SOUTHEAST EUROPE JOURNAL OF SOFT COMPUTING
Available online at www.scjournal.com.ba

Performance Evaluation of Nature-Inspired Algorithms in constrained Optimization

Ali Osman Kusakci [a], Mehmet Can [b]

[a] Faculty of Engineering and Natural Sciences, International University of Sarajevo, Hrasnicka Cesta 15, 71210, Ilidza, akusakci@ius.edu.ba

[b] Faculty of Engineering and Natural Sciences, International University of Sarajevo, Hrasnicka Cesta 15, 71210, Ilidza, mcan@ius.edu.ba

Abstract— (In almost all scientific contributions to the field of Nature-Inspired Algorithms (NIAs), the researchers select some benchmark test suites, which makes possible to draw conclusions on the merit of the proposed algorithm. Hence, it is a vital task to compose comprehensive test suites with the aim of covering variety of different scenarios. Furthermore, while conducting comparative analysis of results obtained with NIAs, selection of the proper performance indicators are of paramount importance. This paper intends to address these two topics with a special stress on NIAs designed for constrained optimization.)

Keywords—constrained optimization, evolutionary algorithms, nonlinear programming, performance measures, benchmark test suites

1. Introduction

A COP in n dimensional space can be defined by two components: an objective function to be maximized or minimized, and several inequality and equality constraints. The general structure is defined as:

min or max $$f(\vec{x}),\quad \vec{x} = [x_1, \dots, x_n]^T \in F \subseteq S \subseteq \mathbb{R}^n \tag{1}$$

subject to

$$g_i(\vec{x}) \le 0,\ \ i = 1, \dots, r \tag{2}$$

$$h_i(\vec{x}) = 0,\ \ i = r+1, \dots, m, \tag{3}$$

where $S = \{\vec{x} \in \mathbb{R}^n | l_j \le x_j \le u_j\}$, $j = 1, \dots, n$ and $F = \{\vec{x} \in S | g_i(\vec{x}) \le 0 \ and \ h_i(\vec{x}) = 0\}$, $\vec{x}$ is solution vector $\vec{x} = [x_1, \dots, x_n]^T$, r is the number of inequality and *m-r* is the number of equality constraints. Some researchers convert equality constraints into inequalities by adding a small

tolerance $\epsilon > 0$.

There have been innumerable benchmark problems introduced in the field of nonlinear constrained optimization by the researchers. In the vast realm of nonlinear programming with nature-inspired algorithms (NIAs), collecting all constrained optimization problems (COPs) is a cumbersome task to be realized. Keeping this fact in mind, this work attempts to bring together the most common COPs while testing the performance of NIAs by the practitioners. Moreover, the composition of test suites should allow evaluating the performance of the algorithm under various conditions.

Second aim of this work is to address the performance measures used in the domain and contrast their informative characteristics. While conducting comparative analysis of results obtained with NIAs, selection of the proper performance indicators are of paramount importance.

The rest of this work is organized as follows: The essential guidelines when selecting a representative benchmark set and the general properties of most common COPs benchmarks are given in Section II. Section III is devoted to setting up proper performance measures to verify the merit of a proposed NIA in constrained optimization domain. Lastly, the findings are summarized in Section IV.

Benchmark Problems

2.1. Selection Guidelines

The benchmark problems in the domain may be grouped into two main classes based on the resource that they originated from (El-Ghazali, 2009):

1. Artificial Problems: From the early stages of the intelligent problem solving with NIAs, different researchers from various backgrounds have introduced countless many COPs. Gradually, their individual contributions have been collected to form benchmark problem sets. In general, the primary goal of such benchmark sets is to cover as many problems variants as possible in order to provide a reliable test basis. On the other hand, the advantage of such collections is that they are evaluated by various methods and the results obtained by these methods are easily accessible (Eiben and Smith, 2008).

Another main source of benchmark problems is the problem generators that have been developed by the researchers. Two most referred examples of such generators can be found in (Spears, 2004) and (Gallagher and Yuan, 2006). Their main drawback is that the problems generated by a certain generator are of the same or similar type since the test problem generators usually rely on certain problem construction principles. Thus, they may not yield to reliable results for test algorithms claiming to be robust (Yu and Gen, 2010).

