Contents lists available at SciVerse ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular paper

A critical assessment of metaheuristics for scheduling emergency infrastructure inspections

Nikos D. Lagaros[a], Matthew G. Karlaftis[b,*]

[a] Institute of Structural Analysis & Seismic Research, National Technical University of Athens, 9, Iroon Polytechniou Str., Zografou Campus, GR-15780 Athens, Greece
[b] Department of Transportation Planning and Engineering, National Technical University of Athens, 9, Iroon Polytechniou Str., Zografou Campus, GR-15780 Athens, Greece

ARTICLE INFO

Article history:
Received 19 December 2010
Received in revised form
4 June 2011
Accepted 10 June 2011
Available online 14 July 2011

Keywords:
Metaheuristics
Deterministic and stochastic districting problem
Scheduling problem
Emergency infrastructure inspections

ABSTRACT

Search and rescue operations following natural disasters have become the cornerstone for minimizing the adverse social effects and the impact of these hazards. Despite their importance, the literature has not adequately dealt with post disaster operations – at least in part – because of the difficulty in rapidly solving the mathematically complex problems involved. In this paper we consider two important issues within the scope of post natural disaster actions; first, we develop deterministic and probabilistic districting and routing problems for scheduling infrastructure inspection crews following a natural disaster in urban areas; second, we assess and compare five metaheuristic optimization algorithms for solving these districting and routing problems. Results suggest that the five approaches examined offer applicable as well as fast solutions but with varying qualitative characteristics; selection of the preferred approach for practical applications will largely depend upon the network's characteristics.

1. Introduction

Natural hazards such as earthquakes, floods, tsunamis and hurricanes can cause invariably enormous damage to both social and infrastructure networks. Following such disasters, local communities and search and rescue crews are faced with rapidly degrading infrastructure networks that may result in much slower response times, delays in population evacuation, and significant complications in infrastructure repair. Because of the obvious importance of minimizing the adverse impacts from natural disasters, the literature has systematically dealt with what are considered the four main steps in disaster response [1]: First, mitigation; this includes assessing seismic hazards [2], probabilistic damage projection [3,4], and decision support systems for integrating the emergency process [5,6]. Second, preparedness, which has mainly focused on preparing infrastructure networks for dealing with potential disasters and for accommodating evacuation needs [7–12]. Third, response, which has evolved around two main research paths; i. planning the response–relief logistics operations [13–16], and, ii. assessing the performance of the infrastructure system following the natural hazard [17–20]. Fourth, recovery operations which have attracted limited attention despite their importance in practice; for example, work has concentrated on infrastructure element protection [21], general assessment of relief performance [22], and fund allocation for infrastructure repairs following disasters [23].

Recovery operations are very important in all natural disasters and particularly for the speedy revitalization of community activities. However, the mathematical complexity of organizing post disaster operations has hampered research efforts. Frequently, following a disaster civil infrastructure elements must be inspected, evaluated, and repairs prioritized. In order to efficiently deal with these problems requires the formulation and solution of complex districting and routing problems. For example, the affected area must be partitioned into districts of responsibility for repair crews and, then, inspection sequences – infrastructure elements to be inspected first, second, and so on – must be determined. In this paper we consider two important issues within the scope of post natural disaster actions; first, we develop deterministic and probabilistic districting and routing problems for scheduling infrastructure inspection crews following a natural disaster in urban areas. We note that the stochastic formulation of the districting problem examined, where the influence of the damages on the structural environment are randomly distributed, has not been previously addressed in the literature.

Second, we assess and compare five metaheuristic optimization algorithms for solving these districting and routing problems. We note that we recently offered a deterministic formulation for the districting problem solved via a particle swarm optimization (PSO) method [24]. Here we extend that work by critically assessing and comparing the efficiency and robustness of five metaheuristic

* Corresponding author.
E-mail addresses: nlagaros@central.ntua.gr (N.D. Lagaros), mgk@central.ntua.gr (M.G. Karlaftis).

2210-6502/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.swevo.2011.06.002

optimization algorithms: differential evolution (DE), harmony search (HS) and particle swarm optimization, covariance matrix adaptation (CMA) evolution strategy and elitist covariance matrix adaptation (ECMA), when dealing with large scale urban networks of infrastructure facilities problems.

The benefits of this work are twofold: first, by assessing the efficiency of different metaheuristic optimization approaches we can offer evidence regarding the implementation of these algorithms in practice; and, second, we can assist in improving infrastructure network restoration times and in minimizing the adverse impacts of natural hazards on civil infrastructure.

2. Background

Following a natural disaster, one of the issues of primary importance is to assess the condition of infrastructure elements so that their rehabilitation can be prioritized and the ability of relief crews to reach the areas hit by the hazard can be assessed. Infrastructure condition assessment is comprised of two very important steps; first, the affected area has to be subdivided into homogeneous subareas of responsibility for each of the available inspection crews; and, second, the optimal routes for each of the crews within the subareas has to be determined.

The first problem discussed is encountered in the literature as a *districting problem* whose aim is to partition a territory into smaller units called districts or zones, with an objective function being optimized and some constraints satisfied. The districting problem, because of its importance in practice (political districting, school districting, police districting, and so on), is one of the well researched problems; its formulation and solution, however, remain a significant challenge and researchers strive to find "better" as well as straightforward approaches to solving it ([25] offer an extensive review of the literature on districting problems). The second problem is known as the *traveling salesman problem* (TSP) whose goal is to find the fastest (or cheapest) way of visiting all the nodes (cities for example) of a network and returning to the starting point (Karlaftis et al. [26] discuss a large number of TSP formulations). To this end, many researchers have striven to obtain fast and "high quality" solutions to the TSP using many heuristic approaches including simulated annealing and genetic algorithms [26].

The complexity of the two problems examined, coupled with their significant computational requirements since we seek to solve the problems for an urban area, lead themselves to an investigation of the performance and solution quality of flexible computational approaches. In the last two decades, new families of computational approaches, denoted as soft computing (SC) methods, have been proposed. These methods are based on heuristic approaches rather than on rigorous mathematics; despite the initial skepticism, the methods have frequently turned out to be surprisingly powerful, while their use in various areas of engineering sciences is continuously growing. Neural networks, metaheuristics and fuzzy logic are the most popular methods. Many SC methods have been inspired by natural phenomena [27–36], frequently offering very good solutions to mathematically challenging problems, while they are adept to immediate and straightforward implementation in emergency response software.

As could be expected, many flexible computational approaches have been implemented in a number of problems in the area of transportation; Perrier et al. [37], Tan et al. [38] and Reche-Lopez et al. [39] dealt with various routing problems, Salazar-Aguilar et al. [40] with commercial districting implementing a bi-objective programming model where dispersion and balancing with respect to the number of customers were used as performance criteria, while Balaprakash et al. [41] with the TSP problem where a flexible solution methodology was proposed transforming the issue of solving a given vehicle routing problem with stochastic demands instance into an issue of solving a small set of capacitated vehicle routing problem instances. Furthermore, a local–global approach for the generalized traveling salesman problem was proposed, while based on this approach we describe a novel hybrid metaheuristic algorithm for solving the problem using genetic algorithms [42,43]. Despite the application of these approaches in problems such as districting and TSP, there has not been a thorough comparison of their efficiency and solution quality on the same problem, although a number of papers have compared their performance for a number of other instances (see, for example [44–50]). In [24] a deterministic formulation of the districting problem was solved with PSO while a TSP dealt with ACO was also presented. In this paper we assess the efficiency of different metaheuristic optimization approaches for the districting and TSP problems. In this manner, we can offer evidence regarding the implementation of these algorithms in practice and improving infrastructure network restoration times.

3. Problem formulation

The main objective of this work is to formulate the problem of inspecting the infrastructure systems of a large urban area as an optimization problem. As previously discussed, we achieve this in two steps: in the first step, the structural blocks to be inspected are optimally assigned into a number of inspection areas (districting problem), while in the second step the scheduling problem (inspection prioritization) is solved for each of the areas obtained from the first step. In formulating the optimization problems, the area examined is composed of N_{SB} structural blocks while N_{IG} is the number of crews available for inspecting the condition of the area's infrastructure system.

3.1. Step 1: Optimal districting problem

3.1.1. Deterministic formulation

The districting problem is defined as a nonlinear programming optimization problem as follows:

$$\begin{aligned}
&\min \sum_{i=1}^{N_{IG}} \sum_{k=1}^{n_{SB}^{(i)}} [d(SB_k, C_i) \cdot D(k)] \\
&x_{C_i} = \frac{1}{n_{SB}^{(i)}} \sum_{k=1}^{n_{SB}^{(i)}} x_k, \qquad y_{C_i} = \frac{1}{n_{SB}^{(i)}} \sum_{k=1}^{n_{SB}^{(i)}} y_k \qquad (1)\\
&D(k) = A(k) \cdot BP(k)
\end{aligned}$$

where $n_{SB}^{(i)}$ is the number of structural blocks assigned to the ith district (frequently referred to in the infrastructure literature as 'inspection group'), $d(SB_k, C_i)$ is the distance between the SB_k building block from the center of the ith group of structural blocks (with coordinates x_{Ci} and y_{Ci}), while $D(k)$ is the 'demand' for inspection for the kth building block defined as the product of the building block total area $A(k)$ times the built-up percentage $\mathrm{BP}(k)$ (i.e. percentage of the area with structures). This is a discrete optimization problem since the design variables $\boldsymbol{s}$ are integers and denote the inspection groups to which each built-up block has been assigned. Therefore the total number of the design variables is equal to the number of structural blocks and the range of the design variables is $[1, N_{IG}]$. To allow for differential damages to the city's infrastructure, the formulation of the optimal assignment problem given in Eq. (1) is modified as follows:

$$\min \sum_{i=1}^{N_{IG}} \sum_{k=1}^{n_{SB}^{(i)}} [d(SB_k, C_i) \cdot D(k) \cdot \mathrm{DF}(k)] \qquad (2)$$

where $\mathrm{DF}(k)$ is the damage factor corresponding to each damage level. This formulation allows for the effects of the natural

Table 1
Damage factor (DF) corresponding to each damage level.

Damage level	Damage factor (DF)
0	1.0
1	1.2
2	1.5
3	2.0

hazard to be different across the city (i.e. the damage level is inhomogeneous).

3.1.2. Probabilistic formulation

To account for the unknown – specific – influence of the hazard across the city, the formulation of the optimal assignment problem given in Eq. (2) is modified as follows:

$$\mathsf{F} = \min \sum_{i=1}^{N_{IG}} \sum_{k=1}^{n_{SB}^{(i)}} [d(SB_k, C_i) \cdot D(k) \cdot r_{\mathrm{DF}}(k)] \qquad (3)$$

$$r_{\mathrm{DF}}(k) \sim N(m_{\mathrm{DF}}(k), \sigma^2_{\mathrm{DF}}(k))$$

where $r_{\mathrm{DF}}(k)$ is a random variable describing the damage factor of the kth structural block, following the normal distribution with mean value equal to $m_{\mathrm{DF}}(k)$ and standard deviation $\sigma_{\mathrm{DF}}(k)$. The mean damage value is given in Table 1, while for calculating the standard deviation a coefficient of variation is set equal to 20%. In the probabilistic formulation, for each realization that corresponds to specific values of the random variables, a different value for the function F is obtained and the objective function to be minimized is the mean value of the function F.

3.2. Step 2: inspection prioritization problem

This is – as previously discussed – a typical TSP, also defined as an integer optimization problem. In TSP the task is to find a Hamiltonian tour of minimal length, i.e. to find a closed tour of minimal length that visits each node of a network, once. For an N cities TSP there are $(N - 1)!$ different tours; the TSP can be represented by a complete weighted graph $G = (N, A)$, with N the set of nodes and A the set of arcs (edges or connections) that fully connects the components of N. A cost function is assigned to every connection between two nodes i and j, represented by the distance between the two nodes $d_{i,j}$ $(i \neq j)$. A solution to the TSP is a permutation $\boldsymbol{p} = \{p(1), \ldots, p(N)\}$ of the node indices $\{1, \ldots, N\}$, as every node must appear only once in a solution. The optimal solution is the one that minimizes the total length $L(\boldsymbol{p})$ given by:

$$\mathsf{L}(p) = \sum_{i=1}^{N-1} \left(d_{p(i),p(i+1)}\right) + d_{p(N),p(1)}. \qquad (4)$$

Thus, the corresponding scheduling problem is defined as follows:

$$\min \left[\sum_{k=1}^{n_{SB}^{(i)}-1} d(SB_k, SB_{k+1}) + d(SB_{n_{SB}^{(i)}}, SB_1) \right], \quad i = 1, \ldots, N_{IG} \qquad (5)$$

where $d(SB_k, SB_{k+1})$ is the distance between the kth and $k+1$th building blocks. The main objective is to define the shortest possible route between the structural blocks that have been assigned to each inspection group in *Step* 1.

4. Monte Carlo simulation

The Monte Carlo simulation (MCS) method is particularly applicable for probabilistic analyses when analytical solutions are not attainable. This occurs in complex problems with a large number of uncertain variables, where all other stochastic analyses approaches are not applicable. Despite the simplicity in formulating the MCS, the method is capable of handling practically every possible case regardless of its complexity and the variation of the uncertain variables. Since MCS is based on the theory of large numbers (N_∞), an unbiased estimator of the probability of violation, the mean value and the variance is given by:

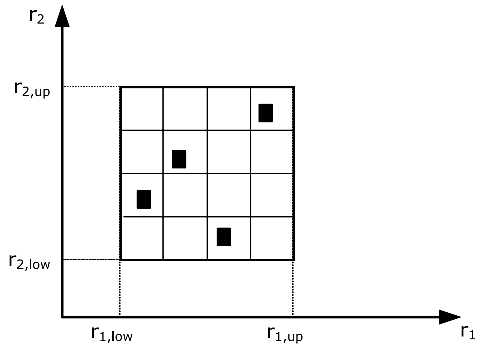


Fig. 1. Latin hypercube sampling in the two-dimensional space.

$$\bar{\mathsf{F}}_i = \frac{1}{N_\infty} \sum_{j=1}^{N_\infty} \mathsf{F}_i(\boldsymbol{r}_j) \qquad (6)$$

$$\sigma_{\mathsf{F}_i} = \frac{1}{N_\infty} \sum_{j=1}^{N_\infty} [\mathsf{F}_i(\boldsymbol{r}_j - \bar{\mathsf{F}}_i)]^2. \qquad (7)$$

In order to estimate $\bar{\mathsf{F}}_i$ and $\sigma^2_{\mathsf{F}_i}$ in Eqs. (6) and (7), an adequate number of N independent random samples is produced.

