Volume 4, Issue 10, October 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Multi-Objective Optimization in R.C.C Structures

Karthiga @ Shenbagam .N, Preetha.V, Loganayagan .S, Jayanthi .V, Chandramohan .N
Assistant Professor, Bannari Amman Institute of Technology,
Sathyamangalam, Erode District, India

Abstract- Nowadays very popular optimization methods, Evolutionary Algorithms (EAs) are presented as one of the possible ways how to solve today's challenging optimization problems.. Traditionally, the EAs have been developed for single-objective problems (SOPs) and therefore they are not so suitable for problems coming from engineering practice where we usually deal with multi-objective, constrained and often mixed integer-continuous optimization problems (CMOPs). Based on the above mentioned notation, four particular examples of EAs that have been developed at the workplace of the author. In particular, three of them are based on the combination of Genetic Algorithms with the Differential Evolution; the last algorithm is the Differential Evolution alone. These optimization algorithms are then used to solve several tasks from engineering practice as well as two test functions and their advantages and disadvantages are shown. The objective function is then the least square error function, which contains differences between values of a known stress-strain curve and values from a micro plane model simulation. Several approaches are tested here to solve the introduced problem. An estimation by an artificial neural network trained on approximations of stress-strain curves shows that some properties can be predicted well but a significant error in other coefficient is obtained. Next, a parallel version of the evolutionary-algorithms-based global optimizer SADE is directly used to obtain required parameters by varying them within a nonlinear finite element analysis. The first main result is that this time consuming analysis can be solved by a parallel analysis in reasonable time. The second outcome is the fact that the objective function corresponding to the identification problem has several local minima, which are characterized by similar values but are far from each other. To solve the above mentioned obstacles and in the view of recent research in this domain, a new methodology is also presented: an application of a Latin Hypercube Sampling method as well as a sensitivity analysis are applied not only to investigate the influence of individual material model parameters, but also to minimize the need of training samples for an artificial neural network. Several promising results along with some concluding remarks are presented.

Keyword- CMOP SOP, IASA, RASA

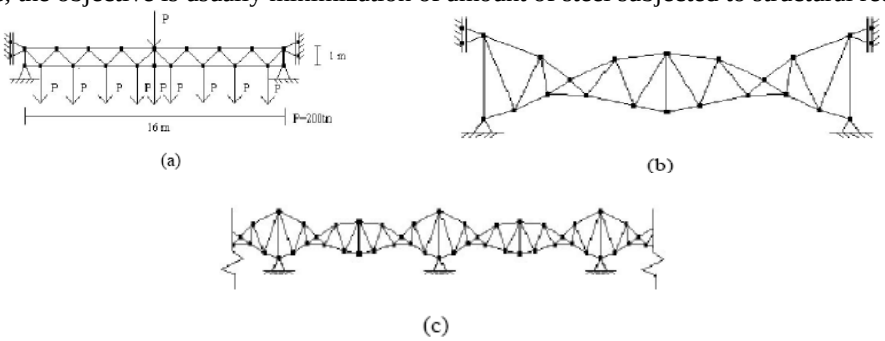
I. INTRODUCTON

When dealing with engineering problems, one may discovered two different areas of optimization.The first is usually called **Global optimization.** The second area, usually called **Structural optimization**, can be described as an applied science, where the methods from the Global optimization area are applied to a model of a structure or a material.

1.1 FORMS OF OPTIMIZATION

1.1.1 Topology optimization

By topology optimization we understand finding a structure without knowing its final form beforehand. Only the environment, optimality criteria and constraints are known. These tasks usually come from the mechanical engineering area, where designing parts of cars or aircraft are the most frequent topics. The major civil engineering representatives serve as a decision tool in selecting an appropriate static scheme of a desired structure. They are mostly applied to the pin-jointed structures, where the nodal coordinates of joints are optimization variables. Based on the position of supports and objective functions, several historically well-known schemes can be discovered (see Fig. 1.1). The typical example of this optimization form within the reinforced concrete area is placement of steel reinforcing bars into a concrete block. In other words, we search for the most suitable strut-and-tie model, in which the position of steel is not known in advance. In this case, the objective is usually minimization of amount of steel subjected to structural requirements.



