DEVELOPMENT AND APPLICATIONS OF METAHEURISTIC ALGORITHMS IN ENGINEERING DESIGN AND STRUCTURAL OPTIMIZATION

ALI SADOLLAH

INSTITUTE OF GRADUATE STUDIES
UNIVERSITY OF MALAYA
KUALA LUMPUR

2013

DEVELOPMENT AND APPLICATIONS OF METAHEURISTIC ALGORITHMS IN ENGINEERING DESIGN AND STRUCTURAL OPTIMIZATION

ALI SADOLLAH

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2013

ABSTRACT

Metaheuristic algorithms have been extensively used in numerous domains especially in engineering. The reason is that for solving complex optimization problems, classical and traditional techniques may not efficiently find global optimum solution.

In this thesis, the applications of a number of well-known metaheuristic algorithms for solving engineering problems have been considered. In addition, two novel optimization methods are developed and presented which are named the mine blast algorithm (MBA) and the water cycle algorithm (WCA).

The fundamental concepts and ideas for MBA are derived from the explosion of mine bombs in real world. Accordingly, the ideas and philosophy of WCA are inspired from water cycle process in the nature and how rivers and streams flow to the sea in the real world. The efficiency of the proposed optimizers was evaluated using numerous well-known unconstrained and constrained benchmark functions which have been widely used in literature.

Optimization of several truss structures (2D and 3D) with discrete variables were carried out using the proposed methods and the results and computational performances were compared with several well-known metaheuristic algorithms. The obtained optimization results shows that the proposed new metaheuristic algorithms are capable of offering faster convergence rate in addition to offering better optimal solutions compared to other optimizers. Furthermore, a comparative study was carried out to show the effectiveness of the proposed algorithms over other well-known methods in terms of computational time (speed) and function values.

As an illustration of statistical optimization results, the MBA and WCA offer minimum weight of 27,532.95 and 29,304.76, respectively, for the complex

200-bar truss in less number of function evaluations (computational time) compared with other optimizers in the literature.

ABSTRAK

Algoritma Metaheuristic telah digunakan secara meluas dalam pelbagai domain terutamanya dalam bidang kejuruteraan. Sebabnya ialah bahawa untuk menyelesaikan masalah pengoptimuman kompleks, teknik klasik dan tradisional mungkin tidak cekap mencari penyelesaian optimum global. Dalam tesis ini, aplikasi beberapa algoritma metaheuristic yang terkenal untuk menyelesaikan masalah kejuruteraan telah dipertimbangkan.

Di samping itu, dua kaedah pengoptimuman novel dibangunkan dan dibentangkan yang dinamakan algoritma letupan lombong (MBA) dan algoritma kitaran air (WCA). Konsep-konsep asas dan idea untuk MBA berasal dari letupan bom lombong dalam dunia sebenar.

Sehubungan dengan itu, idea-idea dan falsafah WCA diilhamkan daripada proses kitaran air dalam sifat dan bagaimana sungai dan aliran sungai ke laut dalam dunia sebenar. Kecekapan daripada pengoptimal yang dicadangkan telah dinilai menggunakan banyak terkenal tidak dikekang dan dikekang fungsi penanda aras yang telah digunakan secara meluas dalam kesusasteraan.

Pengoptimuman beberapa struktur kekuda (2D dan 3D) dengan pembolehubah diskret telah dijalankan menggunakan kaedah yang dicadangkan dan keputusan dan persembahan pengiraan berbanding dengan algoritma metaheuristic beberapa terkenal. Keputusan pengoptimuman diperolehi menunjukkan bahawa algoritma baru yang dicadangkan metaheuristic mampu menawarkan kadar penumpuan yang lebih cepat di samping menawarkan penyelesaian yang lebih baik yang optimum berbanding pengoptimal lain.

Sebagai wakil keputusan pengoptimuman statistik, MBA dan WCA menawarkan berat badan sekurang-kurangnya 27,532.95 29,304.76 dan masing-

masing, bagi kekuda 200-bar di nombor kurang daripada penilaian fungsi (masa pengiraan) berbanding dengan pengoptimal lain dalam kesusasteraan.

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Ali Sadollah **I.C/Passport No.**:

Registration/Matric No.: KHA100051

Name of Degree: PhD of Engineering

Title of Project Paper/ Research Report/ Dissertation/ Thesis:

Development and applications of metaheuristic algorithms in engineering design and structural optimization

Field of Study: Computational intelligence (soft computing methods)

DECLARATION

Chapters 3 to 5 are based on my published papers as follows:

1. **Sadollah, A**., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2012). Mine blast algorithm for optimization of truss structures with discrete variables. *Computers & Structures*, *102-103*, 49-63.
2. Eskandar, H., **Sadollah, A.**, Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm - a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, *110-111*, 151-166.
3. **Sadollah, A.**, Bahreininejad, A., Eskandar, H., & Hamdi, M. (2012). Mine Bomb Algorithm: a new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing*, DOI: http://dx.doi.org/10.1016/j.asoc.2012.11.026.

In these papers, my contribution was to develop and model new optimization engines. In addition, coding and validation of the proposed methods was also carried out.

Associate Professor Dr. Ardeshir Bahreininejad	Signature:
Professor Dr. Mohd Hamdi	Signature:
Hadi Eskandar	Signature:

ACKNOWLEDGEMENTS

The preparation of this work started with my birth. Everything I learned since that day, contributed to this work. I certainly have had many teachers, friends, and advisors who helped shaping my knowledge, skills, and attitudes. I value and thank them all, especially my lovely parents who have given me all they have, their life, and love.

My gratitude goes to Associate Prof. Dr. Ardeshir Bahreininejad, my major supervisor. I thank him for his faith in me and for guiding me on the right track toward a career in my research. I appreciate his open mindedness and vast knowledge, which he always made available for me. He has been a very generous source of knowledge and support, and a role model to follow. I appreciate his tremendous enthusiasm in teaching, coaching, and helping me, and I hope I can live up to his expectation in my scientific career. I certainly am lucky to have learned computational intelligence and metaheuristics from one of the leading pioneers in this field. I am also greatly indebted to all my committee members, who gave me their knowledge, support, and resources. Special appreciation is due to Prof. Dr. Mohd Hamdi Bin Abd Shukor and Associate Prof. Dr. Judha Purbolaksono who have gone a long way beyond duty to support me in the difficult times. At University of Malaya, I have had tremendous support and help, without which this work would have never been possible. Furthermore, I would like to acknowledge and express my gratitude to the Ministry of Higher Education of Malaysia and the University of Malaya, Kuala Lumpur, Malaysia for the all financial support under UM.TNC2/IPPP/UPGP/628/6/ER013/2011A. I am deeply thankful to all the faculty, staff, and colleagues who helped me in every way they could.

Finally, I wish to dedicate this thesis to my loving parents and family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Symbols and Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AIS	Artificial Immune Systems
ASCHEA	Adaptive Segregational Constraint Handling Evolutionary Algorithm
CAEP	Cultural Algorithms with Evolutionary Programming
CDE	Co-Evolutionary Differential Evolution
CPSO	Coevolutionary Particle Swarm Optimization
CRGA	Changing Range Genetic Algorithm
CULDE	Cultured Differential Evolution
DE	Differential Evolution
DEDS	Differential Evolution with Dynamic Stochastic Selection
DELC	Differential Evolution with Level Comparison
DHPSACO	Discrete Heuristic Particle Swarm Ant Colony Optimization
GA	Genetic Algorithm
G-QPSO	Gaussian Quantum-behaved Particle Swarm Optimization
HEAA	Hybrid Evolutionary Algorithm and Adaptive Constraint Handling
HGA	Hybrid Genetic Algorithm
HM	Homomorphous Mappings
HS	Harmony Search
HPSO	Hybrid Particle Swarm Optimization
ICA	Imperialist Competitive Algorithm
ISR	Improved Stochastic Ranking
MBA	Mine Blast Algorithm
MGA	Modified Genetic Algorithm
NM-PSO	Hybrid Nelder-Mead Simplex Search and Particle Swarm Optimization
NSGA-II	Non-dominated Sorting Genetic Algorithm
PESO	Particle Evolutionary Swarm Optimization
PSO	Particle Swarm Optimization
PSOPC	Particle Swarm Optimization with Passive Congregation
PSO-DE	Particle Swarm Optimization with Differential Evolution
QPSO	Quantum-Behaved Particle Swarm Optimization
SA	Simulated Annealing
SAPF	Self Adaptive Penalty Function
SC	Society and Civilization
SGA	Steady State Genetic Algorithms
SR	Stochastic Ranking
SMES	Simple Multi-membered Evolution Strategy
TLBO	Teaching-Learning-Based Optimization
WCA	Water Cycle Algorithm

E	Modulus of elasticity
A	Area of truss member
NC	Initial number of colonies
TC	Total power of an empire
k	Iteration number index
LB	Lower bound of a problem
UB	Upper bound of a problem
rand	Uniformly distributed random number between 0 and 1
randn	Normally distributed pseudorandom number
X	Randomly generated solution/location
F	the function value for the *X*, Actuating force
C	Value of cost function
r	radius
t	Thickness
R	Inner radius
L	Length of the cylindrical section of the vessel
D	Mean coil diameter
d	Wire diameter
P	Number of active coils, Buckling load, Vertical loads in trusses
Z	Number of friction surfaces
N	Number of quantities
d	Distance
m	Direction (slope) of the thrown shrapnel pieces
NS	Number of streams which flow to the specific rivers or sea
f(X)	Value of solution *X*

Greek symbols

ξ	A positive small number
μ	Exploration factor
θ	Angle of the shrapnel pieces
α	Reduction constant
τ	Shear stress of the beam
σ	Bending stress of the beam
δ	End deflection of the beam
ε	Allowable small non-negative value

Subscripts and Superscripts

i	Inner radius, Counter index
o	Outer radius
s	Shrapnel piece, Shell
h	Head of the cylinder
pop	population
vars	Number of design variables
sr	summation of Number of Rivers and sea
max	Maximum allowed value
0	Initial distance
f	Number of first shot point
n	nth empire, Counter index
*	Optimum
col	Colonies
d	Search space dimension
e	Exploding mine bomb
x	*X* direction
y	*Y* direction
z	*Z* direction

CHAPTER 1 : INTRODUCTION

1.1. Introduction

Optimization is the process of making something better. An engineer or scientist comes up with a new idea and optimization improves on that idea. Optimization consists of trying variations on an initial concept and using the information gained to improve on the idea. A computer is the perfect tool for optimization as long as the idea or variable influencing the idea can be input in electronic format (Haupt & Haupt, 2004).

1.1.1. Finding the best solution

The terminology "best" solution implies that there is more than one solution and the solutions are not of equal value. The definition of "best" is relative to the problem at hand, its method of solution, and the tolerances allowed. Thus the optimal solution depends on the person formulating the problem.

Some problems have exact answers or roots, and best has a specific definition. Examples include a solution to a linear first-order differential equation. Other problems have various minimum or maximum solutions known as optimal points or extrema, and best may be a relative definition. Examples include best piece of artwork or best musical composition (Haupt & Haupt, 2004).

1.1.2. What is optimization?

Our lives confront us with many opportunities for optimization. What is the best route to work? When designing something, we shorten the length of this or reduce the weight of that, as we want to minimize the cost or maximize the appeal of a product.

Optimization is the process of adjusting the inputs to or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or results (Haupt & Haupt, 2004). The input consists of variables. The process or function is known as the cost function, objective function, or fitness function, and the output is the cost or fitness. If the process is an experiment, then the variables are physical inputs to the experiment.

Since in engineering applications we usually seek to the minimum values such as minimum stress, weight, cost, etc, the output from the process or function defines as the cost function. Since cost is something to be minimized, optimization becomes minimization. Sometimes maximizing a function makes more sense. To maximize a function, just put a minus sign on the front of the output and minimize it.

Life is interesting due to the many decisions and seemingly random events that take place. Quantum theory suggests there are an infinite number of dimensions, and each dimension corresponds to a decision made. Real life problems are also highly nonlinear, so chaos plays an important role too. A small perturbation in the initial condition may result in a very different and unpredictable solution.

These theories suggest a high degree of complexity faced when studying nature or designing products. Science developed simple models to represent certain limited aspects of nature. Most of these simple (and usually linear) models have been optimized. In the future, scientists and engineers must tackle the unsolvable problems of the past, and optimization is a primary tool needed in the intellectual toolbox (Haupt & Haupt, 2004).

1.1.3. Natural optimization techniques

In complex optimization problem, classical and traditional approaches for optimizing are not efficient and capable of finding the global optimum point (Lee & Geem, 2005). Because they need the derivative of objectoive function and, therefore, the objective function must be continous, while many complex optimization problems have discrerte and combinatorial nature.

Usually, finding the derivate of complex and real life problems is sometimes impossible or takes long time. These reasons reveal many shortfalls of the typical minimum seekers such as exhaustive search, analytical approaches, nelder-mead downhill simplex method (Nelder & Mead, 1965), complex method (Box, 1965), coordinate search method (Schwefel, 1995; Luenberger 1984; Press et al., 1992), steepest descent algorithm (Cauchy, 1847), Davidon-Fletcher-Powell (DFP) algorithm (Powell, 1964), Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (Broyden, 1965; Fletcher, 1963; Goldfarb, 1968; Shanno, 1970), and recursive quadratic programming (Luenberger, 1984).

Since the local optimizers of the past are limited, people have turned to more global methods based upon biological and natural processes. The need of (new) algorithms which works without derivatives and can be applied to combinatorial problems are crucial especially in engineering field.

The metaheuristic algorithms have been extensively used in numerous domains especially in engineering. The advantages of metaheuristic algorithms compared to other traditional approaches are listed as below (Lee & Geem, 2005):

1. They are very flexible in terms of usage and application.
2. Often, they consider as global optimizers.
3. Often robust to the problem size and random variables.
4. May be only practical alternative.
5. No need to calculate the derivative of function.
6. The problem can be continues or discrete.
7. Faster and stronger than other traditional methods.

In this thesis, the main objective is to investigate and develop new metaheuristic algorithms which can outperform (or equally perform) against the existing methods.

1.2. Objectives of thesis

The objective of this thesis is to investigate, model, and develop new metaheuristic algorithms which are based on the ideas of natural phenomena and real life events. This study embarks on the following objectives:

1. To investigate and develope optimization algorithms which are modeled on natural phenomena or real life events.
2. To establish mathematical models for the proposed optimization algorithms.
3. To implement, test, and compare the proposed optimizers with other existing optimization methods for benchmark optimization problems.

CHAPTER 2 : LITERATURE REVIEW ON METAHEURISTICS AND THEIR APPLICATIONS ON ENGINEERING DESIGN

2.1. Introduction

Soft computing became a formal computer science area of study in the early 1990's (Kincaid, 1990). Earlier computational approaches could model and precisely analyze only relatively simple systems. More complex systems arising in biology, medicine, the humanities, management sciences, and similar fields often remained intractable to conventional mathematical and analytical methods.

It should be pointed out that simplicity and complexity of systems are relative, and many conventional mathematical models have been both challenging and very productive. Soft computing deals with imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness, and low solution cost. Components of soft computing include (Kincaid, 1990):

- Neural networks (NNs)
- Fuzzy systems (FSs)
- Evolutionary computation (EC), including:
    - Evolutionary algorithms
- Swarm intelligence

Generally speaking, soft computing techniques resemble biological processes more closely than traditional techniques, which are largely based on formal logical systems, such as sentential and predicate logics, or rely heavily on computer-aided numerical analysis (as in finite element analysis (FEM)) (Duan et al., 1992).

Soft computing techniques are intended to complement each other. In contrast of hard computing schemes, which strive for exactness and full truth, soft computing approaches exploit the given tolerance of imprecision, partial truth, and uncertainty for a particular problem. Another common contrast comes from the observation that inductive reasoning plays a larger role in soft computing than in hard computing.

Computational intelligence (CI) is an offshoot of artificial intelligence. As an alternative to classical artificial intelligence it rather relies on heuristic algorithms such as in fuzzy systems, neural networks, and evolutionary computation. In addition, CI also embraces techniques that use swarm intelligence, fractals and chaos theory, artificial immune systems, and so forth (Golden et al., 1981).

The CI combines elements of learning, adaptation, evolution, and fuzzy logic (fuzzy sets) to create programs that are, in some sense, intelligent. The CI research does not reject statistical methods, but often gives a complementary view (as is the case with fuzzy systems).

Artificial neural networks (ANNs) is a branch of computational intelligence that is closely related to machine learning (Feldman, 1990). The CI is further closely associated with soft computing, connectionist systems, and cybernetics.

Over the last decades, numerous algorithms have been developed to solve a variety of engineering optimization problems. Most of such algorithms are based on the numerical linear and nonlinear programming methods that may require substantial gradient information and usually seek to improve the solution in the neighborhood of a starting point. These numerical optimization algorithms provide a useful strategy to obtain the global optimum solution for simple and ideal models.

However, many real world engineering optimization problems are very complex in nature and quite difficult to solve. If there is more than one local optimum in the problem, the results may depend on the selection of the starting point for which the obtained optimal solution may not necessarily be the global optimum. Furthermore, the gradient search methods may become unstable when the objective function and constraints have multiple or sharp peaks.

Besides, objective function of these problems may have several global minima (i.e. several points in which the value of the objective function is equal to the global minimum value) and it may have some local minima in which the value of the objective function is very close to the global minimum value. In this situation, traditional techniques are not able to find the global optimum point.

The drawbacks (efficiency and accuracy) of existing numerical methods have encouraged researchers to rely on metaheuristic algorithms based on the simulations and nature inspired methods to solve engineering optimization problems. Metaheuristic algorithms commonly operate by combining rules and randomness to imitate natural phenomena (Lee & Geem, 2005).

These phenomena may include the biological evolutionary process such as genetic algorithms (GAs) proposed by Holland (1975) and Goldberg (1989), animal behavior such as particle swarm optimization (PSO) proposed by Kennedy and Eberhart (1995), and the physical annealing which is generally known as simulated annealing (SA) proposed by Kirkpatrick et al. (1983).

Among the optimization methods, the evolutionary algorithms (EAs) which are generally known as general purpose optimization algorithms are known to be capable of finding the near-optimum solution to the numerical real-valued test problems. EAs have been very successfully applied to optimization problems (Coello, 2002).

Metaheuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality (predefined tolerance). Metaheuristics make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions.

Many metaheuristic algorithms implement some form of stochastic optimization. Metaheuristics are used for combinatorial optimization in which an optimal solution is sought over a discrete search space. An example problem is the travelling salesman problem (TSP) (Golden et al., 1981) where the search space of candidate solutions grows more than exponentially as the size of the problem increases which makes an exhaustive search for the optimal solution impossible.

This phenomenon is commonly known as the curse of dimensionality. Popular metaheuristics for combinatorial problems include SA, ant colony optimization (ACO) (Dorigo et al., 1991a), and tabu search (TS) (Glover, 1990).

Metaheuristics are also used for problems over real-valued search-spaces, where the classic way of optimization is to derive the gradient of the function to be optimized and then, employ gradient descent or a quasi-Newton method.

Metaheuristics do not use the gradient or Hessian matrix, hence, their advantage is that the function to be optimized need not be continuous or differentiable and it can also have several constraints (nonlinear). Popular metaheuristic optimizers for real-valued search-spaces include particle swarm optimization (PSO), and evolution strategies (ES) (Beyer & Schwefel, 2002).

Metaheuristics have been extensive used in numerous field of study including engineering. The research questions are based on the suitability of natural phenomena and real life evens for mimicking them as models for optimization procedures.

In order to escape from local optima, metaheuristic algorithms drive some basic heuristics either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one.

The metaheuristic part permits the low-level heuristic to obtain solutions better than those it could have achieved alone, even if iterated. Usually, the controlling mechanism is achieved either by constraining or by randomizing the set of local neighbor solutions to consider in local search, or by combining elements taken by different solutions.

In population based algorithms such as GA and PSO, several random numbers are produced at each iteration named as population of individual. To obtain a reliable solution or test the reliability of an optimization algorithm, several independent runs should be executed. Due to stochastic nature of metaheuristic algorithms, convergence process and probably the final solution may be different in each independent run (Goldberg, 1989; Kennedy & Eberhart, 1995).

It is worth to mention that one of the most important disadvantages in population based algorithms is crowding of the individuals which show the convergence of the algorithm to a point in the crowded region (Ahrari & Aatai, 2010).

If it happens in the early iterations of the algorithm, solution to which the algorithm has converged is probably a local minimum, because the design space has not been explored adequately. Furthermore, in the final population, similar agents do not present different solutions, which can be a disadvantage especially when the objective function has several global minima.

2.2. Genetic algorithms

Genetic algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems.

The GAs exploits historical information to direct the search into the region of better performance within the search space (Holland, 1975). The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution; especially those follow the principles first laid down by Charles Darwin of "survival of the fittest.” Since in nature, competition among individuals for scanty resources, results in the fittest individuals dominating over the weaker ones (Holland, 1975).

The GAs simulates the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we witness in our DNA. Each individual represents a point in a search space and a possible solution.

The individuals in the population are then made to go through a process of evolution. The GAs are based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals using the following foundations (Goldberg, 1989):

- Individuals in a population compete for resources and mates.
- Those individuals most successful in each competition will produce more offspring than those individuals that perform poorly.
- Genes from good individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.
- Thus each successive generation will become more suited to their environment.

A population of individuals is maintained within search space for a GA, each representing a possible solution to a given problem. Each individual is coded as a finite length vector of components, or variables, in terms of some alphabet, usually the binary alphabet [0, 1].

To continue the genetic analogy, these individuals are likened to chromosomes and the variables are analogous to genes. Thus a chromosome (solution) is composed of several genes (variables). A fitness score is assigned to each solution representing the abilities of an individual to compete.

The individual with the optimal (near optimal) fitness score is sought. The GA aims to use selective breeding of the solutions to produce offspring better than the parents by combining information from the chromosomes. The GAs maintains a population of *n* chromosomes (solutions) with associated fitness values.

Parents are selected to mate, based on their fitness, producing offspring via a reproductive plan. Consequently, highly fit solutions are given more opportunities to reproduce, so that offspring inherit characteristics from each parent. As parents mate and produce offspring, room must be made for the new arrivals since the population is kept at a static size (Holland, 1975).

Individuals in the population die and replaced by the new solutions, eventually creating a new generation once all mating opportunities in the old population have been exhausted. In this way it is hoped that over successive generations better solutions will thrive, while the least fit solutions die out.

New generations of solutions are produced containing, on average, better genes than a typical solution in a previous generation. Each successive generation will contain more good partial solutions than previous generations. Eventually, once the population has converged and is not producing offspring noticeably different from those in previous generations, the algorithm itself is said to have converged to a set of solutions to the problem, at hand, which is called stopping criterion (Goldberg, 1989).

The following studies are a number of applications for GAs in different field of study. Haftka and his co-workers, in particular, extensively tested the application of GAs for maximization of the ultimate load of a laminated plate since early 1990s (Leriche & Haftka, 1993; Kogiso et al., 1994; Todoroki and Haftka, 1998; Soremkun et al., 2001).

Nagendra et al. (1996) proposed an improved GA to find the best stacking sequence of the skin and stiffeners laminate, and the stiffener height for minimum weight of a composite stiffened panel under buckling constraint. Xie et al. (2009) applied GA for optimal design of plate fin heat exchangers. The authors considered minimization of total annual cost as an objective function and pressure drop as a constraint.

Mishra et al. (2009) used GA to carry out second law based optimization of cross flow plate-fin heat exchangers. The authors investigated the minimization of entropy generation units as an objective function.

2.3. Ant colony optimization

Ant colony optimization (ACO) is a paradigm for designing metaheuristic algorithms for combinatorial optimization problems. The first algorithm of ACO which can be classified within this framework was presented in 1991 (Dorigo et al., 1991a; Colorni et al., 1991) and, since then, many diverse variations of the basic principle have been developed.

The essential trait of the ACO algorithms is the combination of a priori information about the structure of a promising solution with a posteriori information about the structure of previously obtained good solutions. The characteristic of the ACO algorithms is their explicit use of elements of previous solutions.

In fact, they drive a constructive low-level solution, but including it in a population framework and randomizing the construction in a Monte Carlo way. A Monte Carlo combination of different solution elements is suggested also by GAs, however, in the case of the ACO, the probability distribution is explicitly defined by previously obtained solution components.

The particular way of defining components and associated probabilities is problem- specific, and it can be designed in different ways, facing a trade-off between the specificity of the information used for the conditioning and the number of solutions which need to be constructed before effectively biasing the probability distribution to favor the emergence of good solutions.

Different applications have favored either the use of conditioning at the level of decision variables, thus requiring a huge number of iterations before getting a precise distribution, or the computational efficiency, thus using very coarse conditioning information. ACO (Dorigo et. al, 1999) is a class of algorithms, whose first member, called Ant System, was initially proposed by Dorigo et al. (1991a) Colorni et al. (1991), and Dorigo (1992).

The main underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained results. The collective behavior emerging from the interaction of the different search threads has proved effective in solving combinatorial optimization (CO) problems.

The ACO has been used with success for many combinatorial optimization problems such as travelling salesman person (TSP) (Dorigo et al., 1991b), vehicle routing problem (Bell & McMullen, 2004), set covering problem (Lessing et al., 2004), and graph coloring (Costa & Hertz, 1997). Aymerich and Serra (2008) studied the application of the ACO to the layup optimization of laminated panels for maximum buckling load.

2.4. Particle swarm optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Kennedy and Eberhart (1995) inspired by social behavior of bird flocking or fish schooling. The PSO shares many similarities with evolutionary computation techniques such as GAs. The system is initialized with a population of random solutions and searches for optimal solution by updating generations.

However, unlike GA, the PSO does not possess evolution operators such as crossover and mutation. In the PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far (the fitness value is also stored).

This value is called *pbest* which stands for personal best. Another "best" value that is tracked by the PSO is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest* which stands for local best. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The PSO concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its *pbest* and *lbest* locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations (Kennedy & Eberhart, 1997).

In the PSO, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience (*pbest*), according to the experience of a neighboring particle (*lbest*), and based on the best experience so far (*gbest*) (Bergh & Engelbrecht, 1997).

Thus, as in modern GAs, a PSO algorithm combines local search approaches with global search methods, attempting to balance exploration and exploitation processes. There are some suggestions for choosing the initial parameters used in the PSO (Trelea, 2003).

The PSO has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement. In past several years, the PSO has been successfully applied in many research and application areas. It is demonstrated that the PSO obtains better results in a faster and cheaper way compared with other optimizers.

Ravagnani et al. (2009) applied PSO for optimal design of shells. The authors considered minimization of area and minimization of cost as per the availability of data. Han et al. (2008) used PSO for rolling fin-tube heat exchanger optimization.

Yu et al. (2008) conducted the PSO for fuzzy optimal design of plate fin heat exchanger. The authors considered minimization of weight and minimization of pressure drop as objectives. Miyazaki and Akisawa (2009) utilized PSO to obtain the optimum cycle time of single stage absorption chiller.

2.5. Simulated annealing

In 1953, Metropolis et al. (1953) developed a method for solving optimization problems that mimics the way thermodynamic systems go from one energy level to another (Fleischer, 1995). He thought of this after simulating a heat bath on certain chemicals. In this method, a system of particles exhibit energy levels in a manner that maximizes the thermodynamic entropy at a given temperature value (Fleischer, 1995).

In addition, the average energy level must be proportional to the temperature, which is constant (Fleischer, 1995). This method is called simulated annealing (SA). The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.

The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. Kirkpatrick et al. (1983) originally thought of using the SA on a number of optimization problems

By analogy with this physical process, each step of the SA algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends both on the difference between the corresponding function values and also on a global parameter *T* (temperature), which is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when *T* is large, however, the rate of random changes is decreased as *T* goes to zero (Kirkpatrick et al., 1983).

The allowance for "uphill" moves potentially saves the method from becoming stuck at local optima. Several parameters need to be included in an implementation of the SA which are summarized by Davidson and Harel (1996):

- The set of configurations/states of the system including an initial configuration (which is often randomly chosen).
- A generation rule for new configurations, which is usually obtained by defining the neighborhood of each configuration and choosing the next configuration randomly from the neighborhood of the current one.

- The cost function to be minimized over the configuration space (this is the analogue of the energy).
- The cooling schedule of the control parameter (*T*) including initial values and rules for when and how to change them (this is the analogue of the temperature and its reduction).
- The termination condition which is usually based on the time, number of iterations, and the values of the cost function and/or the control parameter.

In past several years, the SA has been successfully applied in many applications and field of studies. Practical design of reinforced concrete retaining walls is discussed by Ceranic et al (2001) using the SA technique. May and Balling (1992) studied further reduction of heavy computing effort which is usually required by the SA.

2.6. Imperialist competitive algorithm

Imperialist competitive algorithm (ICA) is inspired from the social-political process of imperialism and imperialistic competition. Similar to many optimization methods, ICA starts with an initial population of individuals. Each individual of the population is called a “country”.

Some of the best countries with the minimum cost are considered as the imperialist states and the rest will be the colonies of those imperialist states. All the colonies are distributed among the imperialist countries based on their power.

To define the algorithm, first of all, initial countries of size $N_{Country}$ are produced. Then, some of the best countries (with the size of N_{imp}) in the population are selected as imperialist states. Therefore, the rest with the size N_{col} will form the colonies that belong to imperialists.

Afterwards, the colonies are divided among imperialists according to their power (Atashpaz-Gargari & Lucas, 2007). In such a way that the initial number of each empire's colonies has to be proportional to its power. So, the initial number of colonies of the nth empire will be given as (Khabbazi et al., 2009):

$$NC_n = round(\left| \frac{Cost_n}{\sum_{i=1}^{N_{imp}} Cost_i} \right| \times N_{col}), \quad n = 1, 2, ..., N_{imp} \tag{2.1}$$

where NC_n is the initial number of colonies for the n^{th} empire, N_{col} is the total number of initial colonies, and N_{imp} is the number of imperialist state. To divide the colonies, NC_n of the colonies are randomly chosen and given to the n^{th} imperialist. After dividing all colonies among imperialists and creating the initial empires, these colonies start moving toward their relevant imperialist country.

This movement is a simple model of assimilation policy. Furthermore, the total power of an empire is defined by the sum of the cost of the imperialist, and some percentage of the mean cost of its colonies as given (Khabbazi et al., 2009):

$$\mathrm{TC_n} = \mathrm{Cost\ (imperialist_n)} + \xi\ \mathrm{(mean\ (Cost(colonies\ of\ empire_n)))} \tag{2.2}$$

where TC_n is the total power of the n^{th} empire and ξ is a positive small number. After computing the total power of empires, usually the weakest colony (or colonies) of the weakest empire is (are) chosen by other empires and the competition is started on possessing this colony (colonies).

Each imperialist participating in this competition, based on its power, has a probable chance of possessing the cited colony. To start the competition, at first, the weakest empire is chosen and then the possession probability of each empire is estimated. The possession probability P_p is related to the total power of the empire (TC) (Atashpaz-Gargari & Lucas, 2007).

During the imperialistic competition, the weak empires will slowly lose their power and getting weak by the time. At the end of process, just one empire will remain that governs the whole colonies (Khabbazi et al., 2009).

2.7. Artificial immune systems

The biological immune system is a robust, complex, and adaptive system that defends the body from foreign pathogens. Depending on the type of the pathogen, and the way it gets into the body, the immune system uses different response mechanisms either to neutralize the pathogenic effect or to destroy the infected cells. A detailed overview of the immune system can be found in many textbooks such as Kubi (2002) and Hightower et al. (1995).

Furthermore, it is able to categorize all cells inside the body as self-cells or non-self cells. Using a distributed task force and its network of chemical messengers for communication, the biological immune system can handle this categorization as well.

There are two major branches of the immune system. The innate immune system is an unchanging mechanism that detects and destroys certain invading organisms, whilst the adaptive immune system responds to previously unknown foreign cells and builds a response to them that can remain in the body over a long period of time (Jerne, 1973; Farmer, 1980).

Generally speaking, this remarkable information processing biological system has caught the attention of computer science in recent years. A novel computational intelligence technique, inspired by immunology, has emerged, called artificial immune systems (AISs) (De Castro & Von Zuben, 1999; Nicosia et al., 2004).

In order to imitate the AISs in optimization problems, the antibodies and affinity are considered as the feasible solutions and the objective function, respectively. Real-value number is used to represent the attributes of the antibodies.

Similar to other population-based methods, a population of random individuals is generated which symbolizes a pool of antibodies. Afterwards, these antibodies undergo proliferation and maturation processes. The proliferation of antibodies is realized by cloning each member of the initial pool depending on their affinity (De Castro & Von Zuben, 1999).

In minimization problem, a pool member with lower objective value is considered to have higher affinity. The proliferation rate is directly proportional to the affinity of the antibodies. The maturation process is carried through hyper-mutation which is inversely proportional to the antigenic affinity of the antibodies.

The next step is the application of the aging operator. This aging operator eliminates old antibodies in order to maintain the diversity of the population and to avoid the premature convergence. In this operator, an antibody is allowed to remain in the population for at most τ_B generations (De castro & Von Zuben, 2002).

After this period, it is assumed that this antibody corresponds to local optima and must be eliminated from the current population, no matter what its affinity may be. During the cloning expansion, a clone inherits the age of its parent and is assigned an age equal to zero when it is successfully hyper-mutated i.e. when hyper-mutation improves its affinity.

Several concepts from the immune have been extracted and applied for solution to real world science and engineering problems (Cutello et al., 2005; Cutello et al., 2006; Rahman et al., 2006; Liao, 2006; Cutello et al., 2007; Panigrahi et al., 2007).

2.8. Constrained and unconstrained benchmark problems

As mentioned in subsections earlier in this chapter, among optimization approaches, metaheuristic optimization engines have shown their capabilities for finding the near-optimal solution to the numerical real-valued test problems for which exact and analytical methods may not produce the optimal solution within a reasonable computation time, especially when the global minimum is surrounded by many local minima. These algorithms are usually devised by observing phenomena happening in nature such as GA, SA, ACO, PSO, and so forth.

The GA with floating-point representation (GAF) consists of three genetic operators (selection, crossover, and mutation) which has been carried out for handling multimodal functions. Details of the GAF operators are presented in literature (De Jong, 1975; Michalewicz, 1992; Michalewicz et al., 1994).

The artificial bee colony (ABC) algorithm introduced by Karaboga (2005) is one approach that has been used to find an optimal solution in numerical optimization problems. The ABC is inspired by the behavior of honey bees when seeking a quality food source (Karaboga & Basturk, 2007). In addition, Akay and Karaboga (2010) investigated the application of ABC for constrained optimization problems.