The Latin hypercube sampling (LHS) method was introduced to reduce the required computational cost of purely random sampling methodologies (A schematic representation of the LHS method is given in Fig. 1). Latin hypercube sampling is a strategy for generating random sample points ensuring that all portions of the random space in question are properly represented. LHS is generally recognized as one of the most efficient size reduction techniques. The basis of LHS is a full stratification of the sampled distribution with a random selection inside each stratum. In consequence, sample values are randomly shuffled among different variables. A Latin hypercube sample is constructed by dividing the range of each of the n_r uncertain variables into N non-overlapping segments of equal marginal probability. Thus, the entire parameter space consisting of N parameters is partitioned into N^{n_r} cells; a single value is selected randomly from each interval producing N sample values for each input variable. The values are randomly matched to create N sets from the N^{n_r} space with respect to the density of each interval for the N simulation runs. The advantage of the LHS approach is that the random samples are generated from all the ranges of possible values.

5. Metaheuristic algorithms

The six metaheuristic optimization algorithms tested in this paper appear to be very promising as they have been implemented in various challenging problems with success. We present here a short description of the six algorithms.

5.1. Particle swarm optimization

In particle swarm optimization, multiple candidate solutions coexist and collaborate simultaneously. Each solution is called “a particle” having a position and a velocity in the multidimensional design space while a population of particles is called a swarm. A particle “flies” in the problem search space looking for the optimal position. As “time” passes through its quest, a particle adjusts its

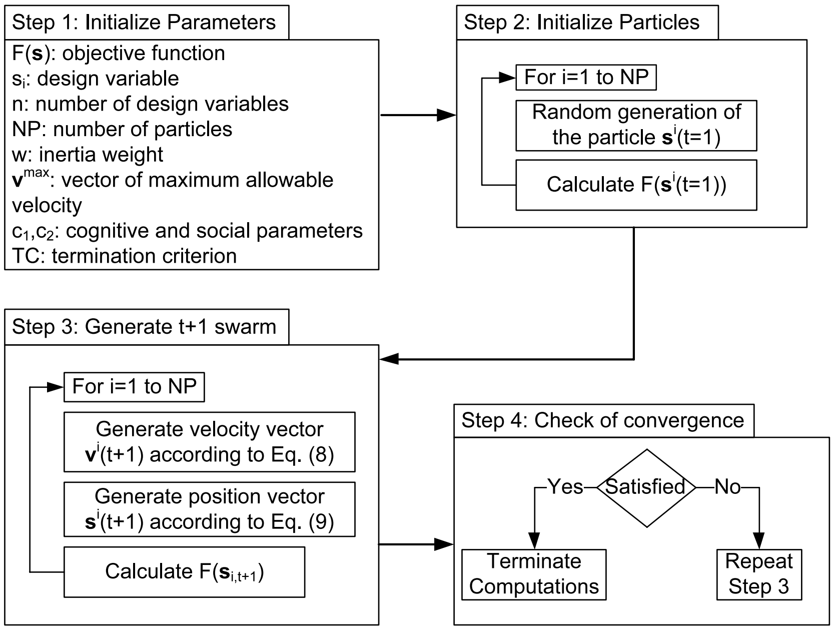

Step 1: Initialize Parameters
F(s): objective function
s_i: design variable
n: number of design variables
NP: number of particles
w: inertia weight
v^max: vector of maximum allowable velocity
c1,c2: cognitive and social parameters
TC: termination criterion
Step 2: Initialize Particles
For i=1 to NP
Random generation of the particle s^i(t=1)
Calculate F(s^i(t=1))
Step 3: Generate t+1 swarm
For i=1 to NP
Generate velocity vector v^i(t+1) according to Eq. (8)
Generate position vector s^i(t+1) according to Eq. (9)
Calculate F(s_i,t+1)
Step 4: Check of convergence
Satisfied
Yes
No
Terminate Computations
Repeat Step 3

Fig. 2. Flowchart of the particle swarm optimization algorithm.

velocity and position according to its own "experience" as well as the experience of other (neighboring) particles. A particle's experience is built by tracking and memorizing the best position encountered. A PSO system combines local search (through self experience) with global search (through neighboring experience), attempting to balance exploration and exploitation. Each particle maintains its two basic characteristics, velocity and position, in the multi-dimensional search space that are updated as follows:

$$\boldsymbol{v}^j(t+1) = w\boldsymbol{v}^j(t) + c_1\boldsymbol{r}_1 \circ \left(\boldsymbol{s}^{\mathrm{Pb},j-\boldsymbol{s}^j(t)}\right) + c_2\boldsymbol{r}_2 \circ \left(\boldsymbol{s}^{\mathrm{Gb}} - \boldsymbol{s}^j(t)\right) \quad (8)$$

$$\boldsymbol{s}^j(t+1) = \boldsymbol{s}^j(t) + \boldsymbol{v}^j(t+1) \quad (9)$$

where $\boldsymbol{v}^j(t)$ denotes the velocity vector of particle j at time t, $\boldsymbol{s}_j(t)$ represents the position vector of particle j at time t, vector $\boldsymbol{s}^{\mathrm{Pb},j}$ is the personal best ever position of the jth particle, and vector $\boldsymbol{s}^{\mathrm{Gb}}$ is the global best location found by the entire swarm. The acceleration coefficients c_1 and c_2 indicate the degree of confidence in the best solution found by the individual particle (c_1—cognitive parameter) and by the whole swarm (c_2—social parameter), respectively, while r_1 and r_2 are two random vectors uniformly distributed in the interval [0, 1]. The symbol "$\circ$" of Eq. (8) denotes the Hadamard product, i.e. the element-wise vector or matrix multiplication. Fig. 2 shows the flowchart of the PSO algorithm, while Fig. 3 depicts a particle's movement in a two-dimensional design space. The particle's current position $\boldsymbol{s}^j(t)$ at time t is represented by the dotted circle at the lower left quadrant of the drawing, while the new position $\boldsymbol{s}^j(t+1)$ at time $t+1$ is represented by the dotted bold circle at the upper right hand of the drawing. Fig. 3 depicts how the particle's movement is affected by: (i) it's velocity $\boldsymbol{v}^j(t)$; (ii) the personal best ever position of the particle, $\boldsymbol{s}^{\mathrm{Pb},j}$, and (iii) the global best location found by the entire swarm, $\boldsymbol{s}^{\mathrm{Gb}}$.

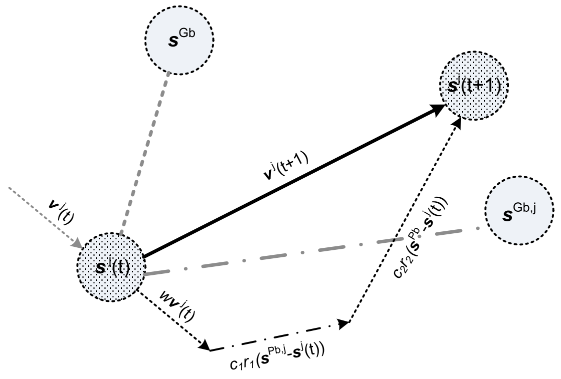


Fig. 3. Visualization of the particle's movement in a two-dimensional design space.

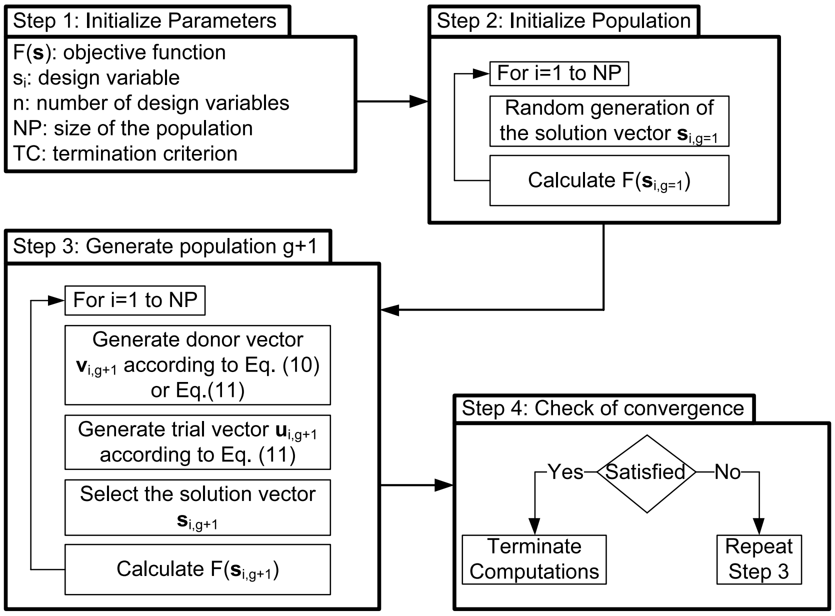
5.2. Differential evolution

In 1995, Storn and Price [33] proposed a new floating point evolutionary algorithm for global optimization and named it differential evolution, by implementing a special kind operator which sought to create new offspring from parent chromosomes. DE is a relatively novel parallel direct search method which utilizes a population of *NP* parameter vectors $\boldsymbol{s}_{i,g}(i = 1, \dots, \mathrm{NP})$ for each generation g, in a recent study by Das and Suganthan [51] a state of the art review on DE is presented. DE generates new vectors by adding the weighted difference vector between two population members to a third member. If the resulting vector corresponds to a better objective function value than a population member, the newly generated vector replaces this member. The comparison is performed between the newly generated vector and all the members of the population excluding the three ones used for its generation. Furthermore, the best parameter vector $\boldsymbol{s}_{\mathrm{best},g}$ is evaluated in every generation in order to keep track of the progress achieved during the optimization process (the DE algorithm is presented in Fig. 4). Several variants of DE have been proposed so far, but the two most widely used are the following.

Scheme DE1. In the first variant, a donor vector $\boldsymbol{v}_{i,g+1}$ is generated first according to:

$$\boldsymbol{v}_{i,g+1} = \boldsymbol{s}_{r_1,g} + F \cdot (\boldsymbol{s}_{r_2,g} - \boldsymbol{s}_{r_3,g}) \quad (10)$$

before the computation of the ith parameter vector $\boldsymbol{s}_{i,g+1}$. This step is equivalent to the mutation operator step of genetic algorithms or evolution strategies. The integers r_1, r_2 and r_3 are chosen randomly from the interval [1, NP] while $i \neq r_1$, r_2 and r_3. F is a real constant value, called mutation factor, which controls the amplification of



Step 1: Initialize Parameters
F(s): objective function
s_i: design variable
n: number of design variables
NP: size of the population
TC: termination criterion
Step 2: Initialize Population
For i=1 to NP
Random generation of the solution vector s_{i,g=1}
Calculate F(s_{i,g=1})
Step 3: Generate population g+1
For i=1 to NP
Generate donor vector v_{i,g+1} according to Eq. (10) or Eq.(11)
Generate trial vector u_{i,g+1} according to Eq. (11)
Select the solution vector s_{i,g+1}
Calculate F(s_{i,g+1})
Step 4: Check of convergence
Satisfied
Yes
No
Terminate Computations
Repeat Step 3

Fig. 4. Flowchart of the differential evolution algorithm.

the differential variation $(\boldsymbol{s}_{r_2,g} - \boldsymbol{s}_{r_3,g})$ and is defined in the range [0, 2]. In the next step the crossover operator is applied by generating the trial vector $\boldsymbol{u}_{i,g+1} = [u_{1,i,g+1}, u_{2,i,g+1}, \ldots, u_{D,i,g+1}]^T$ which is defined from the elements of the vector $\boldsymbol{s}_{i,g}$ and the elements of the donor vector $\boldsymbol{v}_{i,g+1}$ whose elements enter the trial vector with probability CR as follows:

$$u_{j,i,g+1} = \begin{cases} v_{j,i,g+1} & \text{if } \mathrm{rand}_{j,i} \le CR \text{ or } j = I_{\mathrm{rand}} \\ s_{j,i,g} & \text{if } \mathrm{rand}_{j,i} > CR \text{ or } j \ne I_{\mathrm{rand}} \end{cases}$$
$$i = 1, 2, \ldots, \mathrm{NP} \text{ and } j = 1, 2, \ldots, n \tag{11}$$

where $\mathrm{rand}_{j,i} \sim U[0, 1]$, I_{rand} is a random integer from [1, 2, ..., n] that ensures that $\boldsymbol{v}_{i,g+1} \ne \boldsymbol{s}_{i,g}$. The last step of the generation procedure is the implementation of the selection operator where the vector $\boldsymbol{s}_{i,g}$, is compared to the trial vector $\boldsymbol{u}_{i,g+1}$:

$$\boldsymbol{s}_{i,g+1} = \begin{cases} \boldsymbol{u}_{i,g+1} & \text{if } f(\boldsymbol{u}_{i,g+1}) \le f(s_{i,g}) \\ \boldsymbol{s}_{i,g} & \text{otherwise} \end{cases} \quad i = 1, 2, \ldots, \mathrm{NP}. \tag{12}$$

Scheme DE2. In the second variant the donor vector $\boldsymbol{v}_{i,g+1}$ is generated first according to:

$$\boldsymbol{v}_{i,g+1} = \boldsymbol{s}_{i,g} + \lambda \cdot (\boldsymbol{s}_{\mathrm{best},g} - \boldsymbol{s}_{i,g}) + F \cdot (\boldsymbol{s}_{r_2,g} - \boldsymbol{s}_{r_3,g}) \tag{13}$$

before the computation of the ith parameter vector $\boldsymbol{s}_{i,g+1}$, by introducing an additional control variable λ. The idea behind λ is to provide a means to enhance the greediness of the scheme by incorporating the current best vector $\boldsymbol{s}_{\mathrm{best},g}$. The generation of the trial vector $\boldsymbol{u}_{i,g+1}$ as well as the decision process are identical to those of DE1. In this study the second version has been implemented.