(a) (b)

(c)

1.1.2 Shape optimization

In this form of optimization the topology of structure is known a-priori but there can be some part and/or detail of the structure, in which, for instance, high stresses can produce problems. Therefore the objective is usually to find the best shape that will result in the most suitable stress distribution. Parameters of shapes are dimensions of the optimized parts or a set of variables describing the shape, e.g. coefficients of spline functions. Examples for the reinforced concrete area herein can be finding the proper shape of holes within plate members ,the shape of a beam with holes [Yoshimura et al., 2002], the optimal shape of pre-cast retaining structures or the ribs within box columns From the mathematical point of view, two representations of variables - continuous and discrete ones - can be found within the shape optimization area.

1.1.3 Size optimization

In this form of an optimization4 a structure is defined by a set of sizes, dimensions or cross sections. These are combined to achieve the desired optimality criteria. Within this area two main groups of structures can be distinguished.

Discrete structures. Here pin and rigid jointed structures can occur. In the case of steel structures in particular, nearly all possible optimization problems have been subjected to some form of investigation. To list a few successfully solved problems, optimization of structures with semi-rigid connections, optimization against buckling or a finding minimum weight in connection with a minimum number of steel profiles used in a design can be found in the corresponding literature. Many small-size examples from this area serve as benchmarks for different types of optimization algorithms, with the 10-bar truss and the 25-bar space truss, e.g. being the most often cited ones. Again, here all variables are selected from the pre-defined discrete admissible set. But this is not exactly the case of reinforced concrete frame structures which are more likely to be part of the next group of structures:

Continuum structures.

This group contains beam-like structures defined by continuous variables, which are not known in advance in contrast to the previous case. The basic example is a beam with moments of inertia defined as a continuous variable. All reinforced concrete optimization tasks, where the area of reinforcing steel is an unknown, will be the proper representatives of this group, too.

1.1.4 Topography optimization

This form is the least investigated part of structural optimization. Here you can find the search for a proper shape for shell, membrane or tent like structures. Only few papers on this topic can be found in the literature, with even fewer dealing with reinforced concrete structures. And finally, the *Mathematical Programming* methods are known as the only efficient solutions for this type of optimization problems.

1.2 OPTIMIZATON METHODS

From the mathematical point of view, an engineering task can be understood as a **multiobjective**, **constrained** and often **mixed integer-continuous** optimization problem (**CMOP**). Below we offer a formal definition:

1.2.1 Constrained Multi-objective Optimization Problem. A general **CMOP** includes a set of n parameters (decision variables), a set of k objective functions, and a set of m constraints. The optimization goal is to

$$\begin{aligned}
\text{minimize} \quad & \mathbf{y} &&= \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})) \\
\text{subjected to} \quad & g_j(\mathbf{x}) &&= 0\,, \quad j = 1, \ldots, ne\,, \\
& g_j(\mathbf{x}) &&\leq 0\,, \quad j = ne+1, \ldots, m = ne+ni\,, \\
\text{where} \quad & \mathbf{x} &&= (x_1, x_2, \ldots, x_n) \in \mathbf{X}\,, \quad \mathbf{X} \subset \{\mathbb{N}, \mathbb{R}\}^n\,, \\
& \mathbf{y} &&= (y_1, y_2, \ldots, y_n) \in \mathbf{Y}\,, \quad \mathbf{Y} \subseteq \mathbb{R}^n\,,
\end{aligned}$$

ne and ni are the numbers of equalities and inequalities, respectively, x is the decision vector, y is the objective vector, X is denoted as the decision space and Y is called the objective space. Note that the set of all feasible solutions, i.e. all solutions x for which conditions are satisfied, is denoted X_f and its image in the objective spaceis referred to as Yf , i.e. Y_f = f (X_f). Also note that we assume minimization hereafter, the statements for maximization or combined minimization/maximization are similar.

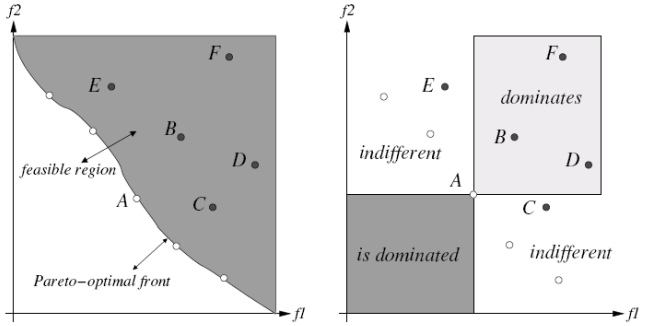


Figure 2: A graphical interpretation of the Pareto dominance.

