Contents lists available at ScienceDirect

Applied Soft Computing

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

Weighted Superposition Attraction (WSA): A swarm intelligence algorithm for optimization problems – Part 2: Constrained optimization

Adil Baykasoğlu*, Şener Akpinar

Dokuz Eylül University, Faculty of Engineering, Department of Industrial Engineering, Izmir, Turkey

ARTICLE INFO

Article history:
Received 14 May 2015
Received in revised form 27 July 2015
Accepted 27 August 2015

Keywords:
WSA algorithm
Non-linear programming
Constrained global optimization
Design optimization
Constraint handling

ABSTRACT

This paper is the second one of the two papers entitled "Weighted Superposition Attraction (WSA) Algorithm", which is about the performance evaluation of the WSA algorithm in solving the constrained global optimization problems. For this purpose, the well-known mechanical design optimization problems, design of a tension/compression coil spring, design of a pressure vessel, design of a welded beam and design of a speed reducer, are selected as test problems. Since all these problems were formulated as constrained global optimization problems, WSA algorithm requires a constraint handling method for tackling them. For this purpose we have selected 6 formerly developed constraint handling methods for adapting into WSA algorithm and analyze the effect of the used constraint handling method on the performance of the WSA algorithm. In other words, we have the aim of producing concluding remarks over the performance and robustness of the WSA algorithm through a set of computational study in solving the constrained global optimization problems. Computational study indicates the robustness and the effectiveness of the WSA in terms of obtained results, reached level of convergence and the capability of coping with the problems of premature convergence, trapping in a local optima and stagnation.

1. Introduction

Optimization consist all of the endeavours within the purpose of systematically improving the effectiveness of a system or designing a system as efficient as possible. In other words, optimization procedures aim at finding the optimum values for the decision variables of the problem on hand by realizing some specific rules. During the optimization process it is possible to use different optimization procedures. The key issue is to determine the proper one thereby the most powerful optimization method, since the classical optimization techniques impose some limitations on solving complex optimization problems [1]. Within this context, many researchers have been interested in developing effective optimization procedures or improving the effectiveness of the existing optimization procedures for different optimization problems, which may arise in such fields of science, engineering and operational research.

The domain of the decision variables of an optimization problem specifies the type of the optimization problem. Optimization problems are divided into three categories by the means of the domain of decision variables: (i) problems having exclusively discrete decision variables, (ii) problems having exclusively continuous decision variables, and (iii) problems having both discrete and continuous decision variables. The optimization problems belonging into these three categories may also be classified into two groups: constrained and unconstrained global optimization problems. This current paper is about to tackle the constrained global optimization problems via Weighted Superposition Attraction (WSA) algorithm, which was developed by Baykasoğlu and Akpınar in the Part-I of this paper as a new member of swarm based meta-heuristic algorithms and used to solve unconstrained global optimization problems [2].

Global optimization, a branch of applied mathematics and numerical analysis, is a challenging research field, since many real word applications could be formulated as a global optimization problem. Engineering design, production management, computational chemistry, and environmental pollution management are some of the application fields of the global optimization. The rapidly developed global optimization techniques has taken interest in different scientific domains such as applied mathematics, operations research, industrial engineering, management science and computer science. Within this context, this current part of this paper

DOI of original article: http://dx.doi.org/10.1016/j.asoc.2015.10.036.
* Corresponding author. Tel.: +90 232 301 16 00; fax: +90 232 301 76 08.
E-mail addresses: adil.baykasoglu@deu.edu.tr (A. Baykasoğlu), sener.akpinar@deu.edu.tr (Ş. Akpinar).

http://dx.doi.org/10.1016/j.asoc.2015.08.052
1568-4946/© 2015 Elsevier B.V. All rights reserved.

Notations of WSA and their definitions

Notation	
Maxiter	Iteration number (stopping condition)
Iteration	Current iteration number
AA	Number of artificial agents
D	Number of dimensions of the problem
τ	User defined parameter
λ	User defined parameter
φ	User defined parameter
UL	Upper limit for the dimensions
LL	Lower limit for the dimensions
$f(i)$	Fitness of the current point of agent i
$f(tar)$	Fitness of the target point
weight	Weight of the current point of an agent
$\vec{x}$	Current position vector of an agent
$\rightarrow tar$	Position vector of the target point
$\rightarrow gap$	Vector combines an agent to target point
$\rightarrow direct$	Move direction vector of an agent
$sign()$	Signum function
sl	Step length

mainly focus on analysing the performance of WSA algorithm on the well-known constrained mechanical design optimization problems, which are formulated as complex nonlinear programming models.

A theoretical background of the engineering design optimization problems was reported by Belegundu [3] and the basic concepts and methods for these problems were described by Arora via some examples [4]. For the global optimization problems, the developed methods should monitor the approach of the algorithms towards the optimum point via a descent function [4] refers to the cost function of the problem. Nevertheless, the descent function must be aware of the feasibility for the constrained global optimization problems. For that reason, it is required to construct the descent function by adding a penalty to the current value of the cost function in case of the constraint violations and researchers would rather derive the penalty value via some formulations named as constraint handling methods.

An improvement for the conventional constraint handling methods was introduced by Coello [5,6]. Their notation uses adaptive penalty factors for a Genetic Algorithm (GA) implementation, however, it has the potential of to be generalizing to any meta-heuristic. Coello and Mezura-Montes proposed another constraint handling method in order to overcome the constraint violations through the selection operator of GA [7,8]. A simple evolution strategy based approach, which does not require a penalty function, was presented by Mezura-Montes et al. [9]. Coello and Becerra proposed a cultural algorithm, which builds the map of the feasible region during the evolutionary process in order to avoid infeasibility and improves the performance of an evolutionary programming technique [10]. Another evolutionary-based approach free from penalty functions was proposed by Mezura and Coello in order to identify and maintain infeasible solution close to feasible region located in promising areas [11]. Furthermore, Akhtar et al. and Ray and Liew presented solution approaches based on the phenomenon of society and civilization for constrained engineering design optimization problems [12,13].

Ray and Sani presented a swarm based approach that realizes a Pareto ranking scheme as a constraint handling method [14]. A feasibility preserving Particle Swarm Optimization (PSO), which is a well-known member of swarm based algorithms, was developed by Hu et al. [15]. A constraint handling approach named as fly-back method, makes any individual to return to its previous position it violates any problem constraint, was realized by He et al. in order to improve the performance of PSO while solving mechanical design optimization problems [16]. Parsopoulos and Vrahatis tested the performance of a unified PSO method by realizing a penalty function and a feasibility preserving modification of the algorithm [17]. A constraint handling technique based on feasibility and sum of constraints violation was realized by Aguirre et al. while tackling the constrained optimization problems via PSO [18]. He and Wang incorporate a co-evolution model into PSO for the first time and their approach realized two types of swarms, multiple swarms for searching satisfactory solutions and a single swarm for evolving suitable penalty factors [19]. A novel hybrid PSO was also presented by He and Wang with a feasibility-based rule has the aim of overcoming the deficiencies of penalty function methods [20]. Tomassetti proposed another hybrid PSO, which was inspired from evolutionary algorithms and realizes multi-start approach, randomly reinitializing the swarm and updating the inertia factor multiplying the previous velocity of the swarm with the aims of enlarging the exploration space, accelerating convergence to the optimal solution and avoiding the algorithm to remain trapped into local minima, respectively [21]. Cagnina et al. proposed a PSO with simple pairwise-comparison based constraint handling method and their method adds an amount of violation, normalized with respect to the largest violation stored far, for the infeasible solutions [22]. A new tool based on PSO proposed by Maruta et al. and it is applicable for a broad class of non-convex problems directly [23]. Kim et al. and Chun et al. proposed a PSO algorithm with a constraint handling method that transforms the given constrained optimization problem into an unconstrained problem without introducing extra problem-dependent parameters such as penalty factors or Lagrange multipliers [24,25].

From the previous two paragraphs, it can be obviously seen that GA, PSO and evolutionary strategies were widely used for solving constrained global optimization such as mechanical design optimization problems. Besides these solution methodologies, some other meta-heuristic algorithms (Artificial Bee Colony Algorithm (ABC) [26,27], Differential Evolution (DE) [28], Harmony Search Algorithm (HSA) [29,30], Cuckoo Search Algorithm (CSA) [31], Bat Algorithm [32], Great Deluge Algorithm (GDA) [1] and Firefly Algorithm (FA) [33,34] were also preferred by varied researchers with the aim of tackling the related problems. Additionally, it must be stated that current competitive market and manufacturing conditions result many problems concerning design optimization and manufacturing parameters estimation. At this point, these problems required to be solved optimally and design optimization/manufacturing parameters estimation via meta-heuristics is a growing and challenging research field as can be realized from the existing literature [35–42].

The main research concern of this current paper is to analyze the performance of WSA algorithm on the well-known constrained mechanical design optimization problems. The WSA algorithm was originally designed for unconstrained continuous global optimization problems in the first part of this paper, thus it requires to be redesigned so as to solve constrained problems and a constraint handling approach must be inserted into WSA. Within this context, we select 6 different constraint handling approaches for implementing WSA through them and analyze if the performance of WSA is strongly related with the used constraint handling approach or not. By the way we will have the ability of providing concluding remarks about the robustness of WSA for solving constrained mechanical design optimization problems at the end of the computational study of this paper.

The remainder of this paper is organized as follows. In section 2, the WSA algorithm and the selected constraint handling methods are depicted. Comparative computational study is given in Section

```
1. Initialize algorithm parameters, best solution and best fitness
2. Generate a pre-defined number of initial solutions
3. Evaluate fitness values of initial solutions, and update best solution
   and best fitness
4. Iteration = 1
     while Iteration <= Maxiter
          •  rank solutions according to their fitness values
          •  assign a weight to each solution by considering their ranks
          •  determine a target point to move the solutions towards it
          •  evaluate the fitness value of the target point
          •  determine search directions for each solution considering
             the target point and its fitness value
          •  move each solution towards its determined direction
          •  evaluate fitness values of each solution, and update best
             solution and best fitness
        Iteration = Iteration + 1
     end while
```

Fig. 1. Main steps of WSA.

3. Finally, the discussions and conclusions are presented in Section 4.

2. Solution methodology

In this paper, we have the aim of investigating the effectiveness of WSA algorithm on constrained global optimization problems. Baykasoğlu and Akpınar introduced WSA as a new member of swarm-based meta-heuristic algorithms in the first part of this paper and they also indicated that WSA has a conspicuous effectiveness on unconstrained continuous global optimization problems [2]. In parallel with the aim of the first part this paper, we will be tested the performance of WSA on the well-known mechanical design problems in the second part of this paper. Additionally, WSA requires a constraint handling mechanism when dealing with these problems, since they formulated as constrained global optimization problems. Within this context, we will be adapted 6 formerly developed constraint handling mechanisms for WSA. The WSA algorithm and the selected 6 constraint handling mechanism will be explained in the following sub-sections in details.

2.1. WSA algorithm

Swarm intelligence has an attractant affect over the optimization research community, since both combinatorial and functional optimization problems were effectively solved by the search algorithms based on the simulation of routine behaviours of different swarms such as birds, fishes, ants, and honey bees. Similarly WSA was based on the superposition principle in combination with the attracted movement of agents that are observable in many systems; the same is also possible for social systems as well. This new member of the swarm-based meta-heuristic algorithms attempts to model and simulate the dynamically changing superposition due to the dynamic nature of the system in combination with the attracted movement of agents as indicated in the first part of this paper. The general flowchart of the WSA algorithm is given in Fig. 1.

2.1.1. Initializing of WSA

At the initializing phase WSA generates a pre-defined number of feasible solutions. Additionally, some parameter values required to be set in order to guide WSA during its search. After that, WSA starts its discovery on the solution space via its special search mechanism, which provides WSA to visit and evaluate different points of the search space and will be explained in the following sub-section with the title of neighbourhood mechanism.

2.1.2. Neighbourhood mechanism realized by WSA

As mentioned in the first part of this paper, WSA is based on the superposition principle in combination with the attractant movement of agents that are observable in many systems. Within this context, WSA realizes artificial agents on a solution space and these artificial agents discover the solution space by moving from one point to another until the optimum point is found. During their movement on the solution space, the agents decide their search direction via the guidance of the superposition of their current positions. An artificial agent is disposed to move towards the superposition, if it is attractive for the related artificial agent. WSA determines the superposition such that the weighted sum of the agents' position vectors forms the superposition and the weights are proportionally determined according to agents' current position finesses. This superposition is a target point, which may attract the agents and they may move towards this target point. Once the move direction was determined by the artificial agents, each agent has to decide the move distance from its current location. Briefly, the neighbourhood mechanism of WSA is based on the determination of a search direction and a step length.

2.1.2.1. Search direction. WSA positions artificial agents at randomly selected points from the solution space. Each artificial agent is employed with collecting data about the fitness value and the position vector of its current location. After that, WSA uses all the collected data in order to determine the aforementioned target point. For this purpose, WSA ranks the artificial agents in ascending order according to their current location's fitness values, since the current paper deals with the minimization problems. After that, WSA assigns some weights to each artificial agent in accordance with their ranks, the lower the rank has an artificial agent, the higher the weight will be assigned to it, and determines the target point with regard to these weights by using the procedure given in Fig. 2.