5.3. Harmony search

The harmony search algorithm was originally inspired by the improvisation process of Jazz musicians [34]. According to the analogy between improvisation and optimization, each musician (saxophonist, bassist, guitarist etc.) corresponds to each decision variable; each musical instrument's pitch range corresponds to a decision variable's value range. Musical harmony at certain times corresponds to a solution vector at certain iterations, and the audience's aesthetics corresponds to the objective function. According to the above algorithmic concept, the HS algorithm consists of the following five steps: parameter initialization; harmony memory initialization; new harmony improvisation; harmony memory update; and termination criterion check (Fig. 5).

Parameter initialization: In the first step, the optimization problem is specified where n is the number of decision variables (equivalent to the number of music instruments), while $s_i^L \le s_i \le s_i^U$, $i = 1, 2, \ldots, n$ determines the range of the ith decision variable's value. The HS algorithm parameters are also specified in this step: HMS is the harmony memory size that corresponds to the number of simultaneous solution vectors stored in harmony memory, HMCR defines the harmony memory considering rate, while PAR is the pitch adjusting rate.

Harmony memory initialization: In the second step, the harmony memory (HM) is initialized with HMS randomly generated solution vectors defining the musician's harmony memory matrix:

$$\mathrm{HM} = \begin{bmatrix} s_1^1 & s_2^1 & s_3^1 \cdots & s_n^1 \\ s_1^2 & s_2^2 & s_3^2 \cdots & s_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ s_1^{\mathrm{HMS}} & s_2^{\mathrm{HMS}} & s_3^{\mathrm{HMS}} \cdots & s_n^{\mathrm{HMS}} \end{bmatrix}. \tag{14}$$

New harmony improvisation: In the third step, a new harmony vector is improvised following three rules: random selection, memory consideration and pitch adjustment. According to the random selection, the value of the decision variable s_i is chosen randomly from the pitches stored in HM $= [s_i^1, s_i^2, \ldots, s_i^{\mathrm{HMS}}]$ with probability of HMCR ($0 \le \mathrm{HMCR} \le 1$) or according to the memory consideration it is randomly chosen with a probability of $(1 - \mathrm{HMCR})$ within its value range, as a musician plays any pitch within the instrument's pitch range:

$$s_i = \begin{cases} s_i \in [s_i^1, s_i^2, \ldots, s_i^{\mathrm{HMS}}] & \text{with probability HMCR} \\ s_i^L \le s_i \le s_i^U & \text{with probability } (1 - \mathrm{HMCR}). \end{cases} \tag{15}$$

After the value s_i is randomly picked according to the above memory consideration process, it can be further adjusted into neighboring values by adding certain amount to the value, with probability of HMCR × PAR ($0 \le \mathrm{PAR} \le 1$) while the original

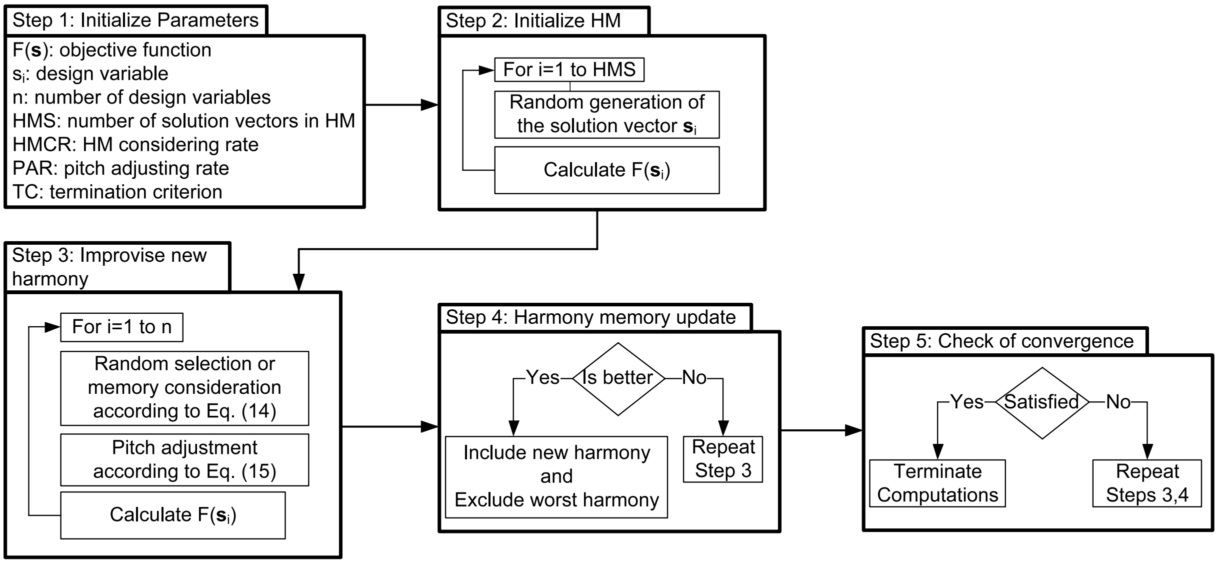

Step 1: Initialize Parameters
F(**s**): objective function
sᵢ: design variable
n: number of design variables
HMS: number of solution vectors in HM
HMCR: HM considering rate
PAR: pitch adjusting rate
TC: termination criterion
Step 2: Initialize HM
For i=1 to HMS
Random generation of the solution vector **s**ᵢ
Calculate F(**s**ᵢ)
Step 3: Improvise new harmony
For i=1 to n
Random selection or memory consideration according to Eq. (14)
Pitch adjustment according to Eq. (15)
Calculate F(**s**ᵢ)
Step 4: Harmony memory update
Is better
Yes
No
Include new harmony and Exclude worst harmony
Repeat Step 3
Step 5: Check of convergence
Satisfied
Yes
No
Terminate Computations
Repeat Steps 3,4

Fig. 5. Flowchart of the harmony search algorithm.

pitch obtained in HM consideration is just kept with a probability of HMCR × (1 − PAR):

$$s_i = \begin{cases} s_i(k+m) \quad \text{with probability HMCR} \times \text{PAR} \\ s_i \text{ with probability HMCR} \times (1 - \text{PAR}). \end{cases} \tag{16}$$

Harmony memory update: If the new generated harmony vector, is better than the worst harmony vector of the HM, with reference to the objective function value, the worst harmony is replaced by the new harmony vector.

5.4. Covariance matrix adaptation

The covariance matrix adaptation, proposed by Hansen and Ostermeier [31] is a completely derandomized self-adaptation scheme. First, the covariance matrix of the mutation distribution is changed in order to increase the probability of producing the selected mutation step again. Second, the rate of change is adjusted according to the number of strategy parameters to be adapted. Third, under random selection the expectation of the covariance matrix is stationary. Further, the adaptation mechanism is inherently independent of the given coordinate system. The transition from generation g to $g+1$, given in the following steps, completely defines the algorithm (Fig. 6).

Generation of offspring: Creation of λ new offspring as follows:

$$\mathbf{s}_k^{(g+1)} \sim N\left(\mathbf{m}^{(g)}, \sigma^{(g)^2}\mathbf{C}^{(g)}\right) \sim \mathbf{m}^{(g)} + \sigma^{(g)} N\left(\mathbf{0}, \mathbf{C}^{(g)}\right) \tag{17}$$

where $\mathbf{s}_k^{(g+1)} \in \Re^n$ is the design vector of the kth offspring in generation $g+1 (k = 1, \ldots, \lambda)$, $N\left(\mathbf{m}^{(g)}, \mathbf{C}^{(g)}\right)$ are normally distributed random numbers where $\mathbf{m}^{(g)} \in \Re^n$ is the mean value vector and $\mathbf{C}^{(g)}$ is the covariance matrix while $\sigma^{(g)} \in \Re_+$ is the global step size. To define a generation step, the new mean value vector $\mathbf{m}^{(g+1)}$, global step size $\sigma^{(g+1)}$, and covariance matrix $\mathbf{C}^{(g+1)}$ have to be defined.

New mean value vector: After selection scheme (μ, λ) operates over the λ offspring, the new mean value vector $\mathbf{m}^{(g+1)}$ is calculated according to the following expression:

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{s}_{i:\lambda}^{(g+1)} \tag{18}$$

where $\mathbf{s}_{i:\lambda}^{(g+1)}$ is the ith best offspring and w_i are weight coefficients defined as follows:

$$w_i = \frac{\ln(\mu+1) - \ln i}{\sum_{j=1}^{\mu} (\ln(\mu+1) - \ln j)}, \qquad \sum_{i=1}^{\mu} w_i = 1, \quad w_i > 0,\ i = 1, \ldots, \mu. \tag{19}$$

Global step size: The new global step size is calculated according to the following expression:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|\mathbf{p}_\sigma^{(g+1)}\right\|}{E\left\|N\left(\mathbf{0}, \mathbf{I}\right)\right\|} - 1\right)\right) \tag{20}$$

where d_σ and c_σ constants are defined as follows:

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 3}$$
$$d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}-1}}{n+1}} - 1\right) + c_\sigma, \tag{21}$$
$$\mu_{\text{eff}} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1}.$$

$E\left\|N\left(\mathbf{0}, \mathbf{I}\right)\right\|$ is the expected value of the Euclidean norm of a normally distributed vector, and $\mathbf{p}_\sigma^{(g+1)}$ is the conjugate evolution path ($\mathbf{p}_\sigma^{(0)} = 0$), that is given by:

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\,\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma\left(2 - c_\sigma\right)\mu_{\text{eff}}}\,\mathbf{C}^{(g)^{-\frac{1}{2}}} \times \frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{22}$$

where the matrix $\mathbf{C}^{(g)^{-\frac{1}{2}}}$ is given by:

$$\mathbf{C}^{(g)^{-\frac{1}{2}}} = \mathbf{B}^{(g)}\mathbf{D}^{(g)^{-1}}\mathbf{B}^{(g)^T} \tag{23}$$

where the columns of $\mathbf{B}^{(g)}$ are an orthogonal basis of the eigenvectors of $\mathbf{C}^{(g)}$ and the diagonal elements of $\mathbf{D}^{(g)}$ are the square roots of the corresponding positive eigenvalues.

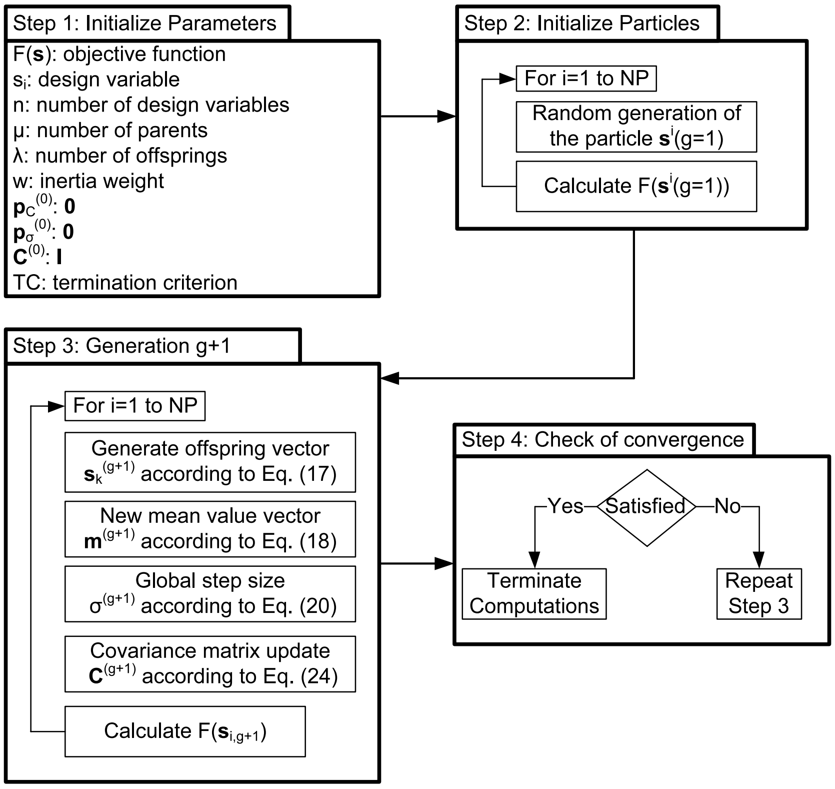

Step 1: Initialize Parameters
F(s): objective function
s_i: design variable
n: number of design variables
μ: number of parents
λ: number of offsprings
w: inertia weight
p_C^(0): 0
p_σ^(0): 0
C^(0): I
TC: termination criterion
Step 2: Initialize Particles
For i=1 to NP
Random generation of the particle s^i(g=1)
Calculate F(s^i(g=1))
Step 3: Generation g+1
For i=1 to NP
Generate offspring vector s_k^(g+1) according to Eq. (17)
New mean value vector m^(g+1) according to Eq. (18)
Global step size σ^(g+1) according to Eq. (20)
Covariance matrix update C^(g+1) according to Eq. (24)
Calculate F(s_i,g+1)
Step 4: Check of convergence
Satisfied
Yes
No
Terminate Computations
Repeat Step 3

Fig. 6. Flowchart of the covariance matrix adaptation algorithm.