The most ambitious task of an optimization method is usually finding the *global* minimum or maximum, which yet fulfills the given constraints. Whether such a solution is "optimal" (from the point of view of achieving the desired practical goal) or not, is another question, which depends on the formulation of the problem, but not on the system of finding the optimum by the search method. Therefore, we will understand the optimization method as a *black box* application and the only "interesting" evaluation of a selected algorithm is its performance, i.e. usually reliability, in finding global optimum. There is no best algorithm, whether or not that algorithm is "evolutionary", and moreover whatever an algorithm gains in performance on one class of problems is necessarily offset by that algorithm's performance on the remaining problems. In other words, there is no best optimization algorithm. On the other hand, it does not mean that for a specific problem you cannot find a superior optimization algorithm. Furthermore,
This is why there are so many available algorithms and tools for global optimization.

1.2.2 Solving multi-objective problems

The main difference between single-objective and multi-objective optimization is that in the solutions can be completely ordered according to the objective function f. the case of single-objective problems (**SOP**s) only one global optimumexists1, but in the case of multi-objective problems (**MOP**s) conflicting objectives can cause situation where no solution is superior to others. For the mathematical expression of the above mentioned statement we need to define the so-called **Pareto dominance**

For any two decision vectors a and b,

$$\begin{aligned}
\mathbf{a} \succ \mathbf{b} \quad & (\mathbf{a} \text{ dominates } \mathbf{b}) \quad \textit{iff} \quad \forall i : f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \wedge \exists i : f_i(\mathbf{a}) < f_i(\mathbf{b}), \\
\mathbf{a} \succeq \mathbf{b} \quad & (\mathbf{a} \text{ weakly dominates } \mathbf{b}) \quad \textit{iff} \quad \forall i : f_i(\mathbf{a}) \leq f_i(\mathbf{b}), \\
\mathbf{a} \sim \mathbf{b} \quad & (\mathbf{a} \text{ is indifferent to } \mathbf{b}) \quad \textit{iff} \quad \exists i : f_i(\mathbf{a}) < f_i(\mathbf{b}) \wedge \exists j : f_j(\mathbf{a}) > f_j(\mathbf{b}).
\end{aligned}$$

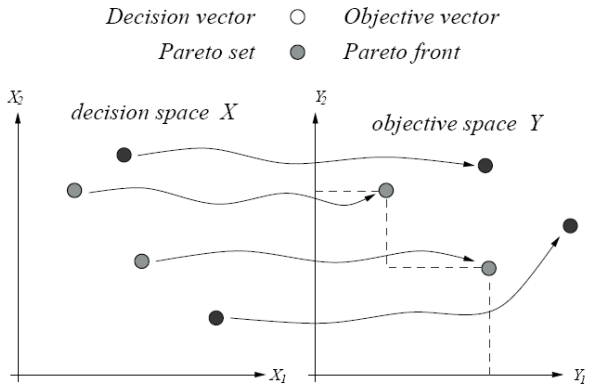


Figure 3: A basic notation for a multi-objective optimization.

1.2.3 Gradient methods from the Mathematical Programming group are the best examples of the (1 , 1)-algorithm. They contain only one operator $x_{t+1} = \mathbf{mut}_1^1(x_t) = x_t + \alpha_t d_t$, where d_t is a direction of the descent and _t is the step in the direction d_t. Since the step d_t is assumed to be in a descent direction, the selection is redundant. Hence the general gradient-based

$$opt_{GRAD}(^{t+1}I) = \mathbf{mut}_1^1(^tI) \ .$$

algorithms can be written as

1.2.4 Simulated Annealing (**SA**)

It is another traditional optimization method, which will be the best example of (1+1)-algorithm. Again, with μ = 1, the only operator is mutation, in this case some random function. The actual implementation of the mutation operator is not important; it must be only ensured that each point in the space is visited at least once in the infinite number of runs. The core of this algorithm is a selection process where the energy difference is given by_E = f(b)−f(a), T is the artificial temperature determined by the so-called cooling schedule. $T \approx \frac{T_0}{\ln t}$

$$\mathbf{sel}_2^1(\mathbf{a}, \mathbf{b}) = \begin{cases} \mathbf{a} \ , & \text{iff} \quad u(0,1) \leq p = \dfrac{1}{1 + e^{\Delta E/T}} \\ \mathbf{b} & \text{otherwise,} \end{cases}$$

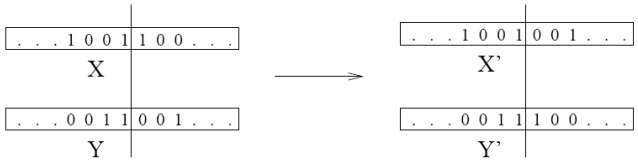


Figure 4: The cross-over operator.