Whenever the target point determined, each artificial agent has to decide whether to move towards this point or not. Every artificial

```
$weight$ = zeros (1, $AA$)
$\overrightarrow{tar}$ = zeros (1, $D$)
for ( $i$ = 1 to $AA$) do
    $weight\ (1, i) = i^{-}$
        for ($j$ = 1 to $D$) do
            $\overrightarrow{tar}(1,j) = \overrightarrow{tar}(1,j) + \vec{x}(i,j) * weight(1,i)$
        endfor
endfor
```

Fig. 2. Target point determination procedure.

```
$\overrightarrow{gap} = zeros(AA, D)$
$\overrightarrow{direct} = zeros(AA, D)$
for ($i$ = 1 to $AA$) do
    if $f(i) >= f(tar)$
        for ($j$ = 1 to $D$) do
            $\overrightarrow{gap}(i,j) = \overrightarrow{tar}(1,j) - \vec{x}(i,j)$
        endfor
        for ($d$ = 1 to $D$) do
            $\overrightarrow{direct}(i,d) = sign(\ \overrightarrow{gap}(i,d))$
        endfor
    elseif $f(i) < f(tar)$
        if $rand() < e^{(f(i)\ -\ f(tar))}$
            for ($j$ = 1 to $D$) do
                $\overrightarrow{gap}(i,j) = \overrightarrow{tar}(1,j) - \vec{x}(i,j)$
            endfor
            for ($d$ = 1 to $D$) do
                $\overrightarrow{direct}(i,d) = sign(\ \overrightarrow{gap}(i,d))$
            endfor
        else
            for ($d$ = 1 to $D$) do
                $\overrightarrow{direct}(i,d) = sign(\ -1 + (1+1) * rand())$
            endfor
        endif
    endif
endfor
```

Fig. 3. Search direction determination procedure.

agent has to select one of two possible alternatives, moving towards the target point or not, by comparing the fitness value of the target point with the fitness value of its current location. If the target point is better than an artificial agent's current location, the related artificial agent certainly moves towards the target point. Otherwise, an artificial agent selects its move direction by comparing a randomly generated number via the value obtained as $e^{(f(i)-f(tar))}$, where $f(i)$ is fitness value of the current position of the related agent i and $f(tar)$ is the fitness value of the target point. If the random number is lower than the obtained value, the related artificial agent decides to move towards the target point. Otherwise, it moves towards a randomly selected direction. The search direction determination procedure realized by WSA is given in Fig. 3.

```
if $rand()$ <=
        $sl = sl - e^{(mod((k\ /\ k\ +\ 1),1))} * \quad * sl$
else
        $sl = sl + e^{(mod((k\ /\ k\ +\ 1),1))} * \quad * sl$
endif
for ($i$=1 to $AA$) do
        for ($d$=1 to $D$) do
                $\vec{x}(i,d) = \vec{x}(i,d) + sl * \overrightarrow{direct}(i,d) * abs(\vec{x}(i,d))$
                if $initial\ (i,d)\ <\ LL$
                        $initial(i,d)\ =\ LL$
                elseif $initial(i,d)\ >\ UL$
                        $initial(i,d)\ =\ UL$
                endif
        endfor
endfor
```

Fig. 4. Position update procedure.

2.1.2.2. Step length. As mentioned earlier, neighbourhood mechanism has two major components, determination a search length and a step length. Once an artificial agent decided to move towards a direction, then it must decide the length of its move. After giving these decisions, each artificial agent updates its position by using a position updating mechanism [43]. By this way, artificial agents move one position to another for discovering its current position's neighbour area. The related position updating mechanism is given by Eq. (1) and agent i ($1 \le i \le AA$) update its position on dimension j ($1 \le j \le D$) at iteration t as below.

$$x_{ij}(t+1) = x_{ij}(t) + sl(t) * d_{ij}(t) * x_{ij}(t) \tag{1}$$

where $x_{ij}(t)$ is the value of the position of agent i on dimension j at iteration t, $sl(t)$ is the value of step length at iteration t, $d_{ij}(t)$ is the search direction of agent i on dimension j at iteration t, $x_{ij}(t)$ is the norm of the position vector of agent i on dimension j at iteration t, and $d_{ij}(t) \in \{-1, 0, 1\}$.

The second term $(sl(t)^{*}d_{ij}(t)^{*}x_{ij}(t))$ of the right hand side of Equation 1 updates the position of an artificial agent. Properly determining the step length makes the WSA algorithm to search solution space intensively and provide a satisfactory performance, from the optimization point of view. WSA realizes the step sizing function developed by [43] on the basis of the one that represented by [1]. This function is given by Eq. (2) and requires an initial step length (sl_0) to be set. As the search progress WSA varies the step length via a proportional rule, which uses a randomly generated number r between 0 and 1 and a user defined parameter λ (*Lambda*).

$$sl(t+1) = \begin{cases} sl(t) - e^{-t/(t+1)} * \varphi * sl(t) & \text{if } r \le \lambda \\ sl(t) + e^{-t/(t+1)} * \varphi * sl(t) & \text{if } r > \lambda \end{cases} \tag{2}$$

where t is the iteration number and φ(*Phi*) is a user defined parameter. WSA decreases the step length as the search progresses with regard to iteration number, although it increases the step length for some randomly generated iterations in order to achieve the ability of jumping out of local extremum points. The position update procedure realized by WSA is depicted in Fig. 4 and this procedure in conjunction with procedure given in Fig. 3 specifies the moving patterns of the artificial agents in the search space during their explorations. On the other hand, it must be noted that the position update procedure may result a feasibility problem by inducing some artificial agents to exceed the search limits for some dimensions of the problem. In such a situation, WSA equalize the related dimension to UL if it exceeds UL and LL if it exceeds LL.

2.2. Constraint handling methods adapted to WSA

As indicated before, the current part of this paper deals with the mechanical design problems as the constrained optimization problems having a general non-linear programming model in the following form.

$$\text{Min} f(\vec{x}) \tag{3}$$

s.t.

$$g_j(\vec{x}) \le 0, \quad j = 1, \ldots, J, \tag{4}$$

$$h_k(\vec{x}) = 0, \quad k = 1, \ldots, K, \tag{5}$$

$$x_i^l \le x_i \le x_i^u, \quad i = 1, \ldots, D. \tag{6}$$

where $\vec{x}$ is an n dimensional vector and ith dimension (variable) can have values in the range of $[x_i^l, x_i^u]$. Additionally, the equality constraints could be transformed into equality constraints thanks to a parameter of δ, which has a small positive value, in such a way that $g_{k+J}(\vec{x}) \equiv |h_k(\vec{x})| - \delta \le 0$ [44]. Thus, the non-linear constrained optimization problem (3)–(6) will have $m = J + K$ number of inequality constraints. A great deal of approaches were developed for solving these type of problems and the crucial component of these approaches is the used constraint handling method for guiding the search towards the satisfactory regions of the solution space. Additionally, constraint handling methods reduce the non-linear constrained optimization problem (3)–(6) into an unconstrained optimization problem. For more detailed information about the literature on constraint handling methods the reader can refer to [45,6,46,44]. As the WSA algorithm originally developed for unconstrained continuous global optimization problems, it also requires a constraint handling method while dealing with the constrained global optimization problems. For this purpose, we adapted six formerly proposed handling methods, Static Penalty Function [45–47], Adaptive Penalty Function [48], Dynamic Penalty Function [49], Inverse Tangent Constraint Handling (ITCH) approach [1,24], Deb's Method [50], and Morales and Quezada's Method [51], to WSA algorithm. As mentioned before, we will have the ability to analyze the effects of the handling methods on WSA and we would produce concluding remarks about the robustness of the WSA algorithm by this way.

2.2.1. Static penalty function

Static penalty function refers to an approach in which the iteration number has no effect over the determination of the penalty factor as the search progresses. Homaifar et al.'s method requires several levels of violation to be set, and penalty factors are chosen with regard to level of violation, the higher level of violation results the higher penalty factor to be selected, for the infeasible solutions [47]. This method evaluates each solution by using the following formula [45].

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m}\left(R_{k,i} \times \max[0, g_i(\vec{x})]^2\right) \tag{7}$$

where $R_{k,i}$ are the specified penalty factors, m is the total number of constraints, $f(\vec{x})$ is the unpenalized objective function and $k = 1, \ldots, l$, where l is the user defined levels of violation. We adapted this method as given by Eq. (8) to WSA, since this methods requires $m(2l+1)$ parameters [46] for m number of constraints, by setting a single penalty factor R instead of $R_{k,i}$ penalty factors.

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m}\left(R \times \max[0, g_i(\vec{x})]^2\right) \tag{8}$$

2.2.2. Adaptive penalty function

Hadj-Alouane and Bean's adaptive penalty method takes feedback from the search process and evaluates each solution by using the following formula [48].

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \lambda(t)\sum_{i=1}^{m}\max[0, g_i(\vec{x})]^2 \tag{9}$$

$\lambda(t)$ is required to be update at each iteration via the following partial rule.

$$\lambda(t+1) = \begin{cases} (1/\beta_1)\lambda(t) & \text{if case \# 1} \\ \beta_2\lambda(t) & \text{if case \#2} \\ \lambda(t) & \text{otherwise} \end{cases} \tag{10}$$

where case #1 refers to the situation where the best individual in the last k iterations was always feasible, and case #2 refers to the situation where the best individual in the last k iterations was never feasible. Note that: $\beta_1, \beta_2 > 1$, $\beta_1 > \beta_2$ and $\beta_1 \neq \beta_2$ (to avoid cycling).

2.2.3. Dynamic penalty function

As distinct from the static penalty function, a dynamic penalty function refers to a method in which the calculation of the penalty factor is affected by the iteration number directly. In general, penalty function must be defined so as to increase the penalty factor over time with the aim of guiding the search towards the feasible region. The dynamic penalty function adapted to WSA is the one that proposed by Joines and Houck and evaluates the solutions at iteration t by the following formula [49].

$$\text{fitness}(\vec{x}) = f(\vec{x}) + (C \times t)^{\alpha} \times \text{SVC}(\beta, \vec{x}) \tag{11}$$

where C, α and β are used defined parameters (the authors set $C = 0.5$, $\alpha = 1$ or 2 and $\beta = 1$ or 2) and $\text{SVC}(\beta, \vec{x})$ is defined in the following form.

$$\text{SVC}(\beta, \vec{x}) = \sum_{j=1}^{J} D_j^{\beta}(\vec{x}) + \sum_{k=1}^{K} D_k(\vec{x}) \tag{12}$$

and

$$D_j(\vec{x}) = \begin{cases} 0 & \text{if } g_j(\vec{x}) \le 0 \\ |g_j(\vec{x})| & \text{otherwise} \end{cases} \quad 1 \le j \le J \tag{13}$$

$$D_k(\vec{x}) = \begin{cases} 0, \text{ if} & -\epsilon \le h_k(\vec{x}) \le \epsilon \\ |h_k(\vec{x})| & \text{otherwise} \end{cases} \quad 1 \le k \le K \tag{14}$$

2.2.4. Inverse tangent constraint handling (ITCH) approach

The ITCH approach for handling the constraints while dealing with the constrained optimization problems was developed by Kim et al. and it is not required to be set any problem-dependent parameters such as penalty factors and Lagrange multipliers [24]. This method evaluates each solution via the following equation.

$$\text{fitness}(\vec{x}) = \begin{cases} \hat{g}(\vec{x}) = g_{\max}(\vec{x}) & \text{if } g_{\max}(\vec{x}) > 0 \\ \hat{f}(\vec{x}) = a\ \tan[f(\vec{x})] - \pi/2 & \text{otherwise} \end{cases} \tag{15}$$

Table 1
Parameter values of WSA.

Parameter	Definition	Value
Maxiter	Maximum number of iterations	5000
AA	Number of artificial agents at each iteration	100
τ	User defined parameter	0.3
λ	User defined parameter	0.75
φ	User defined parameter	0.001
sl_0	Initial step length	0.005

where $g_{\max}(\vec{x}) = \max_{g_i(\vec{x})}[g_1(\vec{x}), g_2(\vec{x}), g_3(\vec{x}), \ldots, g_m(\vec{x})]$ and $a\tan[\cdot]$ denotes the inverse tangent. The reader must note that $\hat{g}(\vec{x}) < 0$ for any $\vec{x}$, and thus $\hat{f}(\vec{x}) < \hat{g}(\vec{x})$ is guaranteed.

2.2.5. Deb's method

Deb's penalty function divides solutions into two group, feasible and infeasible [50]. The objective function value of the feasible solutions are set as their fitness values, while infeasible solutions are evaluated on the basis of constraint violations to guide them towards feasible region. The reader must also note again that this method does not require any penalty factor to be set, and this method evaluates every solution via the following equation.

$$\text{fitness}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if } g_i(\vec{x}) \le 0 \quad \forall i = 1, 2, \ldots, m \\ f_{\text{worst}} + \sum_{i=1}^{m} \max[0, g_i(\vec{x})] & \text{otherwise} \end{cases} \tag{16}$$

where f_{worst} is the objective function value of the worst feasible solution in the population, and if there are no feasible solutions in the population, then f_{worst} is set to zero.

2.2.6. Morales and Quezada's method

Morales and Quezada's method is another type of static penalty function, since the iteration number has no effect over the penalty factor determined by this method as the search progresses [51]. This method evaluates each solution by using the following equation.

$$\text{fitness}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if the solution is feasible} \\ K + \sum_{i=1}^{s} \dfrac{K}{m} & \text{otherwise} \end{cases} \tag{17}$$

where s is the number of constraints satisfied, m is the total number of constraints and K is a large constant (it was set to 1×10^9 by the authors). Additionally, it should be emphasized that, fitness values of the infeasible solutions are not computed and the solutions that violate the same number of constraints receive the same penalty value without considering their distances to feasible region.