Covariance matrix update: The new covariance matrix $\mathbf{C}^{(g+1)}$ is calculated from the following equation:

$$\mathbf{C}^{(g+1)} = (1-c_{\mathrm{cov}})\,\mathbf{C}^{(g)} + \frac{c_{\mathrm{cov}}}{\mu_{\mathrm{cov}}}\boldsymbol{p}_c^{(g+1)}\boldsymbol{p}_c^{(g+1)^T} + c_{\mathrm{cov}}\left(1-\frac{1}{\mu_{\mathrm{cov}}}\right)\sum_{i=1}^{\mu} w_i \mathrm{OP}\left(\frac{s_{i:\lambda}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(g)}}\right). \tag{24}$$

OP denotes the outer product of a vector with itself and $\boldsymbol{p}_c^{(g)} \in \Re^n$ is the evolution path ($\boldsymbol{p}_c^{(0)} = 0$) which is given by equation:

$$\boldsymbol{p}_c^{(g+1)} = (1-c_c)\,\boldsymbol{p}_c^{(g)} + H_\sigma^{(g+1)}\sqrt{c_c\,(2-c_c)\,\mu_{\mathrm{eff}}} \times \frac{\mathbf{m}^{(g+1)}-\mathbf{m}^{(g)}}{\sigma^{(g)}} \tag{25}$$

where:

$$c_c = \frac{4}{4+n}, \qquad \mu_{\mathrm{cov}} = \mu_{\mathrm{eff}}$$
$$c_{\mathrm{cov}} = \frac{1}{\mu_{\mathrm{cov}}}\frac{2}{\left(n+\sqrt{2}\right)^2} + \left(1-\frac{1}{\mu_{\mathrm{cov}}}\right)\min\left(1, \frac{2\mu_{\mathrm{eff}}-1}{(n+2)^2+\mu_{\mathrm{eff}}}\right). \tag{26}$$

5.5. Elitist covariance matrix adaptation

The elitist CMA evolution strategy is a combination of the well known $(1+\lambda)$-selection scheme of evolution strategies [29], with covariance matrix adaptation [32]. The original update rule for the covariance matrix can be reasonably applied in the $(1+\lambda)$-selection. The cumulative step size adaptation (path length control) of the $(\mu/\mu,\lambda)$-CMA is replaced by a success rule based step size control. Every individual a of the ECMA algorithm is comprised of five components:

$$a = \{\boldsymbol{s}, \bar{p}_{\mathrm{succ}}, \sigma, \boldsymbol{p}_c, \mathbf{C}\} \tag{27}$$

where $\boldsymbol{s}$ is the design vector, $\bar{p}_{\mathrm{succ}}$ is a parameter that states the success rate during the evolution process, σ is the step size, $\boldsymbol{p}_c$ is the evolution path, $\mathbf{C}$ is the covariance matrix of the mutation strengths. Contrary to the CMA algorithm, each individual has its own step size σ, evolution path $\boldsymbol{p}_c$ and covariance matrix $\mathbf{C}$. A pseudocode of the ECMA algorithm is shown in Fig. 7(a). In line #1 a new parent $a_{\mathrm{parent}}^{(g)}$ is generated. In lines #4–6, λ new offspring are generated from the parent vector $a_{\mathrm{parent}}^{(g)}$. The new offspring are sampled according to Eq. (17), with variable $\mathbf{m}^{(g)}$ being replaced by the design vector $s_{\mathrm{parent}}^{(g)}$ of the parent individual. After the λ new offspring are sampled, the parent's step size is updated by means of UpdateStepSize subroutine (see Fig. 7(b)). The arguments of the subroutine are the parent $a_{\mathrm{parent}}^{(g)}$ and the success rate $\lambda_{\mathrm{succ}}^{(g+1)}/\lambda$, where $\lambda_{\mathrm{succ}}^{(g+1)}$ is the number of offspring having better fitness function than the parent. The step size update is based upon the 1/5 success rule. When the ratio $\lambda_{\mathrm{succ}}^{(g+1)}/\lambda$ is larger than 1/5, the step size increases, and when it is smaller the step size decreases. If the best offspring has a better fitness value than the parent, it becomes the parent of the next generation (see lines #8–9). If the inequality of line #8 is satisfied, then the covariance matrix of the new parent is updated by means of UpdateCovariance subroutine (see Fig. 7(c)). The arguments of the subroutine are the current parent and the step change:

$$\frac{\boldsymbol{s}_{\mathrm{parent}}^{(g+1)} - \boldsymbol{s}_{\mathrm{parent}}^{(g)}}{\sigma_{\mathrm{parent}}^{(g)}}. \tag{28}$$

The update of the evolution path and the covariance matrix depends on the success rate:

$$\bar{p}_{\mathrm{succ}} = \frac{\lambda_{\mathrm{succ}}}{\lambda}. \tag{29}$$

a

```
Algorithm: $(1+\lambda)$-ECMA

1  g = 0, initialize $a_{parent}^{(g)}$
2  repeat
3      $a_{parent}^{(g+1)} \leftarrow a_{parent}^{(g)}$
4      for $k = 1,\ldots,\lambda$ do
5          $s_k^{(g+1)} \sim N\left(s_{parent}^{(g)}, \sigma^{(g)^2}\mathbf{C}^{(g)}\right)$
6      end do
7      UpdateStepSize$\left(a_{parent}^{(g+1)}, \frac{\lambda_{succ}^{(g+1)}}{\lambda}\right)$
8      if $f\left(s_{1:\lambda}^{(g+1)}\right) < f\left(s_{parent}^{(g)}\right)$ then
9          $\mathbf{x}_{parent}^{(g+1)} \leftarrow \mathbf{x}_{1:\lambda}^{(g+1)}$
10         UpdateCovariance$\left(a_{parent}^{(g+1)}, \frac{s_{parent}^{(g+1)} - s_{parent}^{(g)}}{\sigma_{parent}^{(g)}}\right)$
11     end if
12 until stopping criterion is met
```

b

```
Procedure: UpdateStepSize$\left(a = \left\{s, \bar{p}_{succ}, \sigma, \mathbf{p}_c, \mathbf{C}\right\}, p_{succ}\right)$

1  $\bar{p}_{succ} \leftarrow (1 - c_p)\,\bar{p}_{succ} + c_p p_{succ}$
2  $\sigma \leftarrow \sigma \exp\left(\frac{1}{d}\left(\bar{p}_{succ} - \frac{p_{succ}^{\text{target}}}{1 - p_{succ}^{\text{target}}}(1 - \bar{p}_{succ})\right)\right)$
```

c

```
Procedure: UpdateCovariance$\left(a = \left\{s, \bar{p}_{succ}, \sigma, \mathbf{p}_c, \mathbf{C}\right\}, s_{step} \in \mathbb{R}^n\right)$

1  if $\bar{p}_{succ} < p_{thresh}$ then
2      $\mathbf{p}_c \leftarrow (1 - c_c)\,\mathbf{p}_c + \sqrt{c_c(2 - c_c)}\,\mathbf{x}_{step}$
3      $\mathbf{C} \leftarrow (1 - c_{\text{cov}})\,\mathbf{C} + c_{\text{cov}}\mathbf{p}_c\mathbf{p}_c^T$
4  else
5      $\mathbf{p}_c \leftarrow (1 - c_c)\,\mathbf{p}_c$
6      $\mathbf{C} \leftarrow (1 - c_{\text{cov}})\,\mathbf{C} + c_{\text{cov}}\left(\mathbf{p}_c\mathbf{p}_c^T + c_c(2 - c_c)\,\mathbf{C}\right)$
7  end if
```

Fig. 7. Elitist covariance matrix adaptation: (a) pseudocode, (b) update step size subroutine and (c) update covariance subroutine.

If the success rate is below a given threshold value p_{thresh} then the step size is taken into account and the evolution path and the covariance matrix is updated (see lines #2–3 of Fig. 7(c)). If the success rate is above the given threshold p_{thresh} the step change is not taken into account and the update of the evolution path and the covariance matrix happens (see lines #5–6). The pre-calculated values of the other strategic parameters are given by the following equations:

$$d = 1 + \frac{n}{2\lambda} \tag{30}$$

$$p_{\text{succ}}^{\text{target}} = \frac{1}{5 + \frac{\sqrt{\lambda}}{2}} \tag{31}$$

$$c_p = \frac{p_{\text{succ}}^{\text{target}}\lambda}{2 + p_{\text{succ}}^{\text{target}}\lambda} \tag{32}$$

where d is a damping parameter in subroutine UpdateStepSize, n is the number of design variables, c_p is a normalization constant of the $\bar{p}_{\text{succ}}$parameter in subroutine UpdateStepSize, $p_{\text{succ}}^{\text{target}}$ is a constant that depends on the number of offspring λ.

5.6. Ant colony optimization

The ant colony optimization (ACO) algorithm [52,53] is a population-based probabilistic optimization method, inspired by the behavior of real ants in nature, implemented mainly for finding optimal paths through graphs. In ACO, a set of software agents called artificial ants search for good solutions to the optimization problem of finding the best path on a weighted graph. The ants incrementally build solutions by moving on the graph. Consider a population of m ants where at each iteration every ant defines a "route" by visiting every node sequentially. Initially, ants are set on randomly chosen nodes. At each construction step during an iteration, ant k applies a random action choice rule, called random proportional rule, to decide which node to visit next. While defining the route, an ant k currently at node i, maintains a memory M^k which contains the nodes already visited, in the order they were visited. This memory is used in order to define the feasible neighborhood N_i^k that is the set of nodes that have not yet been visited by ant k. In particular, the probability with which ant k, currently at node i, chooses to go to node j is:

$$p_{i,j}^k = \frac{(\tau_{i,j})^{\alpha} \cdot (\eta_{i,j})^{\beta}}{\sum\limits_{\ell \in \mathsf{N}_i^k}\left((\tau_{i,\ell})^{\alpha} \cdot (\eta_{i,\ell})^{\beta}\right)}, \quad \text{if } j \in \mathsf{N}_i^k \tag{33}$$

where $\tau_{i,j}$ is the amount of pheromone on connection between i and j nodes, α is a parameter to control the influence of $\tau_{i,j}$, β is a parameter to control the influence of $\eta_{i,j}$ and $\eta_{i,j}$ is a heuristic information that is available a priori, denoting the desirability of connection i, j, given by:

$$\eta_{i,j} = \frac{1}{d_{i,j}}. \tag{34}$$

According to Eq. (34), the heuristic desirability of going from node i to node j is inversely proportional to the distance between cities i and j. By definition, the probability of choosing a city outside N_i^k is zero. By this probabilistic rule, the probability of choosing a particular connection i, j increases with the value of the associated pheromone trail $\tau_{i,j}$ and of the heuristic information value $\eta_{i,j}$. The selection of parameters α and β is very important. After all ants have defined their routes, the amount of pheromone for each connection between i and j nodes, is updated for the next iteration

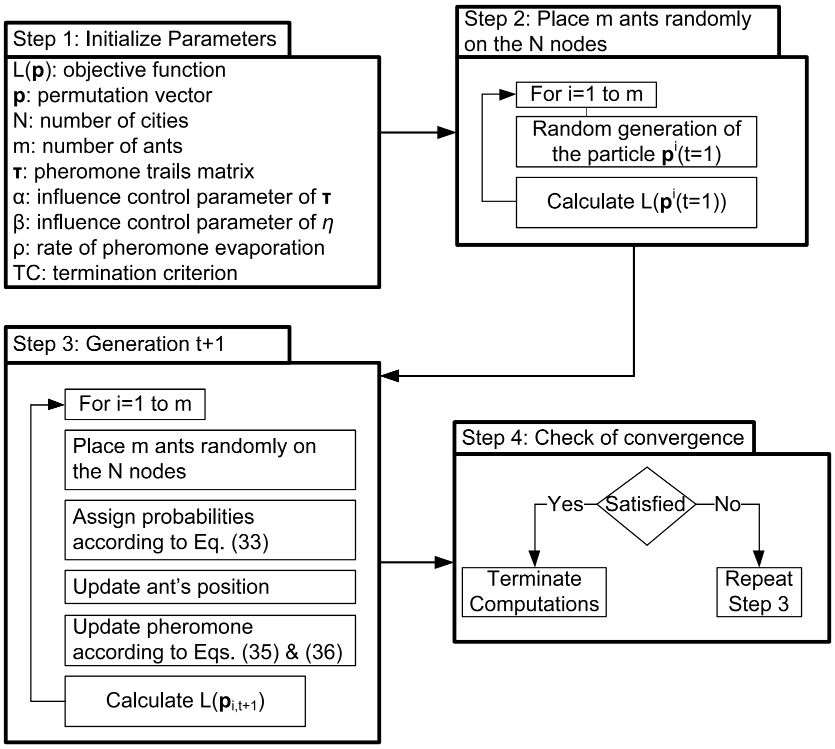

Step 1: Initialize Parameters
L(p): objective function
p: permutation vector
N: number of cities
m: number of ants
τ: pheromone trails matrix
α: influence control parameter of τ
β: influence control parameter of η
ρ: rate of pheromone evaporation
TC: termination criterion
Step 2: Place m ants randomly on the N nodes
For i=1 to m
Random generation of the particle p^i(t=1)
Calculate L(p^i(t=1))
Step 3: Generation t+1
For i=1 to m
Place m ants randomly on the N nodes
Assign probabilities according to Eq. (33)
Update ant's position
Update pheromone according to Eqs. (35) & (36)
Calculate L(p_{i,t+1})
Step 4: Check of convergence
Satisfied
Yes
No
Terminate Computations
Repeat Step 3

Fig. 8. Flowchart of the ant colony optimization algorithm.

$t+1$ as follows:

$$\tau_{i,j}(t+1) = (1-\rho)\cdot\tau_{i,j}(t) + \sum_{k=1}^{m}\Delta\tau_{i,j}^{k}(t), \quad \forall(i,j)\in \mathsf{A} \tag{35}$$

where ρ is the rate of pheromone evaporation, a constant parameter of the method, A is the set of arcs (edges or connections) that fully connects the set of nodes and $\Delta\tau_{i,j}^{k}(t)$ is the amount of pheromone ant k deposits on the connections it has visited through its tour T^k, typically given by:

$$\Delta\tau_{i,j}^{k} = \begin{cases} \dfrac{\mathsf{Q}}{\mathsf{L}(\mathsf{T}^k)} & \text{if connection } (i,j) \text{ belongs to } \mathsf{T}^k \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

The coefficient ρ must be set to a value <1 to avoid unlimited accumulation of trail [54], while Q is a constant. In general, connections that are used by many ants and which are parts of short tours, receive more pheromone and are therefore more likely to be chosen by ants in future iterations of the algorithm. A pseudo code of the ACO procedure is given in Fig. 8.