Pham et al. (2006) developed a metaheuristic method slightly similar to the concept of ABC, called as the bee algorithm (BA). The BA mimics the food foraging behavior of swarms of honey bees. The BA was applied for combinatorial optimization problems (Pham et al, 2006).

Ant colony system (ACS) was derived by the foraging behavior of real ants (Dorigo & Gambardella, 1997). This behavior enables ants to find the shortest path between food sources and their nest. This functionality of real ant colonies is exploited in artificial ant colonies in order to solve unconstrained optimization problems (Aymerich & Serra, 2008).

The idea of the grenade explosion method (GEM) is based on the observation of a grenade explosion, in which the thrown pieces of shrapnel destruct the objects near the explosion location (Ahrari & Aatai, 2010). The loss caused by each piece of shrapnel is calculated and considered as the fitness of the objective function at the object’s location.

Geem et al. (2001) developed a new harmony search (HS) metaheuristic algorithm that was conceptualized using the musical process of searching for a perfect state of harmony. The harmony in music is analogous to the optimization solution vector, and the musician’s improvisations are analogous to local and global search schemes in optimization techniques (Lee & Geem, 2005).

Bacterial foraging optimization (BFO) is based on the foraging (i.e. searching food) strategy of Escherichia coli bacteria (Liu & Passin, 2002). In the BFO, the optimization follows chemo-taxis, swarming, reproduction, elimination, and dispersal events to reach global minima. However, the convergence of original BFO to the optimum value is very slow and its performance is not satisfactory.

Therefore, in synchronous BFO (SBFO) (Bakwad et al., 2010), the best optimum value is updated synchronously after fitness function evaluations of all bacteria. In the SBFO, the optimization follows chemotaxis, swimming, tumbling, and reproduction steps to reach optimum value until computational limitations are exceeded (Bakwad et al., 2010).

The shuffled complex evolution algorithm (SCE-UA) is a general-purpose global optimization algorithm designed to infer the traditional best parameter set and its underlying posterior distribution within a single optimization run (Duan et al., 1992; Vrugt et al., 2003). The goal of the original SCE-UA algorithm is to find a single best parameter set in the feasible space. The modified shuffled complex evolution algorithm (MSCE) introduces the differential evolution algorithm to be used together with the adaptation of the downhill simplex (Mariani et al., 2011).

Differential evolution (DE) is a population-based stochastic function minimizer (or maximize). The DE exhibits an overall excellent performance for a wide range of benchmark multimodal functions (Ursem & Vadstrup, 2003; Vesterström & Thomsen, 2004; Ali & Kajee-Bagdadi, 2009). The DE combines simple arithmetical operators with the operators of recombination, mutation, and selection to evolve from a randomly generated starting population to a final solution.

Ahrari et al. (2010) proposed a covariance matrix adaptation evolution strategy (CMA-ES) for overcoming of getting trapped in local minima for the EAs. To get better performance of the CMA-ES, the Elite search sub-algorithm is introduced and implemented in the basic algorithm. Thereafter, the importance and effects of this modification are illustrated by optimizing a number of unimodal and multimodal benchmark problems (Ahrari et al., 2010).

Zhao et al. (2009) developed an evolutionary optimization engine so called learning algorithm (LA) for solving multimodal optimization. The concept of LA is simple as follows: control parameters, of the length of the list of historical best solutions and the "learning probability" of the current solutions being moved towards the current best solutions and towards the historical ones, are used to assign different search intensities to different parts of the feasible area and to direct the updating of the current solutions (Zhao et al., 2009).

The most multimodal functions considered in the literature are the Schwefel function, Ackley function, Rastrigin function, Sphere function, Rosenbrock function, and Zakharov function with 30 independent variables

(Mariani et al., 2011). These benchmark functions are categorized as high-dimensional problems.

The Schwefel, Ackley, Rastrigin, and Rosenbrock functions are multimodal (various optima) functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms.

It is important to mention that the Rosenbrock function can be treated as a multimodal problem. Rosenbrock function has a narrow parabolic-shaped deep valley from the perceived local optima to the global optimum. To find the valley is trivial, but to achieve convergence to the global minimum is a difficult task. Sphere and Zakharov functions are unimodal (one optimum) (Shang & Qiu, 2006).

Regarding the constrained optimization problems, most researchers have examined their methods with benchmark problems given by Siddall (1982) and Arora (1989). For constrained and engineering problems, the EAs have been successfully applied to constrained optimization problems (Bracken & Mccormick, 1968; Homaifar et al., 1994; Koziel & Michalewicz, 1999; Coello, 2000b; Coello, 2000c; Coello, 2002; Montes & Coello, 2005a; Wang et al., 2009).

Also, GAs was applied for solving engineering and mechanical design (Michalewicz, 1995; Deb & Goyal, 1995; Deb, 2000; Giraud-Moreau & Lafon, 2002; Coello & Montes, 2002; Gupta et al., 2007). Chootinan and Chen (2006) proposed a constraint-handling technique by taking a gradient-based repair method.

The proposed technique is embedded into GAs as a special operator for solving constrained engineering problems (Chootian & Chen, 2006). Recently, Tang et al. (2011) proposed the improved genetic algorithm (IGA) based on a novel selection strategy to handle nonlinear programming constrained problems.

Accordingly, Yuan and Qian (2010) developed a new hybrid genetic algorithm (HGA) to solve twice continuously differentiable nonlinear programming (NLP) problems. The HGA combines the genetic algorithm with local solver differently from some hybrid genetic algorithms (Yuan & Qian, 2010).

Amirjanov (2006) investigated an approach that adaptively shifts and shrinks the size of the search space of the feasible region which is called changing range genetic algorithm (CRGA). The CRGA was successfully optimized engineering constrained problems (Amirjanov, 2006).

Fogel (1995) carried out a comparative study between evolutionary programming (EP) and GA for some selected constrained benchmark functions. Montes and Coello (2005b) proposed a simple multimembered evolution strategy (ES). Later, they improved the efficiency of ES for solving engineering and constrained problems (Montes & Coello, 2008).

He and Wang (2006) proposed an effective co-evolutionary PSO (CPSO) for constrained problems, where the PSO was applied to evolve both decision and penalty factors. In this method, the penalty factors were treated as searching variables and evolved by the GA or PSO to the optimal values.

Coelho (2010) studied quantum-behaved PSO (QPSO) which is derived using mutation operator with Gaussian probability distribution. He and Wang (2007) developed a new hybrid PSO (HPSO) with a feasibility-based rule to solve constrained optimization problems. Other hybridizations of the PSO with other approaches have been studied in the literature (Parsopoulos & Vrahatis, 2005; Renato & Santos, 2006; Zahara & Kao, 2009; Liu et al., 2010).

The DE which is a scheme for generating trial parameter vectors has been widely used for constrained optimization problems (Lampinen, 2002; Zhang et al., 2008; Wang & Cai, 2011; Wang & Cai, 2012b). Furthermore, other variations of

DE have been proposed in the literature (Zavala et al., 2005; Montes et al., 2006b; Huang et al., 2007).

Recently, some hybrid optimization methods have been proposed for handling constrained optimization problems (Montes et al., 2006a; Wang et al., 2007; Wang & Cai, 2009; Wang & Li, 2010; Wang & Cai, 2012a).

Examples of such methods include teaching-learning-based optimization (TLBO) (Rao et al., 2011; Rao & Patel, 2012a, 2012b) which is based on the influence of a teacher on learners, the harmony search (HS) (Lee & Geem, 2005) algorithm which is conceptualized using the musical process of searching for a perfect state of harmony, and the society and civilization (SC) (Ray & Liew, 2003) which is inspired from intra and intersociety interactions within a formal society and the civilization model to solve constrained optimization problems. These algorithms have been applied to numerous engineering optimization problems and have shown the efficiencies in solving some specific kinds of problem.

Stochastic ranking (SR) is an optimization approach trying to balance between objective and penalty functions stochastically and also presents a new view on penalty function methods in terms of the dominance of penalty and objective functions (Runarsson & Xin, 2000; Runarsson & Xin, 2005).

Cultural algorithm with a differential evolution population is proposed by Becerra and Coello (2006). The cultural algorithm uses different knowledge sources to influence the variation operator of the differential evolution algorithm, in order to reduce the number of fitness function evaluations required to obtain competitive results (Coello & Becerra, 2004).

Various other optimization methods have been developed for solving complex and real-life problems, particularly for solving engineering constrained problems (Kannan & Kramer, 1994; Kuang et al., 1998; Coello, 2000a; Hamida &

Schoenauer, 2002; Takahama & Sakai, 2005; Tessema & Yen, 2006; Hedar & Fukushima, 2006; Rao & Savsani, 2012). Furthermore, Table 1.1 represents the applications of considered optimizers for unconstrained and constrained engineering problems. Constrained engineering problems include pressure vessel, spring, welded beam, speed reducer design problem, and so forth.

Table 1.1. Applications of reported methods for unconstrained and constrained problems in this thesis.

Authors	**Methods**	**Constrained problems**	**Unconstrained problems**
Akay & Karaboga (2010)	Artificial Bee Colony (ABS)	×	-
Pham et al. (2006)	Bee Algorithm	×	-
Aymerich & Serra (2008)	Ant Colony System (ACS)	-	×
Ahrari & Aatai (2010)	Grenade Explosion Method (GEM)	-	×
Lee & Geem (2005)	Harmony Search (HS)	×	×
Bakwad et al. (2010)	Synchronous Bacterial Foraging Optimization (SBFO)	-	×
Vrugt et al. (2003)	Shuffled Complex Evolution Algorithm (SCE-UA)	-	×
Mariani et al. (2011)	Modified Shuffled Complex Evolution Algorithm (MSCE)	-	×
Ahrari et al. (2010)	Covariance Matrix Adaptation Evolution Strategy (CMA-ES)	-	×
Zhao et al. (2009)	Learning Algorithm (LA)		×
Gupta et al. (2007)	Genetic Algorithms	×	-
Tang et al. (2011)	Improved Genetic Algorithm (IGA)	×	-
Yuan & Qian (2010)	Hybrid Genetic Algorithm (HGA)	×	-
Amirjanov (2006)	Changing Range Genetic Algorithm (CRGA)	×	-
Montes & Coello (2008)	Evolution Strategy (ES)	×	-
He & Wang	Co-Evolutionary PSO	×	-

(2006)	(CPSO)		
He & Wang (2007)	Hybrid PSO (HPSO)	×	-
Coelho (2010)	Quantum-behaved PSO (QPSO)	×	-
Wang & Cai (2011,2012b)	Differential Evolution (DE)	×	-
Rao & Patel (2012a,2012b)	Teaching-Learning-Based Optimization (TLBO)	×	-
Ray & Liew (2003)	Society and Civilization (SC)	×	-
Runarsson & Xin (2000,2005)	Stochastic Ranking (SR)	×	-

2.9. Truss structures

Over the last decades, various algorithms have been used for truss optimization problems which are very popular in the field of structural optimization. In general, there are three main categories in structural optimization applications: a) sizing optimization (the cross-sectional areas of the members are considered as design variables (Rahami et al., 2008; Kaveh & Talatahari, 2009a), b) shape optimization (nodal coordinates are considered as the design variables (Rahami et al., 2008) and c) topology optimization (the location of links in which nodes are considered as design variables (Rasmussen & Stolpe, 2008; Luh & Lin, 2009).

Metaheuristic methods such as GA (Wu & Chow, 1995), SA (Kirkpatrick et al., 1983), PSO (Perez & Behdinan, 2007) and other stochastic searching methods were used to optimize the trusses.

Goldberg and Samtani (1986) and Rajeev and Krishnamoorthy (1992) have applied sizing optimization on truss structures. Krishnamoorthy et al. (2002) used the GAs to optimize the space truss structure within an object-oriented framework. Sivakumar et al. (2001) presented an optimization technique using the GA for steel lattice towers. Gero et al. (2006) used the GAs for the design optimization of 3D steel structures.

A comprehensive study has been carried out by Adeli and Sarma (2006) for the cost optimization of truss structures using fuzzy logic and GA. Besides, optimization of large steel structures has been investigated using parallel GA (Adeli & Cheng, 1994a, 1994b; Saleh & Adeli, 1994; Soegiarso & Adeli, 1998; Adeli, 2000; Sarma & Adeli, 2001).

Furthermore, for solving structural optimization problems, neural dynamic model, which is a computational method based on the neural network topology and nonlinear dynamic model, was developed (Adeli & Park, 1995a, 1995b). Neural dynamic model was investigated for optimization of truss structures with continuous design variables, bridges, and cold-form steel (Adeli & Saleh, 1997; Adeli & Karim, 1997a, 1997b; Saleh & Adeli, 1998).

Geem et al. (2001) developed a harmony search (HS) metaheuristic algorithm that was conceptualized using the musical process of searching for a perfect state of harmony. The harmony in music is analogous to the optimization solution vector, and the musician's improvisations are analogous to local and global search schemes in optimization techniques (Lee & Geem, 2005). In the sequence, the HS method was applied on truss structures using discrete and continues variables (Lee & Geem, 2004; Lee et al., 2005).

Balling (1991, 1996) studied discrete optimization for three-dimensional steel framed buildings using the SA. The total frame weight was minimized subject to design-code specified constraints on stress, buckling, and deflection.

Kincaid (1990, 1991) optimized a large tetrahedral truss for obtaining minimum surface distortion using the SA and taboo search (TS). Similarly, Chen et al. (1991) applied the SA on large truss structures in which both passive and active vibration suppression was optimized. Bennage and Dhingra (1995) elaborated the application of SA to the design of planar and spatial structures. The authors comprehensively addressed the influence of the SA generic parameters on the results.

Szewczyk and Hajela (1993) examined a neural network approximation of planar and spatial truss structures via a SA search strategy for finding global optimum point. An interesting extension of the SA into simultaneous optimization of size, shape, and topology was developed by Hasancebi and Erbatur (2000).

Recently, the PSO approach is used to optimize the trusses (Perez & Behdinan, 2007). Li et al. (2009) developed a heuristic particle swarm optimization (HPSO) for truss structures, which was proven computationally efficient and reliable, was applied on several truss problems and the obtained results have been compared with hybrid PSO with passive congregation (PSOPC) and standard particle swarm optimization (PSO) (He et al., 2004).

Kaveh and Talatahari (2009b) have combined the PSOPC with ant colony optimization (ACO) and HS to form an efficient algorithm, called heuristic particle swarm ant colony optimization (HPSACO), which was applied on truss optimization with discrete design variables, the so-called discrete HPSACO (DHPSACO) (Kaveh & Talatahari, 2009b).

Also, recently, Gomes (2011) applied the PSO on truss optimization using dynamic constraints. In addition, a summary of applications of reported optimization methods are given in Table 1.2.

Table 2.2. A summary of applications of considered optimizers for the truss structures.

Authors	Methods	Applications
Rahami et al. (2008)	Force Method and Genetic Algorithm	Shape and Sizing Optimizations
Rasmussen & Stolpe (2008); Luh & Lin (2009)	Parallel Cut-and-Branch Method & Ant Colony Optimization	Topology Optimization
Sivakumar et al. (2001)	GA	Steel Lattice Towers Optimization
Krishnamoorthy et al. (2002)	GA	Space Truss Structure Within an Object-Oriented Framework
Gero et al. (2006)	GA	3D Steel Structures
Adeli & Sarma (2006)	Fuzzy Logic and GA	Cost Optimization of Truss Structures
Sarma & Adeli (2001)	Parallel GA	Large Steel Structures
Saleh & Adeli (1998)	Neural Dynamic Model	Truss Structures With Continuous Design Variables, Bridges, and Cold-Form Steel
Lee & Geem (2004)	Harmony Search (HS)	Truss Structures using Discrete and Continues Variables
Balling (1991,1996)	Simulated Annealing (SA)	Discrete Optimization For Three-Dimensional Steel Framed Buildings
Kincaid (1990,1991)	SA and Taboo Search (TS)	Large Tetrahedral Truss
Chen et al. (1991)	SA	Large Truss Structures Having Passive and Active Vibrations
Bennage & Dhingra (1995)	SA	Planar and Spatial Structures
Szewczyk & Hajela (1993)	SA	Neural Network Approximation of Planar and Spatial Truss Structures
Hasancebi & Erbatur (2000)	SA	Size, Shape, and Topology
Kaveh & Talatahari (2009b)	Heuristic Particle Swarm Ant Colony Optimization (HPSACO)	Truss Optimization with Discrete Design Variables
Gomes (2011)	PSO	Truss Optimization using Dynamic Constraints

In summary, this chapter represented the definition of well-known existing metaheuristic methods widely used in the literature. Their fundamental concepts and mathematical formulations also provided in this chapter. In additions, applications of these optimization engines for unconstrained, constrained, and engineering design problems such as truss structures are given in details in this chapter.

In the following chapters (Chapters 4 and 5), our developed methods are described in details as our contribution in this field of research. Afterwards, in Chapter 6, the proposed optimizers are compared with other well-known methods in terms of solution quality and convergence rate (computational time).

CHAPTER 3 : MINE BLAST ALGORITHM

3.1. Basic concepts

The idea of the proposed mine blast algorithm (MBA) is based on the observation of a mine bomb explosion, in which the thrown pieces of shrapnel collide with other mine bombs near the explosion area resulting in their explosion. To understand this situation, consider a mine field where the aim is to clear the mines. Hence, the goal is to find the mines, while the most important is to find the one with the most explosive effect located at optimal point X^*which can cause the most casualties (min or max $f(x)$ per X^*).

The mine bombs of different sizes and explosive powers are planted under the ground. When a mine bomb is exploded, it spreads many pieces of shrapnel and the casualties ($f(x)$) caused by each piece of shrapnel are calculated. A high value for casualties per piece of shrapnel in an area may indicate the existence of other mines which may or may not have higher explosive power.

Each shrapnel piece has definite directions and distances to collide with other mine bombs which may lead to the explosion of other mines due to collision. The collision of shrapnel pieces with other mines may lead us to discover the most explosive mine.

The casualties caused by the explosion of a mine bomb are considered as the fitness of the objective function at the mine bomb's location. The domain (mine field) solution may be divided into infinite grid where there is one mine bomb in each portion of the grid.

3.2. Proposed MBA

The proposed algorithm starts with an initial point(s) called first shot point(s). The first shot point is represented by X_0^f. The superscript f refers to the number of first shot point(s) (f=1,2,3,…), where f can be user defined parameter.

This algorithm requires an initial population of individuals as is the case with some metaheuristic methods.

This population is generated by a first shot explosion producing a number of individuals (shrapnel pieces). The number of initial population (N_{pop}) is considered as the number of shrapnel pieces (N_s). The choice of first shot point(s) may lead the algorithm to search the solution space for different locations.

In addition, it may be no need for entering the first shot point(s). The proposed algorithm can also randomly choose the location(s) of the first shot point(s), without being specified by the user. The algorithm uses the lower and upper bound values given by a problem and create the first shot point value by a small randomly generated value given as:

$$X_0 = LB + rand \times (UB - LB) \tag{3.1}$$

where X_0, LB, and UB are the generated first shot point, lower and upper bounds of the problem, respectively. *rand* is a uniformly distributed random number between 0 and 1. Increasing the number of first shot points increases the initial population and results in an increase in the number of function evaluations (computational cost).

In addition, the increase in first shot points did not offer significant improvement in the optimization process for the problems examined in this thesis. In this thesis, one first shot point was used randomly using Equation (3.1). Suppose that X is the current location of a mine bomb given as:

$$X = \{X_m\}, \quad m = 1, 2, 3, ..., N_d \tag{3.2}$$

in which N_d is the search space dimension equal to the number of independent variables. Consider that N_s shrapnel pieces are produced by the mine bomb explosion causing another mine to explode at X_{n+1} location for 2D space:

$$X_{n+1}^f = X_{e(n+1)}^f + \exp\left(-\sqrt{\frac{m_{n+1}^f}{d_{n+1}^f}}\right) X_n^f \qquad n = 0,1,2,3,... \tag{3.3}$$

where $X_{e(n+1)}^f$, d_{n+1}^f, and m_{n+1}^f are the location of exploding mine bomb collided by shrapnel, the distance and the direction (slope) of the thrown shrapnel pieces in each iteration, respectively. The location of exploding mine bomb $X_{e(n+1)}^f$ is defined as:

$$X_{e(n+1)}^f = d_n^f \times rand \times \cos(\theta) \qquad n = 0,1,2,... \tag{3.4}$$

where *rand* is a uniformly distributed random number and θ is the angle of the shrapnel pieces which is calculated using θ =360/N_s. The exponential term in Equation (3.3) is used to improve the obtained blast point by influencing the information from previous solutions (X_n^f). The distance d_{n+1}^f and the direction of shrapnel pieces m_{n+1}^f are defined as for 2D space:

$$d_{n+1}^f = \sqrt{(X_{n+1}^f - X_n^f)^2 + (F_{n+1}^f - F_n^f)^2} \qquad n = 0,1,2,3,... \tag{3.5}$$

$$m_{n+1}^f = \frac{F_{n+1}^f - F_n^f}{X_{n+1}^f - X_n^f} \qquad n = 0,1,2,3,... \tag{3.6}$$

where *F* is the function value for the *X*. To calculate the initial distance for each shrapnel pieces d_0 = (*UB*-*LB*) in each dimensions is used. The initial distance given by the proposed algorithm is used to search the best solution within a range (*LB* < d_0 < *UB*) that is computed by the product of the initial distance and a randomly generated number (for example *rand* in MATLAB programming software).

Furthermore, in order to conduct exploration of the design space at smaller and larger distances, the exploration factor (μ) is introduced. This constant, which is used in the early iterations of the algorithm, is compared with an iteration number index (k), and if it is higher than k, then the exploration process begins. The formula related to the exploration of the solution space is given as:

$$d_{n+1}^{f} = d_{n}^{f} \times \left(\left|randn\right|\right)^{2} \qquad n = 0, 1, 2, ... \tag{3.7}$$

$$X_{e(n+1)}^{f} = d_{n+1}^{f} \times \cos(\theta) \qquad n = 0, 1, 2, ... \tag{3.8}$$

where *randn* is normally distributed pseudorandom number (*randn* in MATLAB). The square of a normally distributed random number has the advantage of search ability at smaller and larger distances, which offers a better exploration in early iterations. A higher value for the exploration factor (μ) makes it possible to explore more remote regions (better exploration), thus, the value of μ determines the intensity of exploration.

To increase the global search ability of the proposed method, initial distance of shrapnel pieces are reduced gradually to let the mine bombs search the probable global minimum location. A simple formula to reduce is given as:

$$d_{n}^{f} = \frac{d_{n-1}^{f}}{\exp(k / \alpha)} \qquad n = 1, 2, 3, ... \tag{3.9}$$

where α and k are reduction constant which is user parameter and depends on the complexity of the problem and iteration number index, respectively. At the final iteration, the value of distance of shrapnel will be approximately equal to zero (ε=2.2E-16 in MATLAB). The schematic diagram of the algorithm representing two aspects of the MBA (exploration in color lines and exploitation in black color lines) is shown in Figure 3.1.

Based on Figure 3.1, there are two processes for searching the solution domain in order to find the global optimum solution, the exploration and exploitation processes. The difference between these two processes is how they influence the whole search process towards the optimal solution. More specifically, the exploration factor describes the exploration process (color lines in Figure 3.1).

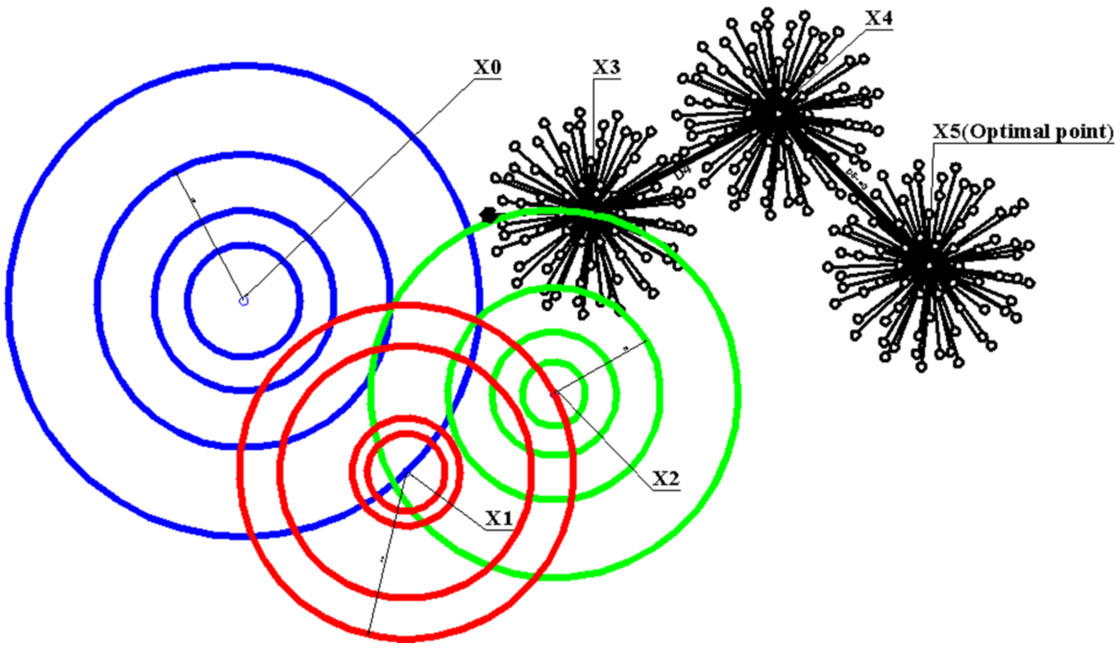


Figure 3.1. Schematic view of the mine blast algorithm including of exploration (color lines) and exploitation (black lines) processes.

Actually, the exploration factor (μ) represents the number of first iterations. Hence, if μ is set to 10, then for 10 iterations the algorithm uses Equations (3.7) and (3.8) for calculating the distance of shrapnel pieces and the location of the exploded mine bomb, respectively.

On the other hand, for the exploitation process (black lines in Figure 3.1), the algorithm is encouraged to focus on the optimal point. In particular, with respect to the exploitation process, the proposed algorithm converges to the global optimum solution using Equations (3.4), (3.5), and (3.6) to calculate the location of exploded mine bomb, distance, and direction of shrapnel pieces, respectively.

The distance of shrapnel pieces is reduced adaptively using Equation (3.9) in exploitation process (i.e., as it converges to the optimal solution). The Pseudocode for the exploration and exploitation processes is as follows:

```
if $\mu > k$
Exploration (Equations (3.7) and (3.8))
else
Exploitation (Equations (3.4), (3.5), (3.6), and (3.9))
end
```

where k is the iteration number index.

3.3. Setting the user parameters

Wrong choice of values for algorithm parameters may result in a low convergence rate, convergence to a local minimum, or undesired solutions. In this thesis, level of complexity is defined with the number of design variables and constraint. In general, problems having up to 4 design variables are considered as simple optimization problem.

For problems having 4 to 20 design variables are categorized as moderate optimization problems, and accordingly, problems with more than 20 design variables are assumed as complex optimization problems. However, this category may not extend for all problems having different number of design variables. It means that there exist problems having only 2 design variables with several local optima and therefore considered as moderate optimization problems (i.e. there are some exceptions). The following guidelines to fine tune the parameters are offered:

• For a simple optimization problem, 10 to 15 pieces of shrapnel per mine bomb can be sufficient. For more complex problems, higher values for the number of shrapnel pieces (N_s) should be selected, since this leads to more mine explosions in the field and, therefore, enables a better search of the design space. For complex problems, N_s may be chosen as 50. On the other hand, increasing the number of shrapnel pieces increases the computation time, in addition to an increase in the number of function evaluations. In other word, the number of shrapnel pieces is the number of population (N_s=N_{pop}).

• Exploration factor (μ) highly depends on the complexity of the problem, the number of independent variables and constraints, and the interval span. Usually, for less than four design variables and moderately complex functions, the value of μ may be taken as zero. Increasing μ may lead the possibility of getting trapped in local minima. In fact, increasing μ means more explorations at each iteration, while an efficient algorithm should balance between exploration and exploitation processes.

• Reduction constant (α) also depends on the complexity of the problem, number of decision variables, and interval span. When the interval span (*LB* and *UB*) is large, large value for α should be chosen for more exploration. That means if we have interval span [-100,100], then α =100 cannot be a good choice, instead α =1000 may be better choice. A larger value for α leads to increase in computational time and also, increases the probability of finding global minimum.

3.4. Constraint handling approach

In the search domain, shrapnel pieces may exceed the constraints of given problem or may violate from upper and lower bounds of design variables. In the current study, a modified feasible-based method is utilized to overcome the constraints which the rules are given as follows (Montes & Coello, 2008):

- Rule 1: Any obtained feasible solution is fancied to any infeasible solution.
- Rule 2: Infeasible solutions having minor violation of the constraints (from 0.01 in the first iteration to 0.001 in the final iteration) are assumed as feasible solutions.
- Rule 3: Between two feasible solutions, the one having the improved objective function value is more interested.

- Rule 4: Between two infeasible solutions, the one with the smaller sum of constraint violation is chosen.

By using the 1th and 4th rules, the search is oriented to the feasible area rather than to the infeasible region, and by applying the 3th rule the search is directed to the feasible district having high quality results (Montes & Coello, 2008). For most structural optimization problems, the global minimum locates on or close to the boundary of a feasible design space. By applying Rule 2, the shrapnel pieces approach the boundaries and can reach the global minimum with a higher probability (Kaveh and Talatahari, 2009b).

Figure 3.2 demonstrates the constraint handling approach for the MBA. As can be seen from Figure 3.2, in the search space, shrapnel pieces may violate either the problem specific constraints or the limits of the design variables. In this case, the distance of infeasible shrapnel piece (e.g. X_3 in Figure 3.2) is reduced adaptively using Equation (3.9) whereas that violated shrapnel piece is also placed in the feasible region.

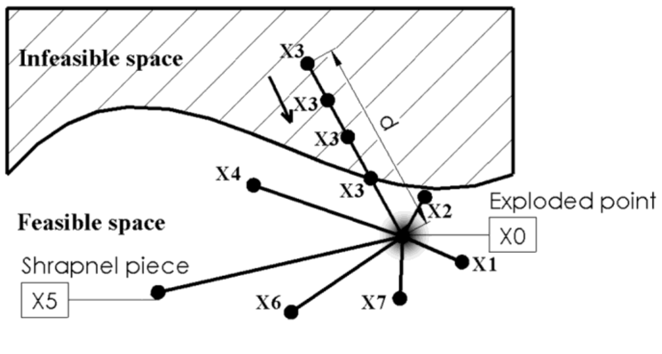


Figure 3.2. Schematic view of constraint handling approach using the proposed method.

3.5. Convergence criteria

For termination criteria, as commonly considered in metaheuristic algorithms, the best result is calculated where the termination condition may be assumed as the maximum number of iterations, CPU time, or ε which is a small value and is defined as an allowable tolerance between the last two results. The MBA proceeds until the above convergence criteria are satisfied.

3.6. Steps and flowchart of MBA

The steps of MBA are summarized as follows:

Step 1: Choose the initial parameters of MBA.

Step 2: Check the condition of exploration factor.

Step 3: If condition of exploration factor is satisfied, calculate the distance of shrapnel pieces and their locations according to Equations (3.7) and (3.8), respectively, and go Step 11. Otherwise, go to Step 4.

Step 4: Calculate the distance of shrapnel pieces and their locations using Equations (3.4) and (3.5).

Step 5: Calculate the direction of shrapnel pieces according to Equation (3.6).

Step 6: Generate the shrapnel pieces and compute their improved locations using Equation (3.3).

Step 7: Check the constraints for generated shrapnel pieces.

Step 8: Save the best shrapnel piece as the best temporal solution.

Step 9: Does the shrapnel piece have the lower function value than the best temporal solution?

Step 10: If true, exchange the position of the shrapnel piece with the best temporal solution.

Step 11: Reduce the distance of the shrapnel pieces adaptively using Equation (3.9).

Step 12: Check the convergence criteria. If the stopping criterion is satisfied, the algorithm will be stopped. Otherwise, return to Step 2.

Figure 3.3 demonstrates the steps of the MBA in form of flowchart. In summary, in this chapter, the detailed explanations and formulations of the proposed method were provided. In addition, setting parameters of the MBA were investigated in this chapter. In summary, in this chapter, the detailed explanations, formulations, and steps of the proposed method were represented.

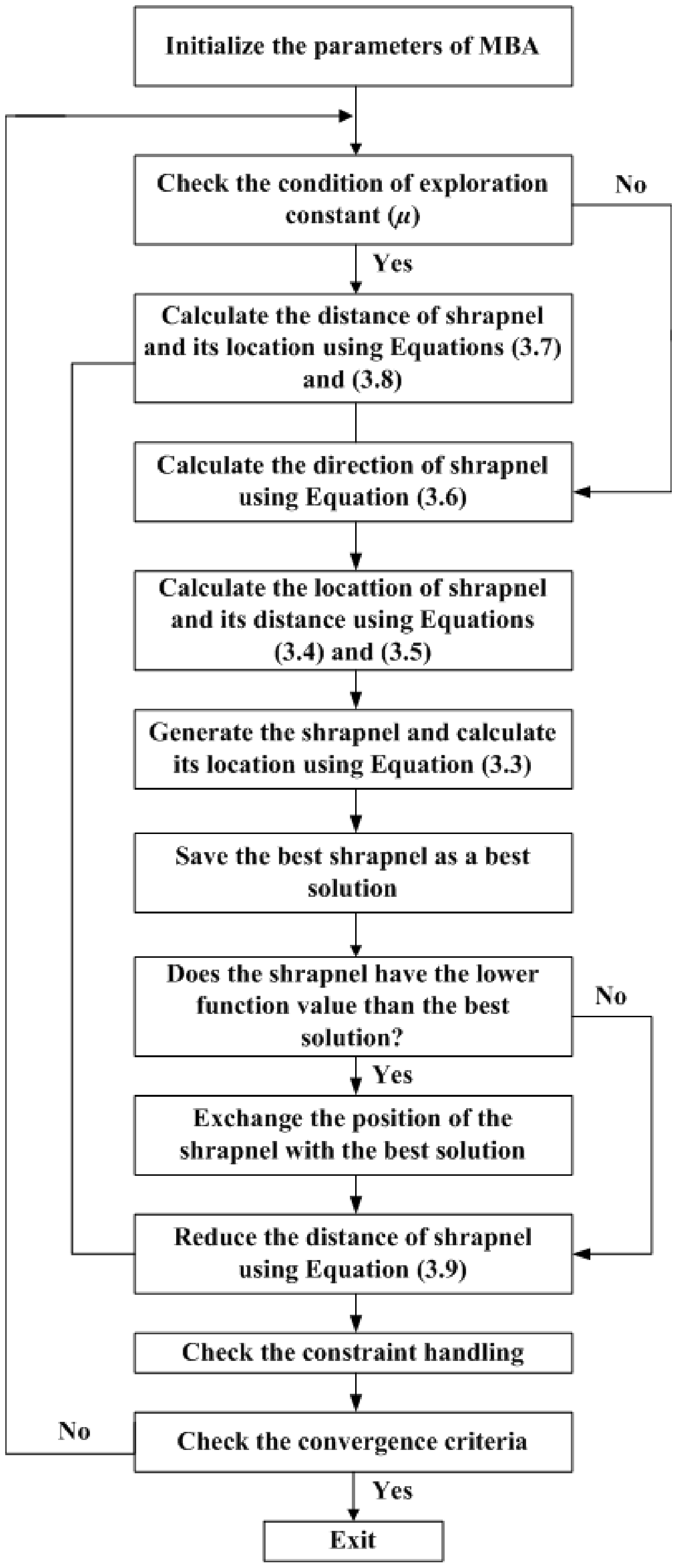

Initialize the parameters of MBA
Check the condition of exploration constant (μ)
No
Yes
Calculate the distance of shrapnel and its location using Equations (3.7) and (3.8)
Calculate the direction of shrapnel using Equation (3.6)
Calculate the locattion of shrapnel and its distance using Equations (3.4) and (3.5)
Generate the shrapnel and calculate its location using Equation (3.3)
Save the best shrapnel as a best solution
Does the shrapnel have the lower function value than the best solution?
No
Yes
Exchange the position of the shrapnel with the best solution
Reduce the distance of shrapnel using Equation (3.9)
Check the constraint handling
No
Check the convergence criteria
Yes
Exit

Figure 3.3. Flowchart of the proposed MBA.