2. Real life problems: This type of problems originates from a real life challenge. They form especially valuable benchmarks while testing the applicability of the algorithm in various field of science. On the other hand, the data of a real life problem is usually unavailable to the whole research community due to the potential copyrights and confidentiality consequences. Thus, repeating the numerical experiments on the data is mostly not a feasible option. This, in turn, makes the comparison of different algorithms on the same problem barely possible (Yu and Gen, 2010). Furthermore, the generalization of the obtained results is highly difficult since such problems usually involve some domain specific aspects.

After introduction of the two main sources of benchmark problems, we would like to specify the main characteristics that a *good* benchmark should possess. Namely, a NIA must be evaluated on various test problems covering different variants of COPs to investigate its convergence behavior. In other words, the set of instances must be diverse in terms of their difficulties and structure (El-Ghazali, 2009). Especially, COPs with challenging properties, including multimodality, sparseness of feasible space, non-separability should be included in the test bed to evaluate the robustness of an algorithm. However, we should keep in mind that robustness is only one of the criterions while proving the merit of an algorithm. Alternatively, a performance evaluation study may reveal the conditions under which the algorithm can be successfully applied.

Throughout the extensive course of research in the domain, some general guidelines for generating test beds for NIAs have been generated by Eiben and Smith (2008) from various resources:

- A few unimodal instances should be provided to test the convergence speed of the algorithm.
- Several multimodal functions with a large number of local optima must be included in the test suite to examine the behavior of the algorithm when dealing with many local optima. Additionally, this type of functions may be good touchstones to evaluate the robustness of the algorithm. Namely, an algorithm reaching the solutions of the same quality among many local optima in each distinct run proves its robustness.
- As an additional robustness measure, the algorithm should be tested on problems with random noise.
- The test suite must contain some scalable problems in terms of problem variables and search range. By testing the algorithm on scalable problems, we may check the possible performance deficiencies due to the enlargement of the search region.
- The convergence performance of the algorithm should be tested also on non-separable objective functions.

In addition to the points stated by them for a general purpose NIA, we extracted some standards for nonlinear constrained optimization test suites. We propose the following guidelines specifically apply for constrained optimization problems:

- The algorithm should provide a proper way of handling not only inequality constraints but also equality type ones. Thus, COPs with high number of equality constraints must be present in the test set. Especially, highly nonlinear equality constraints may be challenging for a constrained handling method.

- Some COPs with sparse feasible domain should be included in the benchmark set. This allows testing the explorative power of the algorithm under such scenario.
- COPs with global optimum lying on the constrained boundary are challenging benchmarks for most NIAs and they should comprise an essential part of the test suite.
- Some NIAs may exploit the above mention property of a COP. Thus, the opposite case, global optimum not on the boundary, must be analyzed.
- Some constrained handling mechanism may easily distract the search process in COPs with small number of constraints and simple structures although they behave well in highly constrained environments. The distraction is usually paid as unnecessary computational cost in such COPs. Thus, some COPs of low complexity may be added to observe this effect.

Obviously, there is still space for more work in this direction. More proper test suites may be composed to test the above-mentioned issues. The next section states the most common test suites used to test NIAs and summarizes the general properties of the COPs included in these test suites.

2.2. Benchmark Problems

CEC2006 Benchmark

The CEC2006 test suite is composed of 24 problems that have been proposed by various researchers (Liang et al., 2006). Since the problems are collected from different resources, the problem set may be considered as a unique case covering a wide range of problem classes with novel properties.

The 24 problems brought together by Liang et al. (2006) and were subject to a competition organized by IEEE community in 2006. The general properties of the benchmark problems are given in Table 1 (Liang et al., 2006) where n indicates the number of variables, and ρ is the ratio of feasible individuals over 1,000,000 random individuals generated with uniform distribution within the definition domain of the problem. LI, NI, LE, and NE stand for the numbers of linear inequalities, nonlinear inequalities, linear equalities, and nonlinear inequalities, respectively. Here, *act* shows the number of active constraints at the optimum point, $\vec{x}^*$.

Mezura-Montes et al. (2010) classifies the CEC2006 problems based on two criterions; number of decision variables, and type of constraints, given in Table 2, and Table 3.