6. Case study

In order to assess the performance of the metaheuristic formulations discussed in the districting and TSP framework, we consider the city of Patras, in Greece. The city of Patras is composed of N_{SB} = 112 structural blocks with different areas and built-up percentages, while three different sets of inspection groups (crews of inspectors) are considered (2, 4 and 6 inspection groups). The subdivision of the city of Patras into 112 structural blocks can be seen in Fig. 9(a). These statistics have been obtained by official census bureau-statistics of Greece [55]. A scenario was considered with respect to the damage level encountered on the structures due to a strong earthquake, where four areas with differential structural damage levels are considered: (i) Level 0—no damage, (ii) Level 1—slight damage, (iii) Level 2—moderate damage and (iv) Level 3—extensive damage (Fig. 9(b)). This study is performed in two steps. In the first one a parametric study is performed with reference to the combination of the parameters of the metaheuristics, while in the second one the best combinations are used for solving the deterministic and stochastic problem formulations defined in Eqs. (2) and (3).

6.1. Definition of the parameters

The performance of the metaheuristics is influenced by the selection of their parameters, in two recent studies parameter tuning of evolutionary algorithms and nonparametric statistical tests as a methodology for the comparing metaheuristics are presented [56,57]. In the first part of this investigation, in order to identify the best combination of the parameters for each metaheuristic algorithm, 30 combinations of the parameters are generated by means of LHS, while for each combination 100 optimization runs are performed in order to calculate the mean and the coefficient of variation with reference to the objective function value. The resulting optimization runs for dealing with the districting problem considered for defining the parameters of the five metaheuristics are equal to 5 metaheuristics × 30 combinations × 100 runs = 15,000 optimization runs. All runs were performed for the case of the deterministic formulation as defined in Eq. (1) while 6 inspection groups are considered. The parameters that are identified for each algorithm are: (i) For PSO, the number of particles NP is defined in the range of [50, 200], the inertia weight w is defined in the range of [−1, 1], while the cognitive parameter c_1 and social parameter c_2 are defined in the range of [−5, 5]. (ii) For DE, the population size NP is defined in the range of [50, 200], the probability CR, the constant F and the control variable λ are defined in the range of [0, 1]. (iii) For HS, the harmony memory size HMS is defined in the range of [50, 200], while the harmony memory consideration rate $HMCR$ and the pitch adjusting rate PAR are defined in the range of [0, 1]. (iv) For CMA, the number of parents μ and the number of offspring λ are defined

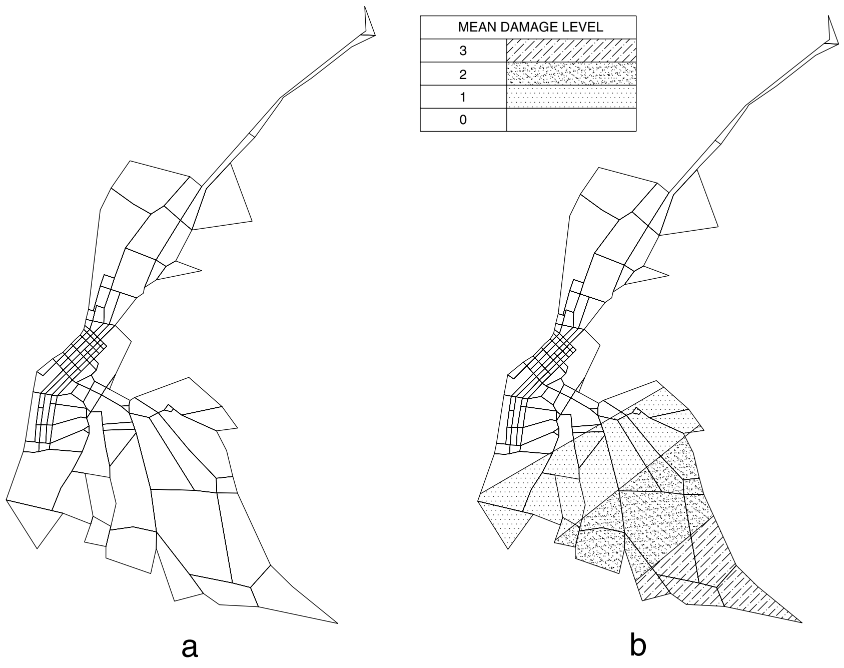


Fig. 9. City of Patras—(a) subdivision into structural blocks and (b) mean damage level distributed over the structural blocks.

in the range of [50, 200]. (v) For ECMA, the number of offspring λ is defined in the range of [50, 200].

The results of this parametric study are presented in Tables 2–6 corresponding to PSO, DE, HS, CMA and ECMA respectively. In these five tables the mean, the coefficient of variation, the best and the worst normalized objective function values are depicted for each combination of the algorithms' parameters. The normalized values are defined by dividing the objective function values with the best one achieved by each optimization algorithm; therefore normalized values equal to unity are encountered for each algorithm. In these five Tables the lower the normalized mean value the better is the coefficient of variation (COV). This is because the normalized mean gives the percentage of increase of the mean optimum value obtained in a pool of 100 independent optimization runs compared to the best optimum value obtained implementing all the combinations of the algorithms; while low COV means that the algorithm is not influenced by the independent runs. As it can be seen in Table 2 corresponding to PSO, for a certain combination of the parameters the mean value is increased up to 130.0% while the COV reaches 12.0%. In the case of DE, as shown in Table 3, for a certain combination of the parameters the mean value is increased up to 45.0% while COV reaches 15%, worth noticing also that every combination reaches the best objective function value in a pool of 100 independent optimization runs. In the case of HS, as shown in Table 4, for a certain combination of the parameters the mean value is increased up to 30% while COV reaches 4.0%, worth noticing also that every combination reaches the best objective function value in a pool of 100 independent optimization runs. In the case of CMA, as shown in Table 5, for a certain combination of the parameters the mean value is increased up to 30% while COV reaches 10.0%, worth noticing also that every combination reaches the best objective function value in a pool of 100 independent optimization runs. Finally, in the case of ECMA, as shown in Table 6, for a certain combination of the parameters the mean value is increased up to 70% while COV reaches 13.0%, worth noticing also that every combination reaches the best objective function value in a pool of 100 independent optimization runs.

The parameters used for implementation of the five meta-heuristic algorithms for solving the districting problem are as follows:

Table 2
Sensitivity analysis of PSO algorithm with reference to the objective function value.

Parameters combination	Mean	Coefficient of variation (%)	Best	Worst
1	1.61	6.14	1.51	1.81
2	1.33	9.27	1.17	1.81
3	2.12	4.09	1.17	2.24
4	2.30	2.15	1.17	2.38
5	1.91	4.13	1.17	2.38
6	1.71	7.70	1.17	2.38
7	1.84	5.44	1.17	2.38
8	1.98	2.98	1.17	2.38
9	1.69	7.00	1.17	2.38
10	2.03	2.64	1.17	2.38
11	1.60	4.53	1.17	2.38
12	1.90	6.67	1.17	2.38
13	1.68	11.55	1.17	2.38
14	2.14	2.14	1.17	2.38
15	2.18	2.31	1.17	2.38
16	1.30	9.15	1.17	2.38
17	2.00	3.79	1.17	2.38
18	1.63	7.66	1.17	2.38
19	1.85	4.20	1.17	2.38
20	2.21	1.72	1.17	2.38
21	2.18	3.77	1.17	2.38
22	1.80	11.84	1.17	2.38
23	1.61	4.72	1.17	2.38
24	1.90	5.10	1.17	2.38
25	1.60	8.96	1.17	2.38
26	1.66	7.43	1.17	2.38
27	**1.09**	**5.75**	**1.00**	**2.38**
28	1.47	7.99	1.00	2.38
29	1.52	6.30	1.00	2.38
30	1.17	8.01	1.00	2.38

(i) PSO method: the number of particles NP = 85, the inertia weight $w = 0.22$, while the cognitive parameter $c_1 = 0.58$ and social parameter $c_2 = 0.99$ based on the parameter study presented in Table 2.

(ii) DE method: the population size NP = 160, the probability $CR = 0.71$, the constant $F = 0.93$, while the control variable $\lambda = 0.2$ based on the parameter study presented in Table 3.

Table 3
Sensitivity analysis of DE algorithm with reference to the objective function value.

Parameter combination	Mean	Coefficient of variation (%)	Best	Worst
1	1.02	1.88	1.00	1.06
2	1.08	6.80	1.00	1.21
3	1.44	14.42	1.00	1.75
4	1.10	8.50	1.00	1.75
5	1.07	5.27	1.00	1.75
6	1.01	0.87	1.00	1.75
7	1.07	3.29	1.00	1.75
8	1.01	0.83	1.00	1.75
9	1.00	0.27	1.00	1.75
10	1.02	3.34	1.00	1.75
11	1.00	0.70	1.00	1.75
12	**1.00**	**0.17**	**1.00**	**1.75**
13	1.08	3.93	1.00	1.75
14	1.03	2.46	1.00	1.75
15	1.19	10.10	1.00	1.75
16	1.00	1.44	1.00	1.75
17	1.00	0.18	1.00	1.75
18	1.13	7.09	1.00	1.75
19	1.12	12.65	1.00	1.75
20	1.01	2.96	1.00	1.75
21	1.00	0.34	1.00	1.75
22	1.00	0.45	1.00	1.75
23	1.37	7.96	1.00	1.75
24	1.46	13.42	1.00	1.75
25	1.34	8.29	1.00	1.75
26	1.15	14.04	1.00	1.75
27	1.24	10.69	1.00	1.75
28	1.24	8.36	1.00	1.75
29	1.46	6.28	1.00	1.75
30	1.44	7.45	1.00	1.75

Table 4
Sensitivity analysis of HS algorithm with reference to the objective function value.

Parameter combination	Mean	Coefficient of variation (%)	Best	Worst
1	1.22	3.16	1.15	1.26
2	1.25	1.93	1.15	1.28
3	1.20	2.11	1.15	1.28
4	1.25	3.34	1.15	1.29
5	1.24	1.41	1.15	1.29
6	1.25	1.22	1.15	1.29
7	1.23	2.88	1.15	1.29
8	1.26	2.11	1.15	1.30
9	1.21	2.73	1.15	1.30
10	1.27	0.94	1.15	1.30
11	1.24	3.24	1.14	1.30
12	**1.08**	**2.78**	**1.00**	**1.30**
13	1.20	1.89	1.00	1.30
14	1.22	2.63	1.00	1.30
15	1.30	3.52	1.00	1.36
16	1.20	2.18	1.00	1.36
17	1.23	4.10	1.00	1.36
18	1.24	3.95	1.00	1.36
19	1.27	2.33	1.00	1.36
20	1.17	1.49	1.00	1.36
21	1.21	1.83	1.00	1.36
22	1.20	2.33	1.00	1.36
23	1.25	2.82	1.00	1.36
24	1.21	2.30	1.00	1.36
25	1.20	2.48	1.00	1.36
26	1.21	3.07	1.00	1.36
27	1.24	3.01	1.00	1.36
28	1.23	2.35	1.00	1.36
29	1.22	3.76	1.00	1.36
30	1.23	2.41	1.00	1.36

Table 5
Sensitivity analysis of CMA algorithm with reference to the objective function value.

Parameter combination	Mean	Coefficient of variation (%)	Best	Worst
1	1.25	5.71	1.08	1.36
2	1.26	7.31	1.03	1.37
3	1.28	2.09	1.03	1.37
4	1.29	1.42	1.03	1.37
5	1.26	2.04	1.03	1.37
6	1.28	2.77	1.03	1.37
7	1.27	1.83	1.03	1.37
8	1.28	1.42	1.03	1.37
9	1.26	4.87	1.03	1.37
10	1.27	2.15	1.03	1.37
11	1.27	1.32	1.03	1.37
12	1.26	3.78	1.03	1.37
13	1.28	2.92	1.03	1.37
14	1.27	2.30	1.03	1.37
15	1.26	2.18	1.03	1.37
16	1.26	1.27	1.03	1.37
17	1.25	4.39	1.03	1.37
18	1.26	2.04	1.03	1.37
19	1.24	5.70	1.03	1.37
20	1.23	3.18	1.03	1.37
21	1.24	4.96	1.03	1.37
22	1.24	4.01	1.03	1.37
23	1.20	6.26	1.03	1.37
24	1.27	1.35	1.03	1.37
25	1.19	8.75	1.03	1.37
26	1.14	9.79	1.00	1.37
27	1.19	7.54	1.00	1.37
28	**1.14**	**8.78**	**1.00**	**1.37**
29	1.16	8.70	1.00	1.37
30	1.19	7.46	1.00	1.37

Table 6
Sensitivity analysis of ECMA algorithm with reference to the objective function value.

Parameter combination	Mean	Coefficient of variation (%)	Best	Worst
1	1.35	13.74	1.01	1.69
2	**1.24**	**11.30**	**1.01**	**1.69**
3	1.37	9.49	1.01	1.69
4	1.36	9.90	1.01	1.69
5	1.38	7.86	1.01	1.69
6	1.24	12.59	1.00	1.69
7	1.35	9.40	1.00	1.69
8	1.36	8.41	1.00	1.69
9	1.33	12.97	1.00	1.69
10	1.53	11.27	1.00	1.78
11	1.32	10.48	1.00	1.78
12	1.54	6.78	1.00	1.78
13	1.29	11.64	1.00	1.78
14	1.34	8.89	1.00	1.78
15	1.32	13.00	1.00	1.78
16	1.60	8.56	1.00	1.78
17	1.57	13.17	1.00	1.78
18	1.35	11.89	1.00	1.78
19	1.59	11.65	1.00	1.78
20	1.65	11.20	1.00	1.78
21	1.63	11.05	1.00	1.78
22	1.43	9.19	1.00	1.69
23	1.66	7.86	1.00	1.69
24	1.26	7.52	1.00	1.69
25	1.63	9.65	1.00	1.69
26	1.56	9.70	1.00	1.69
27	1.26	13.26	1.00	1.69
28	1.59	6.45	1.00	1.69
29	1.28	12.38	1.00	1.69
30	1.27	7.00	1.00	1.69

(iii) HS method: the harmony memory size HMS $= 80$, the harmony memory consideration rate HMCR $= 0.88$, while the pitch adjusting rate *PAR* was taken equal to 0.25 based on the parameter study presented in Table 4.