1.2.5 Parameters tuning

Besides many advantages that Evolutionary Algorithms have, they are connected with several difficulties. In addition to the demand for thousands of evaluations of an objective function, it must be noted, that one of the most important obstacles is the finding of proper parameters – either the cooling scheme in **SA**, or several cross-over and mutation parameters. During last several years, this problem has been solved usually by minimizing needed parameters. However, it can be stated, that there is a dependency between a number of parameters and the popularity of the method - lower a number of parameters, higher a number of satisfied users.

1.2.4 Handling of constraints

Thus far we have supposed that the optimal solution is chosen from the feasible set of solutions, i.e. from X_f. In the case of constrained optimization there arises the need to tackle the problem of promising solutions that, unfortunately, violate some constraints. In the literature several strategies can be found, but we will limit our attention only to methods that are easily applicable to the Evolutionary Algorithms nature and have proved their reliability for engineering optimization tasks. Note that traditional methods come from the **SOP** area and therefore the adopted notation will be for one objective function only. One example of multi-objective approaches will be introduced in the last part of this section.

1.2.5 Death penalty approach

The term "death penalty" stands for the rejection of an infeasible solution from a search process. The advantage of this strategy is its easiness, the disadvantage can occur in problems, where the feasible domain is not convex or is divided into a number of disjoint parts. Also in the case of highly constrained problems, where the problem of finding the first feasible solution can arise, this method usually fails. To overcome these obstacles, the "death penalty" is often combined with repair or problem-dependent search operators.

1.2.6 Penalty function methods

Note that in the present work we use only *exterior* penalty functions which penalize infeasible solutions, which is in contrast with the *interior* penalty approach that penalizes feasible solutions near the boundary of a feasible domain. The former one admits infeasible solutions during the whole optimization process and therefore cannot ensure the feasibility of the found optimum. On the other hand, the big advantage is that the optimization can start everywhere. Therefore this procedure is much more flexible than the other one. The latter works only with feasible vectors, therefore the found optimum as well as intermediate solutions always fulfill the given conditions. The disadvantages are clear - this procedure cannot work with equality constraints (because it is almost impossible not to violate them) and must start in the feasible area. The *exterior* penalty approach is one of the most often used approaches for handling constraints, especially within the Evolutionary Algorithms community. The basic idea is to move the solution from the infeasible to feasible space by adding some value to the objective function, i.e.

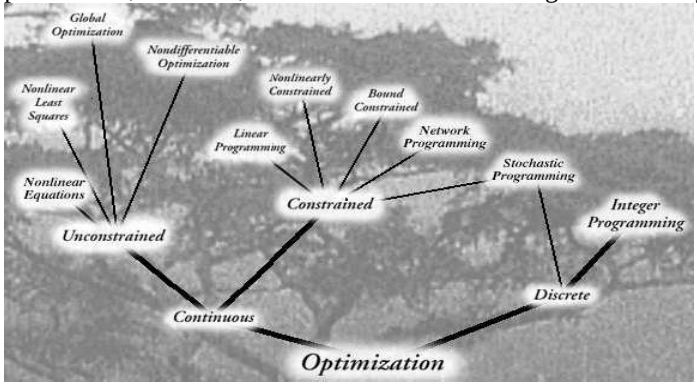
$$f_f(x) = f(x) + Q(x),$$

where Q is equal to zero if the solution is feasible or equals some positive value (in minimization problems) otherwise. The value of Q can be defined on the three different bases:

1. An individual is penalized only for its unfeasibility, with its distance from the feasible set playing no role.
2. The value can be defined as a measure of distance from the feasible domain or
3. as a price or energy spent to repair such a solution.

1.2.7 Constraints as objectives

During an optimization process we admit infeasible solutions but in the end we want to obtain only the feasible ones, i.e the Pareto set. This can be done by minimizing the distance between infeasible and feasible regions. And this is nothing more than next optimization process and, therefore, it can be included in the original multi-objective one.



1.2.8 Integer coding

In the case of a discrete set of real numbers, the indices will play the role of alternative variables. Consider x = {x1, x2, . . . , xn} as a vector of n variables, integer or real numbers xi, defined on a closed interval $min_i \leq xi \leq maxi$ on an appropriate domain $x \in \{N,R\}^n$. Further assume that each variable x_i is to be represented with some required precision pi,

defined as the smallest unit the number xi can attain. Then, each variable x_i can be transformed into a nonnegative integer.