CEC2010 Benchmark

The second test suite worth mentioning is generated by Mallipeddi and Suganthan (2010c), see Table 4. Their intention was to offer an alternative benchmark set to the well-known 24 problems, which have been extensively studied and solved by many algorithms. Thus, it has become almost impossible to demonstrate the superiority of newly designed constrained handling methods. Another motivation was to construct a test suite with scalable problems in terms of decision variables (Mallipeddi and Suganthan, 2010).

Additionally, in this test suite, the objective functions and constraints are rotated by a certain rotation matrix $\boldsymbol{M}$. The rotation operation is justified by the fact that a COP with multiple sparse feasible regions parallel to the coordinate axes can be solved better by algorithms employing line search or difference of two or more solution vectors. Namely, the rotation aims fostering a fair environment of comparison (Mallipeddi and Suganthan, 2010).

TABLE 1: GENERAL PROPERTIES OF CEC2006 BENCHMARK PROBLEMS

Prob.	*n*	*Type of function*	*ρ*	*LI*	*NI*	*LE*	*NE*	*act*
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.9971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000%	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3
g14	10	nonlinear	0.0000%	0	0	3	0	3
g15	3	quadratic	0.0000%	0	0	1	1	2
g16	5	nonlinear	0.0204%	4	34	0	0	4
g17	6	nonlinear	0.0000%	0	0	0	4	4
g18	9	quadratic	0.0000%	0	13	0	0	6
g19	15	nonlinear	33.4761%	0	5	0	0	0
g20	24	linear	0.0000%	0	6	2	12	16
g21	7	linear	0.0000%	0	1	0	5	6
g22	22	linear	0.0000%	0	1	8	11	19
g23	9	linear	0.0000%	0	2	3	1	6
g24	2	linear	79.6556%	0	2	0	0	2

Since all problems in the test set are scalable, a classification based on the number of problem variables is not reasonable. The competition organized by IEE (CEC 2010 Competition of Constrained Real-Parameter Optimization) required the participating researchers to solve the problems for $n = 10$, and $n = 30$. The classification of the problems based on the type of the constraints is shown in Table 5.

TABLE 2:CLASSIFICATION OF CEC2006 PROBLEMS BASED ON THE NUMBER OF DECISION VARIABLES (MEZURA-MONTES, MIRANDA-VARELA AND DEL CARMEN GÓMEZ-RAMÓN, 2010)

Class	***n***	**Problem**
High	10-20	g01, g02, g03, g07, g14, g19, g20, g22
Medium	5-9	g04, g09, g10, g13, g16, g17, g18, g21, g23
Low	2-4	g05, g06, g08, g11, g12, g15, g24

TABLE 3: CLASSIFICATION OF CEC2006 PROBLEMS BASED ON THE CONSTRAINT TYPES (MEZURA-MONTES, MIRANDA-VARELA AND DEL CARMEN GÓMEZ-RAMÓN, 2010)

Class	Problem
Only inequalities	g01, g02, g04, g06, g07, g08, g09, g10, g12, g16, g18, g19, g24
Only equalities	g03, g11, g13, g14, g15, g17
Inequalities and equalities	g05, g20, g21, g22, g23

TABLE 4: GENERAL PROPERTIES OF CEC2010 BENCHMARK PROBLEMS FOR N=10; SEP.: SEPARABLE, NON SEP.: NON SEPARABLE, ROT.: ROTATED (MALLIPEDDI AND SUGANTHAN, 2010)

