *Memetic algorithm for solving resource constrained project scheduling problems.* Automation in Construction, 2020. 111: p. 103052.

[52] Alsmady, A., et al. *Workflow Scheduling in Cloud Computing Using Memetic Algorithm*. in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. 2019.

[53] Yağmur, E. and S.E. Kesen, *A memetic algorithm for joint production and distribution scheduling with due dates.* Computers & Industrial Engineering, 2020. 142: p. 106342.

[54] Jin, L., et al., *A Neutrosophic Number-Based Memetic Algorithm for the Integrated Process Planning and Scheduling Problem With Uncertain Processing Times.* IEEE Access, 2020. 8: p. 96628-96648.

[55] Decerle, J., et al., *A memetic algorithm for multi-objective optimization of the home health care problem.* Swarm and Evolutionary Computation, 2019. 44: p. 712-727.

[56] Abedi, M., et al., *A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines.* Expert Systems with Applications, 2020. 157: p. 113348.

[57] Zhang, W., X. Wang, and K. Yang, *Uncertain multi-objective optimization for the water–rail–road intermodal transport system with consideration of hub operation process using a memetic algorithm.* Soft Computing, 2020. 24(5): p. 3695-3709.

[58] Spencer, K.Y., P.V. Tsvetkov, and J.J. Jarrell, *A greedy memetic algorithm for a multiobjective dynamic bin packing problem for storing cooling objects.* Journal of Heuristics, 2019. 25(1): p. 1-45.

[59] Pistolesi, F. and B. Lazzerini, *TeMA: A Tensorial Memetic Algorithm for Many-Objective Parallel Disassembly Sequence Planning in Product Refurbishment.* IEEE Transactions on Industrial Informatics, 2019. 15(6): p. 3743-3753.

[60] Sun, J., et al., *Interval Multiobjective Optimization With Memetic Algorithms.* IEEE Transactions on Cybernetics, 2019: p. 1-14.

[61] Ghosh, M., et al., *Recursive Memetic Algorithm for gene selection in microarray data.* Expert Systems with Applications, 2019. 116: p. 172-185.

[62] Ruiz, L.G.B., M.I. Capel, and M.C. Pegalajar, *Parallel memetic algorithm for training recurrent neural networks for the energy efficiency problem.* Applied Soft Computing, 2019. 76: p. 356-368.

[63] Wang, S., et al., *Preventing epidemic spreading in networks by community detection and memetic algorithm.* Applied Soft Computing, 2020. 89: p. 106118.

[64] Baliarsingh, S.K., et al., *A memetic algorithm using emperor penguin and social engineering optimization for medical data classification.* Applied Soft Computing, 2019. 85: p. 105773.

[65] Ghosh, M., et al., *Feature Selection for Handwritten Word Recognition Using Memetic Algorithm*, in *Advances in Intelligent Computing*, J.K. Mandal, P. Dutta, and S. Mukhopadhyay, Editors. 2019, Springer Singapore. p. 103-124.

[66] Sadeghiram, S., H. Ma, and G. Chen. *Composing Distributed Data-intensive Web Services Using a Flexible Memetic Algorithm*. in *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2019.

[67] de Lima Corrêa, L. and M. Dorn, *A multi-population memetic algorithm for the 3-D protein structure prediction problem.* Swarm and Evolutionary Computation, 2020. 55: p. 100677.

[68] Gen, M. and R. Cheng, *Genetic Algorithms and Engineering Design*. 2000: John Wiley & Sons, Inc.

[69] Al-Adwan, A., A. Sharieh, and B.A. Mahafzah, *Parallel heuristic local search algorithm on OTIS hyper hexa-cell and OTIS mesh of trees optoelectronic architectures.* Applied Intelligence, 2019. 49(2): p. 661-688.

[70] Yang, K. and B. EI-Haik, *Design for Six Sigma: A Roadmap for Product Development*. 2003: McGraw Hill Professional.

[71] Karimi-Mamaghan, M., et al., *Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art.* European Journal of Operational Research, 2022. 296(2): p. 393-422.

[72] Song, H., I. Triguero, and E. Özcan, *A review on the self and dual interactions between machine learning and optimisation.* Progress in Artificial Intelligence, 2019. 8(2): p. 143-165.

[73] Karimi-Mamaghan, M., et al., *Learning to select operators in meta-heuristics: An integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem.* European Journal of Operational Research, 2022.