3. Computational study

The WSA algorithm with the aforementioned constraint handling methods was coded in MATLAB 8.4, and then the results were obtained by running the coded WSA algorithm on a personal computer having 3.2 GHz processor and 4 GB RAM. The parameter values used by WSA algorithm while solving the mechanical design problems are given in Table 1.

The values for the parameters of *Maxiter* and *AA* were respectively chosen as 5000 and 100 for fairly compare the WSA algorithm to literature about solving the mechanical design problems. By this way the algorithm evaluates 500,000 number of solutions for its one run. The other values for the parameters of λ, φ and were taken from [2] and the values for the parameters of τ and sl_0 were chosen experimentally. Finally, the results were obtained after running the WSA algorithm 20 times for each problem and the following sub-sections present the mechanical design problems with their obtained solutions in depth.

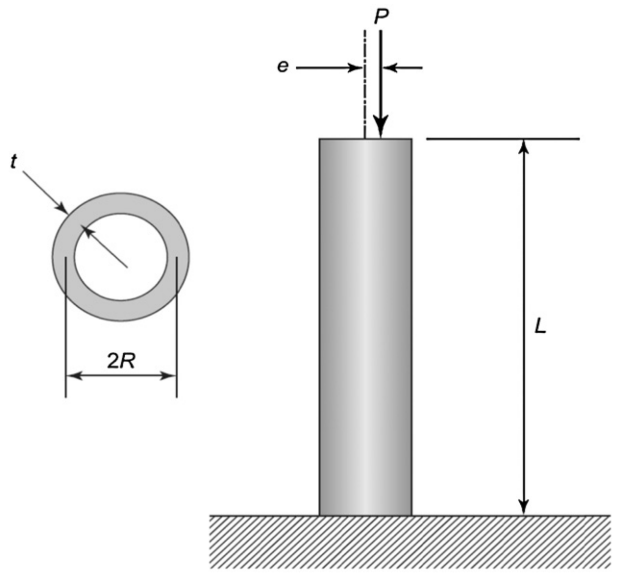

Fig. 5. A vertical column with an eccentric load [52].

3.1. Behavioural analysis of WSA via constraint handling methods

This subsection is about to analyze the effects of the used constraint handling methods over the WSA algorithm during its search. Within this purpose, a two dimensional design problem, column design for minimum mass, was taken from [52]. The objective of this problem is to design a minimum-mass tubular column that is subjected to an eccentric load as represented in Fig. 5 [52]. The design variables of this problem are specified as below.

$$x = (x_1, x_2)^T := (R, t)^T \tag{18}$$

where R is the mean radius of the tube and t is the wall thickness. The constrained non-linear mathematical programming formulation of this problem (19)–(24) was given by Arora in literature [52].

$$\min \ f_5(x) = 2\pi(5)(7850)x_1x_2 \tag{19}$$

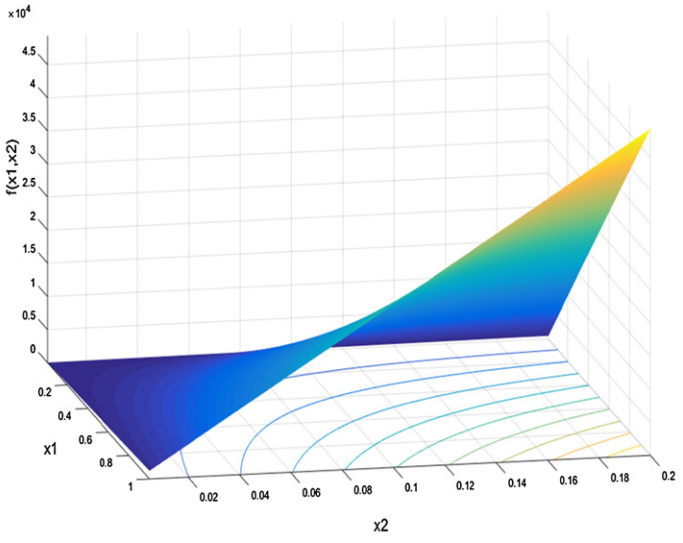

Fig. 6. 3-D plot for the objective function for the column design for minimum mass problem.

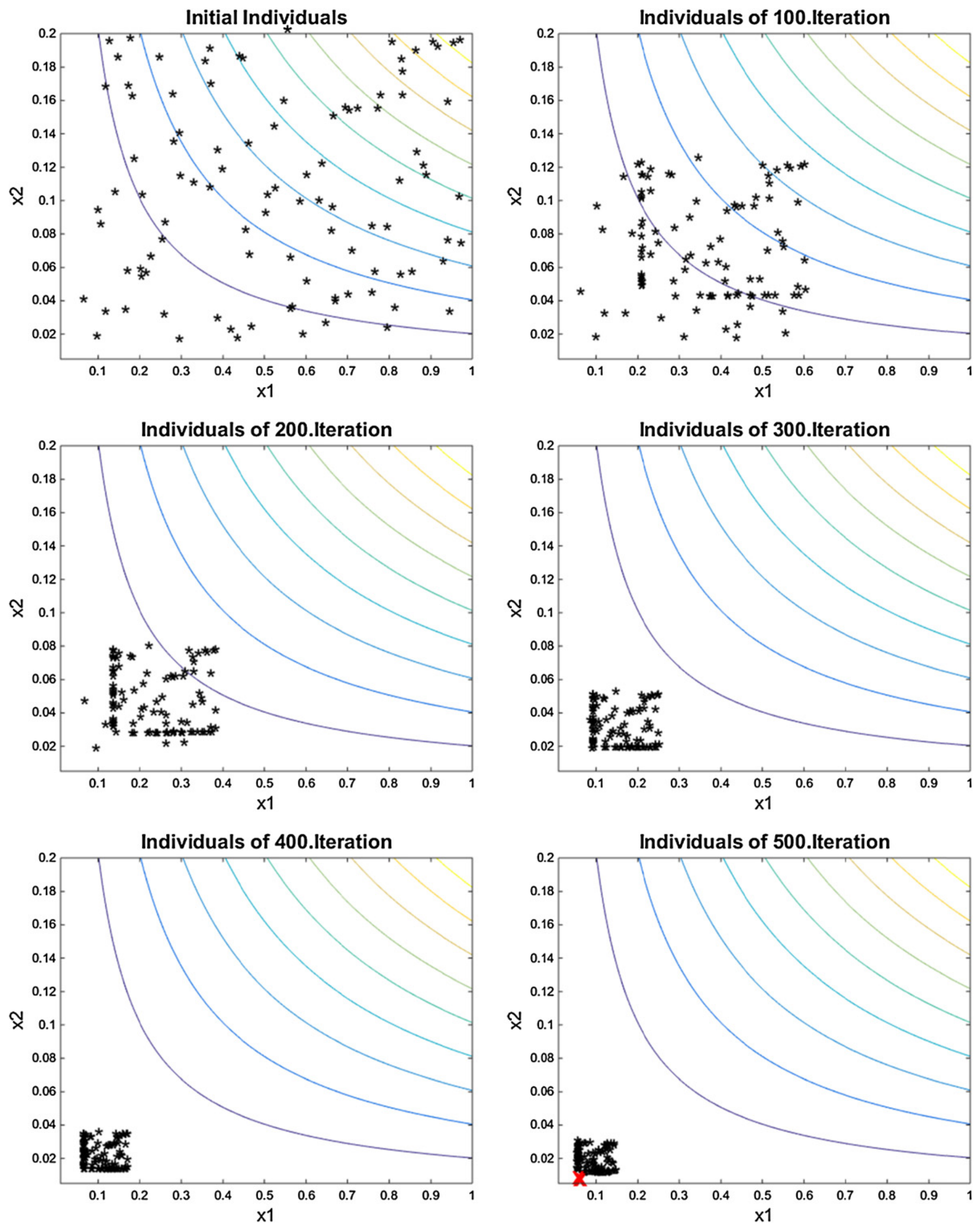

Fig. 7. Behaviour of WSA with static penalty function.

s.t.

$$g_{51}(x) = \frac{P}{2\pi x_1 x_2 \sigma_a}\left[1 + \frac{2(0.02)(x_1+0.5x_2)}{x_1}\sec\left(\frac{\sqrt{2}L}{x_1}\sqrt{\frac{P}{E(2\pi x_1 x_2)}}\right)\right] - 1 \le 0 \tag{20}$$

$$g_{52}(x) = 1 - \frac{\pi^2 E(\pi x_1^3 x_2)}{4L^2 P} \le 0 \tag{21}$$

$$g_{53}(x) = \frac{0.002x_1}{\Delta}\left[\sec\left(L\sqrt{\frac{P}{E(\pi x_1^3 x_2)}}\right) - 1\right] - 1 \le 0 \tag{22}$$

$$g_{54}(x) = \frac{x_1}{50x_2} - 1 \le 0 \tag{23}$$

$$D := \left\{\boldsymbol{x} \in \mathbb{R}^2 : (0.01, 0.005)^T \le x \le (1, 0.2)^T\right\} \tag{24}$$

where$P = 50{,}000$, $E = 210E9$, $L = 5.0$, $\sigma_a = 250E6$, $\Delta = 0.25$.

WSA was initialized for a number of 100 artificial agents with each aforementioned constraint handling method separately and ran for a maximum number of 500 iterations for the design of a

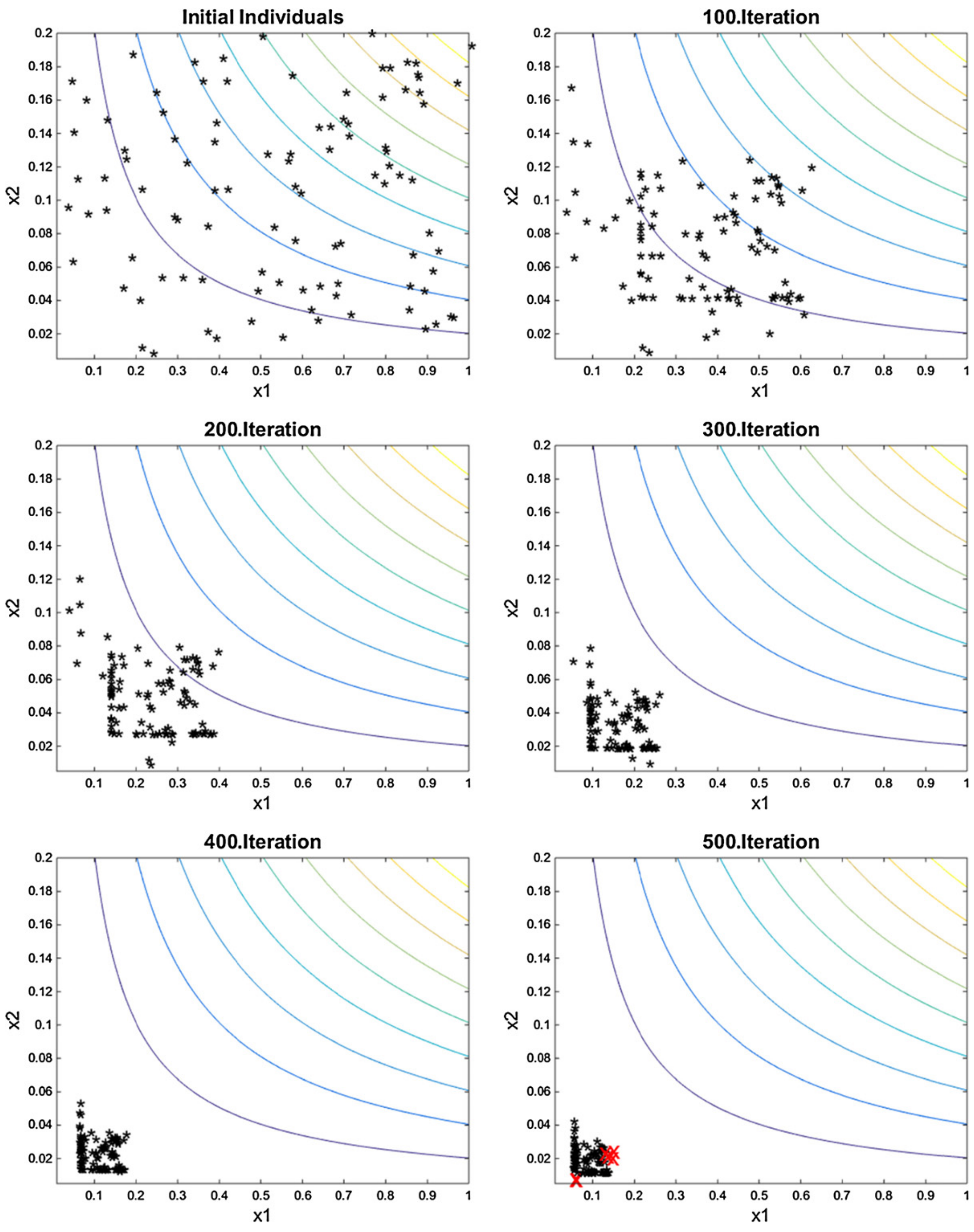

Fig. 8. Behaviour of WSA with adaptive penalty function.

column for minimum mass problem (see Fig. 6 for 3D representation). After that, the initial artificial agents and the artificial agents after every 100 iteration are observed in order to visualize the effects of the constraint handling methods over the WSA algorithm during its execution. Through this visualization, we will achieve the ability to analyze the effects of the constraint handling methods over the WSA algorithm and to analyze how effect these constraint handling methods the convergence behaviour of WSA towards global optimum. Additionally, we would be produced some concluding remarks about the capability of the WSA algorithm in providing feasible solutions with each one of the aforementioned constraint handling methods.