Problem/Search Range	*Type of function*	*Sep.? (0/1)*	*Eq.*	*Ineq.*	*ρ%*
C01 $[0,10]^n$	nonlinear	0	0	2 Non Sep.	99.76
C02 $[-5.12,5.12]^n$	linear	1	1 Sep.	2 Sep.	0
C03 $[-1000,1000]^n$	polynomial	0	1 Non Sep.	0	0
C04 $[-50,50]^n$	linear	1	2 Non Sep., 2 Sep.	0	0
C05 $[-600,600]^n$	linear	1	2 Sep.	0	0
C06 $[-600,600]^n$	linear	0	2 Rot.	0	0
C07 $[-140,140]^n$	polynomial	0	0	1 Sep.	50.51
C08 $[-140,140]^n$	polynomial	0	0	1 Rot.	37.95
C09 $[-500500]^n$	polynomial	0	1 Sep.	0	0
C10 $[-500,500]^n$	polynomial	0	1 Rot.	0	0
C11 $[-100,100]^n$	nonlinear	Rot.	1 Non Sep.	0	0
C12 $[-1000,1000]^n$	nonlinear	1	1 Non Sep.	1 Sep.	0
C13 $[-500,500]^n$	nonlinear	1	0	2 Sep., 1 Non Sep.	0
C14 $[-1000,1000]^n$	polynomial	0	0	3 Sep.	0.31
C15 $[-1000,1000]^n$	polynomial	0	0	3 Rot.	0.32
C16 $[-10,10]^n$	nonlinear	0	2 Sep.	1 Sep., 1 Non Sep.	0
C17 $[-10,10]^n$	quadratic	0	1 Sep.	2 Non Sep.	0
C18 $[-50,50]^n$	quadratic	0	1 Sep.	1 Sep.	0.001

TABLE 5: CLASSIFICATION OF CEC 2010 PROBLEMS BASED ON THE CONSTRAINT TYPES

Class	Problem
Only inequalities	C01, C07, C08, C13, C14, C15
Only equalities	C03, C04, C05, C06, C12
Inequalities and equalities	C02, C09, C10, C11, C16, C17, C18

Engineering Design Problems

The third group of problems introduced here is the engineering design problems collected from various resources. Here, we note that the number of problems possibly put in this set is more than we included in this work (Yiqing, Xigang and Yongjian, 2007; Lin, Hwang and Wang, 2004; Costa and Oliveira, 2001). However, we have selected the most common problems. The first problem is pressure vessel design problem (g40) stated in (Zahara and Kao, 2009). The problem g41 is named in the literature as welded beam design problem. It is firstly formulated in (Kannan and Kramer, 1994) while it attain its standard form in later works. Tension/compression spring (g42) and speed reducer (g43) problems are also the two most cited design problems in the literature (Liu, Cai and Wang, 2010) whereas the car side impact design (g44) and stepped cantilever beam (g45) problems are not frequently used for test purposes (Gandomi, Yang and Alavi, 2011).

Their current formulations we referred are the most common ones stated in the conventional research papers. Namely, there are slight differences in the problem statements in various sources. In general, the engineering design problems have been modified and standardized since their first appearance in the NIA domain. Thus, the collection process was not a straightforward task. The general properties of the selected problems are summarized in Table 6.

TABLE 6: GENERAL PROPERTIES OF SELECTED ENGINEERING DESIGN PROBLEMS

Prob.	*n*	*Type of function*	*ρ*	*LI*	*NI*	*act*
g40	4	polynomial	41.50%	3	1	2
g41	4	polynomial	2.64%	2	5	1
g42	3	polynomial	0.75%	1	3	2
g43	7	nonlinear	0.21%	2	9	4
g44	11	linear	3.56%	0	10	1
g45	10	nonlinear	0.43%	5	6	5

PERFORMANCE MEASURES

Performance evaluation plays an essential role while proving the merit of a heuristic algorithm. Namely, we need objective criterions relying on a sound basis to confirm the effectiveness of the proposed method. Hence, it is highly important to define the performance measures in a systematical manner. This is a necessary step not only when comparing two different algorithms but also when tuning the control parameters of the algorithm during the design stage.

The employed performance measures may be directly obtained from the solutions found by the algorithm or extracted indirectly based on the statistical comparison methods. Needless to say, the performance evaluations should be conducted based on the experiments over several independent runs due to the stochastic nature of the NIAs. In general, the success of an algorithm is measured based on three basic metrics or some indirect values derived from these (Eiben and Smith, 2008):

- Solution quality (effectiveness)
- Speed (efficiency)
- Success rate (robustness)

The first performance measure considers the objective function value achieved within a limited computational time as the main indicator of the success of an algorithm whereas the second metric aims to measure the success based on the computational cost needed to achieve a predefined solution. Namely, the former specifies the computational time and assess the obtained objective function value while the latter one does the reverse.