1.2.9 Binary coding

In accordance with the previous , we may deal with a problem of decoding an integer vector z into a binary string provided that this operation is not covered by the selected programming language. First, the length q of the final binary string is needed and can be computed e.g. by

$$q = \left\lfloor \frac{\ln(max_i - min_i) - \ln(p_i)}{\ln\ 2} \right\rfloor + 1\ ,$$

where the notation is the same as in the previous. Then the binary string $B^i \in \{0; 1\}^q$ pertinent to the integer number z_i can be obtained by

$$B_j^i = z_i / 2^{j-1} \mod 2\ , \quad j = 1, \ldots, q\ .$$

As a result, B_i will contain the list of bits of the number z_i in the ascending order, i.e. from the least to the most important one. The recursive relationship is given by

$$z_i = \sum_{j=1}^{q} B_j^i\ 2^{j-1}\ .$$

1.2.10 Real coding

The beauty of this encoding can be seen in its easiness, because in common programming languages the assignment

$$\mathbf{z} = \mathbf{x}\ , \quad \mathbf{z} \in \mathbb{R}^n\ , \quad \mathbf{x} \in \{\mathbb{N}, \mathbb{R}\}^n$$

will usually work. The opposite direction can be handled with some truncation or rounding procedure. For example,

$$x_i = \lfloor z_i\ /\ p_i + 0.5 \rfloor\ p_i$$

is the most common

II. PROBLEM DESCRIPTION

2.1 Chebychev problem

The Chebychev trial polynomial problem is one of the most famous optimization problems. Our goal is to find such coefficients of a polynomial constrained by the condition that the graph of the polynomial can be fitted into a specified area (see Fig. 5). Thus, the optimized values are the parameters a_i of a polynomial expression:

$$f(x) = \sum_{i=0}^{n} a_i x^i,$$

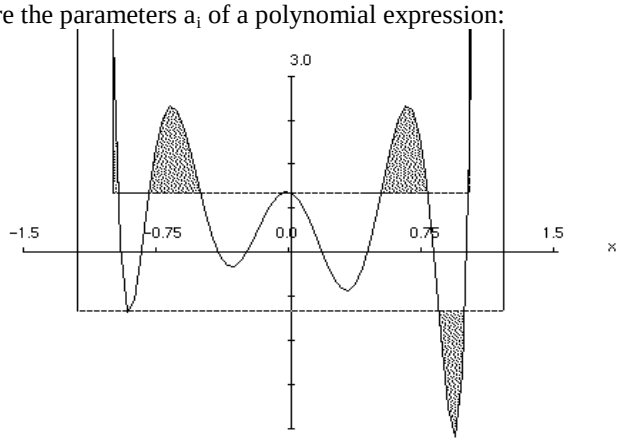


Figure 5: A graph of a Chebychev polynomial (n = 8).

and the value of objective function is determined as a sum of the areas, where the function graph exceeds a given boundary (hatched areas in Fig. 5).

2.2 Type 0 *function*

This trial optimization problem was proposed in [Hrstka and Kuˇcerov´a, 2000] to examine the ability of the optimization method to find a single extreme of a function with a high number of parameters and growth of computational complexity with the problem dimension. For this reason, we used a function with a single extreme on the top of the high and narrow peak:

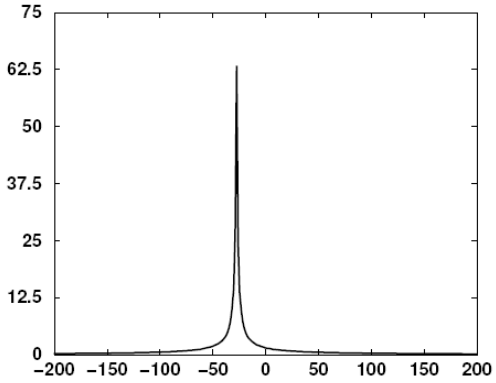


Figure 6: An example of a *type 0* function.

$$f(\mathbf{x}) = y_0 \left(\frac{\pi}{2} - \arctan \frac{\|\mathbf{x} - \mathbf{x}_0\|}{r_0} \right),$$

where x is a vector of unknown variables, x0 is the point of the global extreme (the top of the peak) and y0 and r0 are parameters that influence the height or the width of the peak, respectively.
An example of such a function on one dimensional domain is shown in Fig. 6.