The contour plot of objective function of the design of a column for minimum mass problem was firstly constituted, then the agents of the related iterations were scattered on different contour plots of the objective function of the problem as can be seen through Figs. 7–12. In these figures feasible and infeasible solutions are visualized via the symbols of * and x, respectively.

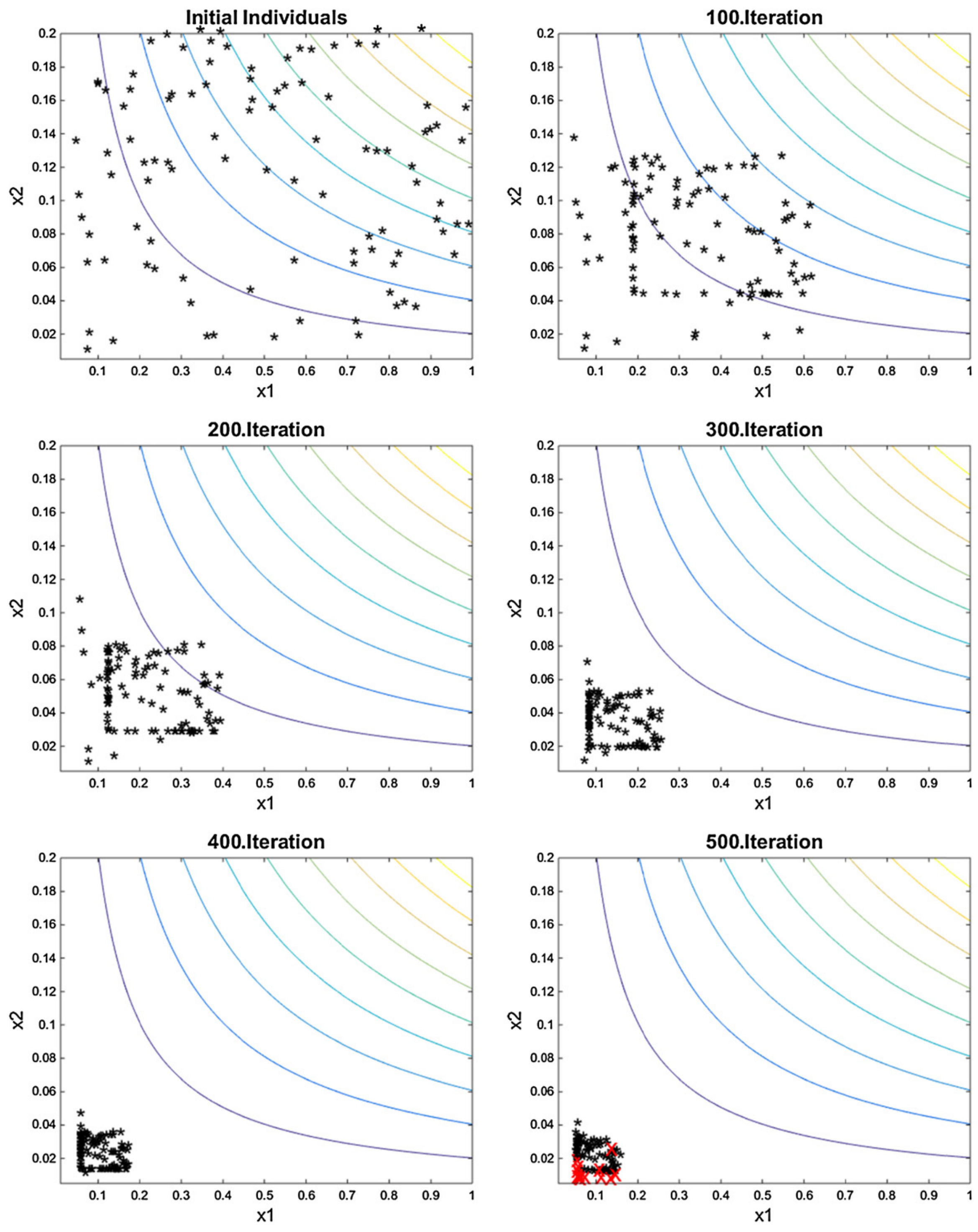

Fig. 9. Behaviour of WSA with dynamic penalty function.

By this way, we have the possibility to visualize the capability of WSA in converging towards the global optima and producing feasible solutions with the related constraint handling method. And, we will establish whether the realized constraint handling method is significantly effective over the performance of WSA algorithm in solving the constrained mechanical design optimization problems or not.

From the observation of the Figs. 7–12, it is clearly seen that WSA has a satisfactory level of convergence and we can conclude that WSA has an intensive search capability around the promising regions of the search space independently from the used constraint handling methods. Additionally, it should be noticed that, the capability of producing feasible solutions of WSA algorithm with the used constraint handling methods is satisfactory. Static, adaptive and dynamic penalty functions and Morales and Quezada's method are capable of producing feasible solutions in high proportions within a population during the search of WSA algorithm, while ITCH approach and Deb's method cause the WSA algorithm to produce feasible solutions in a decreasing proportion during the search process, as illustrated in Figs. 7–12.

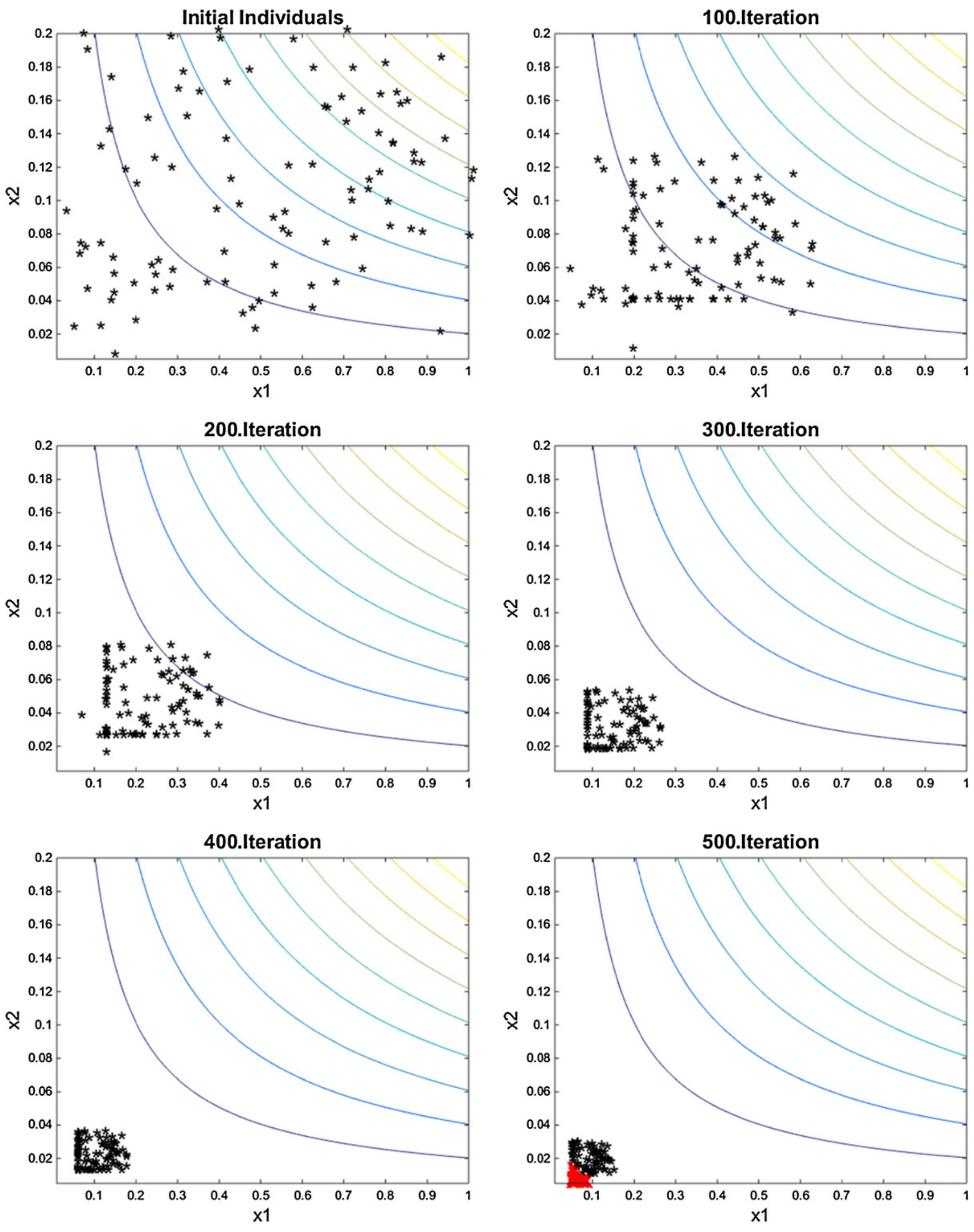

Fig. 10. Behaviour of WSA with ITCH approach.

Furthermore, Fig. 13 illustrates the variation of the artificial agents of WSA with the related constraint handling method having the best fitness as the search progresses for 5000 functions evaluations. Here we can produce some concluding remarks on WSA and the constraint handling methods: WSA converges to a promising area within the search space with the used constraint handling methods, except adaptive penalty function, as illustrated in Fig. 13. Additionally, WSA has the ability to reach a fitness value of 66.1921 better than the fitness value of 66.1922 reported by Arora for the design of a column for minimum mass problem [52]. The reader can refer to http://web.deu.edu.tr/baykasoglu/wsa_constrained.rar for the MATLAB code of this example problem.

3.2. Test problems

As indicated previously, this section is about the performance evaluation of WSA algorithm on the well-known non-linear constrained mechanical design problems, design of a tension/compression coil spring, design of a pressure vessel, design of a welded beam and design of a speed reducer. Additionally, an

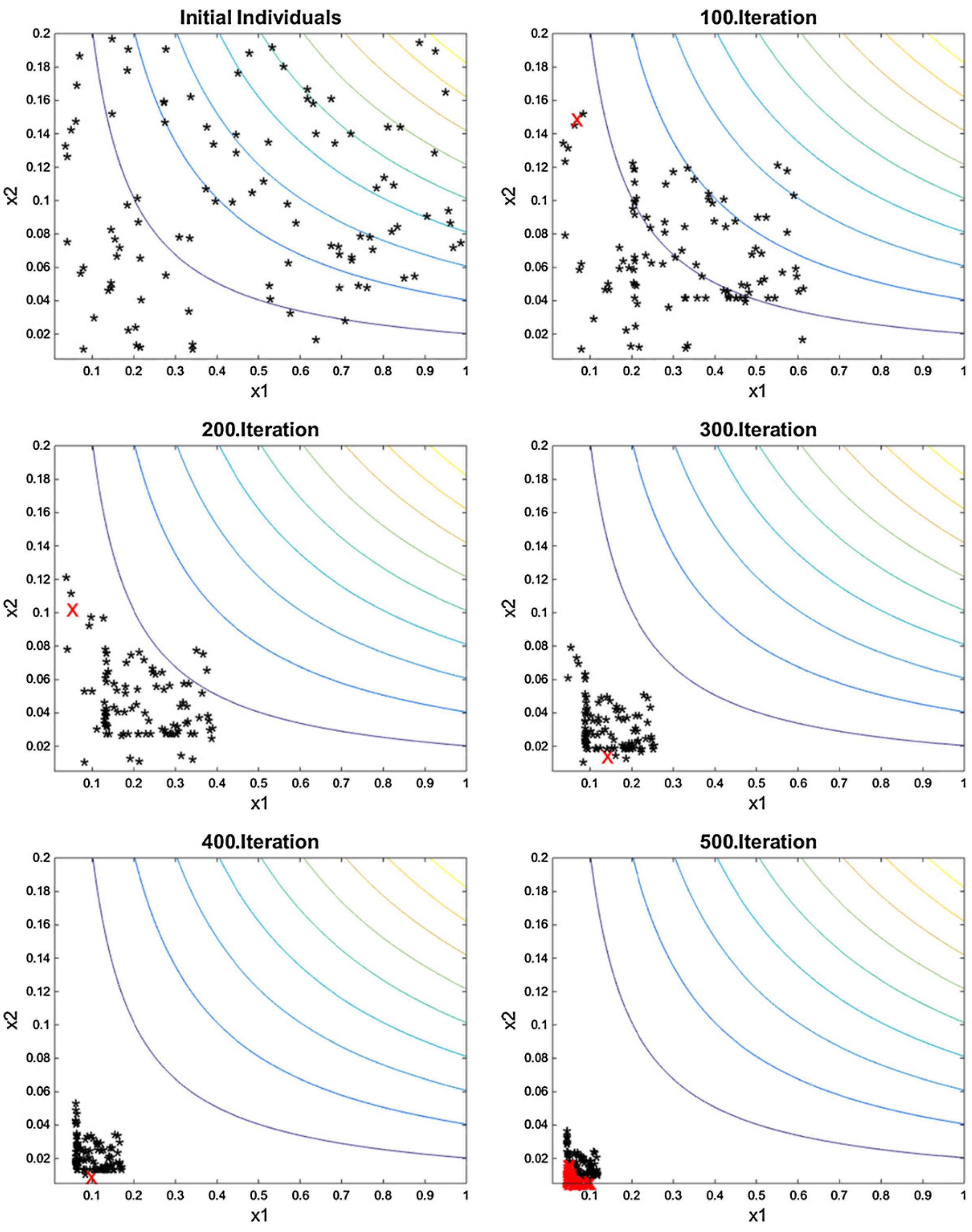

Fig. 11. Behaviour of WSA with Deb's Method.

extra constrained test problem with 13 decision variables is also tackled via WSA in order further demonstrate the effectiveness of WSA. The explanations of these problems and the obtained results via WSA algorithm are given in the following sub-section in details.