In general, the solution quality is defined as the mean best objective function value (MBOV) over a certain number of independent runs. MBOV is calculated by identifying the best objective function value achieved in each run and taking the average of these values (Eiben and Smith, 2008). The best-ever- or the worst-ever-objective function value may be also of great interest for some test cases

A complementary strategy is to identify a satisfactory candidate solution and to measure the computational effort needed to reach that fitness level. A frequently used metric of computational effort is the number of fitness function evaluations (FES) required to find that solution.

Although FES is the most common metric, the direct indicator of computational effort is the CPU time which is defined by El-Ghazali (2009) as the time a processor spends in the execution of the algorithm. However, CPU time may change depending on the specifications of the machine, on which the algorithm is run, operating system, the programming language, and software architecture. Namely, a comparison based on CPU times necessitates the recreation of the algorithms under consideration. Because recreating an algorithm based on the details given in a scientific paper is, in general, very cumbersome, the most common practice in the literature we referred is to report more generic indicators, including the number of function evaluations (FES), the best value obtained so far, percentage of successful runs, and standard deviation, which are indifferent to machine settings.

While employing FES as a performance metric, it is assumed that an essential proportion of the CPU time is used by the evaluation process of fitness function. This assumption holds for most of the real-world optimization problems with computation-intensive objective functions (El-Ghazali, 2009). In this regard, the ratio of the total CPU time needed to execute an algorithm for certain number of iterations, T_2, over the time spent for fitness function evaluations, T_1, can be calculated, which may indicate the complexity of the algorithm, A_{comp} (Mallipeddi and Suganthan, 2010):

$$A_{comp} = (T_2 - T_1)/T_1. \tag{4}$$

To test whether FES is a proper metric in CEC2010 for computational effort, we have extracted the algorithm complexity values, A_{comp}, reported in each paper for 10 dimensional problem set participated in the competition. Surprisingly, the values range from 6.59% to 853%. Namely, the computational overhead induced by an algorithm may vary in a broad interval. Hence, FES may not be good indicator of the speed for some algorithms with very complex structures.

Additional to the criticism above, Eiben and Smith (2008) state that employing FES may be misleading if a NIA uses “hidden labor”, for instance, a time consuming local search heuristics embedded into the algorithm (Pelley, Innocente and Sienz, 2011; Sun and Garibaldi, 2010). In such cases, extra computational costs or additional function evaluations required by the local search procedure remain usually neglected.

The third criterion evaluates the robustness of the algorithm in terms of number of runs in which a predefined objective function value within the specified computation time has been achieved. The success rate (SR) is defined as the ratio of successful runs over total number of runs. The total number of runs should be specified such that the obtained results allow drawing statistical conclusions.

To sum up, a proper performance evaluation method should consider all these three aspects together. Also, we should keep in mind that different performance measures may yield different conclusions (Garcia et al., 2008; Smith, 2007).

Additionally, some graphical tools may be relatively useful to visualize the performance differences between various methods. Convergence graphs and box plots are very common means of visualization. The convergence graphs demonstrate the convergence behavior of the algorithm to a given objective value over the number of fitness function evaluations or the generations. They are usually represented in logarithmic scale. The box plots are used to show the solution quality and reliability of the algorithm over a specified number of independent runs. Obviously, the algorithm with higher mean fitness value and lower deviation is desired.

3.1. Performance Measures for COPs

This section is devoted to the performance measures specifically designed for COPs. In addition to the objective function value, we may define two additional solution quality measures related to the feasibility status of the solution: (i) the sum of constraint violation, and (ii) the number of constraints violated. In some cases, a feasible solution, though not be the optimal one, may be of paramount importance. Thus, finding feasible points quickly in a highly constrained search region is also a desired property.

Mezura-Montes et al. (2010) have utilized four metrics to measure the performance of NIAs for constrained optimization. At this point, we note that the performance measures given below rely on the assumption that a target solution can be identified while it is either the known global optimum or a satisfactory candidate solution. We denote this target solution with $f(\vec{x}^*)$.

Feasibility Rate (FR): With reference to the above discussion, they define feasibility rate (FR) metric showing the percentage of runs where feasible solutions are found. A feasible run is an independent trial at least with one feasible solution. FR is formulated as;

$$FR = \frac{ft}{tr} \tag{5}$$

where ft is the number of feasible trials, and tr is total number of independent runs. Of necessity, FR is between 0, and 1 (Mezura-Montes, Miranda-Varela and Del Carmen Gómez-Ramón, 2010).