2.3 Reinforced concrete beam layout

An effort to create an optimal design of a steel-reinforced concrete structure is as old as the material itself. In present times emphasis is put on this problem due to widespread use of RC structures in Civil Engineering. Frame structures are a major part in this field with beams playing an important role as one of the basic building blocks of this construction system. An objective is to choose the best design from all possible configurations that can create the requested structure – in our case a continuous beam (see Fig. 7). The total cost of the structure is used as a comparison factor. An advantage of the financial rating is its natural meaning to non-experts and easiness of constraints implementation. In our particular case, the objective function reads

$$f(\mathbf{X}) = V_c P_c + W_s P_s + \sum p f_i \,,$$

where V_c is the volume of concrete and Ws is the weight of steel; P_c and P_s are the price of concrete per unit volume and steel per kilogram, respectively. From the mathematical point of view the penalty function pf_i is the distance between a solution and the feasible space, or

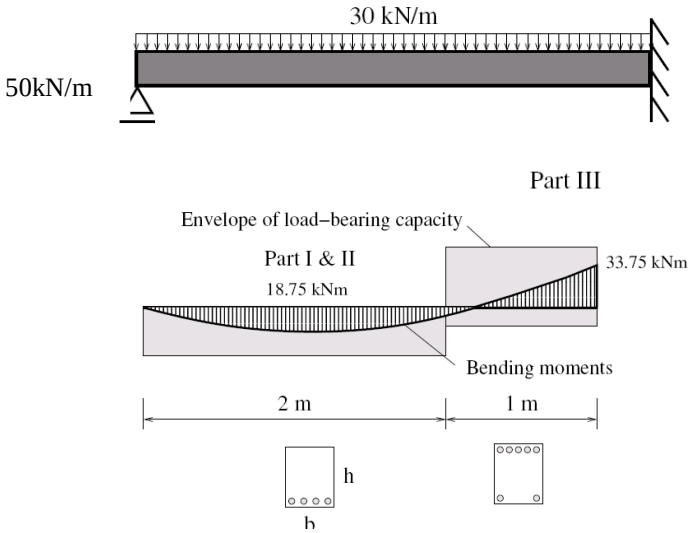


Figure 7: A continuous beam subjected to a uniform loading.

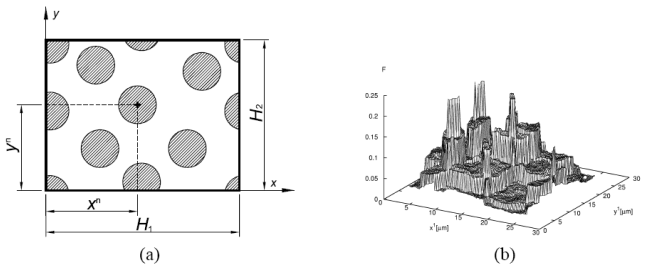


Figure 8: (a) Geometry of a periodic unit cell, (b) An example of the objective function.

The algorithmic scheme is similar to the Genetic
Algorithms but it is much simpler:
1. At the beginning an initial population is randomly created and the fitness function value is assigned to each individual.
2. For each chromosome in the population, its possible replacement is created using the differential operator as described above.
3. Each chromosome in the population has to be compared with its possible replacement and if an improvement occurs, it is replaced.
4. Steps 2 and 3 are repeated until some stopping criterion is reached. As it can be seen, there are certain features that distinguish this method from the Simple Genetic Algorithm, namely:
• the crossing-over is performed by applying the differential operator
• the selection operation like the roulette wheel, for example, is not performed, the individuals that are going to be affected by the differential operator, are chosen purely randomly,
• selection of individuals to survive is simplified to the mentioned fashion: each chromosome has its possible replacement and if an improvement occurs, it is replaced,
• the mutation operator is not introduced as the authors of **DE** claim that the differential operator is able to replace both mutation and uniform crossover known from basic **GA**s.

Contrary to the Differential Evolution, this method uses an algorithmic scheme very similar to
Evolution Strategies:
1. As the first step, the initial population is generated randomly and the fitness function value is assigned to all chromosomes in the population.
2. Several new chromosomes are created using the mutation operators - the mutation and the local mutation (number of them depends on a value a of parameter called *radioactivity*, which gives the mutation probability).
3. Other new chromosomes are created using the simplified differential operator as was described above; the whole amount of chromosomes in the population doubles.
4. The fitness function values are assigned to all the newly created chromosomes.
5. The selection operator is applied to the double-sized population, so the amount of individuals is decreased to its original value.
6. Steps 2-5 are repeated until some stopping criterion is reached.