CHAPTER 4 : WATER CYCLE ALGORITHM

4.1. Basic concepts

The idea of the water cycle algorithm (WCA) is inspired from nature and based on the observation of water cycle and how rivers and streams flow downhill towards the sea in the real world. To understand this further, an explanation on the basics of how rivers are created and water travels down to the sea is given as follows.

A river, or a stream, is formed whenever water moves downhill from one place to another. This means that most rivers are formed high up in the mountains, where snow from the winter, or ancient glaciers, melt. The rivers always flow downhill. On their downhill journey and eventually ending up to a sea, water is collected from rain and other streams.

Figure 4.1 is a simplified diagram for part of the hydrologic cycle. Water in rivers and lakes is evaporated while plants give off (transpire) water during photosynthesis. The evaporated water is carried into the atmosphere to generate clouds which then condenses in the colder atmosphere, releasing the water back to the earth in the form of rain or precipitation. This process is called the hydrologic cycle (water cycle) (David, 1993).

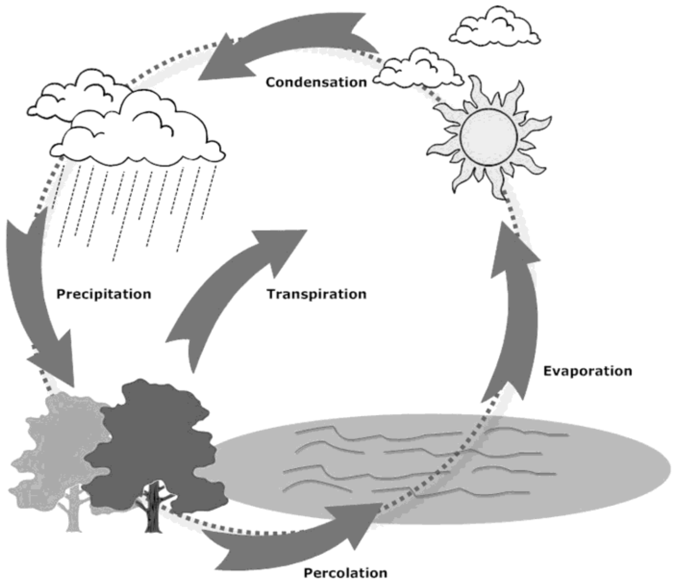


Figure 4.1. Simplified diagram of the hydrologic cycle (water cycle).

In the real world, as snow melts and rain falls, most of water enters the aquifer. There are vast fields of water reserves underground. The aquifer is sometimes called groundwater (see percolation arrow in Figure 4.1). The water in the aquifer then flows beneath the land the same way water would flow on the ground surface (downward).

The underground water may be discharged into a stream (marsh or lake). Water evaporates from the streams and rivers, in addition to being transpired from the trees and other greenery, hence, bringing more clouds and thus more rain as this cycle counties (David, 1993).

Figure 4.2 is a schematic diagram of how streams flow to the rivers and rivers flow to the sea. Figure 4.2 resembles a tree or roots of a tree. The smallest river branches, (twigs of tree shaped figure in Figure 4.2 shown in bright green), are the small streams where the rivers begins to form. These tiny streams are called first-order streams (shown in Figure 4.2 in green colors).

Wherever two first-order streams join, they make a second-order stream (shown in Figure 4.2 in white colors). Where two second-order streams join, a third-order stream is formed (shown in Figure 4.2 in blue colors), and so on until the rivers finally flow out into the sea (the most downhill place in the assumed world) (Strahler, 1952).

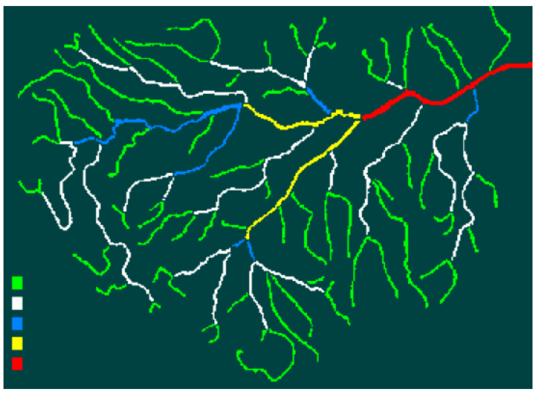


Figure 4.2. Schematic diagram of how streams flow to the rivers and also rivers flow to the sea.

Figure 4.3 shows the Arkhangelsk city on the Dvina River. Arkhangelsk (Archangel in English) is a city in Russia that drapes both banks of the Dvina River, near where it flows into the White Sea. A typical real life stream, river, sea formation (Dvina River) is shown in Figure 4.3 resembling the shape in Figure 4.2.

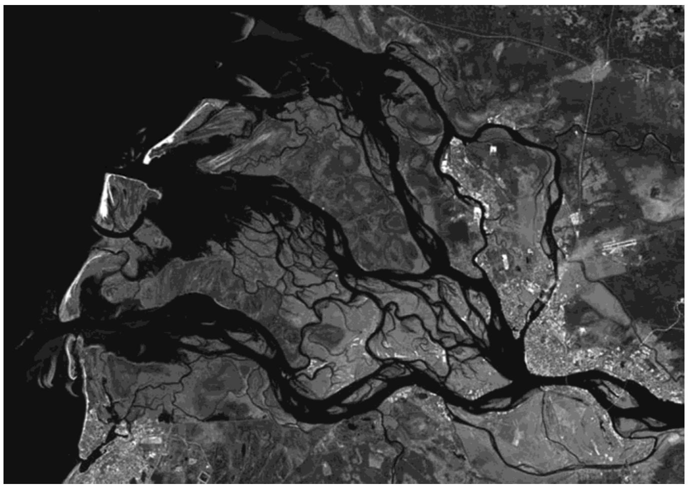


Figure 4.3. Arkhangelsk city on the Dvina River (adopted from NASA, Image Source: http://asterweb.jpl.nasa.gov/gallery-detail.asp?name=Arkhangelsk).

4.2. Proposed WCA

Similar to other metaheuristic algorithms, the proposed method begins with an initial population so called the raindrops. First, we assume that we have rain or precipitation. The best individual (best raindrop) is chosen as a sea. Then, a number of good raindrops are chosen as a river and the rest of the raindrops are considered as streams which flow to the rivers and sea.

Depending on their magnitude of flow which will be described in the following subsections, each river absorbs water from the streams. In fact, the amount of water in a stream entering a rivers and/or sea varies from other streams. In addition, rivers flow to the sea which is the most downhill location.

4.2.1. Create initial population

In order to solve an optimization problem using population-based metaheuristic methods, it is necessary that the values of problem variables be formed as an array. In the GA and PSO terminologies such array is called "Chromosome" and "Particle Position", respectively. Accordingly, in the proposed method it is called "Raindrop" for a single solution. In a N_{var} dimensional optimization problem, an raindrop is an array of $1 \times N_{var}$. This array is defined as follows:

$$Raindrop = [x_1, x_2, x_3, ..., x_N] \qquad (4.1)$$

To start the optimization algorithm, a candidate representing a matrix of raindrops of size $N_{pop} \times N_{var}$ is generated (i.e. population of raindrops). Hence, the matrix X which is generated randomly is given as (rows and column are the number of population and the number of design variables, respectively):

$$Population \quad of \quad raindrops = \begin{bmatrix} Raindrop_1 \\ Raindrop_2 \\ Raindrop_3 \\ \vdots \\ Raindrop_{N_{pop}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_{N_{\text{var}}}^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_{N_{\text{var}}}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{N_{pop}} & x_2^{N_{pop}} & x_3^{N_{pop}} & \cdots & x_{N_{\text{var}}}^{N_{pop}} \end{bmatrix}$$

(4.2)

Each of the decision variable values (x_1, x_2, . . . , x_{Nvar}) can be represented as floating point number (real values) or as a predefined set for continuous and discrete problems, respectively. The cost of a raindrop is obtained by the evaluation of cost function (C) given as:

$$C_i = Cost_i = f(x_1^i, x_2^i, \cdots, x_{N_{\text{var}}}^i) \qquad i = 1, 2, 3, ..., N_{pop} \qquad (4.3)$$

where N_{pop} and N_{vars} are the number of raindrops (initial population) and the number of design variables, respectively. For the first step, N_{pop} raindrops are created. A number of N_{sr} from the best individuals (minimum values) are selected as sea and rivers. The raindrop which has the minimum value among others is considered as a sea.

In fact, N_{sr} is the summation of Number of Rivers (which is a user parameter) and a single sea as given in Equation (4.4). The rest of the population (raindrops form the streams which flow to the rivers or may directly flow to the sea) is calculated using Equation (4.5).

$$N_{sr} = Number\ of\ Rivers + \underbrace{\quad 1 \quad}_{Sea} \tag{4.4}$$

$$N_{Raindrops} = N_{pop} - N_{sr} \tag{4.5}$$

In order to designate/assign raindrops to the rivers and sea depending on the intensity of the flow, the following equation is given:

$$NS_n = round\{\left|\frac{Cost_n}{\sum_{i=1}^{N_{sr}} Cost_i}\right| \times N_{Raindrops}\} \quad , \quad n = 1, 2, ..., N_{sr} \tag{4.6}$$

where NS_n is the number of streams which flow to the specific rivers or sea.

4.2.2. How does a stream flow to the rivers or sea?

As mentioned in Subsection 4.1 in this chapter, the streams are created from the raindrops and join each other to form new rivers. Some of the streams may also flow directly to the sea. All rivers and streams end up in sea (best optimal point). Figure 4.4 shows the schematic view of stream's flow towards a specific river where star and circle represent river and stream, respectively.

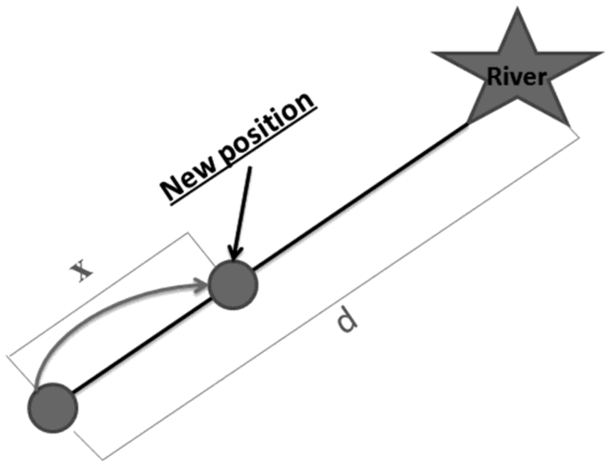


Figure 4.4. Schematic view of stream's flow to a specific river.

As illustrated in Figure 4.4, a stream flows to the river along the connecting line between them using a randomly chosen distance given as follow:

$$X \in (0, C \times d), \qquad C > 1 \tag{4.7}$$

where C is a value between 1 and 2 (near to 2). The best value for C may be chosen as 2. The current distance between stream and river is represented as d. Indeed, same as the PSO, the value of C was chosen using some practical and experimental execution of algorithm (try and error process). The value of X in Equation (4.7) corresponds to a distributed random number (uniformly or may be any appropriate distribution) between 0 and (C×d).

The value of C being greater than one enables streams to flow in different directions towards the rivers. This concept may also be used in flowing rivers to the sea. Therefore, the new position for streams and rivers may be given as:

$$X_{Stream}^{i+1} = X_{Stream}^{i} + rand \times C \times (X_{River}^{i} - X_{Stream}^{i}) \tag{4.8}$$

$$X_{River}^{i+1} = X_{River}^{i} + rand \times C \times (X_{Sea}^{i} - X_{River}^{i}) \tag{4.9}$$

where *rand* is a uniformly distributed random number between 0 and 1. Based on our experiments, uniform random numbers more than 1 lead to diverge motion of rivers to the sea. If the solution given by a stream is better than its connecting river, the positions of river and stream are exchanged (i.e. stream becomes river and river becomes stream). Such exchange can similarly happen for rivers and sea.

Figure 4.5 depicts the exchange of a stream which is the best solution among other streams and the river. From Figure 4.5, star represents river and black color circle shows the best stream among other streams.

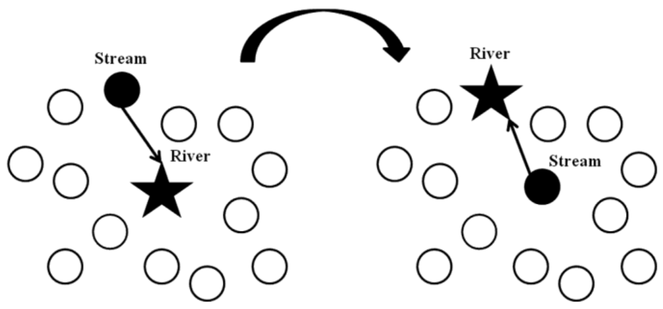


Figure 4.5. Exchanging the positions of the stream and the river.

4.2.3. Evaporation condition

Evaporation is one of the most important factors that can prevent the algorithm from rapid convergence (immature convergence). As can be seen in nature, water evaporates from rivers and lakes while plants give off (transpire) water during photosynthesis.

The evaporated water is carried into the atmosphere to form clouds which then condenses in the colder atmosphere, releasing the water back to the earth in the form of rain. The rain creates the new streams and the new streams flow to the new rivers which flow to the sea. This cycle which was mentioned in Subsection 4.1 is called water cycle.

In the proposed method, the evaporation process causes the sea water to evaporate as rivers/streams flow to the sea. This assumption is proposed in order to avoid getting trapped in local optima. The following Psuocode shows how to determine whether or not river flows to the sea.

```
if $\left| X^{i}_{Sea} - X^{i}_{River} \right| < d_{\max}$        $i = 1, 2, 3, ..., N_{sr} - 1$
    Evaporation and raining process
end
```

where d_{max} is a small number (close to zero). Therefore, if the distance between a river and sea is less than d_{max}, it indicates that the river has reached/joined the sea. In this situation, the evaporation process is applied and as seen in the nature after some adequate evaporation the raining (precipitation) will start.

A large value for d_{max} reduces the search while, a small value encourages the search intensity near the sea. Therefore, d_{max} controls the search intensity near the sea (the optimum solution). The value of d_{max} adaptively decreases as:

$$d_{\max}^{i+1} = d_{\max}^{i} - \frac{d_{\max}^{i}}{\max\, iteration} \qquad (4.10)$$

To further clarify, in order to converge to an optimal point, distance between river and sea should be decreased at each iteration based on the Equation (4.10). This action helps to cover the exploitation process used in the WCA in addition of convergence purposes.

4.2.4. Raining process

After satisfying the evaporation process, the raining process is applied. In the raining process, the new raindrops form streams in the different locations (acting similar to mutation operator in the GA). For specifying the new locations of the newly formed streams, the following equation is used:

$$X_{Stream}^{new} = LB + rand \times (UB - LB) \qquad (4.11)$$

where LB and UB are lower and upper bounds defined by the given problem, respectively. Again, the best newly formed raindrop is considered as a river flowing to the sea. The rest of new raindrops are assumed to form new streams which flow to the rivers or may directly flow to the sea.

In order to enhance the convergence rate and computational performance of the algorithm for constrained problems, Equation (4.12) is used only for the streams which directly flow to the sea. This equation aims to encourage the generation of streams which directly flow to the sea in order to improve the exploration near sea (the optimum solution) in the feasible region for constrained problems.

$$\mathrm{X}_{\mathrm{stream}}^{\mathrm{new}} = \mathrm{X}_{\mathrm{sea}} + \sqrt{\mu} \times \mathrm{randn}(1, \mathrm{N}_{\mathrm{var}}) \qquad (4.12)$$

where μ is a coefficient which shows the range of searching region near the sea. *Randn* is the normally distributed random number. The larger value for μ increases the possibility to exit from feasible region. On the other hand, the smaller value for μ leads the algorithm to search in smaller region near the sea. A suitable value for μ is set to 0.1.

In mathematical point of view, the term $\sqrt{\mu}$ in Equation (4.12) represents the standard deviation and, accordingly, μ defines the concept of variance. Using these concepts, the generated individuals with variance μ are distributed around the best obtained optimum point (sea).

4.3. Constraint handling approach

In the search domain, streams and rivers may exceed the constraints of given problem or may violate from upper and lower bounds of design variables. In the current study, a modified feasible-based method is utilized to overcome the constraints which the rules are given as follows (Montes & Coello, 2008):

- Rule 1: Any obtained feasible solution is fancied to any infeasible solution.
- Rule 2: Infeasible solutions having minor violation of the constraints (from 0.01 in the first iteration to 0.001 in the final iteration) are assumed as feasible solutions.
- Rule 3: Between two feasible solutions, the one having the improved objective function value is more interested.
- Rule 4: Between two infeasible solutions, the one with the smaller sum of constraint violation is chosen.

By using the 1th and 4th rules, the search is oriented to the feasible area rather than to the infeasible region, and by applying the 3th rule the search is directed to the feasible district having high quality results (Montes & Coello, 2008). For most structural optimization problems, the global minimum locates on or close to the boundary of a feasible design space. By applying Rule 2, the streams and rivers approach the boundaries and can reach the global minimum with a higher probability (Kaveh & Talatahari, 2009b).

4.4. Convergence criteria

For termination criteria, as commonly considered in metaheuristic algorithms, the best result is calculated where the termination condition may be assumed as the maximum number of iterations, CPU time, or ε which is a small non-negative value and is defined as an allowable tolerance between the last two results. The WCA proceeds until the maximum number of iterations as a convergence criterion is satisfied.

4.5. Steps and flowchart of WCA

The steps of WCA are summarized as follows:

Step 1: Choose the initial parameters of the WCA: N_{sr}, d_{max}, N_{pop}, *max_iteration*.

Step 2: Generate random initial population and form the initial streams (raindrops), rivers, and sea using Equations (4.2), (4.4), and (4.5).

Step 3: Calculate the value (cost) of each raindrops using Equation (4.3).

Step 4: Determine the intensity of flow for rivers and sea using Equation (4.6).

Step 5: The streams flow to the rivers by Equation (4.8).

Step 6: The rivers flow to the sea which is the most downhill place using Equation (4.9).

Step 7: Exchange positions of river with a stream which gives the best solution, as shown in Figure 4.5.

Step 8: Similar to Step 7, if a river finds better solution than the sea, the position of river is exchanged with the sea (see Figure 4.5).

Step 9: Check the evaporation condition using the Psuocode in Subsection 4.2.3.

Step 10: If the evaporation condition is satisfied, the raining process will occur using Equations (4.11) and (4.12).

Step 11: Reduce the value of d_{max} which is user defined parameter using Equation (4.10).

Step 12: Check the convergence criteria. If the stopping criterion is satisfied, the algorithm will be stopped, otherwise return to Step 5.

The schematic view of the proposed method is illustrated in Figure 4.6 where circles, stars, and the diamond correspond to streams, rivers, and sea, respectively. From Figure 4.6, the white (empty) shapes refer to the new positions found by streams and rivers. In fact, Figure 4.6 is an extension of Figure 4.4.

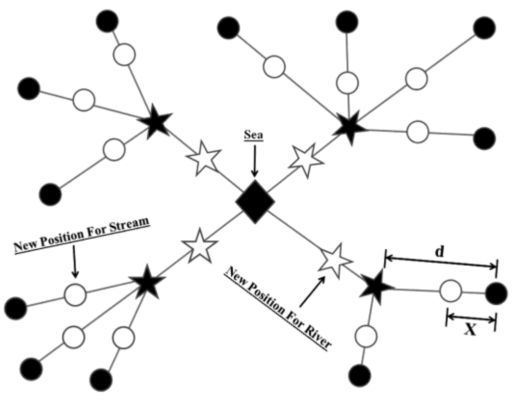


Figure 4.6. Schematic view of WCA.

The procedure for the proposed WCA is shown in Figure 4.7 in the form of a flowchart. In summary, in this chapter, the detailed explanations, formulations, and steps of the proposed method were represented.

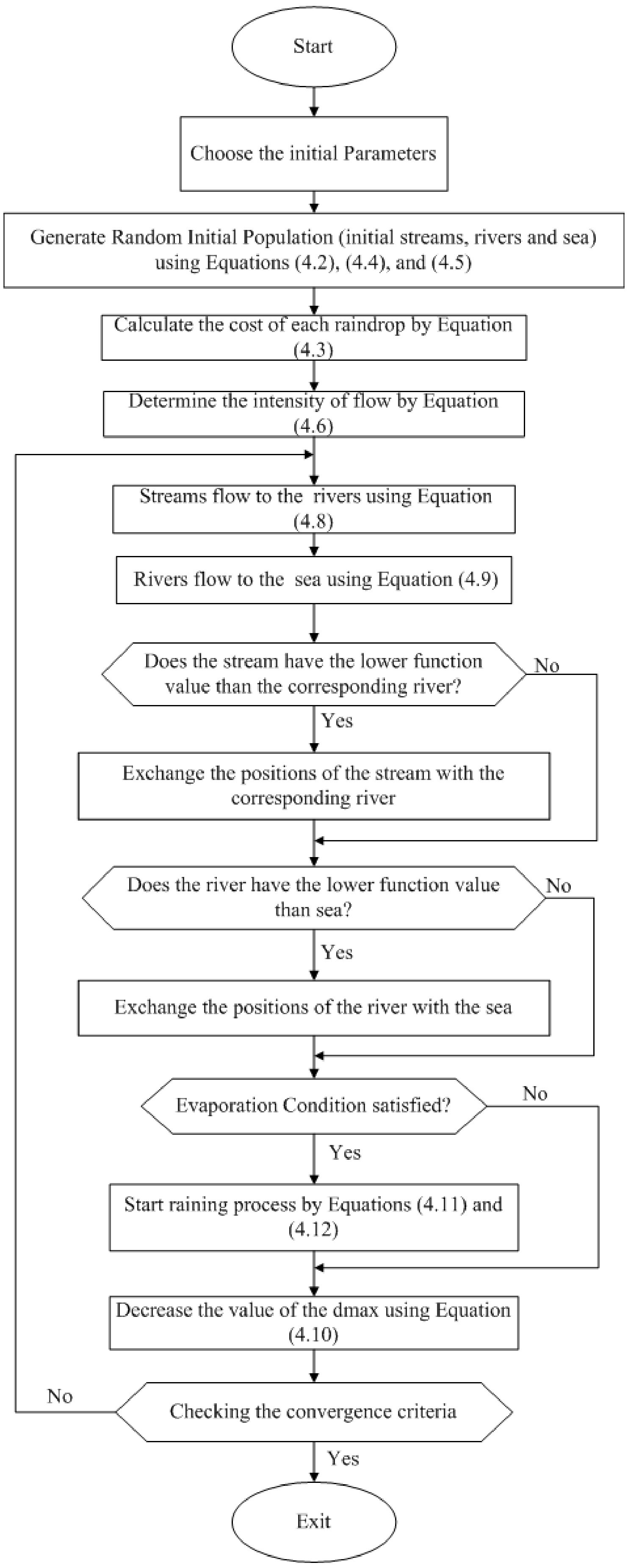

Start
Choose the initial Parameters
Generate Random Initial Population (initial streams, rivers and sea) using Equations (4.2), (4.4), and (4.5)
Calculate the cost of each raindrop by Equation (4.3)
Determine the intensity of flow by Equation (4.6)
Streams flow to the rivers using Equation (4.8)
Rivers flow to the sea using Equation (4.9)
Does the stream have the lower function value than the corresponding river?
No
Yes
Exchange the positions of the stream with the corresponding river
Does the river have the lower function value than sea?
No
Yes
Exchange the positions of the river with the sea
Evaporation Condition satisfied?
No
Yes
Start raining process by Equations (4.11) and (4.12)
Decrease the value of the dmax using Equation (4.10)
No
Checking the convergence criteria
Yes
Exit

Figure 4.7. Flowchart of the proposed WCA.

CHAPTER 5 : VALIDATION OF PROPOSED METHODS

In this chapter, at first, similarities and differences of the proposed methods (MBA and WCA) with other existing and similar metaheuristic algorithms such as PSO (Kennedy & Eberhart, 1995), grenade explosion method (GEM) (Ahrari & Aatai, 2010), water flow algorithm (WFA) (Hieu, 2011), water cycle-like algorithm (WCA) (Zhi-ding & Jie-Kang, 2011), and intelligent water drops (IWD) algorithm (Shah-Hosseini, 2009) are provided in details accompanied with descriptions of their processes.

Afterwards, validation and verification of proposed optimizers are carried out using various types of unconstrained, constrained, and engineering design problems. Comprehensive comparisons are conducted for evaluating the efficiency and performance of the MBA and WCA. Unfortunately, the above methods (except the GEM and PSO) were not applied for the reported problems in this thesis. There are not existing records or publications of applications for these methods for truss structures, constrained, and engineering design problems so far.

5.1. Differences among proposed optimizers with other existing methods

It can be a good question, what are the similarities and differences between the MBA and WCA? The only similarity between the MBA and WCA is that both proposed optimizers are population based methods. Except this similarity, all factors and operators and even their concepts are different. The MBA's concept is from explosion of mine bombs, while the WCA's ideas are inspired from water cycle process and how streams and rivers flow to the sea in nature.

Regarding the constraint handling approach, it is worth to mention that, the approach, given in subsections 3.4 and 4.3, is widely used strategy for controlling constraint violation. It can be implemented for many metaheuristic methods. It cannot count as similarity between the MBA and WCA in terms of concept and performance. In fact, there are many ways for tackling constraint handling in the literature. We used these four rules (known as direct method and given in subsections 3.4 and 4.3) instead of, for example, penalty function approach.

Talking about differences between the WCA and PSO, the updating formulation for positions of rivers and streams are different from given by the PSO (finding and updating the local and global best positions). By observing carefully, we did not use the concept of moving directly to the best solution (global best). In fact, we utilized the concept of moving indirectly from stream to the rivers and from rivers to the sea which is the best temporal obtained optimum point.

In the proposed WCA, rivers (a number of best selected points except the best one (sea), Equation (4.4)) act as guidance points for conducting other individual of populations toward the better positions (see Figure 4.6) and avoid to search on inappropriate regions in near-optimum solutions (Equation (4.8)).

It is worth to mention that rivers, themselves, move toward to the sea (the best solution). They are not fixed points (Equation (4.9)). In fact, this procedure (moving streams to the rivers and, then moving rivers to the sea) leads to indirect move toward the best solution. In other hand, in the PSO, individuals (particles) based on the personal and best experiences find the best solution and the searching approach is moving directly to the best optimal solution.

To mention another distinguish between the WCA and PSO is the existing of evaporation and raining conditions which is act as mutation operator. The evaporation and raining conditions cause to release the proposed algorithm from getting trapped in local optimum solutions, while in the PSO there was not defined such a mechanism.

Talking about the differences between the MBA and GEM, in the MBA, we have different approach for finding an optimal point compared with the GEM. To mention of them, reduction constant and exploration factor, which simulate exploitation and exploration steps for the MBA. The distance of each shrapnel pieces is calculated using Euclidean distance in 2D space and also we have the concept of direction for each shrapnel pieces.

One of the theories of GEM is the agent's territory radius (R_t), which means an agent (in the GEM agents are grenades) does not let other agents come closer than a definite distance, which is specified by R_t. In addition, there is a concept which is determined the intensity of exploration process. In the GEM, L_e is the length of explosion along each coordinate, in which the thrown piece of shrapnel may destruct the objects. The values of R_t and L_e are decreased adaptively in each iteration, while the reduction rate of L_e is slower than R_t for exploration purposes.

As it can be seen, the MBA and GEM are in common only in the basis of explosion concept. It means their ideas of explosion for creating an initial population are the same. In fact, both optimizers (MBA and GEM) are population based methods (population of shrapnel pieces). However, the strategy of MBA to approach towards a global optimum point is totally different. The MBA uses different formulations and strategies to reach its best optimal point.

The MBA does not have radius territory and intensity of exploration operators same as the GEM. In the MBA the new positions of shrapnel pieces calculate using updating formula which is totally different with those given in the GEM. In the MBA, two special operators are defined which do not exist in the GEM: Reduction constant and exploration factor.

The number of initial parameters in the MBA is quit less than those offered by the GEM. In the GEM, besides of common initial parameters for metaheuristic algorithms (i.e. population size and maximum number of iteration), the following values should be selected as user parameters: m_{max}, m_{min}, T_w, N_q, $L_{e\text{-}initial}$, $R_{t\text{-}initial}$, and R_{rd}.

In contrast, for the MBA (except of common user parameters for metaheuristic algorithms), the number of user parameters is comparatively less than the GEM which are reduction constant (α) and exploration factor (μ).

The water flow algorithm (WFA) is inspired by the hydrological cycle in meteorology and the erosion phenomenon in nature. The WFA combines the amount of precipitation and its falling force to form a flexible erosion capability. This helps the erosion process of the algorithm to focus on exploiting promising regions strongly (Hiew, 2011).

In fact, the idea of WCA and WFA is similar to each other inspiring from water cycle process in nature. However, the first idea of WCA was based on how streams and rivers flow to the sea and formulations of WCA are different from those given by the WFA.

By observing the WCA, it focuses on the motion of streams and rivers to the sea and the evaporation condition and updating formula are fully diverse with those suggested in the WFA. In the WFA, the concept of erosion and falling force of raindrops are considered as another differences between the WCA and WFA.

The concept of water cycle-like algorithm (WCA) proposed by Zhi-ding and Jie-Kang (2011) has the same concept given in the WCA by Eskandar et al. (2012). However, for searching mechanism, they utilized the idea of relative gravity of waters to guide particles towards better solutions. It is worth mentioning that the WCA offered in the literature modeled the concepts of confluence, infiltration, and total force which are completely dissimilar with the suggested model in the WCA proposed by Eskandar et al. (2012).

Another similar method to the WCA in terms of concept is intelligent water drops (IWD) algorithm. The IWD algorithm is a swarm-based nature-inspired optimization algorithm (Shah-Hosseini, 2009). This algorithm contains a few essential elements of natural water drops and actions and reactions that occur between river's bed and the water drops that flow within.

The IWD consists of two parts: a graph that plays the role of distributed memory on which soils of different edges are preserved, and the moving part of the IWD algorithm, which is a few number of intelligent water drops. These intelligent water drops (IWDs) both compete and cooperate to find better solutions and by changing soils of the graph, the paths to better solutions become more reachable.

It is mentioned that the IWD needs at least two IWDs to launch. By carefully looking at the processes of the IWD algorithm, one can be seen that the concepts of the WCA and IWD are not the same. The only similarity between the WCA and IWD is using the water drops agent in their populations.

To further clarify, the IWD uses the concepts of soil removal while water drops moving to the rivers, also it gains some velocity and removes some soil from the path it flows on which are totally different with the suggested formulations and concepts offered by the WCA.

It is common to see an algorithm reaches the best solution for some problems and in contrast, for some problems it cannot detect the best optimum point. This is happen for all metaheuristic approaches such as the GA, PSO, SA, and so forth.

For instance, the SA is a suitable optimizer for tackling combinatorial optimization problems, while the PSO performs well for continuous problems. In fact, depends on the nature of a problem being solved, the performance of optimizers may differ from each other.

As for the MBA and WCA, they outperformed other considered optimizers in terms of less computational time and solution accuracy, while for some problems their results were not counted as first ranked solution. In fact, for some problems, they have placed in 2nd rank.

However, in general, for most problems in this thesis, obtained optimization results offered by the proposed optimizers have surpassed other methods. The reason may be depended on their strategies moving to the global optimum point which they can search domain solution better than others using their exploration and exploitation operators.

Regarding the parameter setting, based on our experiments and practical executions, we offered default values for solving problems using the MBA and WCA. If optimization results were not satisfactory, a user would change the default values based on the parameter setting guidance given in this thesis.

It is worth mentioning that the number of initial parameters is comparatively less for the proposed optimizers (2 for the MBA and WCA). The difficulties for tuning the initial parameters are in their minimum level for both methods.

5.2. Unconstrained benchmark problems

The proposed optimizers were implemented in MATLAB programming software and run on Pentium IV, 2500 GHz CPU having 4GB RAM. For validating of the proposed methods, the following criteria were considered in this chapter and the results are shown in tables and figures:

- Comparing WCA and MBA with other optimizers with respect to the number of function evaluations (NFEs) and best function value.
- Finding the global minimum among many local minima.

In the following subsection, various standard unconstrained benchmark function minimization problems have been presented to demonstrate the efficiency and robustness of the proposed algorithms and the obtained results were compared with the results obtained using other efficient optimizers. These examples include 16 well-known unconstrained benchmark functions.

The task of optimizing each of the test functions was executed in 50 independent runs. Different number of iterations was used for different types of benchmark problems. All benchmark functions used in this chapter are given in Appendix *A*.

5.2.1. NFEs and best function value criteria

The number of function evaluations (NFEs) determine the speed (computational time) and the robustness of the algorithm (robustness means fast convergence rate and having the best solution quality). Less NFEs means spending less time to reach the global optimum. This feature returns back to the structure of the algorithm. The best solution represents the accuracy of the method. The NFEs and best solution are dependent on each other. The ideal situation is the less NFEs and more accurate solution.