Success Rate (SR): A successful trial is an independent run, where the absolute difference between the best solution $f(\vec{x})$ and the target/optimal value $f(\vec{x}^*)$ is less than a predefined threshold. $SR \in [0,1]$ is formulated as;

$$SR = \frac{st}{tr} \tag{6}$$

where st is the number of feasible trials (Mezura-Montes, Miranda-Varela and Del Carmen Gómez-Ramón, 2010).

Average Number of Fitness Evaluations for Optimality (AFESO): It is calculated by averaging the number of FES on each successful trial needed to reach the close neighborhood of $f(\vec{x}^*)$. AFESO is formulated as;

$$AFESO = \frac{1}{st}\sum_{i=1}^{st} FES_i. \qquad (7)$$

Average Number of Fitness Evaluations for Feasibility (AFESF): Similarly, we propose an additional metric to measure the average convergence FES required by the algorithm to identify the first feasible solution. AFESF is formulated as;

$$AFESF = \frac{1}{ft}\sum_{i=1}^{ft} FESF_i \qquad (8)$$

where $FESF_i$ denotes the number of fitness function evaluations needed to find the first feasible point in trial i.

Success Performance (SP): Mezura-Montes et al. (2010) combines two metrics, $AFESO$ and SR, to measure the speed and reliability of an algorithm;

$$SP = \frac{AFESF}{SR}. \qquad (9)$$

A low SP measure indicates that the algorithm is able to find the global optimum in less FES with high consistency.

Feasibility Performance (FP): Similar to SP, we may generate a feasibility performance metric which combines two metrics, $AFESF$ and FR, to measure the speed and reliability of an algorithm to identify at least one feasible solution;

$$FP = \frac{AFESF}{FR}. \qquad (10)$$

A low FP measure is preferred as it means that the algorithm requires less FES to find the first feasible solution and it is able to show the same behavior over several runs.

The performance metrics SR, and $AFESO$ employ a stopping criterion, $|f(\vec{x}) - f(\vec{x}^*)| \le \varepsilon$. Namely, the absolute error should be less than a predefined threshold ε before the search is terminated. However, the magnitude of $f(\vec{x}^*)$ is not considered while setting ε. More specifically, a threshold value, $\varepsilon \sim 1.e-4$, is not a proper choice for a target fitness value of similar order of magnitude. Alternatively, we propose to replace the absolute error criterion with relative percentage error (*RPE*) measure to terminate the search process;

$$RPE = |[f(\vec{x}^*) - f(\vec{x})]/f(\vec{x}^*)| \qquad (11)$$

Hence, Equation 4.8 sets a relative comparison metric and the search is terminated when $RPE \le \varepsilon$.

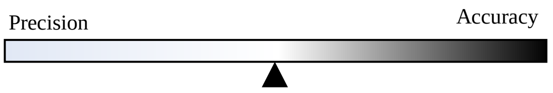

Figure 1: Precision-accuracy trade-off in optimization

Another point that must be highlighted is that the temporal requirements of the optimization problem to be solved may play an essential role during the selection process of a proper set of performance measures. Obviously, *the principle of precision-accuracy trade-off* applies in optimization domain and each optimization problem has a certain focal point that mainly lies on one side of *precision-accuracy*-scale, as depicted in Figure 1.

Namely, certain type of problems may require that the optimization algorithm delivers a *good solution* with high precision in a single run due to the temporal limitations. In general, this may be a case where the optimization problems must be solved repetitively in a dynamic environment within a short time interval (for instance, optimization of the traffic load in a network system). Thus, the available computational budget is finite. Because of this, the algorithm cannot be rerun several times and it should precisely provide good (may not be globally optimal) solutions. For this type of COPs, high average performance and low deviation are vital aspects. Hence, FR, SR, AFES or their combinations might be appropriate measures.

On the other hand, the nature optimization problem may allow exploring thoroughly the whole search region for a global optimum within adequately high temporal budget. The engineering design problems are usually of this type where a *high quality solution* in close vicinity of the global optimum should be found with several trials. In this case, precision is not the main objective since the best candidate point over all runs is selected as the final solution. Best-ever-objective function value is a proper indicator of performance.