3.2.1. *Design of a tension/compression coil spring*

This first test problem, which was introduced by Belegundu for the first time in literature [3], is about the optimal design of the tension/compression coil spring and an example for compression coil spring is represented in Fig. 14 [1,4]. The three design variables (one integer and two continuous) of this problem can be specified as below.

$$x = (x_1, x_2, x_3)^T := (N, D, d)^T \tag{25}$$

where N is the number of spring coils, D is the winding diameter, and d is the wire diameter. Minimization the weight of the spring subject to minimum deflection, shear stress, surge frequency and limits on the outside diameter is objective of this design problem.

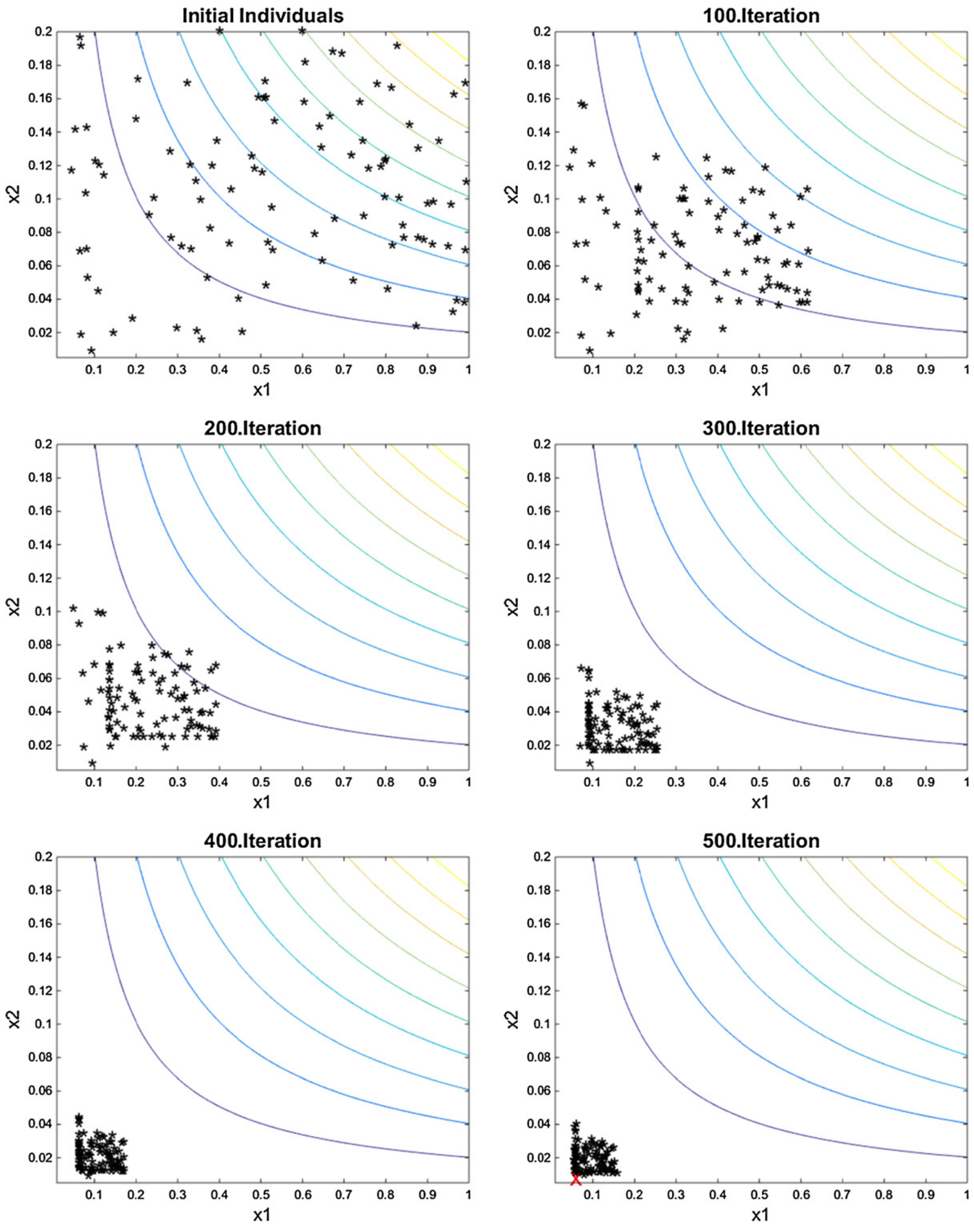

Fig. 12. Behaviour of WSA with Morales and Quezada's method.

The constrained non-linear mathematical programming formulation of this problem (26)–(31) was given by Arora and Kim et al. in literature [4,24].

$$\min f_1(x) = (x_1+2)x_2x_3^2 \tag{26}$$

s.t.

$$g_{11}(x) := 1 - \frac{x_1x_2^3}{71785x_3^4} \leq 0 \tag{27}$$

$$g_{12}(x) := \frac{4x_2^2 - x_2x_3}{12,566(x_2x_3^3 - x_3^4)} + \frac{1}{5108x_3^2} - 1 \leq 0 \tag{28}$$

$$g_{13}(x) := 1 - \frac{140.45x_3}{x_1x_2^2} \leq 0 \tag{29}$$

$$g_{14}(x) := \frac{x_2+x_3}{1.5} - 1 \leq 0 \tag{30}$$

$$D := \{x \in \mathbb{R}^3 : (2.0, 0.25, 0.05)^T \leq \boldsymbol{x} \leq (15.0, 1.3, 2.0)^T\} \tag{31}$$

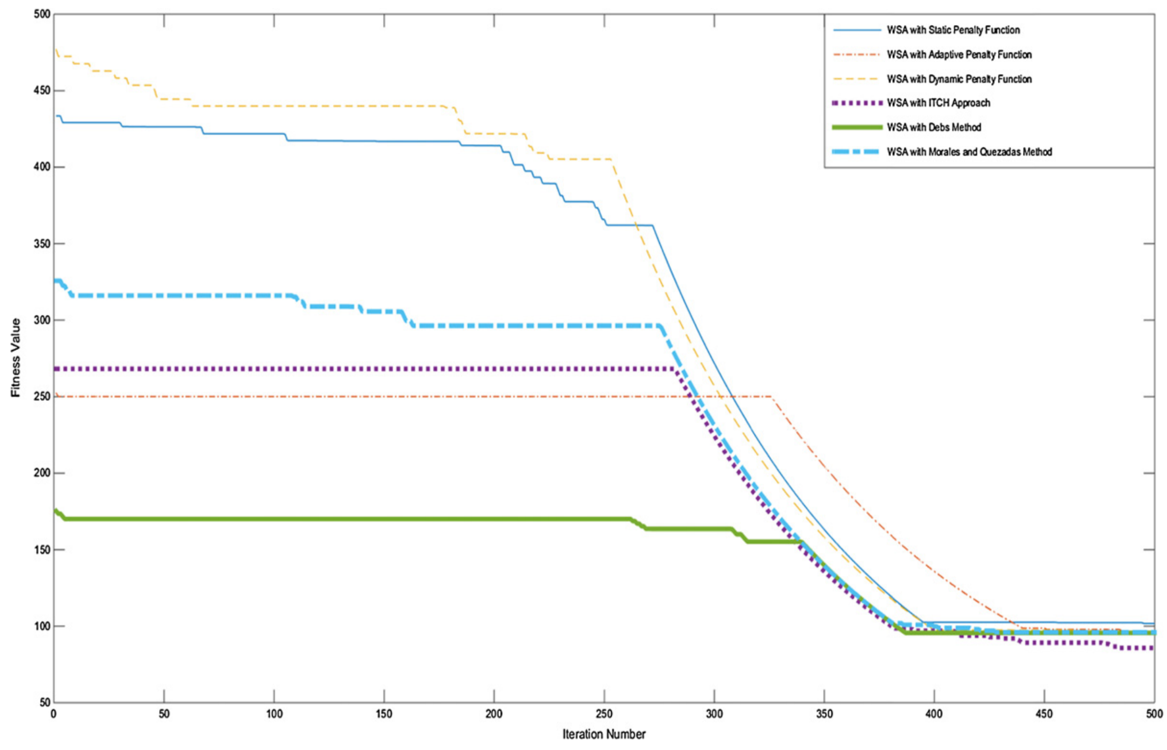

Fig. 13. Moving patterns of the best artificial agents of WSA with the constraint handling methods.

Table 2
The best solutions obtained by WSA for the tension/compression coil spring design problem.

Constraint handling approach	x_1 (best)	x_2 (best)	x_3 (best)	f(best)	f(worst)	f(avg.)	f std.dev.
Static Penalty Function	11.29334620	0.35664398	0.05168599	0.01266528	0.01266975	0.01266728	1.23613E−06
Adaptive Penalty Function	11.28895058	0.35671893	0.05168912	0.01266529	0.01267243	0.01266774	1.92527E−06
Dynamic Penalty Function	11.28408284	0.35680275	0.05169270	0.01266538	0.01267042	0.01266716	1.49193E−06
ITCH Approach	11.29428138	0.35662726	0.05168529	0.01266524	0.01266582	**0.01266548**	2.72618E−07
Deb's Method	11.29291654	0.35665047	0.05168626	**0.01266523**	0.01267781	0.01267061	4.06282E−06
Morales and Quezada's Method	11.28805333	0.35673373	0.05168972	0.01266525	0.01266840	0.01266667	9.43034E−07

The bold values are the best solutions found by the algorithm.

Table 3
The best solutions reported for the tension/compression coil spring design problem.

Reference	x_1 (best)	x_2 (best)	x_3 (best)	f(best)	f(worst)	f(avg.)	f std. dev.
Baykasoglu [1]	11.2835059488	0.3568108568	0.0516929296	0.0126652296	0.0140793687	0.0128750789	0.0002966889
Belegundu [3]	14.25	0.31590	0.05	0.0128334375	–	–	–
Arora. [4]	9.185400	0.399180	0.053396	0.0127302737	–	–	–
Coello [5]	11.632201	0.351661	0.051480	0.0127047834	0.01282208	0.01276920	–
Coello [6]	11.632201	0.351661	0.051480	0.0127047834	–	–	–
Mezura-Montes et al. [9]	–	–	–	0.012688	0.017037	0.013014	0.000801
Coello and Becerra [10]	14.031795	0.317395	0.050000	0.0127210	–	–	–
Mezura and Coello [11]	9.807729	0.384942	0.052836	0.012689	–	0.013165	0.00039
Ray and Liew [13]	10.6484422590	0.3681586950	0.0521602170	0.0126692493	0.01671727	0.01292267	0.000592
Ray and Saini [14]	13.979915	0.321532	0.050417	0.013060	–	–	–
Hu et al. [15]	11.60865920	0.35138394	0.051466369	0.0126661409	–	0.012718975	0.0000644
He et al. [16]	11.28712599	0.35674999	0.05169040	0.0126652812	–	–	–
Parsopoulos and Vrahatis [17]	–	–	–	0.013120	0.0503651	0.0229478	0.00720571
Aguirre et al. [18]	11.28893209	0.35671831	0.05168908	0.012665	–	0.012665	0
He and Wang [19]	11.244543	0.357644	0.051728	0.0126747	0.012924	0.012730	0.000051985
He and Wang [20]	11.265083	0.357126	0.051706	0.0126652	0.0127191	0.0127072	0.000015824
Cagnina et al. [22]	11.438675	0.354190	0.051583	0.012665	–	0.0131	0.00041
Maruta et al. [23]	11.2896874780	0.3567054307	0.0516885495	0.0126652329	0.01461170	0.01275760	0.000269863
Kim et al. [24]	11.2889651961	0.3567177493	0.0516890615	0.0126652328	0.01266523	0.01266523	1.05055E−14
Akay and Karaboga [26]	–	–	–	0.012665	–	0.012709	0.012813
Brajevic and Tuba [27]	11.285988	0.356769	0.051691	0.012665	–	0.012683	0.00000331
Mahdavi et al. [29]	12.0764321	0.34987116	0.05115438	0.0126706	–	–	–
Gandomi et al. [32]	11.2885	0.35673	0.05169	0.01266522	0.0168954	0.01350052	0.001420272
Baykasoglu and Ozsoydan [34]	11.3195613646	0.3561976945	0.0516674837	0.0126653049	0.0000128058	0.0126770446	0.0127116883
WSA	11.29291654	0.35665047	0.05168626	0.01266523	0.01267781	0.01267061	4.06282E−06

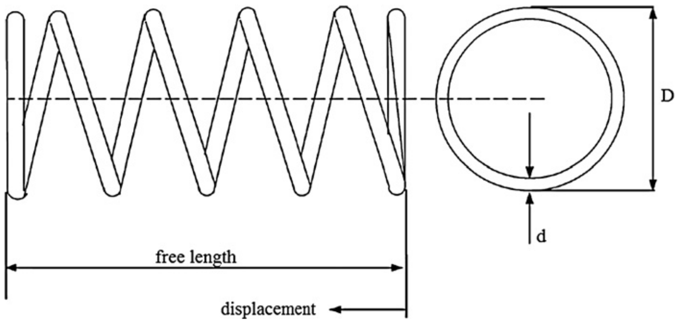

Fig. 14. Tension/compression coil spring [1,4].