(iv) CMA method: the number of parents $\mu = 32$ and the number of offspring $\lambda = 61$ based on the parameter study presented in Table 5 while the rest of the parameters are defined according to the recommendations of [31].

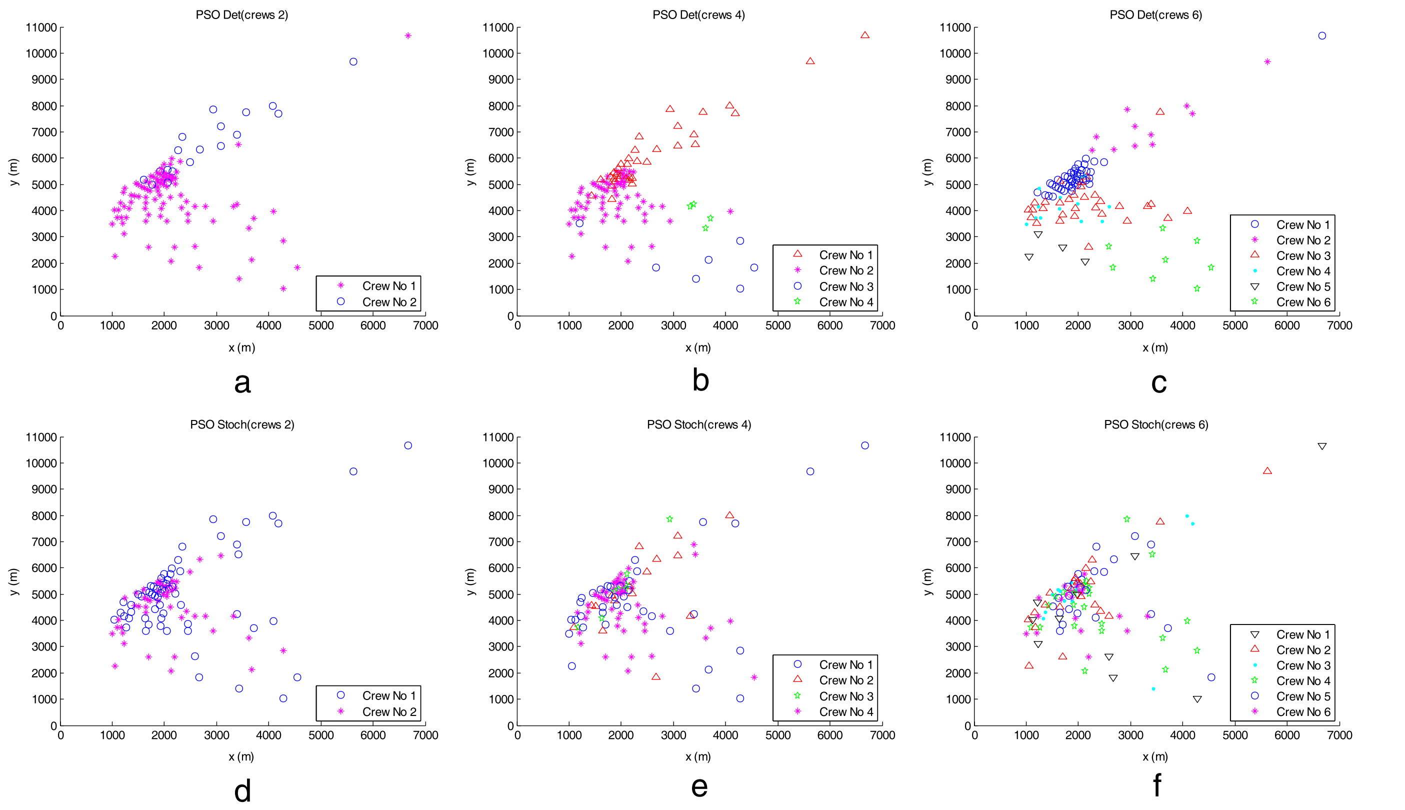


Fig. 10. City of Patras—PSO method subdivision into structural blocks.

(v) ECMA method: the number of parents $\mu = 1$ and the number of offspring $\lambda = 85$ based on the parameter study presented in Table 6 while the rest of the parameters are defined according to the recommendations of [32].

For comparison purposes, the termination criterion is the same for all five metaheuristic optimization algorithms; each procedure is terminated after 10^6 function evaluations have been performed. Comparing the mean time, among the 100 independent optimization runs for a certain combination of parameters, required for performing an independent optimization run PSO requires 1.56 min, DE 15.4 min, HS 0.34 min, CMA 3.12 min while ECMA 12.3 min.

6.2. Deterministic and probabilistic distribution of the damages

In the second part of this study the best combinations of the parameters, found in the previous part of the investigation, are used for solving the problems defined in Eqs. (2) and (3). The resulting optimization runs for dealing with the districting problem were equal to 5 metaheuristics $\times$ 3 inspection groups $\times$ 2 formulations $=$ 30 optimization runs while for the case of the scheduling problem 5 metaheuristics $\times$ i inspection groups $\times$ 2 formulations $=$ 120 optimization runs (where $i = 2$, 4 or 6). In the case of 2 inspection groups 20 runs were performed, for the 4 inspection groups 40 runs were performed, while for the 6 inspection groups 60 runs were performed, thus totaling 120 optimization runs with the ACO algorithm. In a similar procedure to the one described in the previous section we have defined the parameters used for the ACO algorithm. In particular the size of the population of ants m was defined in the range of [10, 100], the parameters α and β were defined in the range of [0, 2] and [−1, 1], respectively; the constant Q was defined in the range of [0, 1] while the rate of evaporation ρ was defined in the range of [0, 1). The best combination of the ACO algorithm is: the size of the population of ants $m = 25$, the parameter $\alpha = 1.2$, the parameter $\beta = 0.8$, the constant $Q = 0.4$ while the rate of evaporation $\rho = 0.15$. All independent runs were performed for the case of the scheduling problem of the first crew as defined by the DE for the deterministic formulation, while 200 optimization iterations were considered.

Table 7
Scheduling applied to the deterministic formulation—mean distance (km).

Method	Crews		
	2	4	6
PSO	43.78	30.89	26.38
DE	40.12	18.11	10.48
HS	48.31	28.54	26.08
CMA	48.10	32.70	25.31
ECMA	50.79	29.54	22.80

Initially, a deterministic distribution of damages is examined as described in Eq. (2). Figs. 10(a)–(c), 11(a)–(c), 12(a)–(c), 13(a)–(c), 14(a)–(c) depict the solutions obtained for the optimal allocation problem for the three different number of inspection groups considered (2, 4 and 6 crews), when the five metaheuristics are implemented. As can be seen from these figures, different solutions are obtained. In order to compare the resulted optimum designs, the scheduling problem has to be solved for each metaheuristic and for each inspection group by means of the ACO method. Therefore, the inspection prioritization problem defined in Eq. (5) is solved by means of the ant colony optimization algorithm. Fig. 15(a)–(d) depict indicative optimal routes achieved, that correspond to the case of DE method when 4 inspection groups are employed, along with the convergence histories of ACO algorithm. The vertical axis is the minimum distance path among the ants in every iteration. These solutions correspond to the least time consuming route required for each inspection crew departing from their base (Table 7 provides the mean distances required by the inspection groups). As can be seen, DE method outperforms all other methods resulting in the least mean traveling distance for all three inspection groups.

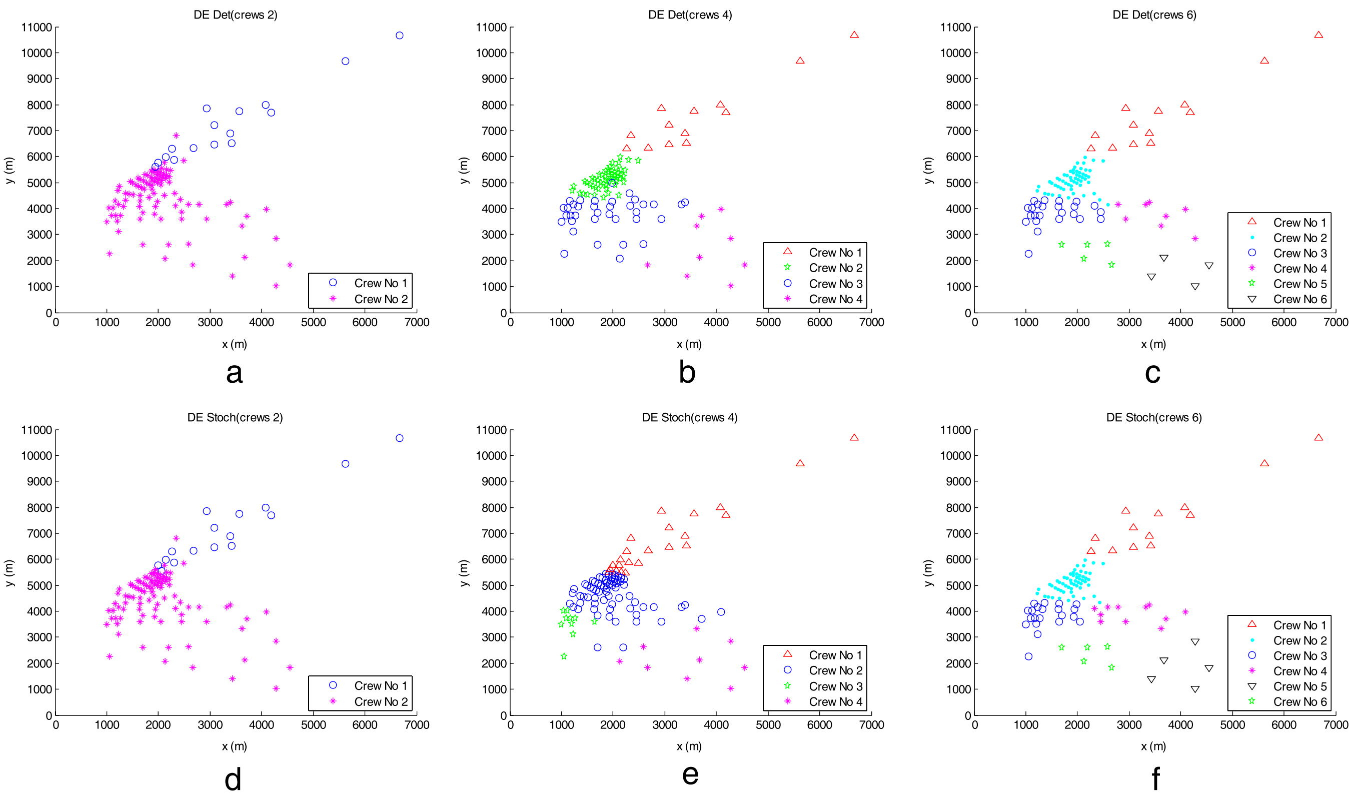


Fig. 11. City of Patras—DE method subdivision into structural blocks.

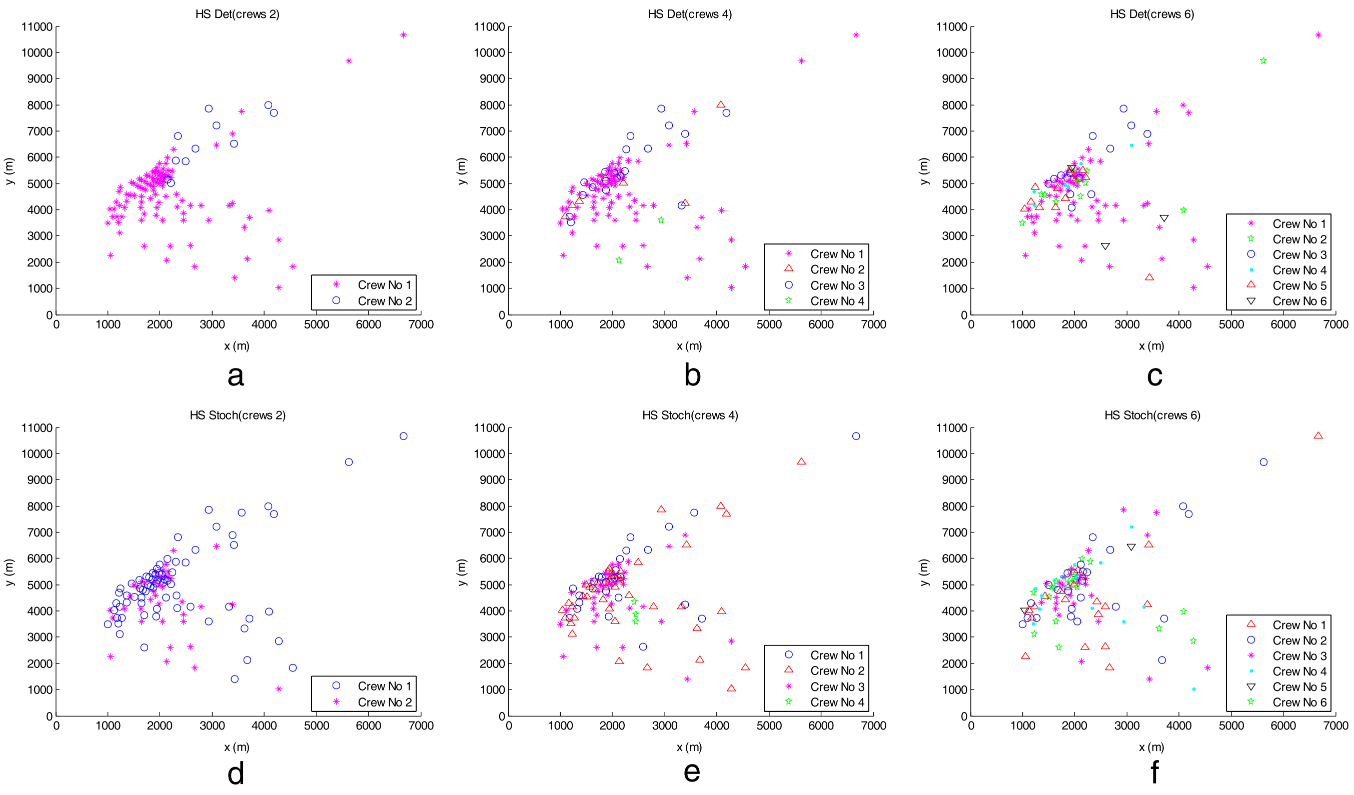


Fig. 12. City of Patras—HS method subdivision into structural blocks.