Table 5.1 presents specifications of seven benchmark functions. For benchmark functions in Table 5.1, the optimization process terminates when the difference between the maximum fitness obtained and the global optimum value is less than 0.1% of the optimum value, or less than 0.001, whichever is smaller. In case the optimum value is zero, the solution is accepted if it differs from the optimum value by less than 0.001 (Pham et al., 2006).

Table 5.1: Specifications of seven unconstrained benchmark functions presented in (Pham et al., 2006; Ahrari et al., 2010). "*N*" stands for the number of design variables.

No.	**Functions**	***N***	**Interval**
1	De Jong	2	$[-2.048,2.048]^N$
2	Goldstein and Price I	2	$[-2,2]^N$
3	Branin	2	$[-5,10]^N$
4	Martin and Gaddy	2	$[0,10]^N$
5a	Rosenbrock	2	$[-1.2,1.2]^N$
5b	Rosenbrock	2	$[-10,10]^N$
5c	Rosenbrock	4	$[-1.2,1.2]^N$
6	Hyper sphere	6	$[-5.12,5.12]^N$
7	Shaffer	2	$[-100,100]^N$

Figure 5.1 shows the surface plot and contour lines for seven benchmark functions given in Table 5.1. Tables 5.2 and 5.3 represent the values which were chosen for parameters used in the MBA and WCA, respectively. Tables 5.4 and 5.5 show the statistical results including worst, mean, best solution, and standard deviation (SD) for seven unconstrained benchmark functions for the MBA and WCA, respectively.

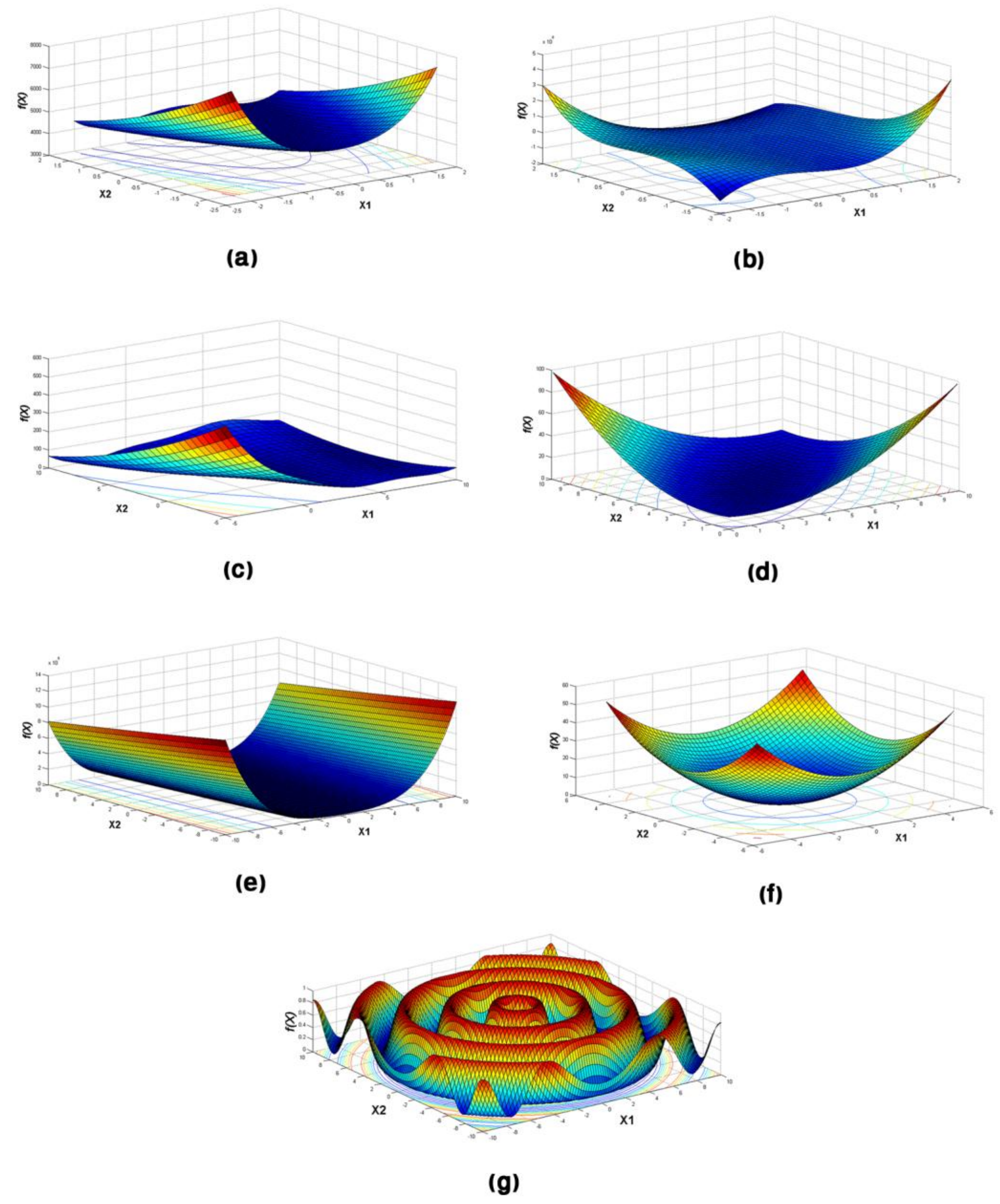


Figure 5.1. Surface plot and contour lines for seven benchmark functions presented in Table 5.1: (a) De Jong, (b) Goldstein and Price I, (c) Branin, (d) Martin and Gaddy, (e) Rosenbrock, (f) Hyper Sphere, (g) Shaffer.

Table 5.2: Initial parameters used for optimization of seven unconstrained benchmark functions using the MBA presented in Table 5.1.

No.	N_s	μ	α	Max_Iteration
1	10	0	500	200
2	10	0	500	100
3	10	0	500	100
4	5	0	500	50
5a	5	0	500	100
5b	10	0	500	500
5c	50	3	500	400
6	10	0	500	100
7	50	0	1000	1000

Table 5.3: User parameters used for optimization of seven unconstrained benchmark functions presented in Table 57.1 using the WCA.

No.	N_{total}	N_{sr}	d_{max}	Max_Iteration
1	10	3	0.01	200
2	10	3	0.01	200
3	10	3	0.01	100
4	5	2	0.01	50
5a	5	2	0.01	100
5b	10	3	0.01	500
5c	50	4	0.01	50000
6	10	3	0.01	100
7	50	4	0.01	1000

Table 5.4: Statistical results of 50 independent runs for seven unconstrained benchmark functions in Table 5.1 using the MBA.

No.	Worst	Mean	Best	SD	Optimum
1	3905.949023	3905.932168	3905.930000	4.45E-03	3905.93
2	3.000126	3.000032	3.0000009	3.45E-05	3
3	0.401670	0.397915	0.3977272	7.86E-04	0.3977272
4	2.27E-03	7.62E-04	3.68E-05	6.37E-04	0
5a	0.102756	0.011318	7.19E-08	2.41E-02	0
5b	7.63E-01	4.68E-02	9.75E-07	0.163767	0
5c1	7.599E-03	1.979E-03	1.21E-06	2.23E-03	0
5c2	3.78E-02	2.89E-03	1.16E-06	9.22E-03	0
6	4.093E-03	9.29E-04	1.34E-05	1.10E-03	0
7	9.715E-03	7.383E-03	1.08E-10	4.234E-03	0

Table 5.5: Statistical results for seven unconstrained benchmark functions given in Table 5.1 using the WCA.

No.	Worst	Mean	Best	SD	Optimum
1	3906.121239	3905.940137	3905.930000	3.82E-02	3905.93
2	3.000968	3.000561	3.000020	2.55E-04	3
3	0.398717	0.398272	0.397731	3.20E-04	0.397727
4	0.000929	0.000416	0.000005	3.11E-04	0
5a	0.014634	0.001345	0.000028	3.11E-03	0
5b	0.000986	0.000432	0.000001	3.20E-04	0
5c	0.000798	0.000212	0.000000	2.29E-04	0
6	0.009223	0.000600	0.000000	1.81E-03	0
7	0.009715	0.001167	0.000026	2.58E-03	0

Tables 5.6 and 5.7 present the results obtained by proposed optimizers and those using deterministic Simplex method (SIMPSA), Stochastic Simulated Annealing optimization procedure (NE-SIMPSA), Genetic Algorithm (GA), Ant Colony System (ACS), Artificial Bees Colony (ABC) (Pham et al., 2006), and Grenade Explosion Method (GEM) (Ahrari & Aatai, 2010). Optimization results for all optimizers except the WCA and MBA were directly driven from (Ahrari & Aatai, 2010; Pham et al., 2006; Ahrari et al., 2010).

The best NFEs in each case has been highlighted in bold as shown in Table 5.7. From Tables 5.6 and 5.7, MNFEs stands for mean number of function evaluations and the Success criterion is in percentage.

Table 5.6: Comparison of results for optimization of seven unconstrained benchmark functions presented in Table 5.1. “N/A” means not available.

No.	SIMPSA		NE-SIMPSA		GA		ACS	
	Succ.	**MNFEs**	**Succ.**	**MNFEs**	**Succ.**	**MNFEs**	**Succ.**	**MNFEs**
1	N/A	N/A	N/A	N/A	100	10,160	100	6000
2	N/A	N/A	N/A	N/A	100	5662	100	5330
3	N/A	N/A	N/A	N/A	100	7325	100	1936
4	N/A	N/A	N/A	N/A	100	2488	100	1688
5a	100	10,780	100	4508	100	10,212	100	6842
5b	100	12,500	100	5007	N/A	N/A	100	7505
5c	99	21,177	94	3053	N/A	N/A	100	8471
6	N/A	N/A	N/A	N/A	100	15,468	100	22,050
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 5.7: Comparison of results for optimization of seven unconstrained benchmark functions presented in Table 5.1.

No.	ABC		GEM		WCA		MBA	
	Succ.	**MNFEs**	**Succ.**	**MNFEs**	**Succ.**	**MNFEs**	**Succ.**	**MNFEs**
1	100	868	100	746	100	684	100	**620**
2	100	999	100	701	100	980	100	**440**
3	100	1657	100	689	100	**377**	100	430
4	100	526	100	258	100	**57**	100	100
5a	100	631	100	572	100	**174**	100	245
5b	100	2306	100	2289	100	**623**	100	830
5c	100	28,529	100	82,188	100	**266**	100	3700
6	100	7113	100	423	100	**101**	100	370
7	100	8456	100	9481	100	8942	100	**6950**

The comparison of results shown in Tables 5.6 and 5.7 reveals that the WCA and MBA have found the global minimum with the desired accuracy faster than (Less NFEs) other optimization engines. This superiority is more evident for functions 3 to 6. By observing Tables 5.6 and 5.7, only the MBA and WCA can compete with each other in terms of NFEs.

Also, Learning Algorithm (LA) given in (Zhao et al., 2009) solved function 7 in Table 5.6 using 19,532 function evaluations for 16-digit accuracy, while the WCA solved the same problem for 17-digit accuracy using 11,550 function evaluations.

One of the advantages of the proposed optimizers is that the function values are reduced to near optimum point in the early iterations. This may be due to the searching criteria and approaches of WCA and MBA where it searches a wide region of problem domain and quickly focuses on the near optimum solutions. In the following subsections in this chapter, this advantage is shown in higher dimensions for benchmark functions (see Figure 5.2).

Another comparison is presented to show the superiority of the proposed algorithms. Table 5.8 shows the specification of seven other unconstrained benchmark functions that were optimized using the HS (Lee & Geem, 2005).

The user parameters which are used for initialization of the proposed WCA and MBA are given in Tables 5.9 and 5.10, respectively. The statistical optimization results for the seven unconstrained benchmark functions from Table 5.8 including the worst, mean, best solutions, and SD are summarized in Tables 5.11 and 5.12 for both proposed WCA and MBA, respectively.

Table 5.8: Specification of seven unconstrained benchmark functions presented in (Lee & Geem, 2005).

No.	**Functions**	**Interval**	***N***
1	Rosenbrock	$[-10,10]^N$	2
2	Goldstein and Price I	$[-5,5]^N$	2
3	Goldstein and Price II	$[-5,5]^N$	2
4	Six Hump Camel Back	$[-10,10]^N$	2
5	Easton and Fenton	$[0,10]^N$	2
6	Wood	$[-5,5]^N$	4
7	Powell Quartic	$[-5,5]^N$	4

Table 5.9: User parameters for the WCA for seven benchmark functions given in Table 5.8.

No.	N_{total}	N_{sr}	d_{max}	Max_Iteration
1	20	4	1E-16	1000
	20	4	1E-03	1000
2	10	3	1E-03	300
3	50	4	1E-03	1000
4	15	3	1E-03	500
5	10	3	1E-03	100
6	50	4	1E-16	500
	50	4	1E-03	1000
7	50	4	1E-16	500
	50	4	1E-03	1000

Table 5.10: Initial parameters used for optimization of seven unconstrained benchmark functions using the MBA presented in Table 5.8.

No.	N_s	μ	α	Max_Iteration
1	10	3	500	1000
2	10	0	500	300
3	50	3	1000	1000
4	15	0	500	500
5	10	0	500	100
6	50	3	1000	1000
7	50	5	1000	5000

Table 5.11: Statistical optimization results for seven unconstrained benchmark functions presented in Table 5.8 using the WCA.

No.	Worst	Mean	Best	SD	Optimum
1	1.12E-01	7.60E-03	0	2.54E-02	0
	4.78E-09	9.54E-10	4.52E-11	1.06E-09	0
2	3.0000	3.0000	3.0000	9.81E-07	3
3	1.1291	1.0118	1.0000	0.0360	1
4	-1.0316	-1.0316	-1.0316	1.38E-08	-1.0316285
5	1.7441	1.7441	1.7441	1.96E-06	1.74
6	8.13E-18	3.25E-19	0	1.62E-18	0
	3.81E-05	1.58E-06	1.30E-10	7.60E-06	0
7	1.41E-11	5.67E-13	4.63E-38	2.83E-12	0
	2.87E-09	6.09E-10	1.12E-11	8.29E-10	0

Table 5.12: Statistical optimization results of 50 independent runs for seven unconstrained benchmark functions given in Table 5.8 using the MBA.

No.	Worst	Mean	Best	SD	Optimum
1 (μ=0)	1.15E-12	4.60E-14	9.79E-27	2.30E-13	0
1 (μ =3)	5.91E-07	5.11E-08	3.60E-14	1.37E-07	0
2	2.999999	2.999999	2.999999	2.60E-12	3
3 (μ =0)	1.037497	1.006012	1.0000002	1.3E-02	1
3 (μ =3)	1.037497	1.010548	1.0000000	1.71E-02	1
4	-1.03162845	-1.03162845	-1.03162845	0	-1.0316285
5	1.744675	1.744180	1.744152	1.07E-04	1.74
6 (μ =0)	2.53E-02	3.86E-03	6.30E-06	6.16E-03	0
6 (μ =3)	1.04E-02	1.17E-03	6.37E-07	2.24E-03	0
7 (μ =0)	1.12E-04	7.11E-05	1.27E-07	3.25E-05	0
7 (μ =3)	1.05E-04	5.88E-06	1.56E-11	2.07E-05	0

Table 5.13 demonstrates the results of optimization in terms of the NFEs and best function value. For all benchmark functions given in Table 5.13, the WCA and MBA shows their superiority over the HS in terms of function evaluations (convergence rate) and best obtained solution (accuracy).

Table 5.13: Comparison of results for the optimization of seven unconstrained benchmark functions presented in Table 5.8.

No.	HS		WCA		MBA	
	Best Solution	NFEs	Best Solution	NFEs	Best solution	NFEs
1	5.68E-10	50,000	0	**820**	3.60E-14	1660
2	3.0000	40,000	3.0000	2400	2.9999	**1190**
3	1.0000	45,000	1.0000	47,500	1.0000	**8700**
4	-1.0316	4870	-1.0316	3105	-1.0316	**1905**
5	1.7441	800	1.7441	650	1.7441	**480**
6	4.85E-09	70,000	0	**1700**	6.37E-07	8500
7	1.25E-11	100,000	4.63E-38	**16,750**	1.56E-11	18,600

5.2.2. Finding the global minimum among many local minima

A special ability of proposed optimizers is finding the global minimum of functions having many local minima without being trapped in local minima. In Subsection 5.1.1 in this chapter, the results showed this ability for the WCA and MBA. For further clarify of this feature, six well-known unconstrained benchmark functions are optimized using the proposed methods.

The multimodal functions considered are the Schwefel function, Ackley function, Rastrigin function, Sphere function, Rosenbrock function, and Zakharov function having 30 independent variables from (Mariani et al., 2011). Table 5.14 presents the specifications of these benchmark functions.

Table 5.14: Specifications of six unconstrained benchmark functions presented in (Ahrari & Aatai, 2010; Mariani et al., 2011).

No.	**Functions**	***N***	**Interval**
1	Schwefel	30	$[-500,500]^N$
2	Ackley	30	$[-32,32]^N$
3	Rastrigin	30	$[-5.12,5.12]^N$
4	Sphere	30	$[-5.12,5.12]^N$
5	Rosenbrock	30	$[-30,30]^N$
6	Zakharov	30	$[-10,10]^N$

Functions 1 to 6 are high-dimensional problems. The Schwefel, Ackley, Rastrigin, and Rosenbrock functions are multimodal (various optima) functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms. It is important to mention that the Rosenbrock function can be treated as a multimodal problem (Shang & Qiu, 2006).

Rosenbrock function has a narrow parabolic-shaped deep valley from the perceived local optima to the global optimum. To find the valley is trivial, but to achieve convergence to the global minimum is a difficult task. The Sphere and Zakharov functions are unimodal (one optimum).

In order to show the complexity and difficulty of mentioned benchmark functions, Figure 5.2 is given for representing these functions having only 2 independent variables. As shown in Figure 5.2, the global minimum is surrounded among many local minima, even for the two-dimensional mode (see Figures 5.2a, 5.2b, and 5.2c).

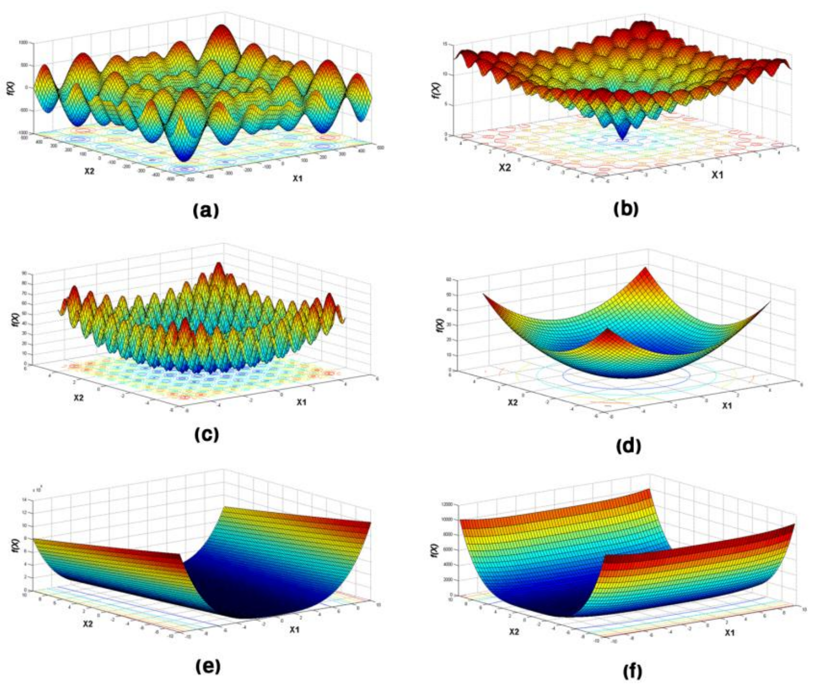


Figure 5.2. Mesh plot and contour lines for six unconstrained benchmark functions in 2 dimensions presented in Table 9: (a) Schwefel, (b) Ackley, (c) Rastrigin, (d) Hyper Sphere, (e) Rosenbrock, (f) Zakharov.

The performance of the proposed methods were compared with Genetic Algorithm with Floating-point representation (GAF), Shuffled Complex Evolution algorithm (SCE-UA), Modified Shuffled Complex Evolution algorithm (MSCE) (Mariani et al., 2011), Differential Evolution (DE), Gregarious Particle Swarm Optimizer (GPSO), and Synchronous Bacterial Foraging Optimization (SBFO) (Bakwad et al., 2010).

The number of function evaluations was chosen as a criterion for measuring computational cost instead of number of iterations and CPU time. Table 5.15 provides selected initial parameters used in the MBA for optimization of functions given in Table 5.14.

Table 5.15: User parameters for optimization of benchmark functions presented in Table 5.14 using the MBA.

No.	N_s	μ	α	Max_Iteration
1	50	10	10000	1000
2	50	10	1000	2000
3	50	10	10000	500
4	50	10	10000	5000
5	50	10	5000	1000
6	50	10	5000	1000

Similarly for the WCA, the number of rivers (N_{sr}), total number of raindrops (N_{total}), d_{max}, and number of maximum iterations for test functions in Table 5.14 were 4, 50, 1E-16, and 500, respectively. Tables 5.16 and 5.17 show the statistical optimization results including the worst, mean, best solution, SD, and NFEs for each benchmark function for the WCA and MBA, respectively.

Table 5.16: Statistical optimization results of WCA for six benchmark functions given in Table 5.14.

No.	Worst	Mean	Best	SD	NFEs
1	3.87E-4	3.82E-4	3.81E-4	1.05E-6	3050
2	4.44E-15	1.03E-15	8.88E-16	7.10E-16	1900
3	4.99E-6	2.00E-7	2.21E-12	9.99E-7	20,350
4	1.05E-17	8.44E-19	2.68E-37	2.92E-18	8000
5	1.75E-4	7.00E-6	3.01E-14	3.50E-5	18,150
6	4.64E-11	1.93E-12	2.26E-36	9.48E-12	17,750

Table 5.17: Statistical optimization results of MBA for six unconstrained benchmark functions presented in Table 5.14.

No.	Worst	Mean	Best	SD	NFEs
1	7.71E-05	7.69E-05	7.63E-05	5.17E-08	7800
2	2.21E-13	4.52E-14	2.22E-14	3.89E-14	48,800
3	3.58E-04	2.67E-05	1.49E-08	5.88E-01	5900
4	1.21E-14	6.59E-16	4.70E-21	2.52E-15	33,950
5	1.75E-02	2.01E-03	1.10E-07	9.80E-01	9300
6	1.44E-02	1.24E-03	1.13E-08	3.11E-03	9600

Tables 5.18, 5.19, and 5.20 represent the statistical optimization results of GAF, SCE-UA, and MSCE for optimization of six unconstrained functions given in Table 5.14, respectively. The WCA and MBA shows their superiority over other considered algorithms in terms of the NFEs for all reported functions.

Table 5.18: Statistical optimization results for the GAF from (Mariani et al., 2011).

No.	Worst	Mean	Best	SD	NFEs
1	6219.6	5434.8	3987.9	552.3	120,000
2	3.1669	1.8585	0.1209	0.6483	120,000
3	1.9902	0.2655	2.13E-13	0.5183	120,000
4	2.294E-4	4.831E-5	9.56E-11	4.292E-5	120,000
5	23.0082	51.7613	27.7946	50.6304	120,000
6	52.8072	30.9811	13.7928	10.5527	120,000

Table 5.19: Statistical optimization results for the SCE-UA from (Mariani et al., 2011).

No.	Worst	Mean	Best	SD	NFEs
1	8594.3853	8042.6031	7394.4199	288.5129	120,000
2	1.6462	0.1068	1.663E-04	0.3407	120,000
3	3.9798	1.5588	5.513E-09	1.1294	120,000
4	5.972E-11	5.92E-12	3.489E-16	1.212E-11	120,000
5	28.2745	27.0576	25.3911	0.6330	120,000
6	0.0393	0.0116	2.603E-04	0.0112	120,000

Table 5.20: Statistical optimization results for the MSCE from (Mariani et al., 2011).

No.	Worst	Mean	Best	SD	NFEs
1	6.1420	1.5598	0.1072	1.4026	120,000
2	8.882E-16	8.882E-16	8.882E-16	1E-15	120,000
3	3.9095	1.5270	5.321E-09	1.1216	120,000
4	0	0	0	0	120,000
5	25.9221	23.4675	20.3137	1.2133	120,000
6	0	0	0	0	120,000

The MSCE used 120,000 function evaluations to found the global optimum point for functions 4 and 6 with standard deviation equal to zero, while the WCA reached its optimal point with 37-digit and 36-digit accuracies, respectively.

Meanwhile, for functions 4 and 6, the NFEs for WCA are 8000 and 17750, respectively.

Hence, as can be seen in Tables 5.16 to 5.20, the proposed optimizers can find the optimum point faster than reported methods compared in this study with good accuracy. The only method that can compete with the WCA and MBA in terms of function value for some functions is the MSCE.

Furthermore, the MBA and WCA were also compared with the DE, GPSO, and SBFO (Bakwad et al., 2010). The obtained results were compared with respect to the best solution and the NFEs. Table 5.21 shows the comparison of optimization results for the proposed methods against other algorithms for a number of benchmark functions presented in Table 5.14.

Table 5.21: Comparison of optimization results for four benchmark functions given in Table 5.14. “ANFEs” stands for average number of function evaluations.

No.	SBFO		GPSO		DE		WCA		MBA	
	Best	ANFEs	Best	ANFEs	Best	ANFEs	Best	ANFEs	Best	ANFEs
2	5.18E-04	100,000	3.70E-02	200,000	8E-04	200,000	8.88E-16	13,217	1.09E-06	31,375
3	4.68E-04	100,000	0.13	200,000	27.43	200,000	2.21E-12	10,425	1.49E-08	6350
4	4.68E-04	100,000	6.60E-02	200,000	3.50E-03	200,000	2.68E-37	3334	4.70E-21	22,350
5	27.6329	100,000	2.46	200,000	34.35	200,000	3.01E-14	9600	1.10E-07	9875

By observing Table 5.21, all methods except the WCA and MBA are given from literature (Bakwad et al., 2010). As shown in Table 5.21, the WCA and MBA outperformed other reported optimizers in terms of NFEs and best function value. In general, for nearly all comparative functions, the proposed algorithms show the advantage of less number of function evaluation and acceptable function value accuracy.

As mention in Subsection 5.1, one of the advantages of the WCA and MBA is that the function values are reduced to near optimum point quickly in the early iteration. Figure 5.3 illustrates the function values with respect to the number of iterations for six benchmark functions presented in Table 5.14. For all six benchmark functions the first 100 iterations are depicted to show the fast convergence of WCA more clearly.

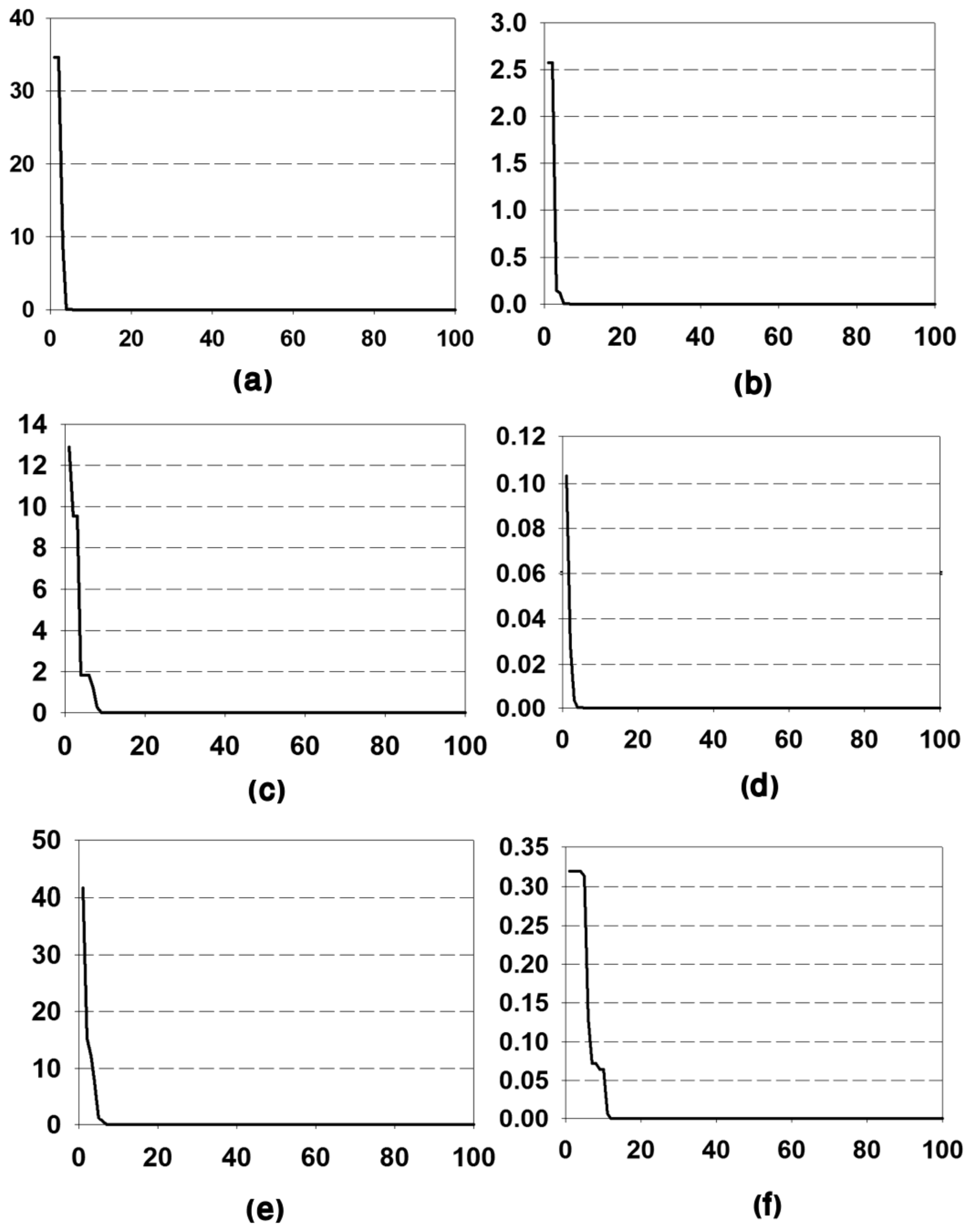


Figure 5.3. Function values versus the number of iterations for six benchmark functions in Table 5.14 using the WCA: (a) Schwefel, (b) Ackley, (c) Rastrigin, (d) Hyper Sphere, (e) Rosenbrock, (f) Zakharov (Vertical and horizontal axes are function values and number of iterations, respectively).

5.3. Constrained and engineering benchmark problems

In this subsection, the performance of the proposed optimizers is tested by solving several constrained and engineering optimization problems. In order to validate the proposed methods for constraint problems, first, two constrained benchmark problems have been applied and then, the performance of the WCA and MBA for five engineering design problems (widely used in literatures) was examined and the optimization results were compared with other optimization engines.

The benchmark problems include the objective functions of various types (quadratic, cubic, polynomial, and nonlinear functions) with various number of the design variables, different types, and number of inequality and equality constraints. The proposed algorithms were written in MATLAB programming software and simulations were run on a Pentium V 2.53 GHz with 4 GB RAM.

The task of optimizing each of the test functions was executed using 50 independent runs. The maximization problems were transformed into minimization ones as $-f(x)$. For all benchmark problems, the initial parameters for the WCA, (N_{total}, N_{sr}, and d_{max}) were chosen as 50, 8, and, 1E-03, respectively. Similarly, for the MBA, the user parameters are given in Table 5.22 for considered constrained and engineering problems in this thesis.

Table 5.22: User parameters used for the MBA for seven constrained and engineering problems.

Problem	N_s	α	μ	Max iteration
Constrained Problem 1	50	20000	5	1000
Constrained Problem 2	50	5000	0	500
Pressure vessel	50	50000	10	2000
Spring design	50	5000	0	1000
Welded beam	30	150,000	5	2000
Speed reducer	50	500	10	500
Rolling element bearing	50	5000	10	1000

Different iteration numbers were used for each benchmark function, with smaller iteration number for smaller number of design variables and moderate functions, while larger iteration number for large number of desicion variables and complex problems. The mathematical formulations and their constraints for the mechanical engineering design problems and constrained benchmark functions are given in Appendix *B*.

5.3.1. Constrained problem 1

This minimization function (see Appendix *B*.1) was previously solved using homomorphous mappings (HM) (Koziel & Michalewicz, 1999), adaptive segregational constraint handling evolutionary algorithm (ASCHEA) (Hamida & Schoenauer, 2002), stochastic ranking (SR) (Runarsson & Xin, 2000), cultural algorithms with evolutionary programming (CAEP) (Coello & Becerra, 2004), hybrid PSO (HPSO) (He & Wang, 2007), changing range genetic algorithm (CRGA) (Amirjanov, 2006), DE (Lampinen, 2002), cultured differential evolution (CULDE) (Becerra & Coello, 2006), PSO with differential evolution (PSO-DE), PSO (Liu et al., 2010), HS, simple multi-membered evolution strategy (SMES) (Montes & Coello, 2005), self adaptive penalty function (SAPF) (Tessema & Yen, 2006), differential evolution with level comparison (DELC) (Wang & Li, 2010), differential evolution with dynamic stochastic selection (DEDS) (Zhang et al., 2008), improved stochastic ranking (ISR) (Runarsson & Xin, 2005), hybrid evolutionary algorithm and adaptive constraint handling technique (HEAA) (Wang et al., 2009), and α constrained simplex method (α simplex) (Takahama & Sakai, 2005).

Table 5.23 compares the reported best solutions for the CULDE, HS, GA (Michalewicz, 1995), WCA, and MBA. The statistical results of different algorithms accompanied with the proposed methods are given in Table 5.24. By observing Table 5.24, the WCA and MBA reached the optimal solution faster and more accurate than other algorithms in this research surpassing the WCA over MBA in terms of number of function evaluations.

Table 5.23: Comparison of the best solution given by various algorithms for the constrained problem 1.

D.V.	CULDE	HS	GA	WCA	MBA	Optimal
X_1	78.00	78.00	78.04	78.00	78.00	78.00000
X_2	33.00	33.00	33.00	33.00	33.00	33.00000
X_3	29.99	29.995	27.081	29.99	29.99	29.99526
X_4	45.00	45.00	45.00	45.00	44.99	45.00000
X_5	36.77	36.77	44.94	36.77	36.77	36.77581
$g_1(X)$	1.35E-08	4.34E-05	1.28	-1.96E-12	1.33E-08	-9.71E-04
$g_2(X)$	-92.00	-92.00	-93.28	-91.99	-91.99	-92
$g_3(X)$	-11.15	-11.15	-9.59	-11.19	-11.19	-1.11E+01
$g_4(X)$	-8.84	-8.84	-10.40	-8.84	-8.84	-8.87
$g_5(X)$	-4.99	-5.00	-4.99	-5.00	-4.99	-5
$g_6(X)$	4.12E-09	6.49E-05	1.91E-03	0.00	-3.06E-09	9.27E-09
$f(X)$	-30665.538	-30665.500	-31020.859	-30665.538	-30665.538	-30665.539

Table 5.24: Comparison of statistical optimization results for several reported algorithms for the constrained problem 1.