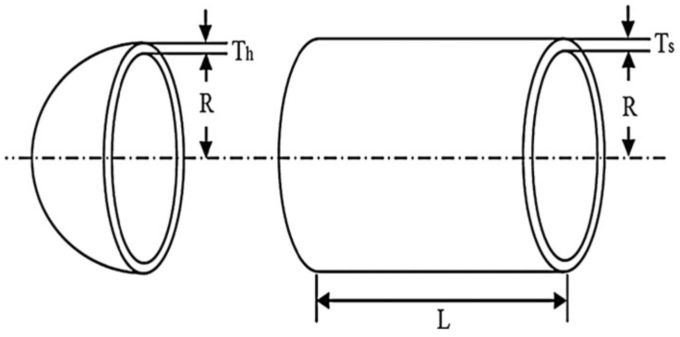

Fig. 15. Pressure vessel [1,9].

The design of a tension/compression coil spring problem was solved via the WSA algorithm by adapting the aforementioned 6 constraint handling methods and the obtained results are reported in Table 2 separately. Additionally, Table 3 compares the best result obtained by WSA algorithm with the results provided by various researchers in literature [34].

From the observation of Table 2, it is obviously seen that WSA algorithm produced the best result for the design of a tension/compression coil spring problem by using Deb's Method as the constraint handling approach. However, WSA has the best performance over the design of a tension/compression coil spring problem via ITCH approach, if a comparison is done in terms of the average objective function value of the 20 runs. If the comparison is extended via the results available in literature, it is clear that WSA algorithm has a satisfactory performance in solving the design of a tension/compression coil spring problem with various constraint handling methods as it can be seen from Table 3. Additionally, the best result produced by WSA algorithm for the design of a tension/compression coil spring problem is differentiated from the best-known results in literature after the seventh digit in the decimal portion.

3.2.2. *Design of a pressure vessel*

Design of a cylindrical vessel capped at both ends by hemispherical heads as represented in Fig. 15 [1,9] is the second test problem and was introduced by Kannan and Kramer for the first time in literature [53]. The objective is to minimize the total cost including the material, forming and welding costs. This problem has four design variables as specified as below.

$$x = (x_1, x_2, x_3, x_4)^T := (T_s, T_h, R, L)^T \quad (32)$$

where T_s is the thickness of the shell, T_h is the thickness of the head, R is the inner radius and L is the length of the cylindrical section of the vessel (not including the head). The first two design variables T_s and T_h are integer multiples of 0.0625, which are the available thicknesses of the rolled steel plates; other two design variables are continuous. The constrained non-linear mathematical programming formulation of this problem (33)–(38) was given by Arora and Kim et al. in literature [4,24].

$$\min\ f_2(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \quad (33)$$

s.t.

$$g_{21}(x) := 0.0193x_3 - x_1 \le 0 \quad (34)$$

$$g_{22}(x) := 0.00954x_3 - x_2 \le 0 \quad (35)$$

$$g_{23}(x) := 1,296,000 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \le 0 \quad (36)$$

$$g_{24}(x) := x_4 - 240 \le 0 \quad (37)$$

$$D := \left\{ \boldsymbol{x} \in \mathbb{R}^4 : (0, 0, 10, 10)^T \le \boldsymbol{x} \le (99, 99, 200, 200)^T \right\} \quad (38)$$

The second test problem, design of a tension/compression coil spring, was also solved via the WSA algorithm by adapting the aforementioned 6 constraint handling methods and the obtained results are reported in Table 4 separately. Furthermore, Table 5 compares the best result obtained by WSA algorithm with the results provided by various researchers in literature [34].

As pointed out in Table 4, WSA has the best performance in terms of the produced best results over the design of a pressure vessel problem when adapting the dynamic penalty function as the constraint handling approach. The average objective function value for 20 replications of WSA algorithm over the design of a pressure vessel problem, reported in Table 4, also encourages the superior performance of WSA algorithm via the dynamic penalty function. Furthermore, the reported results in Table 5 indisputably indicate the superior performance of WSA algorithm over the available results for the design of a pressure vessel problem in literature. Thus, we can conclude that WSA surpasses the existing literature on tackling the design of a pressure vessel problem, since the WSA algorithm has the ability to produce better results than the previous literature with all of the aforementioned constraint handling methods.

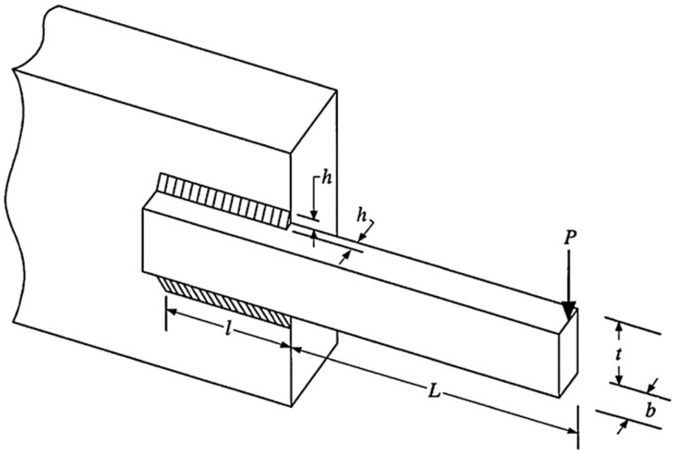

Fig. 16. Welded beam [1,12,17].

3.2.3. *Design of a welded beam*

The third test problem, design of a welded beam, was introduced by Rao [54] with the aim of minimizing cost design of the structural welded beam as it shown in Fig. 16 [1,12,17]. The design purpose is to minimize the structural cost of the beam and the main cost factors (set-up labour cost, welding labour cost and material cost) of such a welded beam under the constraints of weld stress, buckling

Table 4
The best solutions obtained by WSA for the pressure vessel design problem.

Constraint handling approach	x_1 (best)	x_2 (best)	x_3 (best)	x_4 (best)	f(best)	f(worst)	f(avg.)	f std.dev.
Static Penalty Function	0.81895637	0.40369388	42.19594620	175.92544248	5996.92136223	6036.31820729	6024.73151994	9.23814
Adaptive Penalty Function	0.79795960	0.39481590	41.27763424	187.36468155	5936.39305975	5975.60359654	5960.10119926	11.61222
Dynamic Penalty Function	0.78654289	0.39348835	40.75268075	194.78059812	**5929.62188231**	5984.75680753	**5958.41025196**	15.07840
ITCH Approach	0.80589483	0.42652724	41.75341292	180.99106428	6022.85143737	6036.35023309	6029.37715517	4.90533
Deb's Method	0.80794014	0.42691120	41.86181400	179.61407263	6024.60526510	6037.20587604	6030.21145812	2.82231
Morales and Quezada's Method	0.78382085	0.39099967	40.50915776	197.87239308	5929.98060178	6018.50963606	5983.53151376	27.87393

The bold values are the best solutions found by the algorithm.

Table 5
The best solutions reported for the pressure vessel design problem.

Reference	x_1 (best)	x_2 (best)	x_3 (best)	x_4 (best)	f(best)	f(worst)	f(average)	f std. dev.
Baykasoglu [1]	0.8125	0.4375	42.09754674	176.64838674	6059.83905683	6823.60245024	6149.72760669	210.77
Coello [6]	0.8125	0.4375	40.3239	200.0000	6288.7445	–	–	–
Coello and Mezura-Montes [7]	0.8125	0.4375	40.097398	176.654047	6059.946341	–	–	–
Mezura-Montes et al. [9]	0.8125	0.4375	42.098370	176.637146	6059.714355	6846.628418	6355.343115	256.04
Mezura and Coello [11]	0.8125	0.4375	42.098446	176.636596	6059.7143	–	6379.938037	210
Akhtar et al. [12]	0.8125	0.4375	41.9768	182.2845	6171.0	–	–	–
He et al. [16]	0.8125	0.4375	42.0984456	176.63659584	6059.714355	–	–	–
Parsopoulos and Vrahatis [17]	–	–	–	–	6154.7	9387.77	8016.37	745.869
Aguirre et al. [18]	0.8125	0.4375	42.098446	176.636596	6059.714335	–	6071.013366	15.101157
He and Wang [19]	0.8125	0.4375	42.091266	176.746500	6061.0777	6363.8041	6147.1332	86.4545
He and Wang [20]	0.8125	0.4375	42.0984	176.6366	6059.7143	6288.6770	6099.9323	86.2022
Cagnina et al. [22]	0.812500	0.437500	42.098445	176.636595	6.059.714335	–	6092.0498	12.1725
Maruta et al. [23]	0.8125	0.4375	42.0984456	176.63659584	6059.714355	7332.841508	6358.156992	372.71
Kim et al. [24]	0.8125	0.4375	42.0984456	176.63659584	6059.714355	6060.074434	6059.727721	0.065870503
Chun et al. [25]	0.8125	0.4375	42.0984456	176.63659584	6059.714335	6090.52620169	6060.33057699	4.35745530
Akay and Karaboga [26]	–	–	–	–	6059.714736	–	6245.308144	205.00
Brajevic and Tuba [27]	0.8125	0.4375	42.098446	176.636596	6059.714335	–	6192.116211	204
Gandomi et al. [31]								
Gandomi et al. [32]	0.8125	0.4375	42.0984456	176.6365958	6.059.7143348	6318.95	6179.13	137.223
Baykasoglu and Ozsoydan [34]	0.8125	0.4375	42.09844611	176.63658942	6059.71427196	6090.52614259	6064.33605261	11.28785324
WSA	0.78654289	0.39348835	40.75268075	194.78059812	5929.62188231	5984.75680753	5958.41025196	15.07840

load, beam deflection and beam bending stress. The problem has four continuous design variables as specified below.

$$x = (x_1, x_2, x_3, x_4)^T := (h, l, t, b)^T \tag{39}$$

where h is the weld thickness, l is the weld length, t is the bar thickness and b is the bar breadth. The constrained non-linear mathematical programming formulation of this problem (40)–(48) was given by Akhtar et al. and Parsopoulos and Vrahatis in literature [12,17].

$$\min f_3(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \tag{40}$$

s.t.

$$g_{31}(x) := \tau(x) - \tau_{\max} \le 0 \tag{41}$$

$$g_{32}(x) := \sigma(x) - \sigma_{\max} \le 0 \tag{42}$$

$$g_{33}(x) := x_1 - x_4 \le 0 \tag{43}$$

$$g_{34}(x) := 0.125 - x_1 \le 0 \tag{44}$$

$$g_{35}(x) := \delta(x) - \delta_{\max} \le 0 \tag{45}$$

$$g_{36}(x) := P - P_c(x) \le 0 \tag{46}$$

$$g_{37}(x) := 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0 \tag{47}$$

$$D := \left\{ \boldsymbol{x} \in \mathbb{R}^4 : (0.1, 0.1, 0.1, 0.1)^T \le x \le (2, 10, 10, 2)^T \right\} \tag{48}$$

where

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{49}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2} \tag{50}$$

$$\tau'' = \frac{MR}{J} \tag{51}$$

$$M = P\left(L + \frac{x_2}{2}\right) \tag{52}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{53}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{54}$$

$$\sigma(x) = \frac{6PL}{x_3^2x_4} \tag{55}$$

$$\delta(x) = \frac{4PL^3}{Ex_3^3x_4} \tag{56}$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{57}$$

$P = 6000$ lb, $L = 14$ in, $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi.
$\tau_{\max} = 13{,}600$ psi, $\sigma_{\max} = 30{,}000$ psi, $\delta_{\max} = 0.25$ in.

The obtained results for the design of a welded beam problem via the WSA algorithm by adapting the aforementioned 6 constraint handling methods are reported in Table 6 separately. Additionally, comparison between the WSA algorithm's best solution and the

Table 6
The best solutions obtained by WSA for the welded beam design problem.

Constraint handling approach	x_1 (best)	x_2 (best)	x_3 (best)	x_4 (best)	f(best)	f(worst)	f(avg.)	f std.dev.
Static Penalty Function	0.20572978	3.47048732	9.03662239	0.20572971	1.72485264	4.64890038	2.22958516	8.56669E−01
Adaptive Penalty Function	0.20572963	3.47048995	9.03662398	0.20572964	**1.72485254**	3.82321834	2.12575148	6.52914E−01
Dynamic Penalty Function	0.20572963	3.47048898	9.03662538	0.20572964	1.72485262	4.35499421	2.70232154	1.01846
ITCH Approach	0.20571304	3.47096756	9.03662205	0.20572974	1.72489184	1.72501582	**1.72496037**	3.88255E−05
Deb's Method	0.20572984	3.47048621	9.03662220	0.20572984	1.72485356	11.24319715	4.84533201	3.02947
Morales and Quezada's Method	0.20572972	3.47049003	9.03662255	0.20572973	1.72485310	4.81525409	2.46619718	1.06423

Table 7
The best solutions reported for the welded beam design problem.