2.4 Test computations and results

Each of the methods introduced in the previous section has been tested on all of the presented optimization problems. The methodology that has been used for our computations is based on the following criteria:
• For each problem and each method the computation was run 100 times to avoid an influence of random circumstances.
• For all cases, the number of successful runs (which can be traded as the probability of success or the reliability of the method) is presented.
• If the number of successful runs is non-zero, the average number of fitness calls of all successful runs is also presented.
Further details of individual function settings and methodology for results evaluation can be found in the next subsections.

2.5 Results for the Chebychev problem

Table 1: Results for the Chebychev polynomial problem.

Method	IASA	RASA	DE	SADE
Successful runs	100	100	100	100
Average number of fitness calls	10342	47151	25910	24016

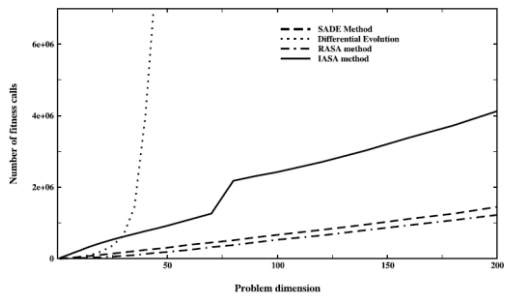


Figure 9: A comparison of results for the *type 0* function.

2.6 Results for the type 0 trial function

Test computations for the *type 0* problem were performed for a wide set of problem dimensions, ranging from 1 to 200. The upper bound on each variable was set to 400, while the lower bound value was −400. For each run, the position of the extreme was randomly generated within these bounds and the height of the peak y_0 was generated from the range 0–50. The parameter r_0 was set to 1. The computation was terminated when the value of the objective function was found with a precision greater than 10^{-3}. The results are given in the form of the growth of computational complexity with respect to the problem dimension. For each dimension, the computation was run 100 times and the average number of fitness calls was recorded (see Fig.9 and Table 2).

Table 2: Average number of fitness calls for the type 0 function

Problem Dimension	IASA	RASA	DE	SADE
10	446,120	14,113	39,440	46,958
30	811,760	75,375	654,650	172,539
50	998,150	188,882	N/A	305,337
100	2,584,590	526,992	N/A	663,085
140	4,193,800	826,046	N/A	949,197
200	4,691,2100	1,330,513	N/A	1,448,540

2.7 Results for the reinforced concrete beam layout problem

The basic parameters subjected to optimization were the beam width b, which was assumed to take discrete values between 0.15 m and 0.45 m with the step 0.025 m and the beam height h ranging from 0.15 m to 0.85 m with the step 0.025 m. For each of the three parts of a beam, the diameter and the number of longitudinal reinforcing bars located at the bottom and the top of the beam, spacing and the diameter of stirrups and the length of the corresponding part were optimized. Lower bounds were selected for the sake of structural requirements; solutions exceeding upper bounds were considered to be irrelevant for the studied examples. However, from the optimization point of view, bounds can be easily adjusted to any reasonable value.The number of longitudinal bars was restricted to the range 0–15, the spacing of stirrups was assumed to vary from 0.05 m to 0.40 m with the 0.025 m step. The profiles of longitudinal bars were drawn from the list of 16 entries while for the stirrups, only 4 diameters were considered. This finally results in 18 independent variables. Note that the maximal number of longitudinal bars presents only the upper bound on the searched variable; the specific restrictions given by Codes of Practice are directly incorporated in the objective function.

Table 4: Results for the periodic unit cell problem

Method	IASA	RASA	DE	SADE
Successful runs	100	100	100	100
Average number of fitness calls	13641	12919	93464	55262

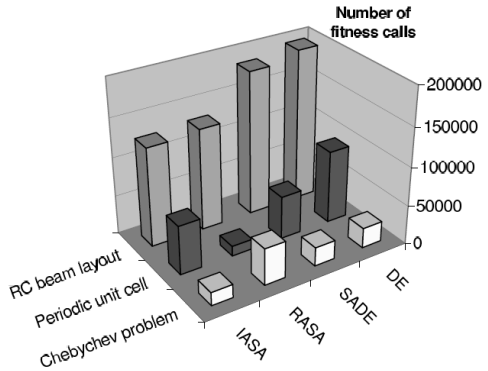


Figure 3.8: A comparison of results for Chebychev polynomial, reinforced concrete beam layout and periodic unit cell problems.