Methods	Worst	Mean	Best	SD	NFEs
HM	-30645.9000	-30665.3000	-30664.500	N.A	1,400,000
ASCHEA	N.A	-30665.5000	-30665.500	N.A	1,500,000
SR	-30665.5390	-30665.5390	-30665.5390	2E-05	88,200
CAEP	-30662.2000	-30662.5000	-30665.5000	9.3	50,020
PSO	-30252.3258	-30570.9286	-30663.8563	81	70,100
HPSO	-30665.5390	-30665.5390	-30665.5390	1.70E-06	81,000
PSO-DE	-30665.5387	-30665.5387	-30665.5387	8.30E-10	70,100
CULDE	-30665.5386	-30665.5386	-30665.5386	1E-07	100,100
DE	-30665.5090	-30665.5360	-30665.5390	5.067E-03	240,000
HS	N.A	N.A	-30665.5000	N.A	65,000
CRGA	-30660.3130	-30664.3980	-30665.5200	1.6	54,400
SAPF	-30656.4710	-30655.9220	-30665.4010	2.043	500,000
SMES	-30665.5390	-30665.5390	-30665.5390	0	240,000
DELC	-30665.5390	-30665.5390	-30665.5390	1.0E-11	50,000
DEDS	-30665.5390	-30665.5390	-30665.5390	2.70E-11	225,000
HEAA	-30665.5390	-30665.5390	-30665.5390	7.40E-12	200,000
ISR	-30665.5390	-30665.5390	-30665.5390	1.10E-11	192,000
α Simplex	-30665.5390	-30665.5390	-30665.5390	4.20E-11	305,343
WCA	-30665.4570	-30665.5270	-30665.5386	2.18E-02	18,850
MBA	-30665.3300	-30665.5182	-30665.5386	5.08E-02	41,750

5.3.2. Constrained problem 2

For this maximization problem (see Appendix *B*.2) which is converted to the minimization problem, the feasible region of the search space consists of 729 disjoint spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist *p*, *q*, *r* such that the inequality holds, as given in Appendix *B* (Zahara & Kao, 2009).

For this problem, the optimum solution is X^*= (5, 5, 5) with $f(X^*)$ = -1. This problem was previously solved using the HM, SR, CULDE, CAEP, HPSO, artificial bee colony (ABC) (Karaboga and Basturk, 2007), particle evolutionary swarm optimization (PESO) (Zavala et al., 2005), CDE (Huang et al., 2007), SMES, and teaching-learning-based optimization (TLBO) (Rao et al., 2011).

The statistical optimization results of twelve optimizers including the MBA and WCA are shown in Table 5.25. From Table 5.25, although the best solution of the WCA and MBA is not as accurate as other considered algorithms, however, they reached the best solution considerably faster than other reported algorithms using 6100 and 14,950 number of function evaluations, respectively.

Table 5.25: Comparison of optimization statistical results given by various algorithms for the constrained problem 2.

Methods	Worst	Mean	Best	SD	NFEs
HM	-0.991950	-0.999135	-0.999999	N.A	1,400,000
SR	-1	-1	-1	0	350,000
CAEP	-0.996375	-0.996375	-1	9.7E-03	50,020
HPSO	-1	-1	-1	1.6E-15	81,000
CULDE	-1	-1	-1	0	100,100
SMES	-1	-1	-1	0	240,000
PESO	-0.994	-0.998875	-1	N.A	350,000
CDE	-1	-1	-1	0	248,000
ABC	-1	-1	-1	0	240,000
TLBO	-1	-1	-1	0	50,000
WCA	-0.999998	-0.999999	-0.999999	2.51E-07	6100
MBA	-0.996539	-0.999147	-0.999813	5.44E-04	14,950

5.3.3. Pressure vessel design problem

In pressure vessel design problem (see Appendix *B*.3), proposed by Kannan and Kramer (1994), the target is to minimize the total cost, including the cost of material, forming, and welding. A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure 5.4.

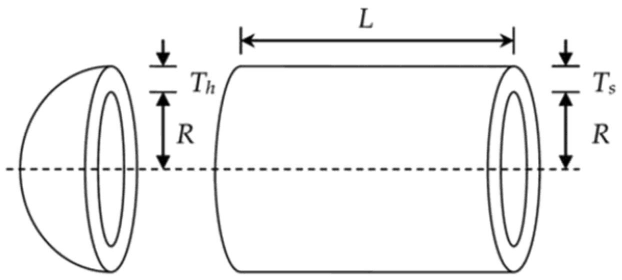


Figure 5.4. Schematic view of pressure vessel problem.

There are four design variables in this problem: T_s (x_1, thickness of the shell), T_h (x_2, thickness of the head), R (x_3, inner radius), and L (x_4, length of the cylindrical section of the vessel). Among the four design variables, T_s and T_h are expected to be integer multiples of 0.0625 *in*, and R and L are continuous design variables.

Table 5.26 shows the comparisons of the best solution for both proposed optimizers and other compared methods. This problem has been solved previously using the GA based co-evolution model (GA1) (Coello, 2000a), GA through the use of dominance-based tour tournament selection (GA2) (Coello & Montes, 2002), co-evolutionary PSO (CPSO) (Renato & Santos, 2006), HPSO, hybrid nelder-mead simplex search and particle swarm optimization (NM-PSO) (Zahara et al., 2009), Gaussian quantum-behaved PSO (G-QPSO), quantum-behaved PSO (QPSO) (Coelho, 2010), PSO, and co-evolutionary differential evolution (CDE) (Huang et al., 2007) and compared with the proposed WCA and MBA as given in Table 5.27.

Table 5.26: Comparison of the best solution obtained from various studies for the pressure vessel problem.

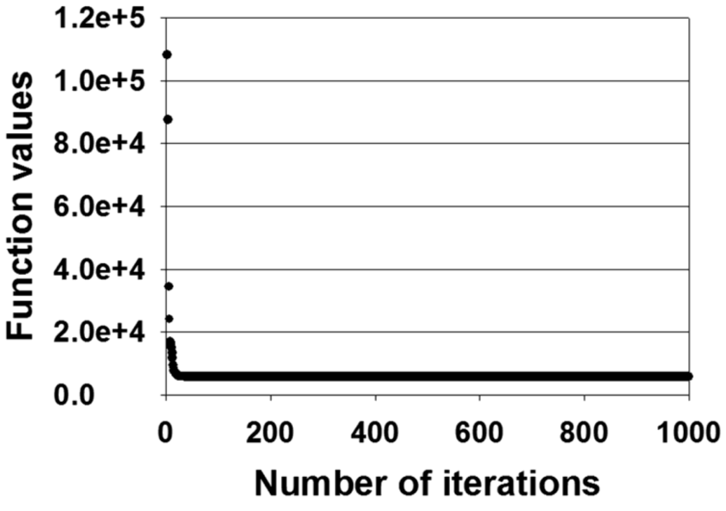
D.V.	CDE	HPSO	NM-PSO	G-QPSO	WCA	MBA
X_1	0.8125	0.8125	0.8036	0.8125	0.7781	0.7802
X_2	0.4375	0.4375	0.3972	0.4375	0.3846	0.3856
X_3	42.0984	42.0984	41.6392	42.0984	40.3196	40.4292
X_4	176.6376	176.6366	182.4120	176.6372	-200.0000	198.4964
$g_1(X)$	-6.67E-07	-8.80E-07	3.65E-05	-8.79E-07	-2.95E-11	0
$g_2(X)$	-3.58E-02	-3.58E-02	3.79E-05	-3.58E-02	-7.15E-11	0
$g_3(X)$	-3.705123	3.1226	-1.5914	-0.2179	-1.35E-06	-86.3645
$g_4(X)$	-63.3623	-63.3634	-57.5879	-63.3628	-40.0000	-41.5035
$f(X)$	6059.7340	6059.7143	5930.3137	6059.7208	5885.3327	5889.3216

Table 5.27: Comparison of statistical results given by different optimizers for the pressure vessel problem.

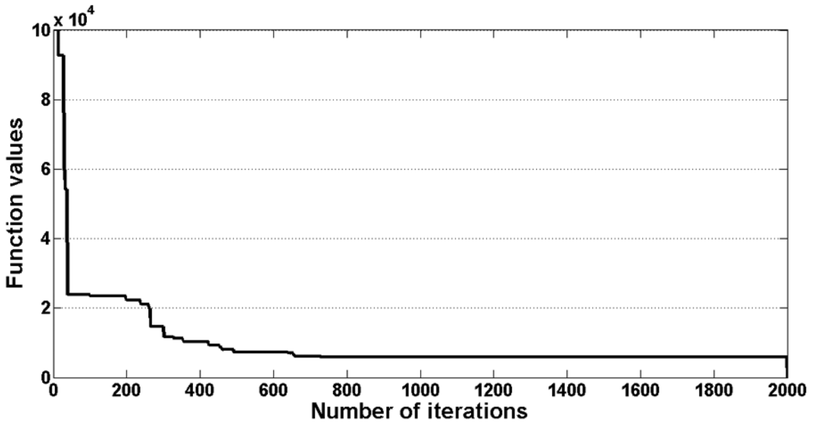
Methods	Worst	Mean	Best	SD	NFEs
GA1	6308.4970	6293.8432	6288.7445	7.4133	900,000
GA2	6469.3220	6177.2533	6059.9463	130.9297	80,000
CPSO	6363.8041	6147.1332	6061.0777	86.45	240,000
HPSO	6288.6770	6099.9323	6059.7143	86.20	81,000
NM-PSO	5960.0557	5946.7901	5930.3137	9.161	80,000
G-QPSO	7544.4925	6440.3786	6059.7208	448.4711	8000
QPSO	8017.2816	6440.3786	6059.7209	479.2671	8000
PSO	14076.3240	8756.6803	6693.7212	1492.5670	8000
CDE	6371.0455	6085.2303	6059.7340	43.0130	204,800
WCA	7319.0197	6230.4247	5885.3711	338.7300	8000
MBA	6392.5062	6200.64765	5889.3216	160.34	70,650

As can be seen from Table 5.27, in terms of best solution and number of function evaluations the proposed WCA is superior to other optimizer, while the MBA has better statistical optimization results than the WCA.

Considering the statistical and comparison results in Table 5.27, it can be concluded that the WCA is more efficient than the other optimization engines for the pressure vessel design problem, in this study. Figure 5.5 depicts the function values versus the number of iterations for the pressure vessel design problem using both proposed methods.



(a)



(b)

Figure 5.5. Function values versus number of iterations for the pressure vessel problem using: (a) WCA, (b) MBA.

One of the advantages of the proposed methods that may be hardly seen in other metaheuristic algorithms is that the function values are reduced to near optimum point in the early iterations (see Figure 5.5). This may be due to the searching criteria and constraint handling approaches of WCA and MBA where it initially searches a wide region of problem domain and rapidly focuses on the optimum solution.

5.3.4. Tension/compression spring design problem

The tension/compression spring design problem (see Appendix *B*.4) is described in Arora (1989) for which the objective is to minimize the weight (*f(x)*) of a tension/compression spring (as shown in Figure 5.6) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The independent variables are the wire diameter *d*(x_1), the mean coil diameter *D*(x_2), and the number of active coils *P*(x_3).

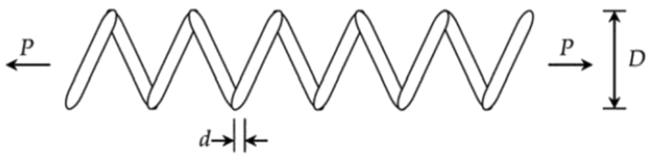


Figure 5.6. Schematic view of tension/compression spring problem.

The comparisons of the best solution among several reported algorithms are given in Table 5.28. This problem has been used as a benchmark problem for testing the efficiency of numerous optimization methods such as GA1, GA2, CAEP, CPSO, HPSO, NM-PSO, G-QPSO, QPSO, PSO-DE, PSO, DELC, DEDS, HEAA, society and civilization (SC) (Ray & Liew, 2003), DE, ABC, and (μ+λ)-ES (Montes & Coello, 2005a). The obtained statistical results using the reported optimizers and the proposed WCA and MBA are given in Table 5.29.

Table 5.28: Comparison of the best solution obtained from various algorithms for the tension/compression spring problem.

D.V.	DEDS	HEAA	NM-PSO	DELC	WCA	MBA
X_1	0.051689	0.051689	0.051620	0.051689	0.051680	0.051656
X_2	0.356717	0.356729	0.355498	0.356717	0.356522	0.355940
X_3	11.288965	11.288293	11.333272	11.288965	11.300410	11.344665
$g_1(X)$	1.45E-09	3.96E-10	1.01E-03	-3.40E-09	-1.65E-13	0
$g_2(X)$	-1.19E-09	-3.59E-10	9.94E-04	2.44E-09	-7.9E-14	0
$g_3(X)$	-4.053785	-4.053808	-4.061859	-4.053785	-4.053399	-4.052248
$g_4(X)$	-0.727728	-0.727720	-0.728588	-0.727728	-0.727864	-0.728268
$f(X)$	0.012665	0.012665	0.012630	0.012665	0.012665	0.012665

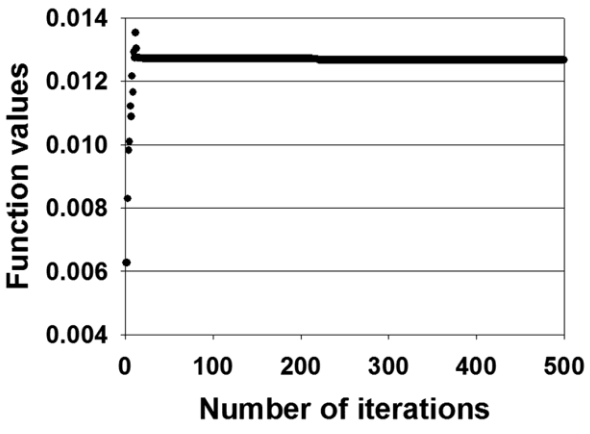
Table 5.29: Comparisons of statistical optimization results obtained from various algorithms for the tension/compression spring problem.

Methods	Worst	Mean	Best	SD	NFEs
GA1	0.012822	0.012769	0.012704	3.94E-05	900,000
GA2	0.012973	0.012742	0.012681	5.90E-05	80,000
CAEP	0.015116	0.013568	0.012721	8.42E-04	50,020
CPSO	0.012924	0.012730	0.012674	5.20E-04	240,000
HPSO	0.012719	0.012707	0.012665	1.58E-05	81,000
NM-PSO	0.012633	0.012631	0.012630	8.47E-07	80,000
G-QPSO	0.017759	0.013524	0.012665	0.001268	2000
QPSO	0.018127	0.013854	0.012669	0.001341	2000
PSO	0.071802	0.019555	0.012857	0.011662	2000
DE	0.012790	0.012703	0.012670	2.7E-05	204,800
DELC	0.012665	0.012665	0.012665	1.3E-07	20,000
DEDS	0.012738	0.012669	0.012665	1.3E-05	24,000
HEAA	0.012665	0.012665	0.012665	1.4E-09	24,000
PSO-DE	0.012665	0.012665	0.012665	1.2E-08	24,950
SC	0.016717	0.012922	0.012669	5.9E-04	25,167
(μ+λ)-ES	N.A	0.013165	0.012689	3.9E-04	30,000
ABC	N.A	0.012709	0.012665	0.012813	30,000
WCA	0.015021	0.013013	0.012665	6.16E-04	2000
MBA	0.012900	0.012713	0.012665	6.30E-05	7650

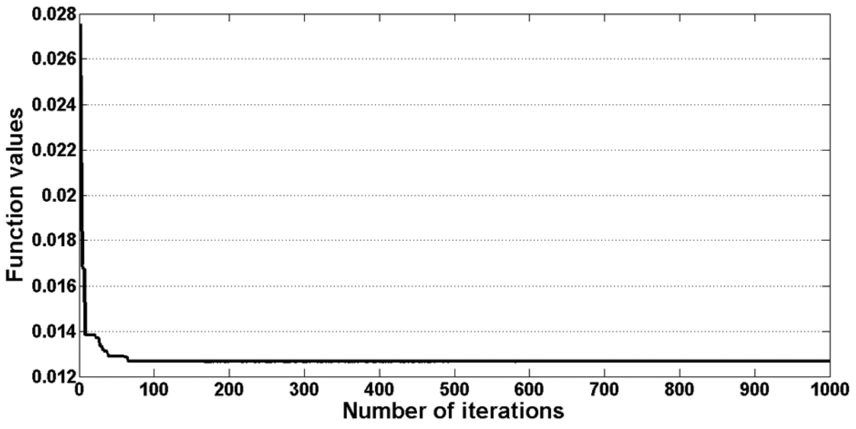
The best function value is 0.012630 with 80,000 function evaluations obtained by the NM-PSO. In terms of the NFEs, both suggested methods have found their best solution in less number of function evaluations compared with the NM-PSO.

From Table 5.29, two proposed methods show their superiority compared with other methods in terms of the number of function evaluations and obtained statistical results. Therefore, the MBA and WCA can identify optimum or very close to optimum solutions for the tension/compression spring design problem faster and/or more accurate than other reported optimizers mentioned in this research.

Figure 5.7 demonstrates the function values with respect to the number of iterations for the tension/compression spring design problem for both proposed methods. From Figure 5.7a, in the early iterations of WCA, the initial population of the algorithm was in the infeasible region. After further iterations, the population was adjusted to the feasible region and the function values were reduced at each iteration.



(a)



(b)

Figure 5.7. Function values with respect to the number of iterations for the tension/compression spring problem using: (a) WCA, (b) MBA.

The constraint violation values with respect to the number of iterations for the tension/compression spring problem are shown in Figure 5.8. From Figure 5.8, the obtained solutions did not satisfy the constraints in the early iterations. As the algorithm continued, the obtained results satisfied the constraints, while the value of constraint violation decreased.

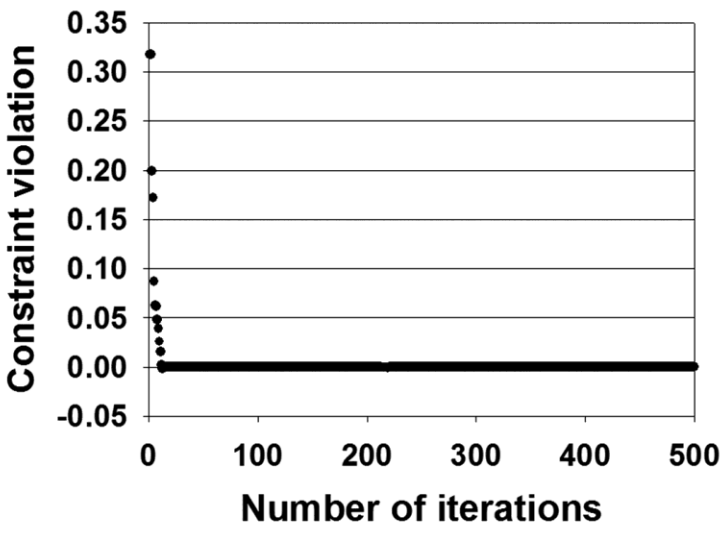


Figure 5.8. Constraint violation values with respect to the number of iterations for tension/compression spring problem using the WCA.

5.3.5. Welded beam design problem

This design problem (see Appendix *B*.5), which has been often used as a benchmark problem, was proposed by Coello (2000a). In this problem, a welded beam is designed for minimum cost subject to constraints on shear stress (τ), bending stress (σ) in the beam, buckling load on the bar (P_b), end deflection of the beam (δ), and side constraints. There are four design variables as shown in Figure 7.9: $h(x_1)$, $l(x_2)$, $t(x_3)$ and $b(x_4)$.

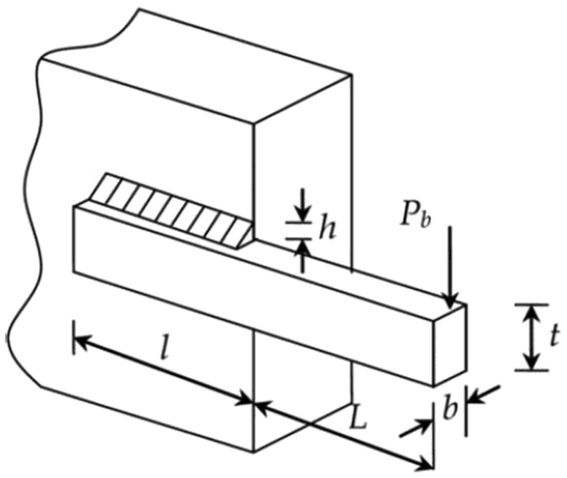


Figure 5.9. Schematic view of welded beam problem.

The optimization engines previously applied to this problem such as GA1, GA2, CAEP, CPSO, HPSO, NM–PSO, hybrid genetic algorithm (HGA) (Yuan & Qian, 2010), modified GA (MGA) (Coello, 2000b), SC, and DE. The comparisons of the best solutions given by different algorithms are presented in Table 5.30. Furthermore, the comparison of the statistical optimization results for several algorithms is given in Table 5.31.

Table 5.30: Comparison of the best solution obtained from various algorithms for the welded beam problem.

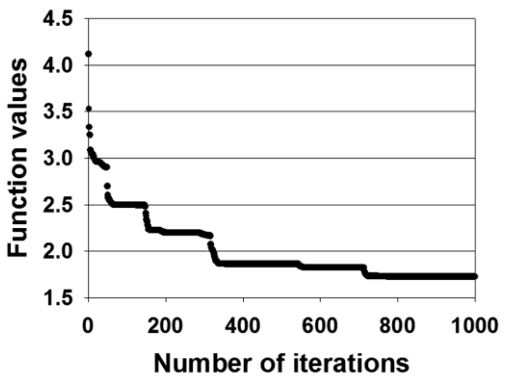
D.V.	**CPSO**	**CAEP**	**HGA**	**NM-PSO**	**WCA**	**MBA**
$X_1(h)$	0.202369	0.205700	0.2057	0.20583	0.205728	0.205729
$X_2(l)$	3.544214	3.470500	3.4705	3.468338	3.470522	3.470493
$X_3(t)$	9.048210	9.036600	9.0366	9.036624	9.036620	9.036626
$X_4(b)$	0.205723	0.205700	0.2057	0.20573	0.205729	0.205729
$g_1(X)$	-13.655547	1.988676	1.988676	-0.02525	-0.034128	-0.001614
$g_2(X)$	-78.814077	4.481548	4.481548	-0.053122	-3.49E-05	-0.016911
$g_3(X)$	-3.35E-03	0	0	0.0001	-1.19E-06	-2.40E-07
$g_4(X)$	-3.424572	-3.433213	-3.433213	-3.433169	-3.432980	-3.432982
$g_5(X)$	-0.077369	-0.080700	-0.080700	-0.08083	-0.080728	-0.080729
$g_6(X)$	-0.235595	-0.235538	-0.235538	-0.235540	-0.235540	-0.235540
$g_7(X)$	-4.472858	2.603347	2.603347	-0.031555	-0.013503	-0.001464
$f(X)$	1.728024	1.724852	1.724852	1.724717	1.724856	1.724853

Table 5.31: Comparison of the statistical results obtained from different optimization engines for the welded beam problem.

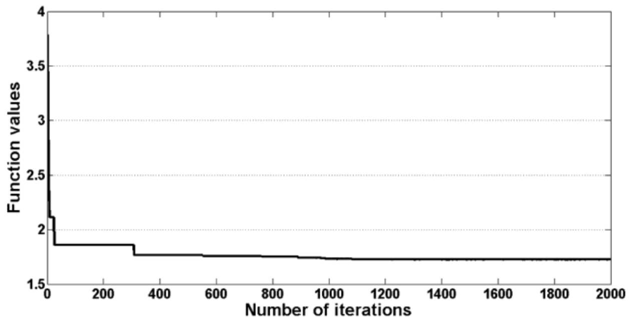
Methods	**Worst**	**Mean**	**Best**	**SD**	**NFEs**
GA1	1.785835	1.771973	1.748309	1.12E-02	900,000
GA2	1.993408	1.792654	1.728226	7.47E-02	80,000
CAEP	3.179709	1.971809	1.724852	4.43E-01	50,020
CPSO	1.782143	1.748831	1.728024	1.29E-02	240,000
HPSO	1.814295	1.749040	1.724852	4.01E-02	81,000
PSO-DE	1.724852	1.724852	1.724852	6.7E-16	66,600
NM-PSO	1.733393	1.726373	1.724717	3.50E-03	80,000
MGA	1.9950	1.9190	1.8245	5.37E-02	N.A
SC	6.399678	3.002588	2.385434	9.6E-01	33,095
DE	1.824105	1.768158	1.733461	2.21E-02	204,800
WCA	1.744697	1.726427	1.724856	4.29E-03	46,450
MBA	1.724853	1.724853	1.724853	6.94E-19	47,340

Among those previously reported studies, the best solution was obtained using the NM-PSO with an objective function value of *f(x)* = 1.724717 after 80,000 function evaluations. Using the proposed WCA and MBA, the best solution of 1.724856 and 1.724853 was obtained using 46,450 and 47,340 number of function evaluations, respectively.

The optimization statistical results obtained by the proposed methods outperformed the obtained results by other considered algorithms, except the NM-PSO, in terms of cost value. However, the WCA and MBA could offer a competitive set of statistical results in less number of function evaluations than the NM-PSO method as shown in Table 5.31. Figure 5.10 illustrates the function values in terms of the number of iterations for the welded beam design problem using both suggested optimizers.



(a)



(b)

Figure 5.10. Function values versus number of iterations for the welded beam problem using: (a) WCA, (b) MBA.

5.3.6. Speed reducer design problem

In this constrained optimization problem (see Figure 5.11), the weight of speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts, and stresses in the shafts (Montes & Coello, 2005a). The variables x_1 to x_7 represent the face width (b), module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2), and the diameter of first (d_1), and second shafts (d_2), respectively.

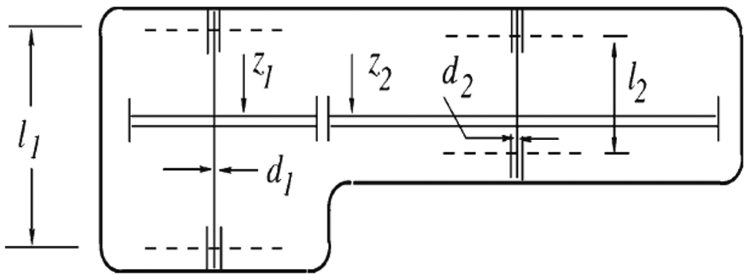


Figure 5.11. Speed reducer design problem.

This is an example of a mixed integer programming problem. The third variable x_3 (number of teeth) is of integer values, while all other variables are continuous. There are 11 constraints in this problem resulting in high complexity of the problem (Kuang et. al, 1998) (the solution reported in (Kuang et al., 1998) is infeasible).

The comparison of best solution by previous methods is given in Table 5.32. The statistical results of nine optimization methods including DELC, DEDS, PSO-DE, ABC, TLBO, modified differential evolution (MDE) (Montes et al., 2006a; Montes et al., 2006b), SC, HEAA, and (μ+λ)-ES is compared with the proposed methods which is given in Table 5.33.

Table 5.32: Comparison of the best solution obtained using different optimizers for the speed reducer design problem.

D.V	DEDS	DELC	HEAA	MDE	WCA	MBA
X_1	3.5000	3.5000	3.5000	3.5000	3.5000	3.5000
X_2	0.7000	0.7000	0.7000	0.7000	0.7000	0.7000
X_3	17	17	17.000	17.0000	17.000	17.0000
X_4	7.3333	7.3333	7.3004	7.3001	7.3000	7.3000
X_5	7.7153	7.7153	7.7153	7.8000	7.7153	7.7157
X_6	3.3502	3.3502	3.3502	3.3502	3.3502	3.3502
X_7	5.2866	5.2866	5.2866	5.2866	5.2866	5.2866
$f(X)$	2994.47106	2994.47106	2994.49910	2996.35668	2994.47106	2994.48245

Table 5.33: Comparison of statistical results using various algorithms for the speed reducer design problem.

Method	Worst	Mean	Best	SD	NFEs
SC	3009.964736	3001.758264	2994.744241	4.0	54,456
PSO-DE	2996.348204	2996.348174	2996.348167	6.4E-06	54,350
DELC	2994.471066	2994.471066	2994.471066	1.9E-12	30,000
DEDS	2994.471066	2994.471066	2994.471066	3.6E-12	30,000
HEAA	2994.752311	2994.613368	2994.499107	7.0E-02	40,000
MDE	N.A	2996.367220	2996.356689	8.2E-03	24,000
(μ+λ)-ES	N.A	2996.348	2996.348	0	30,000
ABC	N.A	2997.058	2997.058	0	30,000
TLBO	N.A	2996.34817	2996.34817	0	10,000
WCA	2994.505578	2994.474392	2994.471066	7.4E-03	15,150
MBA	2999.652444	2996.769019	2994.482453	1.56	6300

From Table 5.33, among the compared optimization algorithms, DELC, DEDS, and WCA have found the best solution so far. Although, MBA could not match the best solution obtained by DELC, DEDS, and WCA, however, it detected its best solution (second best solution) with considerably less NFEs as well as the WCA. Figure 5.12 depicts the reduction of function values versus the number of iterations for the speed reducer design problem using the MBA.

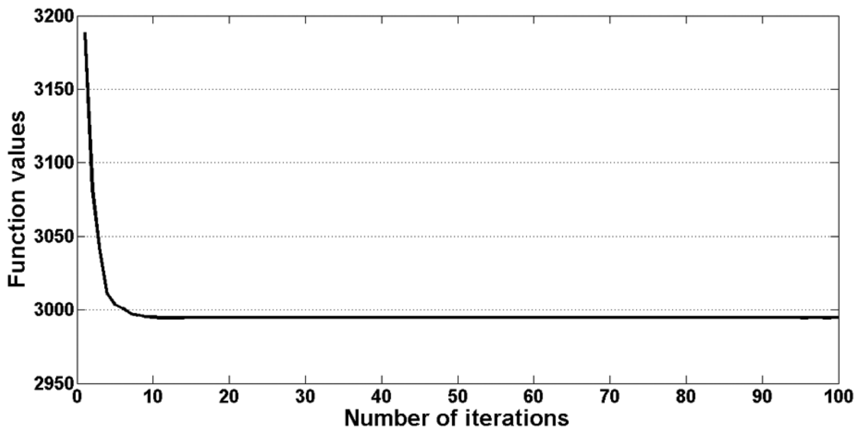


Figure 5.12. Function values versus number of iterations for the speed reducer problem using the MBA.

5.3.7. Rolling element bearing design problem

The objective of this problem is to maximize the dynamic load carrying capacity of a rolling element bearing, as demonstrated in Figure 5.13. This problem has 10 decision variables which are pitch diameter (D_m), ball diameter (D_b), number of balls (Z), inner and outer raceway curvature coefficients (f_i and f_o), KD_{min}, KD_{max}, ε, e, and ζ (see Figure 5.13).

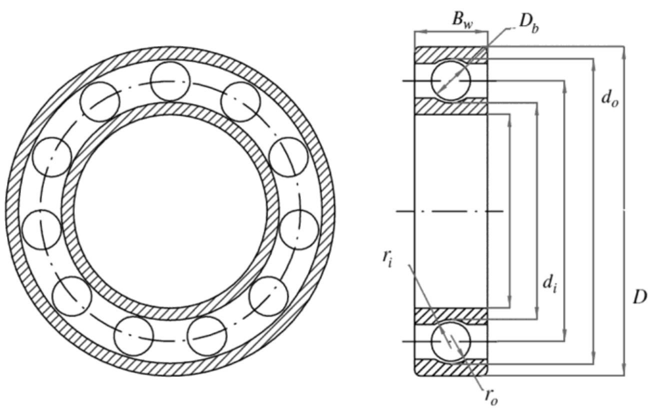


Figure 5.13. Rolling element bearing design problem.

The five latter variables only appear in constraints and indirectly affect the internal geometry. The number of balls (Z) is the discrete design variable and the remainder are continuous design variables. Constraints are imposed based on kinematic and manufacturing considerations.

The problem of the rolling element bearing was studied by GA (Gupta et al., 2007), ABC, and TLBO. Table 5.34 shows the comparison of the best solution for four optimizers in terms of design variables, function values, and constraints accuracy. The statistical optimization results for reported algorithms were compared in Table 5.35.

Table 5.34: Comparison of the best solution obtained using four algorithms for the rolling element bearing problem.

D.V	GA	TLBO	WCA	MBA
X_1	125.7171	125.7191	125.721167	125.7153
X_2	21.423	21.42559	21.423300	21.423300
X_3	11	11	11.001030	11.000
X_4	0.515	0.515	0.515000	0.515000
X_5	0.515	0.515	0.515000	0.515000
X_6	0.4159	0.424266	0.401514	0.488805
X_7	0.651	0.633948	0.659047	0.627829
X_8	0.300043	0.3	0.300032	0.300149
X_9	0.0223	0.068858	0.040045	0.097305
X_{10}	0.751	0.799498	0.600000	0.646095
$g(X_1)$	-0.000821	0	0.000040	0
$g(X_2)$	-13.732999	13.15257	14.740597	-8.630183
$g(X_3)$	-2.724000	1.5252	3.286749	-1.101429
$g(X_4)$	3.606000	0.719056	3.423300	-2.040448
$g(X_5)$	-0.717000	16.49544	0.721167	-0.715366
$g(X_6)$	-4.857899	0	9.290112	-23.611002
$g(X_7)$	-0.003050	0	0.000087	-0.000480
$g(X_8)$	-0.000007	2.559363	0	0
$g(X_9)$	-0.000007	0	0	0
$g(X_{10})$	-0.000005	0	0	0
$f(X)$	81843.3	81859.74	85538.48	85535.9611

Table 5.35: Comparison of statistical results using four optimizers for the rolling element bearing problem.

Method	Worst	Mean	Best	SD	NFEs
GA	N.A	N.A	81843.3	N.A	225,000
ABC	78897.81	81496	81859.7416	0.69	10,000
TLBO	80807.8551	81438.987	81859.74	0.66	10,000
WCA	83942.71	83847.16	85538.48	488.30	3950
MBA	84440.1948	85321.4030	85535.9611	211.52	15,100

From Table 5.35, the proposed methods detected the best solution with considerable improvement over other optimizers in this study. In terms of statistical optimization results, the MBA and WCA offered better results with acceptable NFEs against other considered algorithms.