Reference	x_1 (best)	x_2 (best)	x_3 (best)	x_4 (best)	f(best)	f(worst)	f(average)	f std. dev.
Baykasoglu [1]	0.205730	3.470488	9.036624	0.205730	1.724852	1.724852	1.724852	0.000000
Belegundu [3]	0.208800	3.420500	8.997500	0.210000	1.748309	1.785835	1.771973	0.011220
Arora [4]	0.205986	3.471328	9.020224	0.206480	1.728226	1.993408	1.792654	0.074713
Coello [5]	0.202369	3.544214	9.048210	0.205723	1.728024	1.782143	1.748831	0.012926
Coello [6]	0.2088	3.4205	8.9975	0.2100	1.74830941	–	–	–
Coello and Montes [8]	0.205986	3.471328	9.020224	0.206480	1.728226	–	–	–
Mezura-Montes et al. [9]	–	–	–	–	1.76558	2.844060	1.968200	0.1554150
Coello and Becerra [10]	0.2057	3.4705	9.0366	0.2057	1.7248523	–	–	–
Mezura and Coello [11]	0.205730	3.470489	9.036624	0.205730	1.724852	–	1.7776	0.088
Hu et al., 2003 [15]	0.20573	3.47049	9.03662	0.20573	1.72485	–	1.72485	0
Parsopoulos and Vrahatis [17]	0.2407	6.4851	8.2399	0.2497	2.4426	–	–	–
Aguirre et al. [18]	0.205730	3.470489	9.036624	0.205730	1.724852	–	1.724881	0.000012
He and Wang [19]	0.202369	3.544214	9.048210	0.205723	1.728024	1.782143	1.748831	0.012926
He and Wang [20]	0.205730	3.470489	9.036624	0.205730	1.724852	1.814295	1.749040	0.040049
Cagnina et al. [22]	0.205729	3.470488	9.036624	0.205729	1.724852	–	2.0574	0.2154
Maruta et al. [23]	0.205730	3.470489	9.036624	0.20573	1.724852	1.813471	1.728471	0.0136371
Kim et al. [24]	0.205730	3.470489	9.036624	0.205730	1.724852	1.724852	1.724852	0.000000
Akayand Karaboga [26]	–	–	–	–	1.724852	–	1.741913	0.031
Brajevic and Tuba [27]	0.205730	3.470489	9.036624	0.205730	1.724852	–	1.724853	0.0000017
Yidiz [28]	0.205730	3.470489	9.036624	0.205730	1.7248	1.75322	1.73418	0.00510
Mahdavi et al. [29]	0.20573	3.47049	9.03662	0.20573	1.7248	–	–	–
Fesanghary et al. [30]	0.20572	3.47060	9.03682	0.20572	1.7248	–	–	–
Gandom. et al. [32]	0.2015	3.562	9.0414	0.2057	1.7312	2.3455793	1.8786560	0.2677989
Gandomi et al. [33]	–	–	–	–	1.7312065	2.3455793	1.8786560	0.2677989
Baykasoglu and Ozsoydan [34]	0.205730	3.470489	9.036624	0.205730	1.724852	1.724852	1.724852	0.000000
WSA	0.20572963	3.47048995	9.03662398	0.20572964	1.72485254	3.82321834	2.12575148	6.52914E−01

results provided by various researchers solved this third test problem with various solution procedures in literature [34] is reported in Table 7.

When Table 6 is observed, it is clearly seen that WSA algorithm produced the best result for the design of a welded beam problem by using adaptive penalty function as the constraint handling approach. However, WSA has the best performance over the design of a welded beam problem via ITCH approach, if a comparison is done in terms of the average objective function value of the 20 runs. If the comparison is extended via the results available in literature, it is clear that WSA algorithm has a satisfactory performance in solving the design of a welded beam problem with various constraint handling methods as it can be seen from Table 7. Additionally, the best result produced by WSA algorithm for the design of a tension/compression coil spring problem is differentiated from the best-known results in literature after the sixth digit in the decimal portion. Here, the reader must also be noticed that, the provided best results for design of a welded beam problem are available for six digit in their decimal portion. Hence, we can claim that WSA algorithm is able to produce the best result in literature for the design of a welded beam problem.

3.2.4. *Design of a speed reducer*

The fourth test problem, design of a speed reducer, was introduced by Siddall [55] and deals with the minimum weight design of a speed reducer subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts as shown in Fig. 17 [1,54]. The design of a speed reducer problem has seven design variables (except the third one all of them are continuous) as specified below.

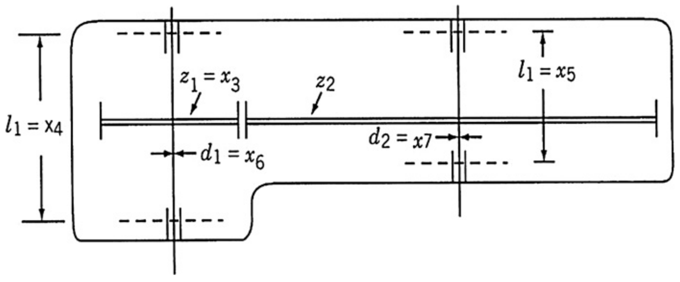

Fig. 17. Speed reducer [1,54].

$$x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)^T := (b, m, z, l_1, l_2, d_1, d_2)^T \tag{58}$$

where b is the face width, m is the module of teeth, z is the number of teeth in the pinion, l_1 is the length of the first shaft between bearings, l_2 is the length of the second shaft between bearings, and d_1 and d_2 are the diameters of the first shaft and second shaft, respectively. The constrained non-linear mathematical programming formulation of this problem (59)–(71) was given by Golinski [56] and Rao [54] in literature.

$$\begin{aligned} \min\ f_4(x) = {} & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ & - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\ & + 0.7854(x_4x_6^2 + x_5x_7^2) \end{aligned} \tag{59}$$

s.t.

$$g_{41}(x) := \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0 \tag{60}$$

$$g_{42}(x) := \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0 \tag{61}$$

$$g_{43}(x) := \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0 \tag{62}$$

$$g_{44}(x) := \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0 \tag{63}$$

$$g_{45}\,(x) := \frac{[(745 x_4 / x_2 x_3)^2 + 16.9 \times 10^6]^{1/2}}{110 x_6^3} - 1 \leq 0 \tag{64}$$

$$a g_{46}(x) := \frac{[(745 x_4 / x_2 x_3)^2 + 157.5 \times 10^6]^{1/2}}{85 x_7^3} - 1 \leq 0 \tag{65}$$

$$g_{47}(x) := \frac{x_2 x_3}{40} - 1 \leq 0 \tag{66}$$

$$g_{48}(x) := \frac{5 x_2}{x_1} - 1 \leq 0 \tag{67}$$

$$g_{49}(x) := \frac{x_1}{12 x_2} - 1 \leq 0 \tag{68}$$

$$g_{410}(x) := \frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0 \tag{69}$$

$$g_{411}(x) := \frac{1.1 x_7 + 1.9}{x_5} - 1 \leq 0 \tag{70}$$

$$\begin{aligned} D := \Big\{ \boldsymbol{x} \in \mathbb{R}^7 : & (2.6, 0.7, 17, 7.3, 7.8, 2.9, 5.0)^T \\ & \leq \boldsymbol{x} \leq (3.6, 0.8, 28, 8.3, 8.3, 3.9, 5.5)^T \Big\} \end{aligned} \tag{71}$$

The aforementioned 6 constraint handling methods were also adapted into WSA for solving the design of a speed reducer problem, and the obtained results are reported in Table 8 separately. Additionally, the best result obtained via WSA algorithm is compared in Table 9 with the results provided by various researchers [34] in literature.

Table 8 obviously indicates that WSA algorithm produced the best result for the design of a speed reducer problem by using ITCH approach as the constraint handling approach. Also WSA performs well via ITCH approach over the design of a speed reducer problem in terms of the average objective function value of the 20 runs. When WSA is via the existing literature on tackling this problem, the satisfactory performance of WSA can be clearly seen from Table 9. Additionally, the best result produced by WSA algorithm for the design of a speed reducer problem is differentiated from the best-known results, available for seven digit in their decimal portion, in literature after the seventh digit in the decimal portion.

3.2.5. *Single-objective test function*

The fifth test problem is also a minimization problem of single objective function and this problem has 13 decisions variables and 9 inequality constraints [57]. This problem is a familiar benchmark used by several researchers [39,58–63] with the aim of testing the performance of their algorithms. The formulation of this problem was given by the equation set (72)–(82) in literature and the problem's global optimum has the value of $f(x^*) = 15$ at the point of $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$.

$$\min f_5(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{72}$$

Table 8
The best solutions obtained by WSA for the speed reducer design problem.

Constraint handling approach	x_1 (best)	x_2 (best)	x_3 (best)	x_4 (best)	x_5 (best)	x_6 (best)	x_7 (best)	f (best)	f (worst)	f (avg.)	f std. dev.
Static Penalty Function	3.50051013	0.7	17	7.3	7.8	3.35039995	5.28681840	2996.68171805	3002.49631808	2999.40319946	1.44597
Adaptive Penalty Function	3.50135581	0.7	17	7.3	7.8	3.35326955	5.28809987	2998.56098876	3006.36129273	3001.97896317	2.09954
Dynamic Penalty Function	3.50051202	0.7	17	7.3	7.8	3.35113112	5.28704818	2997.01498544	3003.89247758	3000.46377102	1.60476
ITCH Approach	3.50000001	0.7	17	7.3	7.8	3.35021489	5.28668322	**2996.34822249**	2996.34824388	**2996.34823295**	9.10931E−06
Deb's Method	3.49999996	0.7	17	7.3	7.8	3.35021496	5.28668338	2996.34832473	2997.49347232	2996.46017157	2.47867E−01
Morales and Quezada's Method	3.50000276	0.7	17	7.3	7.8	3.35022158	5.28668559	2996.35253207	2996.42334312	2996.36352230	1.48567E−02

Table 9
The best solutions reported for the speed reducer design problem.

Reference	x_1 (best)	x_2 (best)	x_3 (best)	x_4 (best)	x_5 (best)	x_6 (best)	x_7 (best)	f (best)	f (worst)	f (avg.)	f std. dev.
Baykasoglu [1]	3.5000	0.7000	17	7.30000	7.80000	3.350215	5.286683	2996.348072	3094.556809	3016.492651	24.48
Mezura-Montes et al. [9]	–	–	–	–	–	–	–	2998.011963	3162.881104	3056.206999	49.40
Mezura and Coello [11]	3.499999	0.699999	17	7.300000	7.800000	3.350215	5.286683	2996.348094	–	2996.348094	0
Akhtar et al. [12]	3.506122	0.700006	17	7.549126	7.859330	3.365576	5.289773	3008.08	3028.28	3012.12	–
Ray and Liew [13]	–	–	–	–	–	–	–	2994.744241	–	3001.758264	4.0
Aguirre et al. [18]	3.5	0.7	17	7.3	7.8	3.3502146	5.2866832	2996.348165	–	2996.408525	0.028671
Tomassetti [21]	3.5	0.7	17	7.3	7.8	3.350215	5.286683	2996.348165	–	–	–
Cagnina et al. [22]	3.5	0.7	17	7.3	7.8	3.350214	5.286683	2996.348165	–	2996.3482	0.0000
Akay and Karaboga [26]	3.499999	0.7	17	7.3	7.8	3.350215	5.287800	2997.058412	–	2997.058412	–
Brajevic and Tuba [27]	3.500000	0.7	17	7.3	7.71532	3.350215	5.286654	2994.471066	–	2994.471072	0.00000.598
Gandomi et al. [31]	3.5015	0.7000	17	7.6050	7.8181	3.3520	5.2875	3000.9810	3.0090	3007.1997	4.9634
Baykasoglu and Ozsoydan [34]	3.500000	0.7000	17	7.302489	7.800067	3.350219	5.286683	2996.372698	2996.669016	2996.514874	0.09
WSA	**3.50000001**	**0.7**	**17**	**7.3**	**7.8**	**3.35021489**	**5.28668322**	**2996.34822249**	**2996.34824388**	**2996.34823295**	**9.10931E−06**

Table 10a
p-Values of the paired t-test ($\alpha = 0.05$).

	p-Value
Static Penalty Function – Adaptive Penalty Function	0.412647742
Static Penalty Function – Dynamic Penalty Function	0.403880381
Static Penalty Function – ITCH Approach	0.876355061
Static Penalty Function – Deb's Method	0.525959085
Static Penalty Function – Morales and Quezada's Method	0.355499139
Adaptive Penalty Function – Dynamic Penalty Function	0.324908043
Adaptive Penalty Function – ITCH Approach	0.441382930
Adaptive Penalty Function – Deb's Method	0.415212204
Adaptive Penalty Function – Morales and Quezada's Method	0.531939072
Dynamic Penalty Function – ITCH Approach	0.431954015
Dynamic Penalty Function – Deb's Method	0.406430087
Dynamic Penalty Function – Morales and Quezada's Method	0.494999064
ITCH Approach – Deb's Method	0.255497670
ITCH Approach – Morales and Quezada's Method	0.400196153
Deb's Method – Morales and Quezada's Method	0.362765966

s.t.

$$g_{51}(x) := 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \tag{73}$$

$$g_{52}(x) := 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \tag{74}$$

$$g_{53}(x) := 2x_2 + 2x_3 + x_{12} + x_{12} - 10 \leq 0 \tag{75}$$

$$g_{54}(x) := -8x_1 + x_{10} \leq 0 \tag{76}$$

$$g_{55}(x) := -8x_2 + x_{11} \leq 0 \tag{77}$$

$$g_{56}(x) := -8x_3 + x_{12} \leq 0 \tag{78}$$

$$g_{57}(x) := -2x_4 - x_5 + x_{10} \leq 0 \tag{79}$$

$$g_{58}(x) := -2x_6 - x_7 + x_{11} \leq 0 \tag{80}$$

$$g_{59}(x) := -2x_8 - x_9 + x_{12} \leq 0 \tag{81}$$

$$D := \left\{ \boldsymbol{x} \in \mathbb{R}^{13} : (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T \leq x \leq (1, 1, 1, 1, 1, 1, 1, 1, 1, 100, 100, 100, 1)^T \right\} \tag{82}$$

This problem was also tackled by WSA algorithm and the obtained results compared with the results produced by the formerly published papers in literature and provided by [39] in Table 10. From the observation of Table 10, we can conclude again that WSA algorithm is an effective search algorithm for constrained global optimization problems. In Table 10, we only reported the best results obtained by WSA with different constraint handling methods, since there was not a significantly difference between them. It must also be stated that the results of WSA, reported in Table 10, were obtained after running WSA 30 times independently with 100 artificial agents for a maximum number of 1000 iterations.