III. CONCLUSIONS FOR COMPARISON

3.1 Differential Evolution

The Differential Evolution algorithm showed to be very efficient and robust for moderate-sized problems, but its performance for higher dimensions deteriorated. Moreover, the small number of parameters, see Table C.2, is another advantage of this method3. However, the results suggest that the absence of mutation-type operator(s) is a weak point of the algorithm.

3.2 Simplified Atavistic Differential Evolution

The **SADE** algorithm was able to solve all problems of our test set with high reliability and speed. Although it needed a larger number of function calls than other methods (see Table 3.5), the differences are only marginal and do not present any serious disadvantage. Another attractive feature of this method is the relatively small number of parameters, see Table C.3.

3.3 Real-valued Augmented Simulated Annealing The **RASA** algorithm was successful for all presented problems; the average number of function calls was comparable to the other methods. The obvious disadvantage of this algorithm is a large number of parameters (Table C.1), which can result in a tedious tuning procedure. On the other hand, as follows from Appendix C, only two types of parameter settings were necessary – one for the continuous and one for the discrete functions.

3.4 Integer Augmented Simulated Annealing The **IASA** algorithm was the most successful and fastest method on problems with small dimensions. But on the problems with larger dimensions and with a higher number of local minima, the algorithm suffers from premature convergence and limited precision due to integer coding of variables. In addition, initial tuning of individual parameters, see Table 4, presents another drawback of this method.

Table 5: Overall performance of methods.

Method	IASA	RASA	DE	SADE
Chebychev Problem	1	4	3	2
Type 0 Test function	3	1	4	2
Concrete beam layout	1	2	4	3
Periodic unit cell	3	1	4	2
Summation	8	8	14	9

The summary results are given in Table 3.5 to quantify the overall performance of the individual methods. Each of the method is ranked primarily with respect to its success rate and secondary with respect to the average number of fitness calls. The sum then reveals the overall performance of the method.

3.5 Final comments In our opinion, several interesting conclusions and suggestions can be made from the presented results. Each of them is discussed in more detail.

• The performance and robustness of the **SADE** method was distinguishly better than for the **DE** algorithm. This supports an important role of a mutation operator(s) in the optimization process.

• Although algorithms were developed independently, all use some form of differential operator. This shows the remarkable performance of this operator for both real-valued and discrete optimization problems.

• The most successful methods, the **SADE** and **RASA** algorithms, both employ a variant of "local mutation". This operator seems to be extremely important for higher-dimensional *type-0* functions, where these methods clearly outperform the others.

• Slightly better results of the **RASA** method can be most probably attributed to the reannealing/ restarting phase of the algorithm (a trivial but efficient tool for dealing with local minima) and to the search for an identical individual. The procedure for local minima assessment was implemented to the **SADE** method, incorporation into the **IASA** algorithm is under development.

• When comparing methods based on the discrete coding of variables with real-encoded ones it becomes clear that for continuous functions the methods with the real coding perform better. Nevertheless, after implementing new features, like those mentioned before, the performance is expected to be similar. On the other hand, the advantage of the **IASA** algorithm is the possibility of its use for discrete combinatorial problems like the Traveling salesman problem. Therefore, from the practical point of view, the **SADE** method seems to be the most flexible alternative due to its simplicity and small number of parameters.

REFERENCES

[1] [Abdallaa et al., 1996] Abdallaa, K.M., Alshegeirb, A., and Chenc,W. F. (1996). Analysis and design of mushroom slabs with a strut-tie model. *Computers & Structures*, 58(2):429–434.

[2] [Adeli and Kamal, 1986] Adeli, H. and Kamal, O. (1986). Efficient optimization of space trusses. *Computers & Structures*, 24(3):501–511.

[3] [Adleman, 1994] Adleman, L.M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024.

[4] [Alander, 1995] Alande8r, J. T. (1995). Indexed bibliography of distributed genetic algorithms. Report 94-1-PARA, University of Vaasa, Department of Information Technology and Production Economics. Available at Springer-Verlag, New-York Berlin Heidelberg.

[5] [Andre et al., 2000] Andre, J., Siarry, P., and Dognon, T. (2000). An improvement of the standard genetic algorithmfighting premature convergence in continuous optimization. *Advances in Engineering Software*, 32(1):49–60.

[6] [Annicchiarico, 2003] Annicchiarico, W. D. (2003). Three dimensional shape optimization of complex model by using distributed micro genetic algorithms. In [Bugeda et al., 2003].