Figure 5.14 compares the convergence rate for used optimizers. From Figure 5.14a it is seen that the convergence rate of ABC and TLBO is nearly same with a slightly higher mean searching capability for the TLBO. However, the MBA and WCA reached the best solution at 302 and 79 iterations, respectively, offering the best solution so far as shown in Figures 5.14b and 5.14c (see Table 5.35).

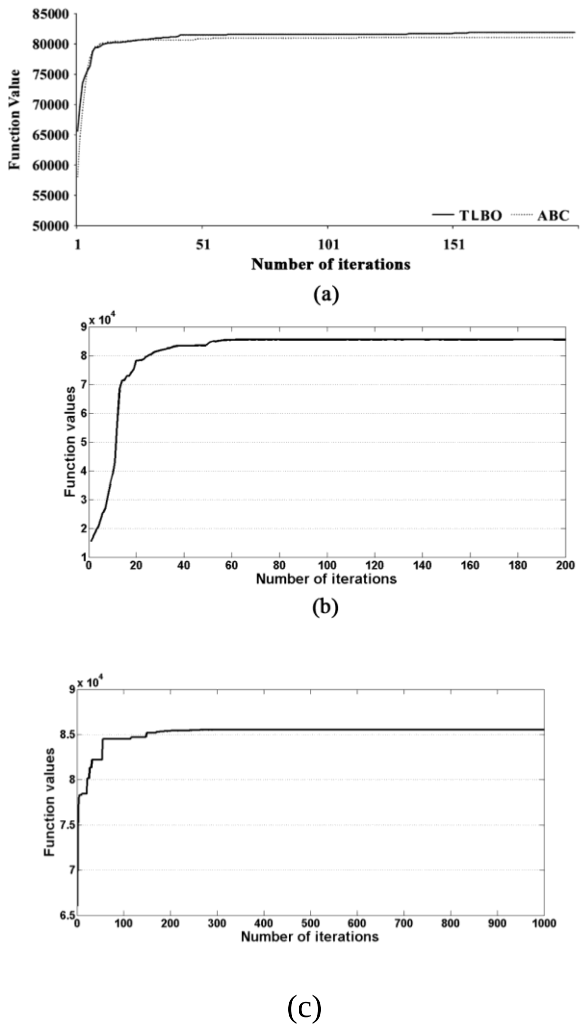


Figure 5.14. Comparison of convergence rate for the rolling element bearing design problem using: (a) TLBO and ABC, (b) WCA, (c) MBA.

These overall engineering optimization results indicate that the proposed methods have the capability in handling various combinatorial optimization problems (COPs) and can offer optimum solutions (near or better than to the best-known results) under lower computational efforts (measure as number of function evaluations). Therefore, it can be concluded that the MBA and WCA may be attractive alternative optimizers for constrained and engineering optimization challenging other metaheuristic methods.

5.4. Truss Structures

In this subsection, the MBA and WCA were tested in a number of discrete optimization benchmark problems. The examples include four well-known truss structures. The proposed MBA and WCA were implemented in MATLAB programming software and runs were performed on Pentium IV 2500 GHz CPU with 4 GB RAM.

For all truss structures, number of population (N_{total}), number of rivers (N_{sr}), and d_{max} (maximum distance between sea and river) were chosen 50, 8, and 1e-5, respetivley, as user parameters for the WCA. Accordingly, for the MBA, the initial parameters were set to 50, 10, and 50,000 for population size, exploration factor (μ), and reduction constant (α), respectively.

Different iteration numbers were used for each structure, with smaller iteration number for smaller number of variables and larger values for large number of variables. The analysis of all trusses has been performed via the finite element method.

The number of design variables for 25, 52, 72, and 200-bar is 8, 12, 16, and 96, respectively. Similarly, the number of constraints for 25, 52, 72, and 200-bar is 80, 144, 198, and 550, respectively.

Based on the dimensions of design variables and constraints, 25 independent runs were performed for the 25-bar truss. However, due to high dimensionality of problems for 52, 72, and 200-bar and the high CPU time for computations, only 20, 15, and 15 independent runs were performed, respectively.

5.4.1. 52-bar planar truss

The 52-bar planar truss, shown in Figure 5.15, has been studied by Wu and Chow (1995), Lee et al. (2005), Li et al. (2009), and Kaveh and Talatahari (2009b). The material density and the modulus of elasticity are 7860 *kg/m³* and E=2.07×10⁵ MPa, respectively.

The stress limitation for each member of this structure is equal to ±180 MPa. This truss has 12 design variables, since its members were divided into 12 groups: (1) A_1-A_4, (2) A_5-A_{10}, (3) A_{11}-A_{13}, (4) A_{14}-A_{17}, (5) A_{18}-A_{23}, (6) A_{24}-A_{26}, (7) A_{27}-A_{30}, (8) A_{31}-A_{36}, (9) A_{37}-A_{39}, (10) A_{40}-A_{43}, (11) A_{44}-A_{49}, and (12) A_{50}-A_{52}.

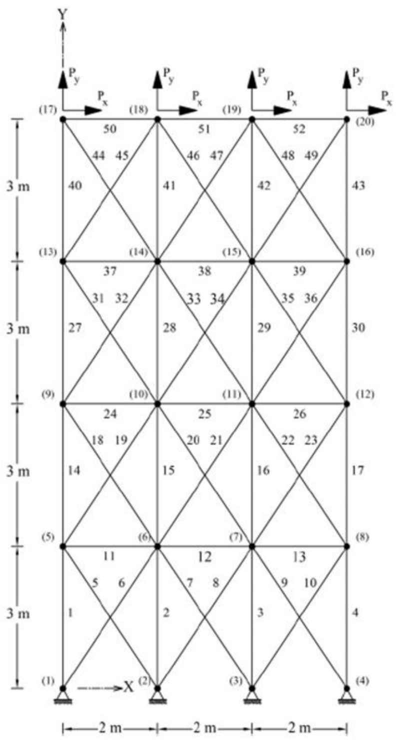


Figure 5.15. 52-bar planar truss.

The discrete variables were selected using American institute of steel construction (AISC) data, which are shown in Table 5.36. Vertical loads were set equal to P_x=100 kN and P_y=200 kN. In general, the problem has a variable dimensionality of 12 and constraint dimensionality of 144 (52 tension constraints, 52 compression constraints, and 40 displacement constraints).

A maximum number of 500 iterations was imposed. The statistical results of the 52-bar truss using the WCA include worst, mean, best solution, and standard deviation which are 1912.646, 1909.856, 1902.995, and 7.09, respectively. Also, the statistical results for the MBA in terms of worst, mean, best solution and standard deviation, namely, 1912.646, 1906.076, 1902.605 and 4.09, respectively.

Table 5.36: Available cross-section areas of the AISC norm.

No.	**in.^{2}**	**mm^2**	**No.**	**in.^{2}**	**mm^2**
1	0.111	71.613	33	3.840	2477.414
2	0.141	90.968	34	3.870	2496.769
3	0.196	126.451	35	3.880	2503.221
4	0.250	161.290	36	4.180	2696.769
5	0.307	198.064	37	4.220	2722.575
6	0.391	252.258	38	4.490	2896.768
7	0.442	285.161	39	4.590	2961.284
8	0.563	363.225	40	4.800	3096.768
9	0.602	388.386	41	4.970	3206.445
10	0.766	494.193	42	5.120	3303.219
11	0.785	506.451	43	5.740	3703.218
12	0.994	641.289	44	7.220	4658.055
13	1.000	645.160	45	7.970	5141.925
14	1.228	792.256	46	8.530	5503.215
15	1.266	816.773	47	9.300	5999.988
16	1.457	939.998	48	10.850	6999.986
17	1.563	1008.385	49	11.500	7419.340
18	1.620	1045.159	50	13.500	8709.660
19	1.800	1161.288	51	13.900	8967.724
20	1.990	1283.868	52	14.200	9161.272
21	2.130	1374.191	53	15.500	9999.980
22	2.380	1535.481	54	16.000	10322.560
23	2.620	1690.319	55	16.900	10903.204
24	2.630	1696.771	56	18.800	12129.008
25	2.880	1858.061	57	19.900	12838.684
26	2.930	1890.319	58	22.000	14193.520
27	3.090	1993.544	59	22.900	14774.164
28	3.130	2019.351	60	24.500	15806.420
29	3.380	2180.641	61	26.500	17096.740
30	3.470	2283.705	62	28.000	18064.480
31	3.550	2290.318	63	30.000	19354.800
32	3.630	2341.931	64	33.500	21612.860

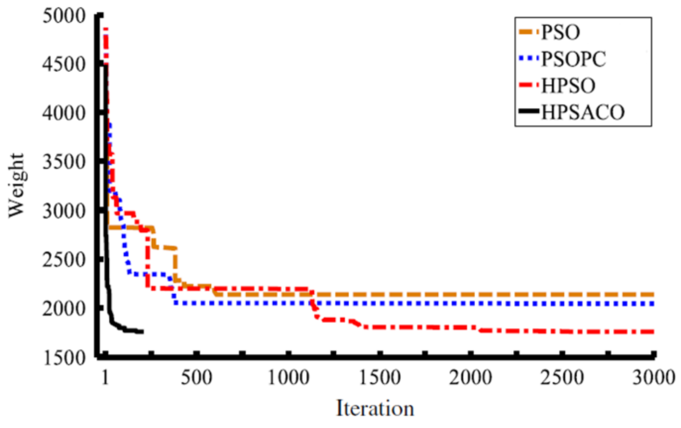
The results obtained using the proposed methods for the 52-bar truss have been compared with the results of SGA (Wu & Chow, 1995), HS, DHPSACO (Kaveh & Talatahari, 2009b), PSO, PSOPC, and HPSO (Li et al., 2009) as shown in Table 5.37. The best optimal design is highlighted in bold in Table 5.37 and it is obvious that the MBA and WCA, both, obtained the better final design than other reported methods.

Table 5.37: Comparison of results for the 52-bar truss obtained using various algorithms.

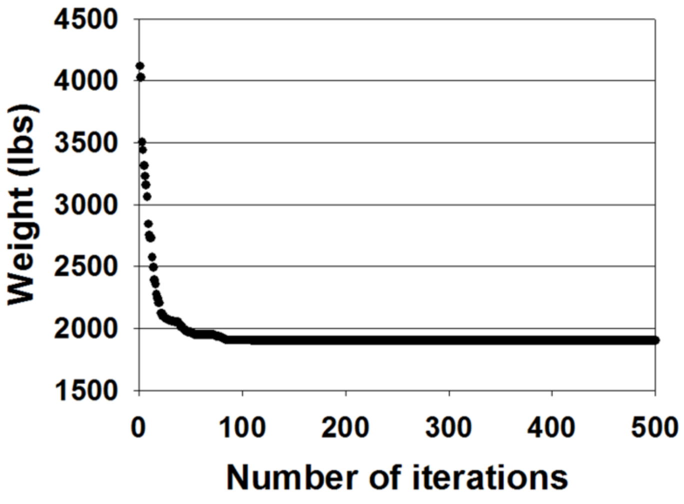
Variables (mm^2)	**SGA**	**HS**	**PSO**	**PSOPC**	**HPSO**	**DHPSACO**	**MBA**	**WCA**
A_1-A_4	4658.05	4658.05	4658.05	5999.98	4658.05	4658.05	4658.05	4658.05
A_5-A_{10}	1161.28	1161.28	1374.19	1008.38	1161.28	1161.28	1161.28	1161.28
A_{11}-A_{13}	645.16	506.45	1858.06	2696.77	363.22	494.19	494.19	494.19
A_{14}-A_{17}	3303.21	3303.21	3206.44	3206.44	3303.21	3303.21	3303.21	3303.21
A_{18}-A_{23}	1045.15	940.00	1283.87	1161.29	940.00	1008.38	940.00	940.00
A_{24}-A_{26}	494.19	494.19	252.26	729.03	494.19	285.16	494.19	494.19
A_{27}-A_{30}	2477.41	2290.31	3303.22	2238.71	2238.70	2290.31	2283.70	2283.70
A_{31}-A_{36}	1045.15	1008.38	1045.16	1008.38	1008.38	1008.38	1008.38	1008.38
A_{37}-A_{39}	285.16	2290.31	126.45	494.19	388.38	388.38	494.19	494.19
A_{40}-A_{43}	1696.77	1535.48	2341.93	1283.87	1283.86	1283.86	1283.86	1283.86
A_{44}-A_{49}	1045.15	1045.15	1008.38	1161.29	1161.28	1161.28	1161.28	1161.28
A_{50}-A_{52}	641.28	506.45	1045.16	494.19	729.25	506.45	494.19	494.19
Weight (kg)	1970.142	1906.76	2230.16	2146.63	1905.495	1904.83	**1902.605**	**1902.605**

Figure 5.16 illustrates the comparison of convergence rates for the 52-bar truss for the PSO, PSOPC, HPSO, DHPSACO, MBA, and WCA. The WCA derived the best solutions at 140 iterations (7100 function evaluations as shown in Figure 5.16c), while MBA detected its best solution at 109 iterations (5450 function evaluations as shown in Figure 5.16b).

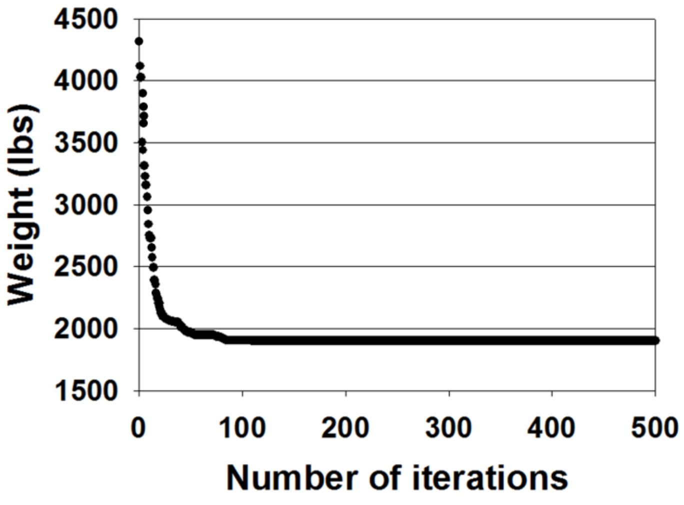
The DHPSACO and HPSO obtained the best solution, while are not as accurate as the results given by the proposed optimizers (MBA and WCA) at 222 and almost 2100 iterations (11100 and almost 105,000 function evaluations), respectively (see Figure 5.16a).



(a)



(b)



(c)

Figure 5.16. Comparison of convergence rates for the 52-bar truss using: (a) DHPSACO (Kaveh & Talatahari, 2009b), (b) MBA, (c) WCA.

In addition, the PSO and PSOPC did not reach the best solution after 3000 iterations (150,000 number of function evaluations), as shown in Figure 5.16a. It is worth to mention that Figures 5.16b and 5.16c represent the weight values for 500 iterations. It is obvious that the MBA and WCA converged to their best optimal designs much faster than competing optimizers outperforming the MBA over WCA in terms of convergence rate and statistical results for the 52-bar truss.

5.4.2. 25-bar spatial truss

The next problem considers the weight minimization of a 25-bar transmission tower (as shown in Figure 5.17) which was studied by Wu and Chow (1995), Rajeev and Krishnamoorthy (1992), Ringertz (1988), Lee et al. (2005), Li et al. (2009), and Kaveh and Talatahari (2009b). The material density and the modulus of elasticity are 0.1 *lb/in*^{3} (0.0272 *N/cm*^{3}) and E=10^4 *ksi* (68947.57 MPa), respectively.

The stress limitation for each member of this structure is equal to ±40,000 *psi* (±275.79 MPa). The allowable displacement for each node in three directions is ±0.35 *in* (±0.00889 *m*). In general, the problem has a variable dimensionality of 8 and a constraint dimensionality of 80 (25 tension constraints, 25 compression constraints and 30 displacement constraints).

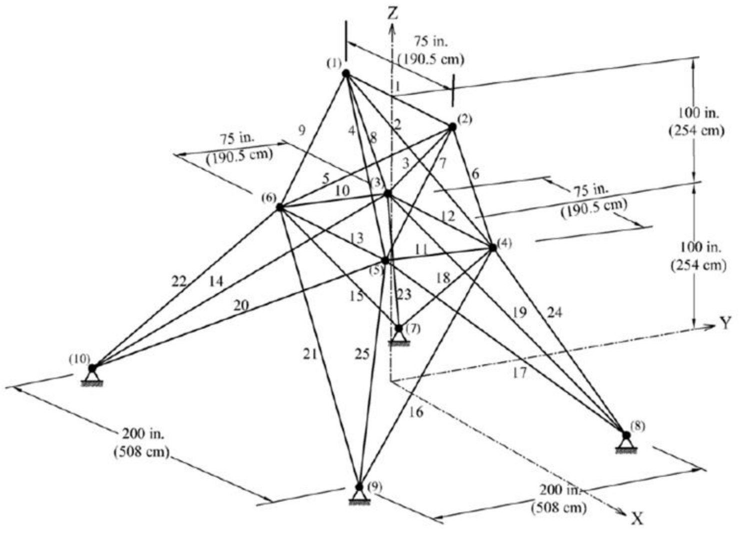


Figure 5.17. 25-bar spatial truss.

The cross-sectional areas of the 25 members were divided into 8 groups: (1) A_1, (2) A_2-A_5, (3) A_6-A_9, (4) A_{10}-A_{11}, (5) A_{12}-A_{13}, (6) A_{14}-A_{17}, (7) A_{18}-A_{21} and

(8) A_{22}-A_{25}. Three optimization cases have been examined: Case 1: the discrete variables are selected from the set D = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4] (in^2); Case 2: the discrete variables are selected from the set D = [0.01, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0] (in^2); Case 3: the design variables are selected from Table 5.36. The load cases applied to the 25-bar truss are described in Table 5.38.

Table 5.38: Load cases for the 25-bar truss.

		Loads		
Load cases	**Nodes**	**P_x (kips)**	**P_y (kips)**	**P_z (kips)**
1	1	1	-10	-10
	2	0	-10	-10
	3	0.5	0	0
	6	0.6	0	0
2	1	0	20	-5
	2	0	-20	-5
3	1	1	10	-5
	2	0	10	-5
	3	0.5	0	0
	6	0.5	0	0

A maximum number of 500 iterations was imposed for all cases. The obtained statistical results of the 25-bar truss structure for Case 1 include worst, mean, best solution, and SD which are 485.379, 484.874, 484.854, and 0.103, respectively, using the WCA.

Similarly, the statistical results of the 25-bar truss for Case 1 for the MBA include worst, mean, best solution, and SD which are 485.048, 484.885, 484.854, and 7.2E-02, respectively. The best and mean numbers of function evaluations (NFEs) are 2100 and 9900, respectively, for the Case 1 using the WCA.

The comparison of optimization results obtained using the SGA (Wu & Chow, 1995), GA (Rajeev & Krishnamoorthy, 1992), Ringertz (1988), HS (Lee et al., 2005), PSO, PSOPC, HPSO (Li et al., 2009), and DHPSACO (Kaveh &

Talatahari, 2009b) for the 25-bar truss structure (for all cases) is given in Tables 5.39 to 5.41.

Table 5.39: Comparison of optimization results obtained using various methods for the 25-bar truss for Case 1.

Variables (in^2)	SGA	GA	HS	PSO	PSOPC	HPSO	MGA	MBA	WCA
A_1	0.1	0.1	0.1	0.4	0.1	0.1	0.1	0.1	0.1
A_2-A_5	0.5	1.8	0.3	0.6	1.1	0.3	0.3	0.3	0.3
A_6-A_9	3.4	2.3	3.4	3.5	3.1	3.4	3.4	3.4	3.4
A_{10}-A_{11}	0.1	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
A_{12}-A_{13}	1.5	0.1	2.1	1.7	2.1	2.1	2.1	2.1	2.1
A_{14}-A_{17}	0.9	0.8	1	1	1	1	1	1	1
A_{18}-A_{21}	0.6	1.8	0.5	0.3	0.1	0.5	0.5	0.5	0.5
A_{22}-A_{25}	3.4	3	3.4	3.4	3.5	3.4	3.4	3.4	3.4
Weight (lb)	486.29	546.01	484.85	486.54	490.16	484.85	484.85	484.85	484.85

Table 5.40: Comparison of results obtained using various methods for the 25-bar truss for Case 2.

Variables (in^2)	SGA	Ringertz	HS	PSO	PSOPC	HPSO	MBA	WCA
A_1	0.4	0.01	0.01	0.01	0.01	0.01	0.01	0.01
A_2-A_5	2	1.6	2	2	2	2	2	2
A_6-A_9	3.6	3.6	3.6	3.6	3.6	3.6	3.6	3.6
A_{10}-A_{11}	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
A_{12}-A_{13}	0.01	0.01	0.01	0.4	0.01	0.01	0.01	0.01
A_{14}-A_{17}	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
A_{18}-A_{21}	2	2	2	1.6	1.6	1.6	1.6	1.6
A_{22}-A_{25}	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Weight (lb)	563.52	568.69	560.59	566.44	560.59	560.59	560.59	560.59

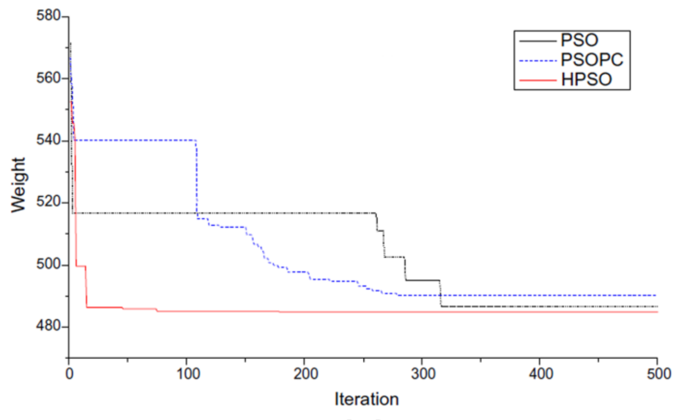
Table 5.41: Comparison of optimization results obtained using different methods for the 25-bar truss for Case 3.

Variables (in^2)	SGA	PSO	PSOPC	HPSO	DHPSACO	MBA	WCA
A_1	0.307	1	0.111	0.111	0.111	0.111	0.111
A_2-A_5	1.99	2.62	1.563	2.13	2.13	2.13	2.13
A_6-A_9	3.13	2.62	3.38	2.88	2.88	2.88	2.88
A_{10}-A_{11}	0.111	0.25	0.111	0.111	0.111	0.111	0.111
A_{12}-A_{13}	0.141	0.307	0.111	0.111	0.111	0.111	0.111
A_{14}-A_{17}	0.766	0.602	0.766	0.766	0.766	0.766	0.766
A_{18}-A_{21}	1.62	1.457	1.99	1.62	1.62	1.62	1.62
A_{22}-A_{25}	2.62	2.88	2.38	2.62	2.62	2.62	2.62
Weight (lb)	556.43	567.49	556.9	551.14	551.14	551.14	551.14

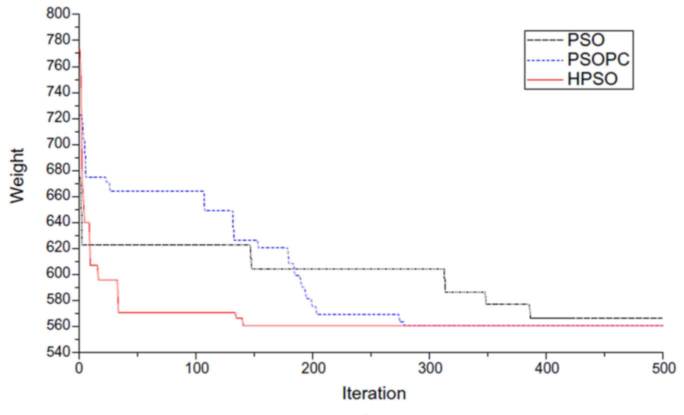
By inspecting Table 5.39, it is evident that the WCA, similarly to the MBA, HS, MGA, and HPSO, reached the best solution. For Cases 2 and 3, most algorithms obtained the best solution as shown in Tables 5.40 and 5.41, respectively. The standard deviation of the WCA, similar to the MBA, for Case 2 is zero, i.e., the worst, means and best solutions have the same values.

The best and averaged NFEs of WCA are 850 and 1900, respectively, for the second case. Similarly, for the Case 3, the best and averaged NFEs are 1450 and 12400, respectively, using the WCA. The gained statistical results of the WCA optimizer for the 25-bar truss for Case 3 include worst, mean, best solution and standard deviation which are 554.743, 552.010, 551.14, and 1.358, respectively. Similarly, the statistical results of the MBA optimizer for the 25-bar truss for Case 3 are 554.067, 551.540, 551.14, and 0.987, respectively.

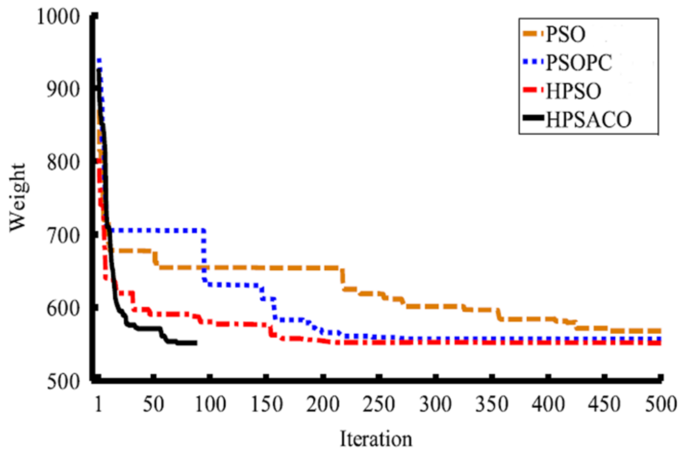
Figure 5.18 shows the comparison of convergence rates of the 25-bar truss for the PSO, PSOPC, HPSO, and DHPSACO for all considered cases. The graph in Figures 5.19 and 5.20 depict the weight values (in lb) with respect to the number of iterations for the three cases for the WCA and MBA, respectively. In order to further clarify the convergence rate results, Figure 5.19a represents the weight values for 100 iterations.



(a)

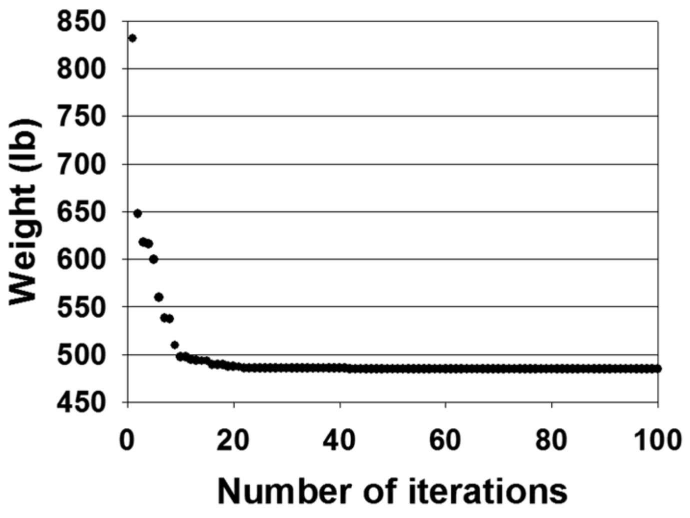


(b)

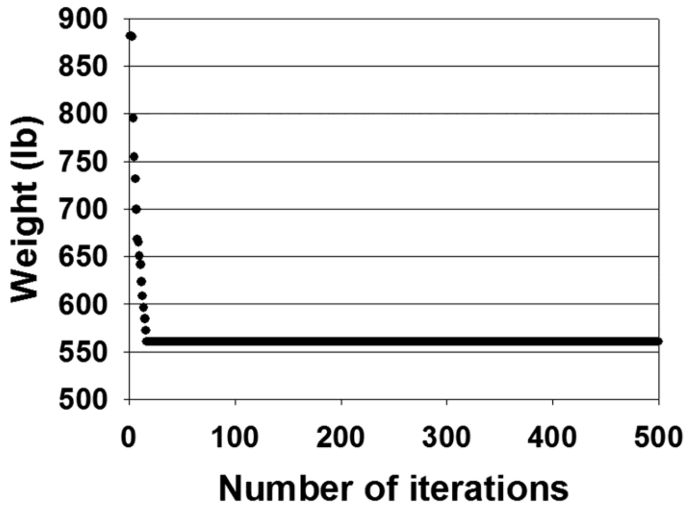


(c)

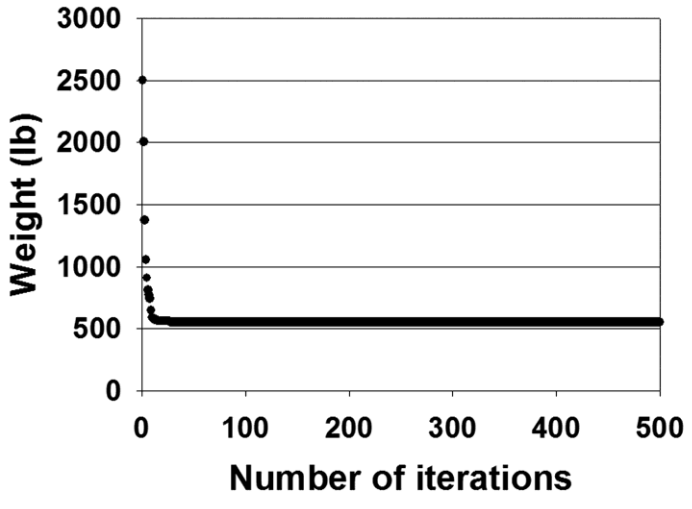
Figure 5.18. Comparison of convergence rates for the 25-bar truss using PSO, PSOPC, HPSO, and DHPSACO (Kaveh & Talatahari, 2009b): (a) Case 1, (b) Case 2, (c) Case 3.



(a)

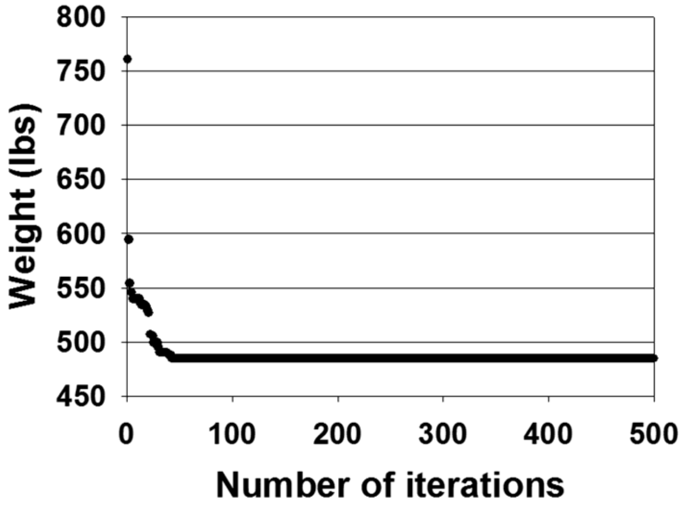


(b)

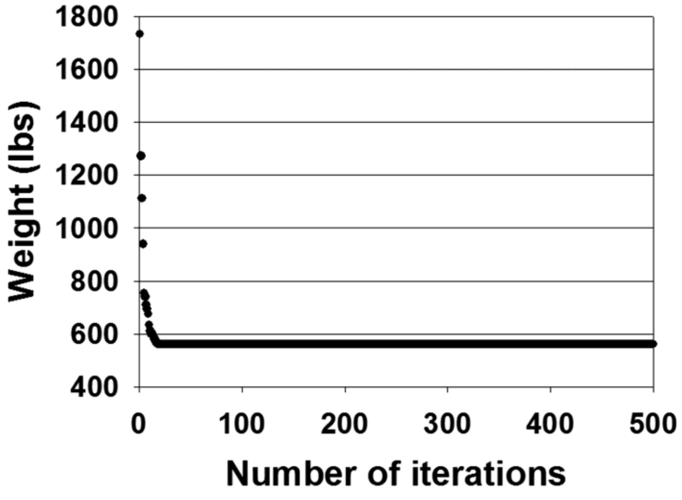


(c)

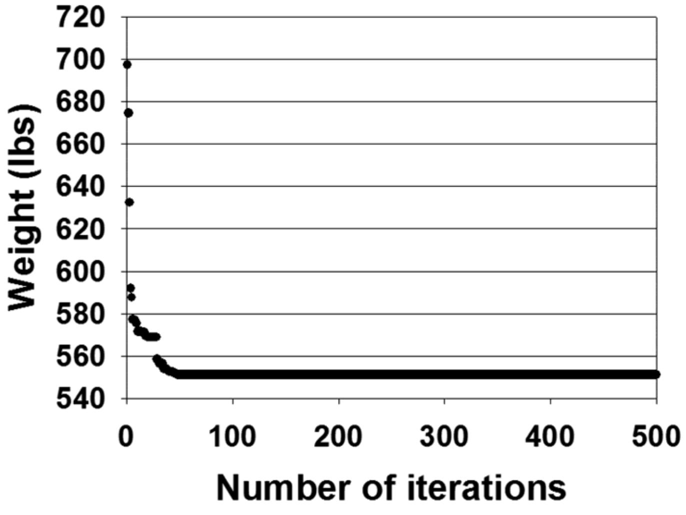
Figure 5.19. Weight (lb) evolution history for the 25-bar truss using WCA: (a) Case 1, (b) Case 2, (c) Case 3.



(a)



(b)



(c)

Figure 5.20. Weight (lb) evolution history for the 25-bar truss using MBA: (a) Case 1, (b) Case 2, (c) Case 3.

By observing Figures 5.18a, 5.19a, and 5.20a (Case 1), the WCA derived the best solution at 42 iterations (2100 function evaluations), while MBA and HPSO needed 43 and 75 iterations (2150 and 3750 function evaluations). In contrast, the PSO and PSOPC did not reach the best solution after 500 iterations as depicted in Figure 5.18a.

From Figures 5.18b, 5.19b, and 5.20b (Case 2), the WCA obtained the best solution at 17 iterations (850 function evaluations), while the MBA, HPSO, and PSOPC reached their best solution at 19, less than 150, and 300 iterations (950, less than 7500, and 1500 function evaluations), respectively. The PSO did not find the best solution after 500 iterations compared to other algorithms as shown in Figure 5.18b.