CONCLUSION

Firstly, this chapter illustrated the principal characteristics of a *good* benchmark. Based on the abstracted guidelines, the benchmark problems are selected to test the different variants of the proposed algorithm.

Precisely defined performance metrics are of paramount importance during the design stage of a NIA dealing with COPs, Also, a sound basis is necessary while comparing the proposed algorithm with other methods. Thus, the most common performance measures are discussed in this chapter. Moreover, we highlighted that the set of proper performance measures may change due to the specifications of the COP. Based on the findings of this chapter, a parameters analysis on the proposed algorithms will be conducted and the resulting algorithm will be compared with state-of-the-art methods addressing the same set of benchmark problems.

REFERENCES

Costa, L. and Oliveira, P., 2001. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, [online] 25(2-3), pp.257–266. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0098135400006530> [Accessed 4 Mar. 2013].

Eiben, A.E. and Smith, J.E., 2008. *Introduction to evolutionary computing*. 2nd ed. Springer, p.304.

El-Ghazali, T., 2009. *Metaheuristics: from design to implementation*. [online] *Jonh Wiley and Sons Inc., Chichester*. Hoboken: John Wiley & Sons. Available at: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:

Metaheuristics:+from+design+to+implementation#0> [Accessed 13 Dec. 2012].

Gallagher, M. and Yuan, B., 2006. *A general-purpose tunable landscape generator*. *Evolutionary Computation, IEEE Transactions on*, .

Gandomi, A.H., Yang, X.-S. and Alavi, A.H., 2011. Mixed variable structural optimization using Firefly Algorithm. *Computers & Structures*, [online] 89(23-24), pp.2325–2336. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0045794911002185> [Accessed 7 Nov. 2012].

Garcia, S. et al., 2008. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, [online] 13(10), pp.959–977. Available at: <http://www.springerlink.com/index/10.1007/s00500-008-0392-y> [Accessed 2 Mar. 2013].

Kannan, B.K. and Kramer, S.N., 1994. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design. *Journal of Mechanical Design*, 116(June), pp.405–411.

Liang, J.J. et al., 2006. *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization*. *Evolutionary Computation*, pp.251–256.

Lin, Y.-C., Hwang, K.-S. and Wang, F.-S., 2004. A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems. *Computers & Mathematics with Applications*, [online] 47(8-9), pp.1295–1307. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S089812210490123X>.

Liu, H., Cai, Z. and Wang, Y., 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, [online] 10(2), pp.629–640. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1568494609001550> [Accessed 6 Oct. 2012].

Mallipeddi, R. and Suganthan, P.N., 2010. *Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real- Parameter Optimization*. Singapore.

Mezura-Montes, E., Miranda-Varela, M.E. and Del Carmen Gómez-Ramón, R., 2010. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, [online] 180(22), pp.4223–4262. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0020025510003415> [Accessed 29 Mar. 2012].

Pelley, C., Innocente, M. and Sienz, J., 2011. Memetic Particle Swarm for Continuous Unconstrained and Constrained Optimization Problems. In: *ICSI 2011: International conference on swarm intelligence*. [online] pp.1–9. Available at: <http://icsi11.eisti.fr/papers/paper_9.pdf> [Accessed 9 May 2012].

Smith, J., 2007. On replacement strategies in steady state evolutionary algorithms. *Evolutionary computation*, [online] 15(1), pp.29–59. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/17388778> [Accessed 17 Dec. 2012].

Spears, W.M., 2004. *Evolutionary algorithms: The role of mutation and recombination*. Berlin, Heidelberg: Springer, p.222.

Sun, J. and Garibaldi, J.M., 2010. A novel memetic algorithm for constrained optimization. *IEEE Congress on Evolutionary Computation*, [online] pp.1–8. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5585938>.

Yiqing, L., Xigang, Y. and Yongjian, L., 2007. An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints. *Computers & Chemical Engineering*, [online] 31(3), pp.153–162. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0098135406001281> [Accessed 4 Mar. 2013].

Yu, X. and Gen, M., 2010. *Introduction to Evolutionary Algorithms*. London: Springer.

Zahara, E. and Kao, Y.-T., 2009. Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, [online] 36(2), pp.3880–3886. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0957417408001735> [Accessed 28 Apr. 2012].