In the second part, a non-uniform distribution of damages is examined as described in the problem formulation of Eq. (3). The mean damage level for each region is shown in Fig. 9(b). The damages follow the Gaussian distribution with mean value of 0, 1, 2 and 3 for the four zones (Fig. 9(b)). In this formulation and in order to calculate the objective function given in Eq. (6), the LHS method is implemented considering 100 simulations. Figs. 10(d)–(f), 11(d)–(f), 12(d)–(f), 13(d)–(f), 14(d)–(f) depict

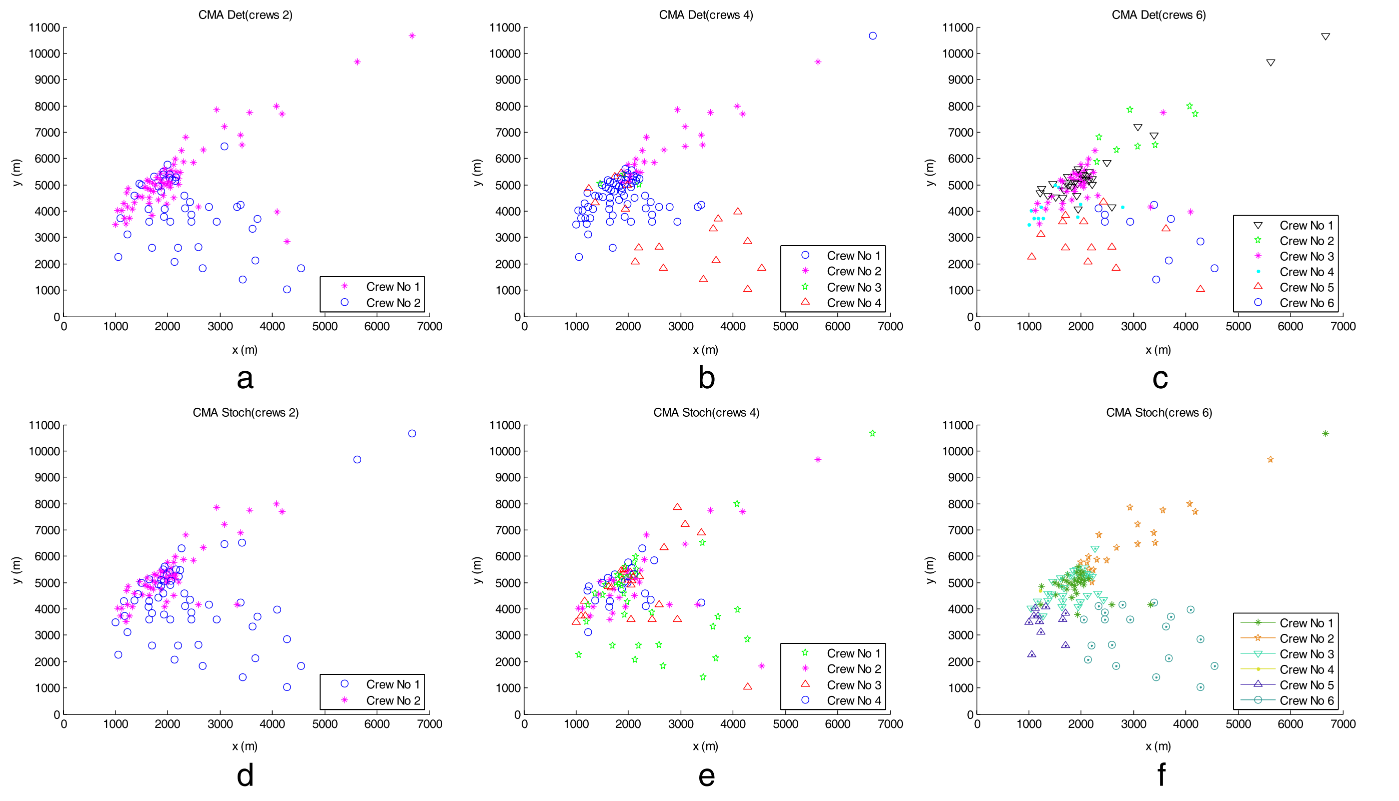


Fig. 13. City of Patras—CMA method subdivision into structural blocks.

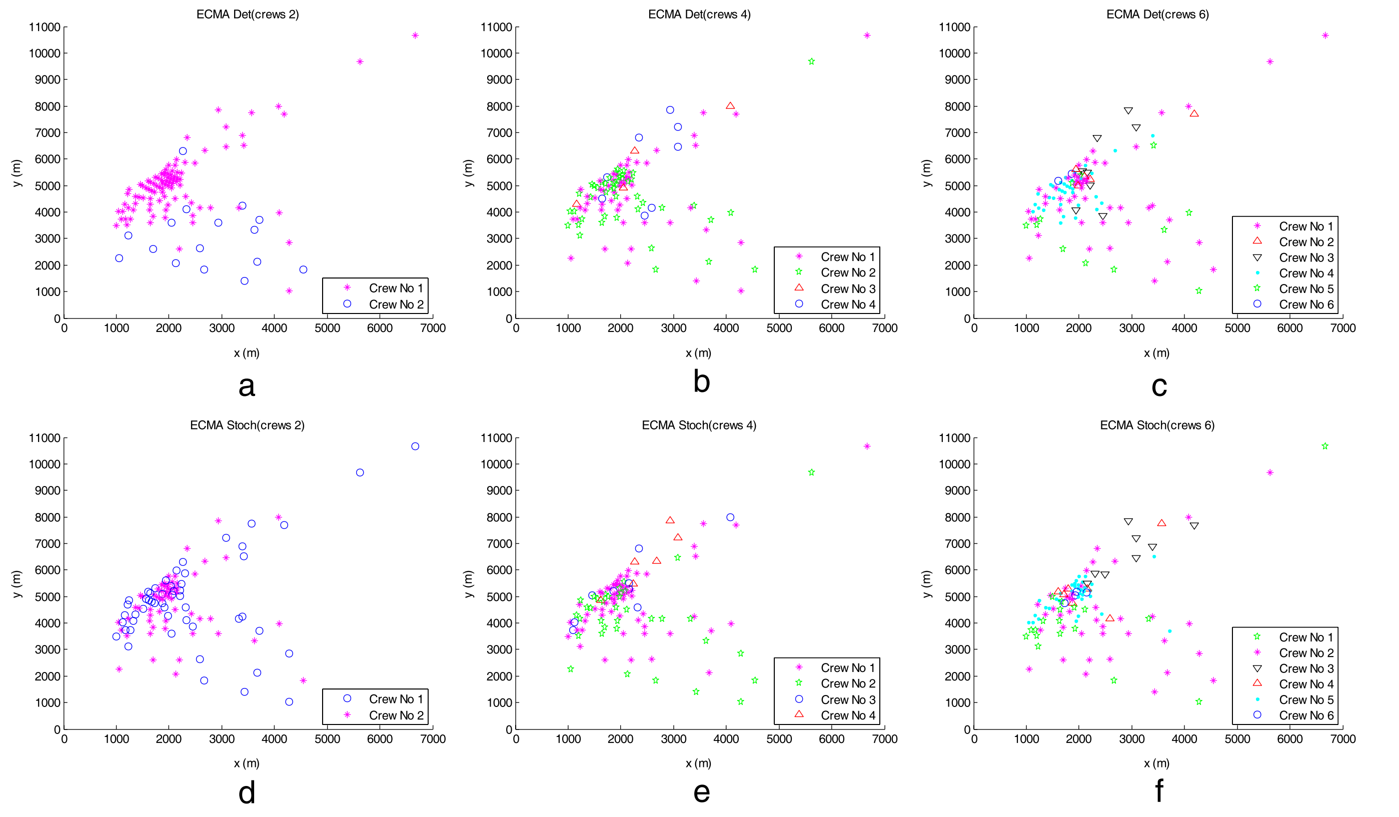


Fig. 14. City of Patras—ECMA method subdivision into structural blocks.

the solutions obtained for the optimal allocation problem for the three different number of inspection groups when the five metaheuristics are implemented. As can be seen from these figures, similar to the deterministic formulation, different solutions are obtained. In order to compare the resulting optimal designs, the scheduling problem has also been solved for each metaheuristic

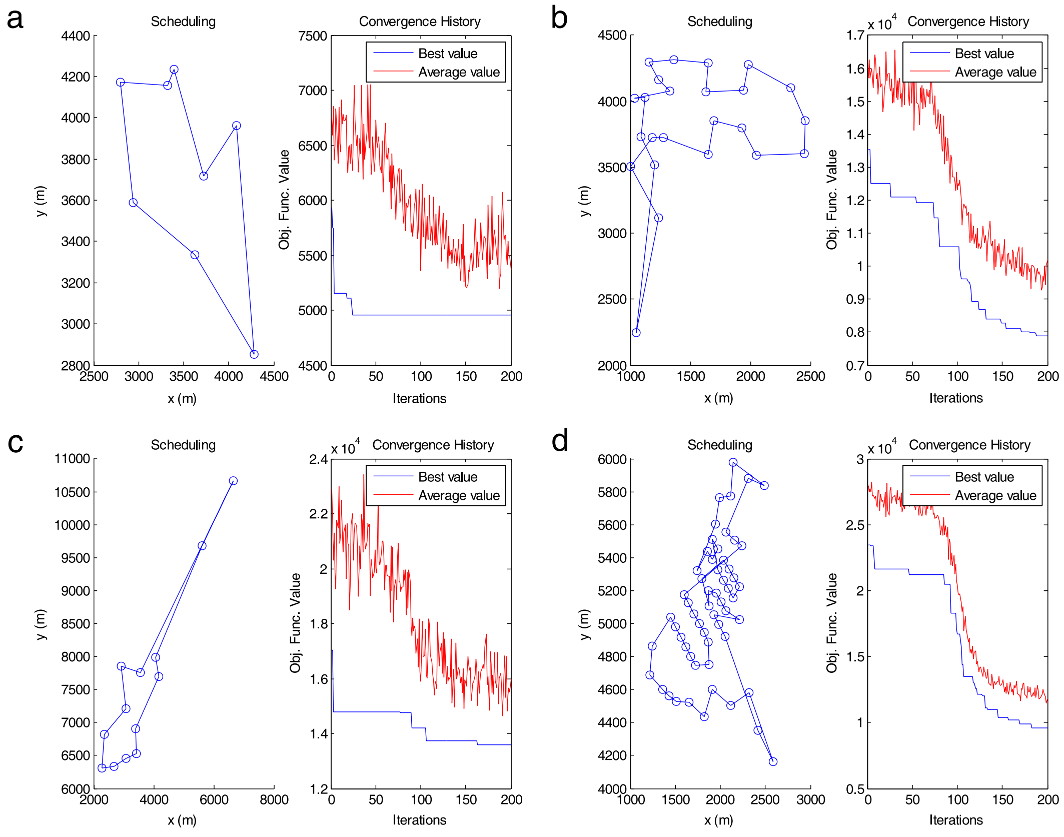


Fig. 15. City of Patras—best route for DE method for four inspection groups (a) group A, (b) group B, (c) group C and (d) group D for the deterministic formulation.

Table 8
Scheduling applied to the stochastic formulation—mean distance (km).

Method	Crews		
	2	4	6
PSO	47.69	19.57	17.01
DE	39.25	16.86	10.98
HS	39.79	26.55	22.84
CMA	46.12	23.13	17.40
ECMA	42.62	31.26	19.93

and for each inspection group by means of the ACO method. Therefore, the inspection prioritization problem defined in Eq. (5) is solved by means of the ant colony optimization algorithm. Fig. 16(a)–(d) depict indicative optimal routes achieved that corresponds to the case of DE method when 4 inspection groups are employed, along with the convergence histories of the ACO algorithm. The vertical axis is the minimum distance path among the ants for every iteration. These solutions correspond to the least time consuming route required for each inspection crew departing from their base (Table 8 provides the mean distances required by the inspection groups). As can be seen, similar to the deterministic formulation DE method outperforms all other methods resulting to the least required mean traveling distance for all three inspection groups.

7. Conclusions

In this work we offered an approach for scheduling critical infrastructure inspection crews following an earthquake in densely populated metropolitan regions, and considered two important issues within the scope of post natural disaster actions. First, we developed deterministic and probabilistic districting and routing problems for scheduling infrastructure inspection crews following a natural disaster in urban areas. Furthermore, two formulations were implemented: in the first, the structural blocks were assigned to different inspection groups in order to homogeneously distribute the workload between the groups; in the second, the optimal route within each group was determined so as to minimize the distance that each inspection group has to cover. Second, we assessed and compared five metaheuristic optimization algorithms for solving these districting and routing problems. In particular the differential evolution, harmony search, particle swarm optimization, covariance matrix adaptation evolution strategy and elitist covariance matrix adaptation. The ant colony approach was implemented for dealing with the routing problem; both steps resulted in tractable and rapid response models.

The paper was composed of two parts. In the first, various combinations of the parameters that affect the performance of the metaheuristics are tested with reference to their performance for the solution of the districting (deterministic formulation) and scheduling problems. In particular, the particle swarm optimization method depicts a significant influence on the parameters since the mean best value is increased by up to 130%, while only few combinations of the parameters converge to the best optimum. In the case of the harmony search method, the mean best value is increased by up to 30%, while only half of the parameters' combinations lead to the best optimum. In the case of the covariance matrix adaptation evolution strategy method, the mean best value is increased by up to 30%, while all the parameters' combinations lead to an almost best optimum. In the case of the elitist covariance matrix adaptation evolution strategy method, the mean best value is increased by up to 70%, while all the parameters' combinations lead to an almost best optimum. On the other hand in the case of the differential evolution method, the mean best value is increased by up to 45%, while all combinations of the parameters lead to the best optimum.