As it can be observed from Figures 5.18c, 5.19c, and 5.20c (Case 3), the WCA detected the best solution at 29 iterations (1450 function evaluations), while MBA, DHPSACO, and HPSO found their best solution at 48, less than 100, and at almost 200 iterations (2400, less than 5000, and at almost 10000 function evaluations), respectively. Conversely, the PSO and PSOPC did not find the best solution after 500 iterations, as shown in Figure 5.18c. In this problem, the WCA slightly outperformed MBA in terms of convergence rate (computational effort).

5.4.3. 72-bar spatial truss

The 72-bar spatial truss, shown in Figure 5.21, has been studied by Wu and Chow (1995), Lee et al. (2005), Kaveh and Talatahari (2009b), and Li et al. (2009). The material density is 0.1 *lb/in*^{3} and the modulus of elasticity is 10,000 ksi. The members are subjected to stress limitations of ±25 ksi.

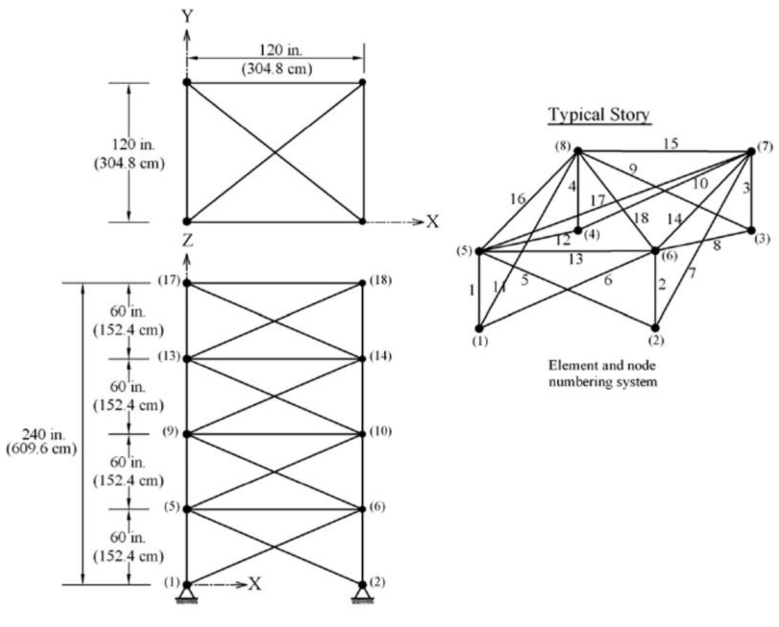


Figure 5.21. 72-bar spatial truss.

The uppermost nodes are subjected to displacement limits of ±0.25 in both in *x* and *y* directions. Hence, the problem has a variable dimensionality of 16 and constraint dimensionality of 198 (72 tension constraints, 72 compression constraints, and 54 displacement constraints). Two load cases were considered as described in Table 5.42.

Table 5.42: Load cases for the 72-bar spatial truss.

Nodes	**Load case 1**			**Load case 2**		
	P_x **(kips)**	P_y **(kips)**	P_z **(kips)**	P_x **(kips)**	P_y **(kips)**	P_z **(kips)**
17	5	5	-5	0	0	-5
18	0	0	0	0	0	-5
19	0	0	0	0	0	-5
20	0	0	0	0	0	-5

The 72 members were divided into 16 groups as follows: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}–A_{22}, (6) A_{23}–A_{30} (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15) A_{67}–A_{70}, and (16) A_{71}–A_{72}. Two optimization cases have been studied: Case 1: the discrete variables are selected from the set D = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1,3.2] (*in*^{2}), Case 2: the discrete variables were selected from Table 5.36. For comparison with other algorithms, a maximum number of 1000 iterations was imposed. The comparison of obtained statistical optimization results using the WCA and MBA are preseneted in Table 5.43.

Table 5.43: Comparison of statistical results using the WCA and MBA for the 72-bar truss for Cases 1 and 2.

Methods	**Best Solution**		**Mean Solution**		**Worst Solution**		**SD**	
	Case 1	**Case 2**	**Case 1**	**Case 2**	**Case 1**	**Case 2**	**Case 1**	**Case 2**
MBA	385.542	390.739	387.665	395.432	390.615	399.49	1.62	3.04
WCA	385.542	389.334	385.842	389.941	386.80	393.778	0.55	1.43

The minimum and averaged NFEs using the WCA for the Case 1 are 3200 and 19750, respectively. Accordingly for the Case 2, the best and mean NFEs by WCA are 4600 and 26050, accordingly. Tables 5.44 and 5.45 show the comparisons of results obtained by the SGA, HS, PSO, PSOPC, HPSO, DHPSACO, MBA, and WCA for the 72-bar truss for Cases 1 and 2, respectively.

Table 5.44: Comparison of the best results obtained using various methods for Case 1 for the 72-bar truss.

Varibales (*in*^{2})	**SGA**	**HS**	**PSO**	**PSOPC**	**HPSO**	**DHPSACO**	**MBA**	**WCA**
A_1-A_4	1.5	1.9	2.6	3.0	2.1	1.9	2.0	1.9
A_5-A_{12}	0.7	0.5	1.5	1.4	0.6	0.5	0.6	0.5
A_{13}-A_{16}	0.1	0.1	0.3	0.2	0.1	0.1	0.4	0.1
A_{17}-A_{18}	0.1	0.1	0.1	0.1	0.1	0.1	0.6	0.1
A_{19}-A_{22}	1.3	1.4	2.1	2.7	1.4	1.3	0.5	1.4
A_{23}-A_{30}	0.5	0.6	1.5	1.9	0.5	0.5	0.5	0.5
A_{31}-A_{34}	0.2	0.1	0.6	0.7	0.1	0.1	0.1	0.1
A_{35}-A_{36}	0.1	0.1	0.3	0.8	0.1	0.1	0.1	0.1
A_{37}-A_{40}	0.5	0.6	2.2	1.4	0.5	0.6	1.4	0.5
A_{41}-A_{48}	0.5	0.5	1.9	1.2	0.5	0.5	0.5	0.5
A_{49}-A_{52}	0.1	0.1	0.2	0.8	0.1	0.1	0.1	0.1
A_{53}-A_{54}	0.2	0.1	0.9	0.1	0.1	0.1	0.1	0.1
A_{55}-A_{58}	0.2	0.2	0.4	0.4	0.2	0.2	1.9	0.2
A_{59}-A_{66}	0.5	0.5	1.9	1.9	0.5	0.6	0.5	0.6
A_{67}-A_{70}	0.5	0.4	0.7	0.9	0.3	0.4	0.1	0.4
A_{71}-A_{72}	0.7	0.6	1.6	1.3	0.7	0.6	0.1	0.6
Weight (lb)	400.66	387.94	1089.88	1069.79	388.94	385.54	385.54	385.54

Table 5.45: Comparison of the optimum results obtained using different optimizers for Case 2 for the 72-bar truss.

Varibales (*in*^{2})	**SGA**	**PSO**	**PSOPC**	**HPSO**	**DHPSACO**	**MBA**	**WCA**
A_1-A_4	0.196	7.22	4.49	4.97	1.800	0.196	1.99
A_5-A_{12}	0.602	1.80	1.457	1.228	0.442	0.563	0.442
A_{13}-A_{16}	0.307	1.13	0.111	0.111	0.141	0.442	0.111
A_{17}-A_{18}	0.766	0.196	0.111	0.111	0.111	0.602	0.111
A_{19}-A_{22}	0.391	3.09	2.620	2.88	1.228	0.442	1.228
A_{23}-A_{30}	0.391	0.785	1.130	1.457	0.563	0.442	0.563
A_{31}-A_{34}	0.141	0.563	0.196	0.141	0.111	0.111	0.111
A_{35}-A_{36}	0.111	0.785	0.111	0.111	0.111	0.111	0.111
A_{37}-A_{40}	1.800	3.09	1.266	1.563	0.563	1.266	0.563
A_{41}-A_{48}	0.602	1.228	1.457	1.228	0.563	0.563	0.563
A_{49}-A_{52}	0.141	0.111	0.111	0.111	0.111	0.111	0.111
A_{53}-A_{54}	0.307	0.563	0.111	0.196	0.250	0.111	0.111
A_{55}-A_{58}	1.563	1.990	0.442	0.391	0.196	1.800	0.196
A_{59}-A_{66}	0.766	1.620	1.457	1.457	0.563	0.602	0.563
A_{67}-A_{70}	0.141	1.563	1.228	0.766	0.442	0.111	0.391
A_{71}-A_{72}	0.111	1.266	1.457	1.563	0.563	0.111	0.563
Weight (lb)	427.20	1209.48	941.82	933.09	393.380	390.73	**389.334**

By observing Table 5.44, the WCA, similarly to the DHPSACO and MBA, outperformed the rest of considered methods with respect to the best solution for the Case 1. Nevertheless, the design variables of the WCA were different from those of the DHPSACO and MBA.

As shown in Table 5.45, the WCA is superior to the other reported algorithms with respect to the derived solutions for Case 2. The best obtained solution by WCA is highlited in bold in Table 7.45. Figures 7.22 to 7.24 depict the convergence rate of the 72-bar truss for the two cases obtained by Li et al. (2009) and WCA, and MBA, respectively.

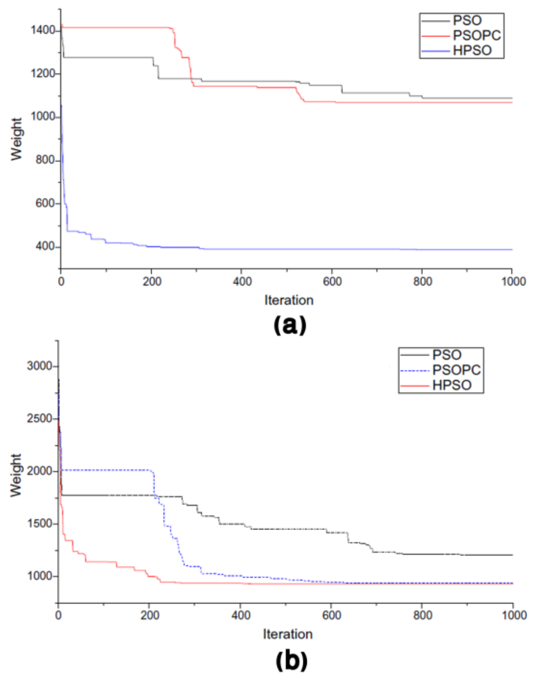


Figure 5.22. Comparison of convergence rates for the 72-bar truss using PSO, PSOPC, and HPSO: (a) Case 1, (b) Case 2.

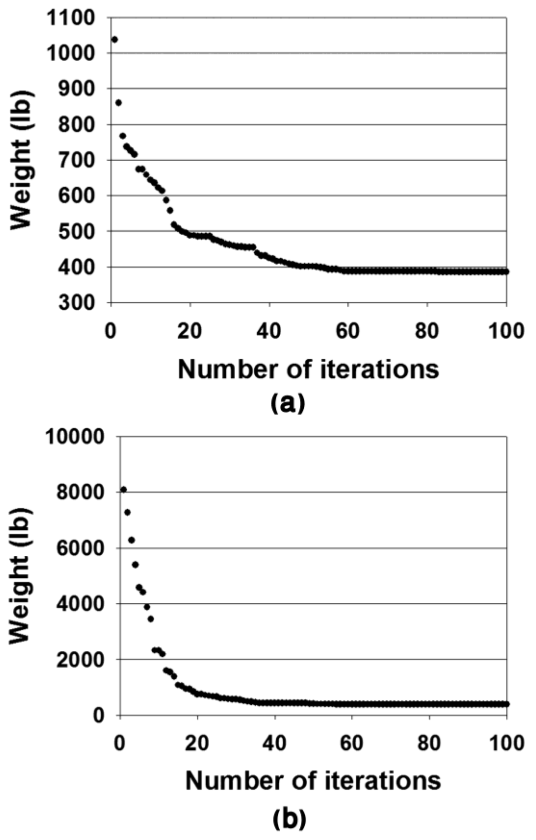
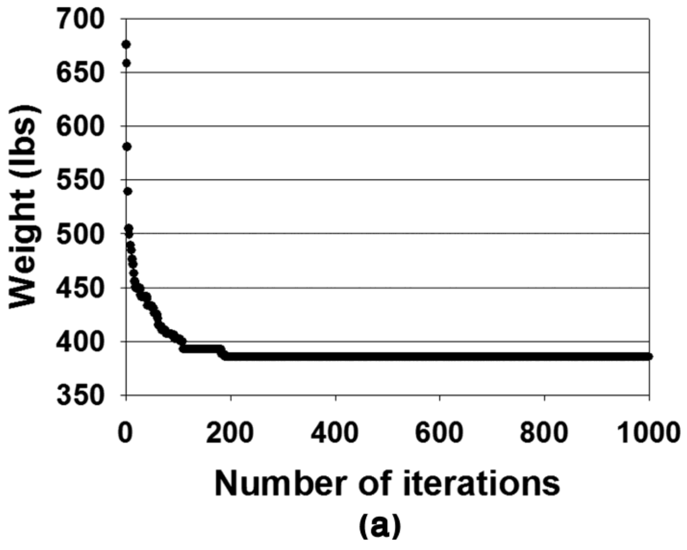


Figure 5.23. Weight (lbs) evolution history for the 72-bar truss using the WCA: (a) Case 1, (b) Case 2.



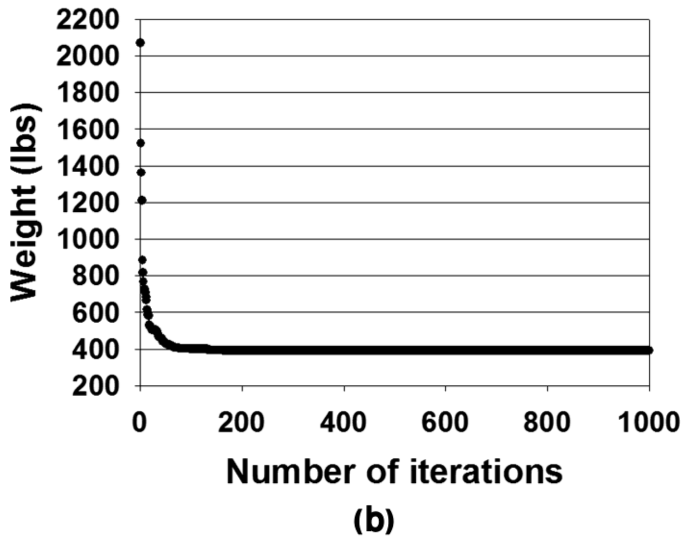


Figure 5.24. Weight (lbs) evolution history for the 72-bar truss using the MBA: (a) Case 1, (b) Case 2.

As it can be seen in Figures 5.22a, 5.23a, and 5.24a (Case 1), the WCA obtained the best solution at 64 iterations (3200 function evaluations), while the MBA, DHPSACO and HPSO found the best solution at 189, 213, and almost 250 iterations (9450, 10650 and almost 12500 function evaluations), respectively. In contrast, the PSO and PSOPC, as shown in Figure 5.22a, did not get the best solution after 1000 iterations.

From Figures 5.22b, 5.23b, and 5.24b for Case 2, the WCA obtained the best solution at 92 iterations (4600 function evaluations), while the MBA and

DHPSACO found the optimum (which is not as optimal as the WCA) at 232 and more than 250 iterations (11600 and more than 12500 function evaluations), respectively.

Conversely, the HPSO, PSO, and PSOPC did not reach the best solution after 1000 iterations as shown in Figure 5.22b. For more clarification on the convergence rate results, Figures 5.23a and 5.23b present the weight evolution history only for 100 iterations. For the 72-bar truss, the WCA is superior to the MBA having faster convergence rate and high quality solutions.

5.4.4. 200-bar truss

Schematic view of 200-bar truss structure is shown in Figure 5.25. The 200-bar truss is proposed and optimized under various types of constraints and several design variables. In this research, the elements of this truss are grouped into 96 sets (design variables) as given in Ghasemi et al. (1999).

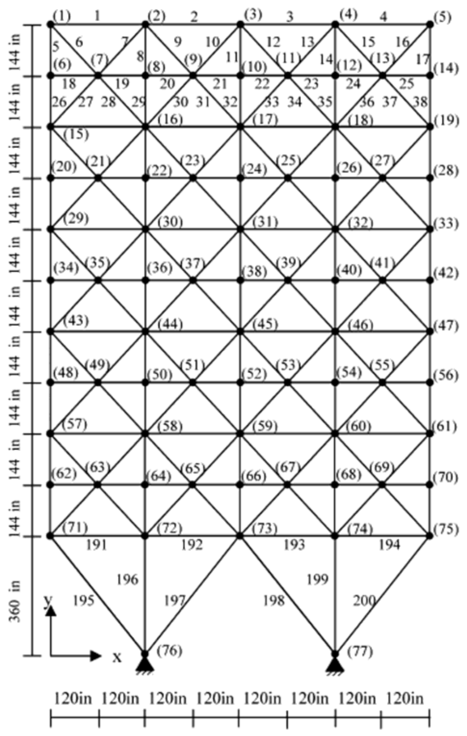


Figure 5.25. 200-bar planar truss.

The detail of grouping for the 200-bar truss is specified in Table 5.46. In terms of mechanical and material properties, modulus of elasticity of 30,000 *ksi* and density of material of 0.283 lb/in^3 are considered for this truss structure. The acceptable displacement is restricted to 0.5 *in* and the permissible stress is set to ±30 ksi.

Table 5.46: Group membership for the 200-bar truss.

No.	Members	No.	Members	No.	Members	No.	Members
1	1,4	25	46,52	49	102,114	73	146
2	2,3	26	47,51	50	103,113	74	153,156
3	5,17	27	48,50	51	104,112	75	154,155
4	6,16	28	49	52	105,111	76	157,169
5	7,15	29	57,58,61,62	53	106,110	77	158,168
6	8,14	30	59,60	54	107,109	78	159,167
7	9,13	31	64,76	55	108	79	160,166
8	10,12	32	65,75	56	115,118	80	161,165
9	11	33	66,74	57	116,117	81	162,164
10	132,139,170,177,18,25,56,63	34	67,73	58	119,131	82	163
11	19,20,23,24	35	68,72	59	120,130	83	171,172,175,176
12	21,22	36	69,71	60	121,129	84	173,174
13	26,38	37	70	61	122,128	85	178,190
14	27,37	38	77,80	62	123,127	86	179,189
15	28,36	39	78,79	63	124,126	87	180,188
16	29,35	40	81,93	64	125	88	181,187
17	30,34	41	82,92	65	133,134,137,138	89	182,186
18	31,33	42	83,91	66	135,136	90	183,185
19	32	43	84,90	67	140,152	91	184
20	39,42	44	85,89	68	141,151	92	191,194
21	40,41	45	86,88	69	142,150	93	192,193
22	43,55	46	87	70	143,149	94	195,200
23	44,54	47	95,96,99,100	71	144,148	95	196,199
24	45,53	48	97,98	72	145,147	96	197,198

The next is a list of 30 discrete values for decision variables implemented to solve this truss: A = [0.100, 0.347, 0.440, 0.539, 0.954, 1.081, 1.174, 1.333, 1.488, 1.764, 2.142, 2.697, 2.800, 3.131, 3.565, 3.813, 4.805, 5.952, 6.572, 7.192, 8.525, 9.300, 10.850, 13.330, 14.290, 17.170, 19.180, 23.680, 28.080, 33.700 in^2]. The 200-bar truss is imposed to three various load cases which they are given as follows: Load case 1: 1 *kip* operating in the positive *x* direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71. Load case 2: 10 *kips* imposing in the negative *y* direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60, 61, 62, 64, 66, 68, 70, 71,72, 73,74, and 75. For load Case 3: Cases 1 and 2 are mingled. In this research, similar to other researches, load Case 3 is considered.

This benchmark problem was considered for optimization purposes using different methods such as modified GA (MGA) (Dede et al., 2011), GA (Ghasemi et al., 1999), evolution strategies (ES) (Cai & Thierauf, 1993). The WCA and MBA were applied for the optimization of the 200-bar truss and the obtained optimization results and comparisons are given.

The optimum configurations and the best obtained weight found by the MBA and WCA are given in Tables 5.47 and 5.48, respectively. Table 5.49 represents the comparisons of obtained statistical results (best, mean, worst, and SD) for the WCA and MBA.

Table 5.47: Best optimum results obtained using the MBA for the 200-bar truss.

No.	**Area (in^2)**	**No.**	**Area (in^2)**	**No.**	**Area (in^2)**	**No.**	**Area (in^2)**
1	0.347	25	2.697	49	7.192	73	9.3
2	0.1	26	0.44	50	0.1	74	1.764
3	5.952	27	0.347	51	2.697	75	1.333
4	0.347	28	3.813	52	7.192	76	4.805
5	0.1	29	0.1	53	0.347	77	4.805
6	2.697	30	0.1	54	0.1	78	0.1
7	0.347	31	6.572	55	7.192	79	13.33
8	0.347	32	0.1	56	0.1	80	1.764
9	2.697	33	2.142	57	0.1	81	0.539
10	0.1	34	5.952	58	7.192	82	8.525
11	0.1	35	0.1	59	0.1	83	0.1
12	0.44	36	0.347	60	1.764	84	0.1
13	4.805	37	7.192	61	10.85	85	3.813
14	0.1	38	1.333	62	1.333	86	0.1
15	0.539	39	0.347	63	0.1	87	5.952
16	3.813	40	6.572	64	6.572	88	14.29
17	0.1	41	2.142	65	0.539	89	0.954
18	0.347	42	0.1	66	0.347	90	2.142
19	8.525	43	7.192	67	8.525	91	10.85
20	1.081	44	0.1	68	2.142	92	3.565
21	1.174	45	0.44	69	0.347	93	1.488
22	7.192	46	4.805	70	14.29	94	5.952
23	1.488	47	0.1	71	0.44	95	19.18
24	0.1	48	0.1	72	1.333	96	6.572
Weight = **27532.95 lb** (12488.73 Kg)							
Maximum Constraint Violation = -2.9048e-005							

Table 5.48: Best configurations obtained by the WCA for the 200-bar truss.

No.	Area (in^2)	No.	Area (in^2)	No.	Area (in^2)	No.	Area (in^2)
1	0.347	25	4.805	49	10.85	73	7.192
2	0.347	26	0.1	50	0.1	74	0.1
3	5.952	27	0.347	51	0.539	75	0.539
4	0.1	28	7.192	52	5.952	76	13.33
5	0.1	29	0.347	53	0.539	77	1.174
6	2.697	30	0.1	54	0.539	78	0.44
7	0.539	31	13.33	55	7.192	79	10.85
8	0.954	32	0.1	56	0.1	80	1.333
9	2.142	33	0.44	57	0.1	81	0.1
10	0.1	34	8.525	58	14.29	82	9.30
11	0.1	35	0.539	59	0.1	83	0.347
12	0.1	36	0.347	60	0.1	84	0.954
13	3.813	37	6.572	61	10.85	85	10.85
14	0.44	38	0.1	62	0.1	86	0.347
15	0.1	39	0.1	63	1.174	87	1.174
16	4.805	40	8.525	64	7.192	88	7.192
17	0.1	41	0.347	65	0.44	89	0.1
18	0.539	42	0.44	66	0.954	90	2.697
19	3.813	43	5.952	67	13.33	91	8.525
20	0.347	44	1.488	68	0.954	92	7.192
21	0.347	45	0.1	69	0.539	93	8.525
22	13.33	46	10.85	70	9.3	94	10.85
23	0.1	47	0.347	71	0.954	95	9.30
24	0.1	48	0.347	72	0.347	96	9.30
Weight = 29304.76 lb (13292.41 Kg)							
Maximum Constraint Violation = -3.1556e-04							

Table 5.49: Comparison of statistical results obtained using the WCA and MBA.

Methods	Best Solution	Mean Solution	Worst Solution	SD	NFEs
WCA	29,304.76	29,885.78	30,188.52	409.75	30,000
MBA	27,532.95	28,667.09	29,742.63	312.68	30,000

By observing Table 5.49, in this case, the MBA is superior to the WCA in terms of statistical results obtaining minimum weight and convergence rate (see Figure 5.27b). Using the similar information for this truss, Ghasemi et al. (1999) obtained the minimum weight of the 200-bar truss as 30,905 lb and 31,109 lb by GA2-800 and GA2-100, respectively.

Cai and Thierauf (1993) has detected the minimum weight of the 200-bar truss as 31,014 lb. The weight obtained by Dede et al. (2011) is 30,868.45 lb. The

WCA has found its best solution of 29,304.76 lb, while the MBA has detected the minimum weight of 27,532.95 lb for the 200-bar truss structure.

Figure 5.26 demonstrates the comparisons of convergence rate for considered methods. As shown in Figure 5.26a, the convergence is obtained around 612th iteration (122,400 function evaluations) using the MGA. The MBA and WCA have converged to their best solutions faster and more accurate than the MGA after 600 iterations (30,000 function evaluations), as shown in Figure 5.26b.

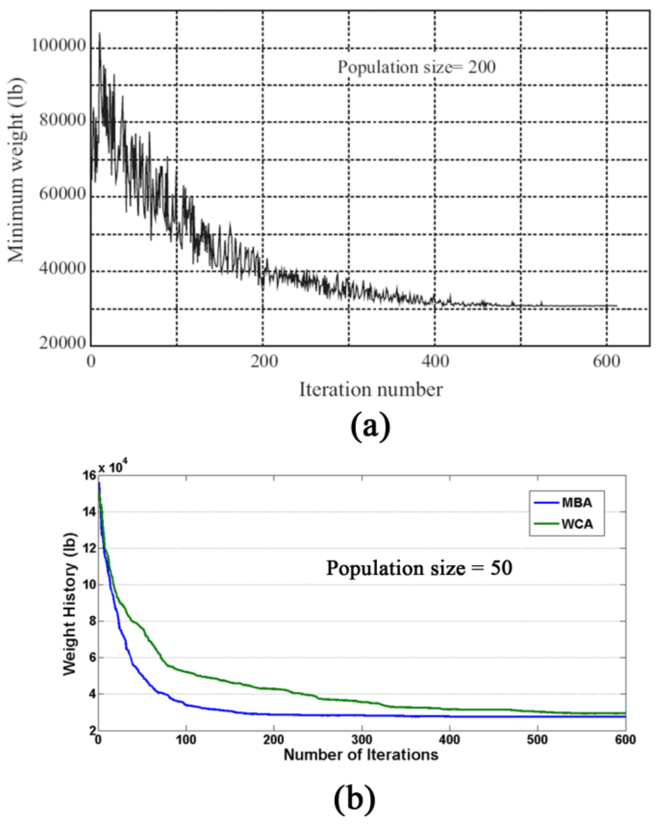


Figure 5.26. Comparisons of convergence rate for the 200-bar truss using: (a) GA (Dede et al., 2011), (b) WCA and MBA.

From Figure 5.26b, the MBA (represented by blue line) is converged to its near optimum solution after almost 200 iterations, while the WCA (represented by green line) has reached the best configuration after almost 550 iterations. Interestingly, in terms of computational efforts, the number of population (individuals) for the MBA and WCA was set to 50, while for the MGA the population size was taken 200.

Population size of 150 were also chosen for the MBA and WCA, however, the optimization resulted were the same optimum solutions as of the 50 population. Hence, the MBA and WCA are capable of solving complex problems using smaller population size which results in less number of function evaluations (computational effort).

By comparing the 72-bar truss (16 design variables and 198 constraints) and 200-bar truss (96 design variables and 550 constraints) problems, one may conclude that when the number of design variables and number of constraints increase, the MBA and WCA offer better efficiency, performance, and reliability in finding best optimal design compared with other considered algorithms needing less number of function evaluations and having fast convergence rate.

In summary, the applications of the MBA and WCA were tested on several benchmark constrained and engineering design problems in this chapter. Comprehensive comparisons were carried out in order to have fair judgment about the performance and efficiency of the proposed optimizers. In general, for most considered problems, the MBA and WCA offered better statistical optimization results having less number of function evaluations (computational time) compared with other reported optimizers.

CHAPTER 6 : CONCLUSIONS

6.1. Conclusions

In this thesis, two novel optimization engines introduced, the so-called mine blast algorithm (MBA) and water cycle algorithm (WCA). The fundamental concepts and ideas to formulate the MBA are derived from the explosion of mine bombs in real world. Accordingly, the fundamental concepts and ideas which underlie the WCA are inspired from nature and based on the water cycle process in real world.

Thereafter, the WCA and MBA with embedded constraint handling approaches are proposed for solving a number of unconstrained, constrained benchmark optimization, engineering design problems, and truss structures (2D and 3D). The statistical optimization results based on the comparisons of the efficiency of the proposed optimizers against numerous other optimization methods, illustrate the attractiveness of the proposed methods for handling numerous types of constraints.

The obtained optimization results show that the proposed algorithms generally offer better solutions than other optimizers considered in this thesis in addition to their efficiencies in terms of having less number of function evaluations (computational time) for almost every problem. In general, the WCA and MBA offer competitive solutions compared with other metaheuristic optimizers based on the reported and experimental results in this research.

However, the computational efficiency and quality of solutions given by the WCA and MBA may depend on the nature and complexity of the underlined problem. This also applies to the performance of most metaheuristic methods. The proposed methods may be used for solving the real world optimization problems which require significant computational efforts efficiently with acceptable degree of accuracy for the solutions.

6.2. Future researches

Although the proposed methods (MBA and WCA) at their present format show good potential to be used as a global optimization algorithm, they may be improved in terms of mathematical formulation. For instance, other mathematical modeling for the calculating the location of mine bombs, and the reduction of distance for shrapnel pieces (for the MBA) may be considered as future research. Furthermore, the effects of hybridization of MBA with WCA and/or other methods may also be investigated.

In light of the needs of industry and the nature of real-life problems that are highly-dimensioned, the proposed optimizers can be applied to large-scale optimization and multi-objective problems. The objective of these problems may be the cost of consumed materials, the weight of highly-bar trusses, the layout of a factory from a high-dimensional point of view, and also other objectives which can be considered, simultaneously.

REFERENCES

Adeli, H. (2000). High-performance computing for large-scale analysis, optimization, and control. *Journal of Aerospace Engineering, ASCE, 13(1),* 1-10.

Adeli, H., & Park, H. S. (1995a). A neural dynamics model for structural optimization – Theory. *Computers and Structures, 57(3),* 383-390.

Adeli, H., & Park, H. S. (1995b). Optimization of space structures by neural dynamics. *Neural Networks, 8(5),* 769-781.

Adeli, H., & Karim, A. (1997a). Neural Dynamics model for optimization of cold-formed steel beams. *Journal of Structural Engineering, ASCE, 123(11),* 1535-1543.

Adeli, H., & Karim, A. (1997b). Scheduling/cost optimization and neural dynamics model for construction. *Journal of Construction Management and Engineering, ASCE, 123(4),* 450-458.

Adeli, H., & Saleh, A. (1997). Optimal control of adaptive/smart bridge structures. *Journal of Structural Engineering, ASCE, 123(2),* 218-226.

Adeli, H., & Sarma, K. (2006). *Cost Optimization of Structures – Fuzzy Logic, Genetic Algorithms, and Parallel Computing*. John Wiley and Sons, West Sussex, United Kingdom.

Adeli, H., & Cheng, N. T. (1994a). Concurrent genetic algorithms for optimization of large structures. *Journal of Aerospace Engineering, ASCE, 7(3),* 276-296.

Adeli, H., & Cheng, N. T. (1994b). Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering, ASCE, 7(1),* 104-118.

Ahrari, A., & Aatai, A. A. (2010). Grenade Explosion Method-A novel tool for optimization of multimodal functions. *Applied Soft Computing, 10,* 1132-1140.

Ahrari, A., Saadatmand, M. R., Shariat-Panahi, M., & Atai, A. A. (2010). On the limitations of classical benchmark functions for evaluating robustness of evolutionary algorithms. *Applied Mathematics and Computation, 215,* 3222-3229.

Akay, B., & Karaboga, D. (2010). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing,* DOI:10.1007/s10845-010-0393-4.

Ali, A. A., & Z. Kajee-Bagdadi, Z. (2009). A local exploration-based differential evolution algorithm for constrained global optimization. *Applied Mathematics and Computation, 208,* 31-48.

Amirjanov, A. (2006). The development of a changing range genetic algorithm. *Computer Methods in Applied Mechanics and Engineering, 195,* 2495-2508.

Arora, J. S. (1989). *Introduction to optimum design*. New York, McGraw-Hill.

Atashpaz-Gargari E., & Lucas, C. (2007). Imperialist Competitive Algorithm: An Algorithm for Optimization Inspires by Imperialistic Competition. *IEEE Congress on Evolutionary Computation,* Singapore.

Aymerich, F., & Serra, M. (2008). Optimization of laminate stacking sequence for maximum buckling load using the ant colony optimization (ACO) metaheuristic. *Composites Part A: Applied Science and Manufacturing, 39(2),* 262-272.

Bakwad, K. M., Pattnaik, S. S., Sohi, B. S., Devi, S., Gollapudi, S., Sagar, C. V., & Patra, P. K. (2010). Multimodal function optimization using synchronous bacterial foraging optimization technique. *IETE journal of research, 56(2),* 80-87.

Balling, R. J. (1991). Optimal steel frame design by simulated annealing. *Journal of Structural Engineering-ASCE, 117,* 1780-1795.

Balling, R. J. (1996). Application of the simulated annealing algorithm to structural design. In Grierson DE, Hajela P,editors. *Emergent computing methods.*

Becerra, R., & Coello, C. A. C. (2006). Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering, 195,* 4303-4322.

Bell, J. E., & McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics, 18(1),* 41-48.

Bennage, W. A., & Dhingra, A. K. (1995). Single and multi-objective structural optimization in discrete–continuous variables using simulated annealing. *International Journal for Numerical Methods in Engineering, 38,* 2753-2773.

Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories, Inform. *Sciences, 176,* 937-971.

Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing, 1,* 3-52.

Box, M. J. (1965). A comparison of several current optimization methods and the use of transformations in constrained problems. *Computer Journal, 8*, 67-77.

Bracken, J., & Mccormick, G. P. (1968). *Selected applications of nonlinear programming*. John Wiley & Sons, New York.

Broyden, G. C. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematic of Computation, 19,* 577-593.

Cai, J., & Thierauf, G. (1993). Discrete structural optimization using evolution strategies. In B.H.V. Topping, A.I. Khan (Eds.), *Neural Networks and Combinatorial Optimization in Civil and Structural Engineering* (pp. 95-100). Civil-Comp, Edinburg.

Cauchy, A. (1847). Methodes generales pour la resolution des syst‘emes dequations simultanees. *Comptes Rendus Hebd. S´eances Acad. Science, Paris, 25,* 536-538.