3.3. Statistical analysis

A final analyses, a paired t-test, is executed on Excel 2013 in order to understand whether the differences in obtained results of the four test problems with WSA by using the constraint handling methods are due to random chance or not. This test is executed on the average values for 20 independent replications represented in Tables 2, 4, 6 and 8, and the results of the executed paired t-test are presented in Table 10.

From the observation of the p-values presented in Table 10, we can conclude that, the used constraint handling methods have not a significant effect over performance of the WSA algorithm. Furthermore, Table 10 obviously indicates that the WSA is a robust approach in solving the constrained mechanical design optimization problems.

Table 10b
Comparison of WSA with different methods for the fifth test problem.

Method	Best	Mean	Worst	Std. Dev.	Func. Eval.
WSA	−14.99992	−14.99965	−14.99915	2.2717E−4	100,000
PSRE [39]	−15	−14.876	−14.6819	0.113	100,000
Coello and Cortes [58]	−14.7841	−14.5266	−13.8417	0.2335	150,000
Yoo and Hajela [59]	−5.2735	−3.7435	−2.4255	0.9696	150,000
Hamida and Schoenauer [60]	−15	−14.84	N.A.	N.A.	1,500,000
Koziel and Michalewicz [61]	−14.7864	−14.7082	−14.6154	N.A.	1,000,000
Hadj-Alouane and Bean [62]	−5.16559	−3.64004	−2.72518	0.60624	N.A.
Michalewicz and Attia [63]	−734,334	−5.07136	−3.59536	0.77247	N.A.

4. Conclusions

In this paper, we have tested the performance of the WSA algorithm, which is a new swarm based search algorithm and inspired by the superposition and the attracted movement of agents, in solving the well-known constrained mechanical design optimization problems. Since these problems are constrained optimization problems, WSA must cope with the feasibility problem during its search. Within this context, some constraint handling methods were required to adapt into WSA in order to make the algorithm to have the capability of overcoming feasibility problem. For this purpose, we have selected 6 formerly developed constraint handling methods and adapted them into WSA algorithm, then we have analyzed the effects of these methods over the performance of the WSA algorithm. The obtained results indicate how effective the WSA algorithm in solving the constrained mechanical design problems. Additionally, the robustness of the WSA algorithm stood out, since the used constraint handling methods were not significantly effective on the performance of the WSA algorithm in solving the constrained mechanical design problems. Computational results also indicate the effectiveness of the WSA algorithm by solving the constrained mechanical design problems in terms of obtained results, reached level of convergence and the capability of coping with the problems of premature convergence, trapping in a local optima and stagnation.

References

[1] A. Baykasoğlu, Design optimization with chaos embedded great deluge algorithm, Appl. Soft Comput. 12 (2012) 1055–1567.
[2] A. Baykasoğlu, Ş. Akpınar, Weighted Superposition Attraction (WSA): a swarm intelligence algorithm for optimization problems – Part 1: unconstrained optimization, Appl. Soft Comput. (2015), submitted for publication).
[3] A.D. Belegundu, A Study of Mathematical Programming Methods for Structural Optimization, Dept. of Civil Environ. Eng., Iowa Univ., 1982.
[4] J.S. Arora, Introduction to Optimum Design, McGraw-Hill, New York, 1989.
[5] C.A.C. Coello, Self-adaptive penalties for GA based optimization, Proc. Congr. Evolut. Comput. 1 (1999) 573–580.
[6] C.A.C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127.
[7] C.A.C. Coello, E. Mezura-Montes, Use of dominance-based tournament selection to handle constraints in genetic algorithms, in: Intelligent Engineering Systems through Artificial Neural Networks (ANNIE'2001), ASME Press, St. Louis, MI, 2001, pp. 177–182.
[8] C.A.C. Coello, E. Mezura-Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inf. 16 (2002) 193–203.
[9] C.A.C. Coello, E. Mezura-Montes, R.L. Becerra, Engineering optimization using a simple evolutionary algorithm, in: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, 2003.
[10] C.A.C. Coello, R.L. Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, Eng. Optim. 36 (2) (2004) 219–236.
[11] E. Mezura-Montes, C.A.C. Coello, Useful infeasible solutions in engineering optimization with evolutionary algorithms, in: A. Gelbukh, A.D. Albornoz, H. Terashima-Marín (Eds.), Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 652–662.
[12] S. Akhtar, K. Tai, T. Ray, A socio-behavioural simulation model of engineering design optimization, Eng. Optim. 34 (2002) 341–354.
[13] T. Ray, K.M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behaviour, IEEE Trans. Evolut. Comput. 7 (4) (2003) 386–396.
[14] T. Ray, P. Saini, Engineering design optimization using a swarm with an intelligent information sharing among individuals, Eng. Optim. 33 (6) (2001) 735–748.
[15] X. Hu, R. Eberhart, Y. Shi, Engineering optimization with particle swarm, in: Swarm Intelligence Symposium, SIS '03. Proceedings of the 2003 IEEE, 2003, pp. 53–57.
[16] S. He, E. Prempain, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Eng. Optim. 36 (5) (2004) 585–605.
[17] K.E. Parsopoulos, M.N. Vrahatis, Unified particle swarm optimization for solving constrained engineering optimization problems, in: L. Wang, K. Chen, Y.S. Ong (Eds.), Advances in Natural Computation, Springer, Berlin, 2005, pp. 582–591.
[18] H. Aguirre, A.M. Zavala, E.V. Diharce, S.B. Rionda, COPSO: constrained optimization via PSO algorithm. Technical Report No. I-07-04/22-02-2007, Center for Research in Mathematics (CIMAT), 2007.
[19] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (1) (2007) 89–99.
[20] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2007) 1407–1422.
[21] G. Tomassetti, A cost-effective algorithm for the solution of engineering problems with particle swarm optimization, Eng. Optim. 42 (2010) 471–495.
[22] L. Cagnina, S. Esquivel, C.A.C. Coello, Solving engineering optimization problems with the simple constrained particle swarm optimizer, Informatica 32 (3) (2008) 319–326.
[23] I. Maruta, T.H. Kim, T. Sugie, Fixed-structure H∞ controller synthesis: a meta-heuristic approach using simple constrained particle swarm optimization, Automatica 45 (2009) 553–559.
[24] T.H. Kim, I. Maruta, T. Sugie, A simple and efficient constrained particle swarm optimization and its application to engineering design problems, Proc. Inst. Mech. Eng. C: J. Mech. Eng. Sci. 224 (2) (2010) 389–400.
[25] S. Chun, Y.T. Kim, T.H. Kim, A diversity-enhanced constrained particle swarm optimizer for mixed integer-discrete-continuous engineering design problems, Adv. Mech. Eng. (2013), http://dx.doi.org/10.1155/2013/130750.
[26] B. Akay, D. Karaboga, Artificial bee colony algorithm for large-scale problems and engineering design optimization, J. Intell. Manuf. 23 (4) (2012) 1001–1014.
[27] I. Brajevic, M. Tuba, An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems, J. Intell. Manuf. 24 (2013) 729–740.
[28] A.R. Yildiz, Comparison of evolutionary-based optimization algorithms for structural design optimization, Eng. Appl. Artif. Intell. 26 (2013) 327–333.
[29] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2007) 1567–1579.
[30] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems, Comput. Methods Appl. Mech. Eng. 197 (2008) 3080–3091.
[31] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (2013) 17–35.
[32] A.H. Gandomi, X.S. Yang, A.H. Alavi, S. Talatahari, Bat algorithm for constrained optimization tasks, Neural Comput. Appl. 22 (6) (2013) 1239–1255.
[33] A.H. Gandomi, X.S. Yang, A.H. Alavi, Mixed variable structural optimization using firefly algorithm, Comput. Struct. 89 (2011) 2325–2336.
[34] A. Baykasoğlu, F.B. Ozsoydan, Adaptive firefly algorithm with chaos for mechanical design optimization problems, Appl. Soft Comput. 36 (2015) 152–164.
[35] A.R. Yildiz, N. Öztürk, N. Kaya, F. Öztürk, Hybrid multi-objective shape design optimization using Taguchi's method and genetic algorithm, Struct. Multidisc. Optim. 34 (2007) 317–332.
[36] A.R. Yildiz, A new hybrid particle swarm optimization approach for structural design optimization in automotive industry, Proc. Inst. Mech. Eng. D – J. Automob. Eng. 226 (2012) 1340–1351.
[37] A.R. Yildiz, Hybrid Taguchi-harmony search algorithm for solving engineering optimization problems, Int. J. Ind. Eng – Theor. 15 (2008) 286–293.
[38] A.R. Yildiz, A new hybrid artificial bee colony optimization algorithm for robust optimal design and manufacturing, Appl. Soft Comput. 13 (2013) 2906–2912.
[39] A.R. Yildiz, A novel particle swarm optimization approach for product design and manufacturing, Int. J. Adv. Manuf. Technol. 40 (2009) 617–628.
[40] A.R. Yildiz, K. Saitou, Topology synthesis of multicomponent structural assemblies in continuum domains, J. Mech. Des. 133 (2011).

[41] A.R. Yildiz, A novel hybrid immune algorithm for global optimization in design and manufacturing, Robot. Cim-Int. Manuf. 25 (2009) 261–270.
[42] A.R. Yildiz, An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry, J. Mater. Process. Technol. 209 (2009) 2773–2780.
[43] S. Akpinar, A. Baykasoğlu, Multiple colony bees algorithm for continuous spaces, Appl. Soft Comput. 24 (2014) 829–841.
[44] J.H. Lee, P.T. Chang, A survey and numerical comparison of factor-free penalty function constraint-handling techniques in genetic algorithms, J. Chin. Inst. Ind. Eng. 29 (1) (2012) 61–86.
[45] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolutionary Programs, Springer, New York, 1992.
[46] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (2002) 1245–1287.
[47] A. Homaifar, S.H.Y. Lai, X. Qi, Constrained optimization via genetic algorithms, Simulation 62 (4) (1994) 242–254.
[48] A.B. Hadj-Alouane, J.C. Bean, A Genetic Algorithm for the Multiple-choice Integer Program. Technical Report TR 92-50, Department of Industrial and Operations Engineering, The University of Michigan, 1992.
[49] J. Joines, C. Houck, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Gas, in: D. Fogel (Ed.), Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, FL, 1994, pp. 579–584.
[50] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2000) 311–338.
[51] A.K. Morales, C.V. Quezada, A universal eclectic genetic algorithm for constrained optimization, in: Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, September 7–10, 1998, pp. 518–522.
[52] J.S. Arora, Introduction to Optimum Design, 3rd ed., Elsevier, Amsterdam, 2012.
[53] B.K. Kannan, S.N. Kramer, An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, J. Mech. Des. 116 (1994) 318–320.
[54] S.S. Rao, Engineering Optimization, 3rd ed., Wiley, New York, 1996.
[55] J.N. Siddall, Analytical Decision-making in Engineering Design, Prentice-Hall, Englewood Cliffs, 1972.
[56] J. Golinski, An adaptive optimization system applied to machine synthesis, Mech. Mach. Synth. 8 (1973) 419–436.
[57] C. Floudas, P. Pardalos, A Collection of Test Problems for Constrained Global Optimization Algorithms, vol. 455 of LNCS, Springer Verlag, Berlin, 1987.
[58] C.A.C. Coello, N.C. Cortes, Hybridizing a genetic algorithm with an artificial immune system for global optimization, Eng. Optim. 36 (2004) 607–634.
[59] J. Yoo, P. Hajela, Immune network simulations in multi-criterion design, Struct. Multidisc. Optim. 1 (1999) 85–94.
[60] S.B. Hamida, M. Schoenauer, ASCHEA: new results using adaptive segregational constraint handling, Proc. Congr. Evolut. Comput. 1 (2002) 884–889.
[61] S. Koziel, Z. Michalewicz, Evolutionary algorithms homomorphous mappings and constrained parameter optimization, Evol. Comput. 71 (1999) 19–44.
[62] A.B. Hadj-Alouane, J.C. Bean, A genetic algorithm for the multiple-choice integer program, Oper. Res. 45 (1997) 92–101.
[63] Z. Michalewicz, N. Attia, Evolutionary optimization of constrained problems, in: Proc. 3rd Annual Conference on Evolutionary Programming World Scientific, 1994, pp. 98–108.