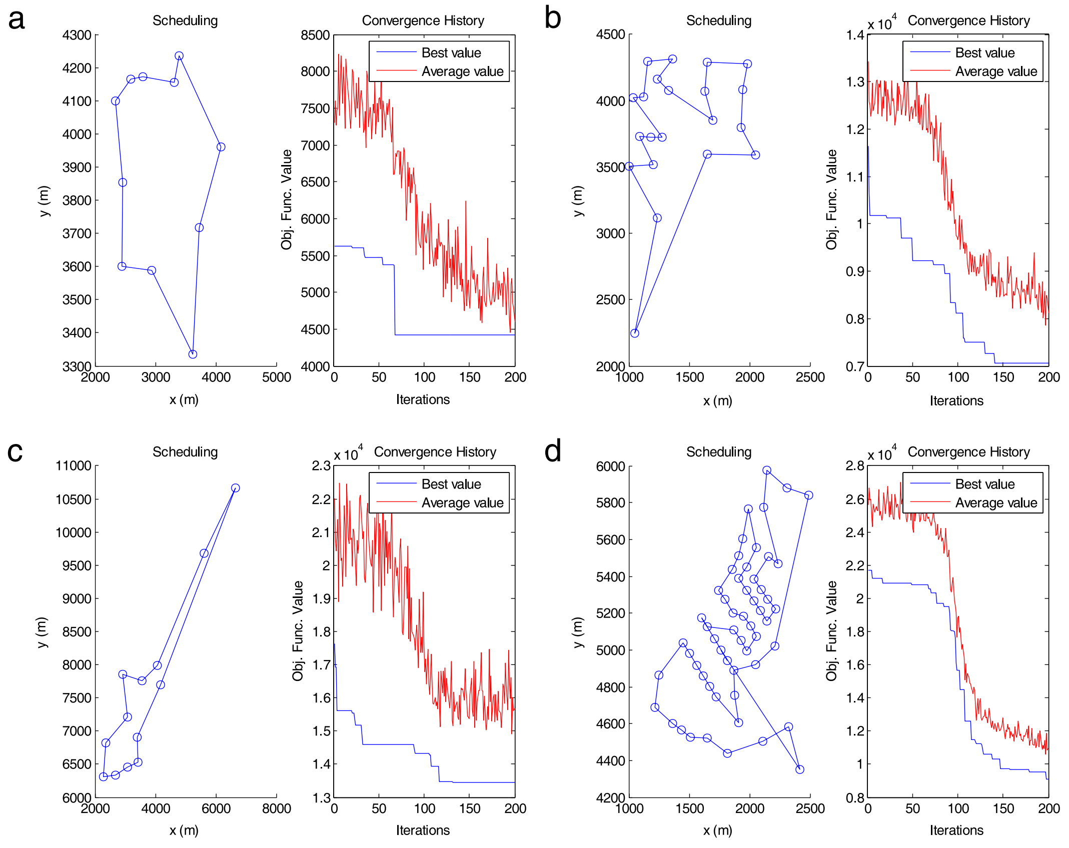


Fig. 16. City of Patras—best route for DE method for four inspection groups (a) group A, (b) group B, (c) group C and (d) group D for the stochastic formulation.

In the second, the best combination of parameters is used for solving both deterministic and probabilistic formulations of the districting problem. Results suggest that for both the deterministic and the probabilistic formulations the differential evolution method outperformed all other approaches, resulting to the lowest required mean travel distance regardless of the number of inspection groups available.

It should be stated that the authors have implemented the variants of the metaheuristics presented in the previous sections of this study and therefore the results and the superiority of one algorithm compared to the others refers to these variants. Recently various variants of the metaheuristics have been proposed; like those of differential evolution implementing trigonometric mutation, using arithmetic recombination, with neighborhood-based mutation, with adaptive selection of mutation strategies, or the adaptive with current-to-pbest mutation presented in [51]; the variants of harmony search like the explorative one, the self-adaptive global best algorithm proposed in [58,59] or the variants of particle swarm optimization like the improved particle swarm optimization with differentially perturbed velocity, the hybrid cooperative algorithm, the adaptive algorithm and the dynamic multi-swarm particle swarm optimizer with harmony search presented in [51,60,61]. The conclusions can be different if these or other variants are implemented.

References

[1] N. Altay, W.G. Greene, OR/MS research in disaster operations management, European Journal of Operational Research 175 (2006) 475–493.
[2] W.M. Dong, W.L. Chiang, H.C. Shah, Fuzzy information processing in seismic hazard analysis and decision making, Soil Dynamics and Earthquake Engineering 6 (4) (1987) 2202–2226.
[3] W. Peizhuangm, L. Xihui, E. Sanchez, Set-valued statistics and its application to earthquake engineering, Fuzzy Sets and Systems 18 (3) (1986) 347–356.
[4] H. Tamura, K. Yamamoto, S. Tomiyama, I. Hatono, Modelling and analysis of decision making problem for mitigating natural disaster risks, European Journal of Operational Research 122 (2) (2000) 461–468.
[5] D. Mendonca, G.E.G. Beroggi, W.A. Wallace, Decision support for improvisation during emergency response operations, International Journal of Emergency Management 1 (1) (2001) 30–38.
[6] D. Mendonca, G.E.G. Beroggi, D. van Gent, W.A. Wallace, Designing gaming simulations for the assessment of group decision support systems in emergency response, Safety Science 44 (2006) 523–535.
[7] K. Viswanath, S. Peeta, Multicommodity maximal covering network design problem for planning critical routes for earthquake response, Transportation Research Record 1857 (2003) 1–10.
[8] A. Nicholson, Z.-P. Du, Degradable transportation networks systems: an integrated equilibrium model, Transportation Research Part B 31 (3) (1997) 209–223.
[9] H. Sakakibara, Y. Kajitani, N. Okada, Road network robustness for avoiding functional isolation in disasters, Journal of Transportation Engineering 130 (5) (2004) 560–567.
[10] J. Sohn, Evaluating the significance of highway network links under the flood damage: an accessibility approach, Transportation Research Part A 40 (6) (2006) 491–506.
[11] J. Song, T.J. Kim, G.J.D. Hewings, J.S. Lee, S.-G. Jang, Retrofit priority of transport network links under an earthquake, Journal of Urban Planning and Development 129 (4) (2003) 195–210.
[12] V. Verter, S. Lapierre, Location of preventive healthcare facilities, Annals of Operations Research 110 (2002) 123–132.
[13] G. Barbarosoglou, Y. Arda, A two-stage stochastic programming framework for transportation planning in disaster response, Journal of the Operational Research Society 55 (1) (2004) 43–53.
[14] G. Barbarosoglou, L. Ozdamar, A. Cevik, An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations, European Journal of Operational Research Society 140 (1) (2002) 118–133.
[15] F. Fiedrich, F. Gehbauer, U. Rickers, Optimized resource allocation for emergency response after earthquake disasters, Safety Science 35 (1–3) (2000) 41–57.
[16] L. Ozdamar, E. Ekinci, B. Kucukyazici, Emergency logistics planning in natural disasters, Annals of Operations Research 129 (1–4) (2004) 217–245.
[17] M.G.H. Bell, A game theory approach to measuring the performance reliability of transportation networks, Transportation Research Part B 34 (6) (2000) 533–545.

[18] S.E. Chang, N. Nojima, Measuring post-disaster transportation system performance: the 1995 Kobe earthquake in comparative perspective, Transportation Research Part A 35 (6) (2001) 475–494.
[19] F. Karaouchi, Y. Lida, H. Shimada, Evaluation of road network reliability considering traffic regulation after a disaster, in: M.G.H. Bell, Y. Lida (Eds.), The Network Reliability of Transport: Proceedings of the 1st International Symposium on Transportation Network Reliability, INSTR, Elsevier, Oxford, UK, 2001.
[20] Y. Li, H. Tsukaguchi, Improving the reliability of street networks in highly densely populated urban areas, in: M.G.H. Bell, Y. Lida (Eds.), The Network Reliability of Transport: Proceedings of the 1st International Symposium on Transportation Network Reliability, INSTR, Elsevier, Oxford, UK, 2001.
[21] L. Cret, F. Yamakazi, S. Nagata, T. Katayama, Earthquake damage estimation and decision-analysis for emergency shutoff of city gas networks using fuzzy set theory, Structural Safety 12 (1) (1993) 1–19.
[22] B. Song, S. Hao, S. Murakami, S. Sadohara, Comprehensive evaluation method on earthquake damage using fuzzy theory, Journal of Urban Planning and Development 122 (1) (1996) 1–17.
[23] M.G. Karlaftis, K.L. Kepaptsoglou, S. Lampropoulos, Fund allocation for transportation network recovery following natural disasters, Journal of Urban Planning and Development 133 (1) (2007) 82–89.
[24] V. Plevris, M.G. Karlaftis, N.D. Lagaros, A swarm intelligence approach for emergency infrastructure inspection scheduling, in: K. Gopalakrishnan, S. Peeta (Eds.), Sustainable Infrastructure Systems: Simulation, Imaging, and Intelligent Engineering, Springer, 2010, pp. 201–230.
[25] N. Geroliminis, M.G. Karlaftis, A. Skabardonis, A spatial queuing model for the emergency vehicle districting and location problem, Transportation Research Part B 43 (2009) 798–811.
[26] M.G. Karlaftis, K. Kepaptsoglou, E. Sambracos, Containership routing with time deadlines and simultaneous deliveries and pickups, Transportation Research Part E: Logistics and Transportation Review 45 (2009) 210–221.
[27] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley, 1966.
[28] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley, 1989.
[29] I. Rechenberg, Evolutionsstrategie—Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution, Fromman-Holzboog, 1973.
[30] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. IV, 1995.
[31] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computations 9 (2) (2001) 159–195.
[32] C. Igel, N. Hansen, S. Roth, Covariance matrix adaptation for multi-objective optimization, Evolutionary Computations 15 (1) (2007) 1–28.
[33] R.M. Storn, K.V. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[34] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2001) 60–68.
[35] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 4661–4667.
[36] O. Bozorg Haddad, A. Afshar, M.A. Mariño, Honey bees mating optimization algorithm (HBMO): a new heuristic approach for engineering optimization, in: Proceeding of the First International Conference on Modelling, Simulation and Applied Optimization, ICMSA0/05, Sharjah, UAE, 2005, pp. 1–3.
[37] N. Perrier, A. Langevin, J.F. Campbell, A survey of models and algorithms for winter road maintenance. Part III: vehicle routing and depot location for spreading, Computers & Operations Research 34 (2007) 211–257.
[38] K.C. Tan, Y.H. Chew, L.H. Lee, A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems, European Journal of Operational Research 172 (2006) 855–885.
[39] P. Reche-Lopez, N. Ruiz-Reyes, S. Garcia Galan, F. Jurado, Comparison of metaheuristic techniques to determine optimal placement of biomass power plants, Energy Conversion and Management 50 (2009) 2020–2028.
[40] M.A. Salazar-Aguilar, R.Z. Rvos-Mercado, J.L. Gonzalez-Velarde, A bi-objective programming model for designing compact and balanced territories in commercial districting, Transportation Research Part C: Emerging Technologies 19 (5) (2011) 885–895.
[41] P. Balaprakash, M. Birattari, T. Stutzle, M. Dorigo, Estimation-based metaheuristics for the probabilistic travelling salesman problem, Computers & Operations Research 37 (2010) 1939–1951.
[42] P.C. Pop, New integer programming formulations of the generalized travelling salesman problem, American Journal of Applied Sciences 4 (11) (2007) 932–937.
[43] P.C. Pop, O. Matei, C. Sabo, A new approach for solving the generalized travelling salesman problem, in: Lecture Notes in Computer Science, Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, in: LNCS, vol. 6373, 2010, pp. 62–72.
[44] I. Lari, F. Ricca, A. Scozzari, Comparing different metaheuristic approaches for the median path problem with bounded length, European Journal of Operational Research 190 (2008) 587–597.
[45] W. Bo-Zejko, M. Wodecki, Solving permutational routing problems by population-based metaheuristics, Computers & Industrial Engineering 57 (2009) 269–276.
[46] M. Mastrolilli, C. Blum, On the use of different types of knowledge in metaheuristics based on constructing solutions, Engineering Applications of Artificial Intelligence 23 (2010) 650–659.
[47] A.L. Jourdan, E.G. Talbi, Metaheuristics and cooperative approaches for the bi-objective ring star problem, Computers & Operations Research 37 (2010) 1033–1044.
[48] E. Vallada, Rubén Ruiz, Cooperative metaheuristics for the permutation flowshop scheduling problem, European Journal of Operational Research 193 (2009) 365–376.
[49] M. Lozanoa, C. García-Martínez, Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report, Computers & Operations Research 37 (2010) 481–497.
[50] P.R. Bergamaschi, S.F.P. Saramago, L.S. Coelho, Comparative study of SQP and metaheuristics for robotic manipulator design, Applied Numerical Mathematics 58 (2008) 1396–1412.
[51] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.
[52] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, New York, USA, 1985.
[53] M. Dorigo, Optimization, learning and natural algorithms, Politecnico di Milano, Milano, 1992.
[54] M. Dorigo, T. Stützle, Ant Colony Optimization, The MIT Press, 2004.
[55] http://www.statistics.gr/.
[56] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.
[57] A.E. Eiben, S.K. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 19–31.
[58] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41 (1) (2011) 89–106.
[59] Q.-K. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems, Applied Mathematics and Computation 216 (3) (2010) 830–848.
[60] B.K. Panigrahi, V. Ravikumar Pandi, S. Das, Adaptive particle swarm optimization approach for static and dynamic economic load dispatch, Energy Conversion and Management 49 (6) (2008) 1407–1415.
[61] S.-Z. Zhao, P.N. Suganthan, Q.-K. Pan, M. Fatih Tasgetiren, Dynamic multi-swarm particle swarm optimizer with harmony search, Expert Systems with Applications 38 (4) (2011) 3735–3742.