Ceranic, B., Fryer, C., & Baines, R. W. (2001). An application of simulated annealing to the optimum design of reinforced concrete retaining structures. *Computers & Structures, 79,* 1569-1581.

Chen, G. S., Bruno, R. J., & Salama, M. (1991). Optimal placement of active/passive members in truss structures using simulated annealing. *AIAA Journal, 29,* 1327-1334.

Chootinan, P., & Chen, A. (2006). Constraint handling in genetic algorithms using a gradient-based repair method. *Computers & Operation Research, 33,* 2263-2281.

Chu, C. L., Xue, X. Y., Zhu, J. C., & Yin, Z. D. (2006). In vivo study on biocompatibility and bonding strength of hydroxyapatite-20vol%Ti composite with bone tissues in the rabbit. *Biomedical Materials and Engineering, 16(3),* 203-213.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191,* 1245-1287.

Coello, C. A. C. (2000a). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41,* 113-127.

Coello, C. A. C. (2000b). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems, 17,* 319-346.

Coello, C. A. C. (2000c). Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization, 32(3),* 275-308.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16,* 193-203.

Coello, C. A. C., & Becerra, R. L. (2004). Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization, 36,* 219-236.

Coelho, L. D. S. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications, 37,* 1676-1683.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies, Proceedings of ECAL'91, *European Conference on Artificial Life,* Elsevier Publishing, Amsterdam.

Costa, D., & Hertz, A. (1997). Ants can color graphs. *Journal of the Operational Research Society, 48,* 295-305.

Cutello, V., Morelli, G., Nicosia, G., & Pavone, M. (2005). Immune algorithms with aging operators for the string folding problem and the protein folding problem. EvoCOP, LNCS, 3448 (pp. 80–90), Heidelberg, Springer.

Cutello, V., Narzisi, G., Nicosia, G., & Pavone, M. (2006). Real coded clonal selection algorithm for global numerical optimization using a new inversely proportional hypermutation operator. The *21st annual ACM symposium on applied computing,* vol. 2, (pp. 950–954), Dijon, France.

Cutello, V., Nicosia, G., Romeo, M., & Oliveto, P. S. (2007). On the convergence of immune algorithms. In *the first IEEE symposium on foundations of computational intelligence,* Los Alamitos: IEEE Computer Society Press.

David, S. (1993). *The water cycle*. Illus. John Yates, New York, Thomson Learning.

Davidson, R., & Harel, D. (1996). Drawing Graphs Nicely Using Simulated Annealing. *ACM Transactions on Graphics, 15(4),* 301-33.

De Castro D., L. N., & Von Zuben, F. J. (1999), "Artificial Immune Systems: Part I – Basic Theory and Applications", *Technical Report – RT DCA 01/99.*

De Castro, L. N., & Zuben, F. J. (2002). Learning and optimization using through the clonal selection principle, *IEEE Transactions Evolutionary Computation, 6(3),* 239-251.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186,* 311-338.

Deb, K., & Goyal, M. (1995). Optimizing engineering designs using a combined genetic search. In Eshelman, L.J. (Ed.), Proceedings of the *sixth international conference in generic algorithms* (pp. 521-528). University of Pittsburgh, Morgan KauKman Publishers, San Mateo, California.

Deb, K., & Srinivasan, A. (2006). Innovization: innovative design principles through optimization. *Proceedings of the 8th annual conference on Genetic and evolutionary computation,* New York, USA, 1629-1636.

Dede, T., Bekiroglub, S., & Ayvazc, Y. (2011). Weight minimization of trusses with genetic algorithm, *Applied Soft Computing, 11,* 2565-2575.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. PhD Thesis, University of Michigan, Michigan, USA.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991a). The ant system: an autocatalytic optimizing process. *Technical Report TR91-016*, Politecnico di Milano.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991b). Positive feedback as a search strategy. *Technical report 91-016*, Dipartimento di ElettronicaPolitecnico diMilano, Italy.

Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant Algorithms for Discrete Optimization. *Artificial Life, 5(2),* 137-172.

Dorigo, M. (1992). Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milano, Milano.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions Evolutionary Computation, 1(1),* 53-66.

Duan, Q., Gupta, V. K., & Sorooshian, S. (1992). Effective and efficient global optimization for conceptual rainfall-runoff models, *Water Resources Research, 28,* 1015-1031.

Farmer, J. D., Packard, N. H., & Perelson, A. S. (1986), The immune system, adaptation, and machine learning. *Physica, 22,* 187-204.

Feldman, J. A. (1990). Neural networks, artificial intelligence and computational reality. *Computers in Industry, 14(1-3),* 145-148.

Fletcher, R. (1963). Generalized inverses for nonlinear equations and optimization. In R. Rabinowitz (ed.), *Numerical Methods for Non-linear Algebraic Equations.* London: Gordon and Breach.

Fleischer, M. (1995). Simulated Annealing: Past, Present, and Future. *Proceedings of the 1995 Winter Simulation Conference,* Department of Engineering Management, Old Dominion University, Norfolk, VA.

Fogel, D. B. (1995). A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems. *Simulation, 64,* 397-404.

Geem, Z., Kim, J., & Loganathan, G.V. (2001). A new heuristic optimization algorithm: Harmony Search. *Simulation, 76,* 60-68.

Gero, M. B. P., Garcia, A. B., & Diaz, J. J. D. C. (2006). Design optimization of 3D steel structures: genetic algorithms vs. classical techniques. *Journal of Constructional Steel Research, 62,* 1303-1309.

Ghasemi, M. R., Hinton, E., & Wood, R. D. (1999). Optimization of trusses using genetic algorithms for discrete and continuous variables, *Engineering Computations, 16(3),* 272-301.

Glover, F. (1990). Tabu search – a tutorial. *10.1287/inte.20.4.74, 20(4),* 74-94.

Giraud-Moreau, L., & Lafon, P. (2002). Comparison of evolutionary algorithms for mechanical design components. *Engineering Optimization, 34,* 307-322.

Goldfarb, D., & Lapidus, B. (1968). Conjugate gradient method for nonlinear programming problems with linear constraints. *Industrial & Engineering Chemistry Fundamentals, 7,* 142-151.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, MA.

Goldberg, D. E., & Samtani, M. P. (1986). Engineering optimization via genetic algorithms. *Proceedings of the Ninth Conference on Electronic Computations,* ASCE, Birmingham, Alabama, 471-482.

Golden, B., Levy, L., & Dahl, R. (1981). Two generalizations of the traveling salesman problem. *Omega, 9(4),* 439-441.

Gomes, H. M. (2011). Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Systems with Applications, 38,* 957-968.

Gupta, S., Tiwari, R., & Shivashankar, B. N. (2007). Multi-objective design optimization of rolling bearings using genetic algorithm. *Mechanism and Machine Theory, 42,* 1418-1443.

Hamida, S. B, & Schoenauer, M. (2002). ASCHEA: new results using adaptive segregational constraint handling. *IEEE Transaction on Evolutionary Computation, 1,* 884-889.

Han, W. T., Tang, L. H., & Xie, G. N. (2008). Performance comparison of particle swarm optimization and genetic algorithm in rolling fin-tube heat exchanger optimization design. *Proceedings of the ASME Summer Heat Transfer Conference,* Jacksonville, FL, 5-10.

Han, X., Xu, D., & Liu, G. R. (2003). A computational inverse technique for material characterization of a functionally graded cylinder using a progressive neural network. *Neurocomputing, 51,* 341-360.

Hasancebi, O., & Erbatur, F. (2000). Layout optimization of trusses using simulated annealing. In Topping BHV, editor. *Computational Engineering using metaphors from nature* (pp. 175-190). Civil-Comp Press.

Haupt, R. L., & Haupt, S. E. (2004). *Practical Genetic Algorithms*. (2nd ed.), John Wiley & Sons, Inc. publication, doi:10.1002/0471671746.

He, Q., & Wang, L. (2006). An effective co-evolutionary particle swarm optimization for engineering optimization problems. *Engineering Applications of Artificial Intelligence, 20,* 89-99.

He, Q., & Wang, L. (2007). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation, 186,* 1407-1422.

He, S., Wu, Q. H., Wen, J. Y., Saunders, J. R., & Paton, R. C. (2004). A particle swarm optimizer with passive congregation. *Biosystems, 78,* 135-147.

Hedar, A. R., & Fukushima, M. (2006). Derivative-Free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization, 35,* 521-549.

Hedia, H. S. (2005). Design of functionally graded dental implant in the presence of cancellous bone. *Journal of Biomedical Materials Research - Part B Applied Biomaterials, 75,* 74-80.

Hedia, H. S., & Mahmoud, N. A. (2004). Design optimization of functionally graded dental implant. *Bio-Medical Materials and Engineering, 14*, 133-143.

Hedia, H. S., Shabara, M. A. N., El-midany, T. T., & Fouda, N. (2006). Improved design of cementless hip stems using two-dimensional functionally graded materials. *Journal of Biomedical Materials Research - Part B Applied Biomaterials, 79,* 42-49.

Hieu, T. T. (2011). A water flow algorithm for optimization problems. *PhD thesis*, National University of Singapore.

Hightower, R. R., Forrest, S., & Perelson, A. S. (1995). The evolution of emergent organization in immune system gene libraries. *Proceedings of the 6th Conference on Genetic Algorithms,* 344-350.

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI.

Homaifar, A., Qi, C. X., & Lai, S. H. (1994). Constrained optimization via genetic algorithms. *Simulation*, *62,* 242-253.

Huang, F. Z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation, 186(1),* 340-356.

Huiskes, R., Dalstra, M., Vondervenne, R., Grootenboer, H., & Slooff, T. J. (1987). A hypothesis concerning the effect of implant rigidity on adaptive cortical bone remodeling in the femur. *Journal of Biomechanics*, *20 (8),* 808-809.

Jerne, N. K. (1973). Towards a network theory of the immune system. *Annals of Immunology*, *125(c),* 373-389.

Kannan, B. k., & Kramer, S. N. (1994). An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanichal Design, 116,* 405-411.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization, *Technical Report-TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey.

Karaboga, D., & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. *Foundations of Fuzzy Logic and Soft Computing, 4529,* 789-798.

Kaveh, A., & Talatahari, S. (2009a). Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers & Structures, 87,* 267-283.

Kaveh, A., & Talatahari, S. (2009b). A particle swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research, 65,* 1558-1568.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *ICNN - IEEE International Conference on Neural Networks*, Perth, Australia, 4, 1942 1948.

Kennedy, J., & Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. *IEEE International Conference on Systems, Man, and Cybernatics, Computational Cybernetics and Simulation*, Orlando, FL, 5, 4104-4108.

Khabbazi, A., Atashpaz-Gargari, E., & Lucas, C. (2009). Imperialist competitive algorithm for minimum bit error rate beam forming. *International Journal of Bio-Inspired Computation, 1,* 125-133.

Kirkpatrick, S., Gelatt Jr. C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220,* 671-680.

Kim, C., Park, H. B., Jin, T. E., Jeong, I. S., & Seok, C. S. (2004). Prediction of material properties ofCF8M cast stainless steel by thermal embrittlement using neural network. *Key Engineering Materials, 270–273,* 102-107.

Kincaid, R. K. (1990). Minimizing distortion and internal forces in truss structures by simulated annealing. *Proceedings of 31th AIAA/ASME/ASCE/AHS/ASC Structural Materials and Dynamics Conference,* Long Beach, USA, AIAA, 327-333.

Kincaid, R. K. (1991). Minimizing distortion in truss structures: a comparison of simulated annealing and taboo search. *Proceedings of 32th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference,* Baltimore, USA, AIAA, 424-430.

Kogiso, N., Watson, L. T., Gurdal Z., & Haftka R. T. (1994). Genetic algorithms with local improvement for composite laminate design. *Structural and Multidisciplinary Optimization, 7,* 207-218.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *IEEE Transactions on Evolutionary Computation, 7,* 19-44.

Krishnamoorthy, C. S., Venkatesh, P. P., & Sudarshan, R. (2002). Object-oriented framework for genetic algorithms with application to space truss optimization. *Journal of Computing in Civil Engineering, 16(1),* 66-75.

Kuang, J. K., Rao, S. S., & Chen, L. (1998). Taguchi-aided search method for design optimization of engineering systems. *Engineering Optimization, 30,* 1-23.

Kubi, J. (2002). Immunology. (Fifth ed.) by Richard A. Goldsby, Thomas J. Kindt, BarbaraOsborne, W H Freeman.

Lampinen, J. (2002). A constraint handling approach for the differential evolution algorithm. *IEEE Transactions on Evolutionary Computation, 2,* 1468-1473.

Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers and Structures, 82*, 781-798.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Method in Applied Mechanics and Engineering, 194,* 3902-3933.

Lee, K. S., Geem, Z. W., Lee, S. H., & Bae, K. W. (2005). The harmony search heuristic algorithm for discrete structural optimization. *Engineering Optimization, 37(7),* 663-684.

Leriche, R., & Haftka, R. T. (1993). Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. *AIAA Journal, 31(5),* 951-956.

Lessing, L., Dumitrescu, I., & Stützle, T. (2004). A comparison between ACO algorithms for the set covering problem. *ANTS Workshop*, 1–12.

Li, L. J., Huang, Z. B., & Liu, F. (2009). A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures, 87,* 435-443.

Li, L. J., Huang, Z. B., & Liu, F. (2009). A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures, 87,* 435-443.

Liao G. C. (2006). Short-term thermal generation scheduling using improved immune algorithm. *Electric Power Systems Research, 76,* 360–373.

Lin, D., Li, Q., Li, W., Zhou, S., & Swain, M. V. (2009). Design optimization of functionally graded dental implant for bone remodeling. *Composites Part B: Engineering, 40,* 668-675.

Lin, D., Li, Q., Li, W., & Swain, M. V. (2008a). Dental implant induced bone remodeling and associated algorithms. *Journal of Mechanical Behavior of Biomedical Materials,* doi:10.1016/ j.jmbbm.2008.11.007.

Lin, D., Li, Q., Li, W., Swain, M. V. (2008b). Functionally graded implant and its effect on bone remodeling. *Advance Materials Research, 47(50),* 1035–1038.

Liu, G. R., Han, X., Xu, Y. G., & Lam, K. Y. (2001). Material characterization of functionally graded material by means of elastic waves and a progressive-learning neural network. *Composites Science and Technology, 61(10),* 1401-1411.

Liu, G. R., Lam, K. Y., & Han, X. (2002). Determination of elastic constants of anisotropic laminated pipes using elastic waves and a progressive neural network. *Journal of Sound and Vibration, 252(2),* 239-259.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10,* 629-640.

Liu, Y., & Passino, K. M. (2002). Biomimicry of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors. *Journal of Optimization Theory and Applications, 115,* 603-628.

Luenberger, D. G. (1984). *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley.

Luh, G. C., & Lin, C. Y. (2009). Structural topology optimization using ant colony optimization algorithm. *Applied Soft Computing, 9,* 1343-1353.

Mariani, V. C., Luvizotto, L. G. J., Guerra, F. A., & Coelho, L. D. S. (2011). A hybrid shuffled complex evolution approach based on differential evolution for unconstrained optimization. *Applied Mathematics and Computation, 217,* 5822-5829.

May, S. A., & Balling, R. J. (1992). A filtered simulated annealing strategy for discrete optimization of 3D steel frame works. *Structural Optimization, 4,* 142-148.

McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5,* 115-133.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller A. H., & E. Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21,* 1087-1092.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, AI Series, New York, NY, USA.

Michalewicz, Z. (1995). Genetic algorithms, numerical optimization, and constraints. *Proceedings of 6th International Conference on Genetic Algorithms,* Morgan Kauffman, San Mateo, 151-158.

Michalewicz, Z., Logan, T. D., & Swaminathan, S. (1994). Evolutionary operators for continuous convex parameter spaces. Proceedings of *3rd Annual Conference on Evolutionary Programming, World Scientific*, 84-97.

Mishra, M., Das, P. K., & Sarangi, S. (2009). Second law based optimization of cross flow plate-fin heat exchanger design using genetic algorithm. *Applied Thermal Engineering, 29,* 2983-2989.

Mishra, M., Das, P. K., & Sarangi, S. (2009). Second law based optimization of cross flow plate-fin heat exchanger design using genetic algorithm. *Applied Thermal Engineering, 29,* 2983-2989.

Miyazaki, T., & Akisawa, A. (2009). The influence of heat exchanger parameters on the optimum cycle time of adsorption chillers. *Applied Thermal Engineering, 29(13)*, 2708-2717.

Montes, E. M., & Coello, C. A. C. (2005a). In MICAI: Lect. Notes Artif. Int, *Useful infeasible solutions in engineering optimization with evolutionary algorithms* (pp. 652-662), DOI:10.1007/11579427-66.

Montes, E. M., & Coello, C. A. C. (2005b). A simple multimembered evolution strategy to solve constrained optimization problems. IEEE *Transactions on Evolutionary Computation, 9,* 1-17.

Montes, E. M., & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems, 37,* 443-473.

Montes, E. M., Coello, C. A. C., & Velazquez-Reyes, J. (2006a). Increasing successful offspring and diversity in differential evolution for engineering design. In Proceedings of *7th international conference on adaptive computing in design and manufacture,* 131-139.

Montes, E. M., Velazquez-Reyes, J., & Coello, C. A. C. (2006b). Modified differential evolution for constrained optimization. *Evolutionary Computation, CEC, IEEE Congress*, 25-32.

Nagendra, S., Jestin, D., Guradal, Z., Haftka, R. T., & Watson, L. T. (1996). Improved genetic algorithm for the design of stiffened composite panels. *Computers & Structures, 58(3),* 543-555.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal, 7,* 308-313.

Nicosia G, Cutello V, Bentley PJ, Timmis J. Artificial immune systems. In: *3rd international conference* (ICARIS 2004), Catania, Italy, LNCS, vol. 3239, Springer-Verlag; 2004.

O'Mahony, A. M., Williams, J. L., & Spencer, P. (2001). Anisotropic elasticity of cortical and cancellous bone in the posterior mandible increases peri-implant stress and strain under oblique loading. *Clinic Oral Implants Research, 12(6),* 648-657.

Osyczka, A. (2002). Evolutionary algorithms for single and multicriteria design optimization: studies in fuzzyness and soft computing. Heidelberg, Physica-Verlag

Panigrahi B. K., Yadav, S. R., Agrawal, S., & Tiwari, M. K. (2007). A clonal algorithm to solve economic load dispatch. *Electric Power Systems Research, 77,* 1381-1389.

Parsopoulos, K., & Vrahatis, M. (2005). In Advances in Natural Computation, LNCS, *Unified particle swarm optimization for solving constrained engineering optimization problems* (pp. 582-591). Berlin, Springer-Verlag.

.

Perez, R. E., & Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers & Structures, 85,* 1579-1588.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm-A novel tool for complex optimisation problems. *Intelligent Production Machines and Systems,* 454-459.

Powell, M. J. D. (1964). An efficient way for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal, 7,* 155-162.

Press, W. H., Teukolsky, S. A., Vettering, W. T., & Flannery, B. P. (1992). *Numerical Recipes.* New York: Cambridge University Press.

Qian, H. C., Xia, B. C., Li, S. Z., & Wang, F. G. (2002). Fuzzy neural network modeling of material properties. *Journal of Materials Processing Technology, 122(2-3),* 196-200.

Rahman, T. K. A., Suliman, S. I., & Musirin, I. (2006). Artificial immune-based optimization technique for solving economic dispatch in power systems. Berlin, Heidelberg: Springer-Verlag. pp. 338–45.

Rahami, H., Kaveh, A., & Gholipour, Y. (2008). Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures, 30,* 2360-2369.

Rajeev, S., & Krishnamoorthy, C. S. (1992). Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering-ASCE, 118(5),* 1233-1250.

Rao, R. V., & Savsani, V. J. (2012). *Mechanical design optimization using advanced optimization techniques.* Springer-Verlag, London.

Rao, R. V., & Patel, V. (2012a). An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *International Journal of Industrial Engineering Computation, 3,* 535-560.

Rao, R. V., & Patel, V. (2012b). Multi-objective optimization of heat exchangers using a modified teaching-learning-based-optimization algorithm. *Applied Mathematical Modeling,* doi:10.1016 /j.apm.2012.03.043.

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43,* 303-315.

Rasmussen, M. H., & Stolpe, M. (2008). Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. *Computers & Structures, 86,* 1527-1538.

Ravagnani, M. A. S. S., Silva, A. P., Biscaia Jr. E. C., & Caballero J. A. (2009). Optimal design of shell and tube heat exchangers using particle swarm optimization. *Industrial & Engineering Chemistry Research, 48 (6),* 2927-2935.

Ray, T., & Liew, K. M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation, 7,* 386-396.

Renato, A. K., & Santos, C. L. D (2006). Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 36*, 1407-1416.

Rho, J. Y., Hobatho, M. C., & Ashman, R. B. (1995). Relations of mechanical-properties to density and Ct numbers in human bone. *Medical Engineering & Physics, 17(5),* 347-355.

Ringertz, U. T. (1988). On methods for discrete structural constraints. *Engineering Optimization, 13(1),* 47-64.

Rouge, F., & Laussane (1896). *V. Pareto, Cours D'Economie Politique*. volume I and II.

Runarsson, T. P., & Xin, Y. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation, 4,* 284-294.

Runarsson, T. P., & Xin, Y. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 35,* 233-243.

Saleh, A., & Adeli, H. (1994). Parallel algorithms for integrated structural and control optimization. *Journal of Aerospace Engineering, ASCE, 7(3),* 297-314.

Saleh, A., & Adeli, H. (1998). Optimal control of adaptive/smart building structures. *Computer-Aided Civil and Infrastructure Engineering, 13(6),* 389-403.

Sarma, K. C., & Adeli, H. (2001). Bi-Level parallel genetic algorithms for optimization of large steel structures. *Computer-Aided Civil and Infrastructure Engineering, 16 (5),* 295-304.

Schwefel, H. (1995). *Evolution and Optimum Seeking.* New York, Wiley.

Shah-Hosseini, H. (2009). The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation, 1*, 71-79.

Shang, Y. W., & Qiu, Y. H. (2006). A note on the extended rosenbrock function. *Evolutionary Computation, 14(1),* 119-126.

Shanno, D. F. (1970). An accelerated gradient projection method for linearly constrained nonlinear estimation. *SIAM Journal on Applied Mathematics, 18*, 322-334.

Siddall, J. N. (1982). *Optimal engineering design, principles and applications*. Marcel Dekker, New York.

Sivakumar, P., Rajaraman, A., Natajan, K., & Samuel, K. G. M. (2001). Artificial intelligence techniques for optimization of steel lattice towers, recent developments in structural engineering. *Proceeding of Structural Engineering Convention,* 435-445.

Soegiarso, R., & Adeli, H. (1998). Parallel-vector algorithms for optimization of large steel structures. *Computer-Aided Civil and Infrastructure Engineering, 13(3),* 207-217.

Soremekun, G., Gurdal, Z., Haftka, R. T., & Watson, L. T. (2001). Composite laminate design optimization by genetic algorithm with generalized elitist selection. *Computers & Structures, 79,* 131-143.

Strahler, A. N. (1952). Dynamic basis of geomorphology. *Geological Society of America Bulletin, 63,* 923-938.

Suman, B. (2004). Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers & Chemical Engineering, 28,* 1849-1871.

Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society, 15,* 1143-1160.

Suppapitnarm, A., Seffen, K. A., Parks, G. T., & Clarkson, P. J. (2000). Simulated annealing: an alternative approach to true multiobjective optimization. *Engineering Optimization,* 33-59.

Szewczyk, Z., & Hajela, P. (1993). Neural network approximations in a simulated annealing based optimal structural design. *Structural Optimization, 5,* 159-165.

Takahama, T., & Sakai, S. (2005). Constrained optimization by applying the *α*; constrained method to the nonlinear simplex method with mutations. IEEE *Transactions on Evolutionary Computation, 9,* 437-451.

Tang, K. Z., Sun, T. K., & Yang, J. Y. (2011). An improved genetic algorithm based on a novel selection strategy for nonlinear programming problems. *Computers & Chemical Engineering, 35,* 615-621.

Tessema, B., & Yen, G. G. (2006). A self adaptive penalty function based algorithm for constrained optimization. *IEEE Transactions on Evolurionary Computation,* 246-253.

Todoroki, A., & Haftka, R. T. (1998). Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy. *Composite Part B: Engineering, 29(3),* 277-285.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters, 85,* 317-325.

Turner, C. H., Anne, V., & Pidaparti, R. M. V. (1997). A uniform strain criterion for trabecular bone adaptation: do continuum level strain gradients drive adaptation?. *Journal of Biomechanics, 30(6),* 555-563.

Turner, C. H. (1998). Three rules for bone adaptation to mechanical stimuli. *Bone, 23(5),* 399-407.

Ursem, R., & Vadstrup, R. (2003). Parameter identification of induction motors using differential evolution. Proceedings of the *5th Congress on Evolutionary Computation,* Piscataway, NJ, USA, 1, 790-796.

Vesterström, J., & Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization and evolutionary algorithms on numerical benchmark problems. Proceedings of *the Sixth Congress on Evolutionary Computation,* Piscataway, NJ, USA, 2, 1980-1987.

Vrugt, J. A., Gupta, H. V., Bouten, W., & Sorooshian, S. (2003). A shuffled complex evolution Metropolis algorithm for optimization and uncertainty assessment of hydrological model parameters. *Water Resources Research, 39,* 1201-1218.

Wang, F., Lee, H. P., & Lu, C. (2007). Thermal-mechanical study of functionally graded dental implants with the finite element method. *Journal of Biomedical Materials Research - Part A, 80,* 146-158.

Wang, Y., & Cai, Z. (2009). A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. *Frontiers of Computer Science, 3(1),* 38-52.

Wang, Y., & Cai, Z. (2012a). A dynamic hybrid framework for constrained evolutionary optimization. *IEEE Transactions on Systems Man and Cybernetics Part B, 42(1),* 203-217.

Wang, Y., & Cai, Z. (2012b). Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Transaction Evolutionary Computation, 16(1),* 117-134.

Wang, Y., & Cai, Z. (2011). Constrained evolutionary optimization by means of $(\mu+\lambda)$-differential evolution and improved adaptive trade-off model. *Evolutionary Computation, 19(2),* 249-285.

Wang, Y., Cai, Z., Guo, G., & Zhou, Y. (2007). Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. *IEEE Transactions on Systems Man and Cybernetics Part B, 37(3),* 560-575.

Wang, Y., Cai, Z., Zhou, Y., & Fan, Z. (2009). Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint handling technique. *Structural and Multidisciplinary Optimization, 37,* 395-413.

Wang, L., & Li, L. P. (2010). An effective differential evolution with level comparison for constrained engineering design. *Structural Multidisciplinary Optimization, 41,* 947-963.

Watari, F., Yokoyama, A., Omori, M., Hirai, T., Kondo, H., Uo, M., & Kawasaki, T. (2004). Biocompatibility of materials and development to functionally graded implant for bio-medical application. *Composites Science and Technology, 64,* 893-908.

Weinans, H., Huiskes, R., & Grootenboer, H. J. (1992). The behavior of adaptive bone remodeling simulation models. *Journal of Biomechanics, 25(12),* 1425-1441.

Wu, S. J., & Chow, P. T. (1995). Steady-state genetic algorithms for discrete optimization of trusses. *Computers & Structures, 56,* 979-991.

Xie, G. N., Sunden, B., & Wang, Q. W. (2008). Optimization of compact heat exchangers by a genetic algorithm. *Applied Thermal Engineering, 28,* 895-906.

Xu, B., Shen, Z., Ni, X., Wang, J., Guan, J., & Lu, J. (2004). Determination of elastic properties of a film substrate system by using the neural networks. *Applied Physics Letters, 85(25),* 6161-6163.

Yang, J., & Cheng, J. C. (2001). Inversion of elastic constants for orthotropic pipe by artificial neural network. *Progress in Natural Science, 11,* S75-8.

Yang, J., Cheng, J. C., & Berthelot, Y. H. (2002). Determination of the elastic constants of a composite pipe using wavelet transforms and neural networks. *Journal of the Acoustical Society of America, 111(3),* 1245-1250.

Yang, J., & Xiang, H.-J. (2007). A three-dimensional finite element study on the biomechanical behavior of an FGBM dental implant in surrounding bone. *Journal of Biomechanics, 40,* 2377-2385.

Yu, X. C., Cui, Z. Q., & Yu, Y. (2008). Fuzzy optimal design of the plate-fin heat exchangers by particle swarm optimization. Proceedings of *the Fifth International Conference on Fuzzy Systems and Knowledge Discovery,* Jinan, China, 574-578.

Yuan, Q., & Qian, F. (2010). A hybrid genetic algorithm for twice continuously differentiable NLP problems. *Computers & Chemical Engineering, 34,* 36-41.

Zavala, A. E. M., Aguirre, A. H., & Diharce, E. R. V. (2005). Constrained optimization via evolutionary swarm optimization algorithm (PESO). Proceedings of *the 2005 Conference on Genetic and Evolutionary Computation,* New York, USA, 209-216.

Zahara, E., & Kao, Y. T. (2009). Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications, 36,* 3880-3886.

Zhao, X., Yao, Y., & Yan L. (2009). Learning algorithm for multimodal optimization. *Computers & Mathematics with Applications, 57,* 2016-2021.

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences, 178,* 3043-3074.

Zhi-ding, W., & Jie-Kang, W. (2011). Water cycle-like algorithm for unconstrained optimization problems. *Computer Engineering, 37(22),* 187-190.

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms: a comperative case study. *Parallel Problem Solving from Nature V.* Springer, Berlin, Germany, 292–301.

APPENDICES

Appendix A: Mathematical formulations for unconstrained benchmark problems

N: Number of design variables.

Rastrigin function

$$f(x)=\sum_{i=1}^{N}(x_i^2-10\cos(2\pi ix)+10)$$

Ackley function

$$f(x)=-20\exp(-0.2\sqrt{\frac{\sum_{i=1}^{N}x_i^2}{N}})-\exp(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i))+20+e$$

Zakharov function

$$f(x)=\sum_{i=1}^{N}x_i^2+(\sum_{i=1}^{N}0.5ix_i)^2+(\sum_{i=1}^{N}0.5ix_i)^4$$

Schwefel function

$$f(x)=418.9829\times N-\sum_{i=1}^{N}x_i\sin(\sqrt{|x_i|})$$

Rosenbrock function

$$f(x)=\sum_{i=1}^{N-1}\left\{100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right\}$$

Hyper Sphere function

$$f(x)=\sum_{i=1}^{N}x_i^2$$

Martin and Gaddy function

$$f(x)=(x_1-x_2)^2+\left[{(x_1+x_2-10)}\big/{3}\right]^2$$

Branin function

$$f(x)=a(x_2-bx_1^2+cx_1-d)^2+e(1-f)\cos(x_1)+e$$

$$a=1,b=\frac{5.1}{4}\left(\frac{7}{22}\right)^2,c=\frac{5}{22}\times 7,d=6,e=10,\ f=\frac{1}{8}\times\frac{7}{22}$$

Goldstein and Price I function

$$f(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \\ \times\left[3 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$$

Goldstein and Price II function

$$f(x) = \exp\left\{\frac{1}{2}(x_1^2 + x_2^2 - 25)^2\right\} + \sin^4(4x_1 - 3x_2) + \frac{1}{2}(2x_1 + x_2 - 10)^2$$

De Jong function

$$f(x) = 3905.93 - 100(x_1^2 - x_2)^2 - (1 - x_1)^2$$

Six Hump Camel Back function

$$f(x) = 4x_1^2 - 2.1x_1^4 + \tfrac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

Shaffer function

$$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001(x_1^2 + x_2^2)\right)^2}$$

Wood function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ + 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)$$

Powell Quartic function

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

Easton and Fenton function

$$f(x) = \left\{12 + x_1^2 + \frac{1 + x_2^2}{x_1^2} + \frac{x_1^2x_2^2 + 100}{(x_1x_2)^4}\right\}\left(\tfrac{1}{10}\right)$$

Appendix B: Mathematical Formulations for constrained engineering problems

B.1. Constrained problem 1

$$f(x) = 5.3578547x_3^3 + 0.8356891x_1x_5 + 37.293239x_1 + 40729.141$$

subject to:

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 0.0022053x_3x_5 \le 0$$

$$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0$$

$$g_4(x) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0$$

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$$

$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$

$$78 \le x_1 \le 102$$

$$33 \le x_2 \le 45$$

$$27 \le x_i \le 45 \quad i = 3,4,5$$

B.2. Constrained problem 2

$$f(x) = -\frac{100 - (x_1-5)^2 - (x_2-5)^2 - (x_3-5)^2}{100}$$

subject to:

$$g(X) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$

$$0 \le x_i \le 10 \quad i = 1,2,3 \quad , \quad p,q,r = 1,2,3,\ldots,9$$

B.3. Pressure vessel design problem

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

subject to:

$$g_1(x) = -x_1 + 0.0193x \le 0$$

$$g_2(x) = -x_2 + 0.00954x_3 \le 0$$

$$g_3(x) = -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1,296,000 \le 0$$

$$g_4(x) = x_4 - 240 \le 0$$

$$0 \le x_i \le 100 \qquad i = 1,2$$

$$10 \le x_i \le 200 \qquad i = 3,4$$

B.4. Tension/compression spring design problem

$$f(x) = (x_3 + 2)x_2 x_1^2$$

subject to:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71,785 x_1^4} \le 0$$

$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12,566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \le 0$$

$$g_3(x) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0$$

$$g_4(x) = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

$$0.05 \le x_1 \le 2.00$$

$$0.25 \le x_2 \le 1.30$$

$$2.00 \le x_3 \le 15.00$$

B.5. Welded beam design problem

$$f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2)$$

subject to:

$$g_1(x) = \tau(x) - \tau_{\max} \le 0$$

$$g_2(x) = \sigma(x) - \sigma_{\max} \le 0$$

$$g_3(x) = x_1 - x_4 \le 0$$

$$g_4(x) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0$$

$$g_5(x) = 0.125 - x_1 \le 0$$

$$g_6(x) = \delta(x) - \delta_{\max} \le 0$$

$$g_7(x) = P - P_c(x) \le 0$$

$$0.1 \le x_i \le 2 \qquad i = 1,4$$

$$0.1 \le x_i \le 10 \qquad i = 2,3$$

where,

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}\,, \quad \tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}$$

$$M = P(L + \frac{x_2}{2}), \quad R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2}\,, \quad J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2\right]\right\}$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2}, \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4}, \quad P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000\,lb, \quad L = 14\,in, \quad E = 30\times10^6\,psi, \quad G = 12\times10^6\,psi$$

$\tau_{\max} = 13,600\,psi, \quad \sigma_{\max} = 30,000\,psi, \quad \delta_{\max} = 0.25\,in$