A Preliminary Study of Fitness Inheritance in Evolutionary Constrained Optimization

Efrén Mezura-Montes[1], Lucía Muñoz-Dávila[2], and Carlos A. Coello Coello[3]

[1] Laboratorio Nacional de Informática Avanzada (LANIA A.C.), Rébsamen 80, Centro, Xalapa, Veracruz, 91000, MÉXICO emezura@lania.mx
[2] Instituto Tecnológico de Apizaco, Av. Instituto Tecnológico S/N, Apizaco, Tlaxcala, MÉXICO lucy@itapizaco.edu.mx
[3] Departamento de Computación, Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F., 07300, MÉXICO ccoello@cs.cinvestav.mx

Summary. This document presents a proposal to incorporate a fitness inheritance mechanism into an Evolution Strategy used to solve the general nonlinear programming problem. The aim is to find a trade-off between a lower number of evaluations of each solution and a good performance of the approach. A set of test problems taken from the specialized literature was used to test the capabilities of the proposed approach to save evaluations and to maintain a competitive performance.

1 Introduction

The general nonlinear programming problem (NLP) is formally defined as follows: Find $\mathbf{x}$ which minimizes $f(\mathbf{x})$ subject to: $g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m$, and $h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p$ where $\mathbf{x} \in \mathbb{R}^n$ is the vector of solutions $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, where each $x_i, \quad i = 1, \ldots, n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$ which define the search space $\mathcal{S}$, $\mathcal{F}$ is the feasible region and $\mathcal{F} \subseteq \mathcal{S}$; m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear).

Evolutionary Algorithms (EAs) are widely used as alternative techniques (mathematical programming approaches are always the first choice) to solve the NLP [1].

However, three shortcomings can be identified when applying EAs to solve the NLP:

- A set of parameter values must be defined by the user and the behavior of the EA in the search process depend of these values.
- In the presence of constraints, a constraint-handling mechanism must be added to the EA in order to incorporate feasibility information in its selection and/or replacement process(es) and this mechanism may involve additional parameters to be fine-tuned by the user.

- Usually, the EA requires to perform several evaluations of the objective function of the problem and its constraints as to find a "good" solution. Furthermore, for some real-world problems, the evaluation of a single solution may require a high computational cost.

This work is focused on the last disadvantage. A common approach to deal with it is the use of fitness approximation models which prevent the EA to use the original (and maybe costly) model of the problem every time a new solution is evaluated [2]. Polynomial models, Kriging, Neural Networks and Support Vector Machines are the main approaches used for fitness approximation [2]. In fact, some applications of them are reported in the specialized literature [3]. But, despite the fact that the use of these models decreases the number of evaluations required by an EA, they indeed add an extra computational cost related to its generation and updating process.

On the other hand, this work aims to propose a simpler approximation mechanism, known as fitness inheritance [4], which prevents a new solution of being evaluated. Instead, it inherits the fitness value from its parents. This mechanism is added to an Evolution Strategy [5] which is used to solve the NLP problem.

This document is organized as follows: Section 2 presents a brief summary of approaches for evaluation savings used to solve the general NLP problem adopting EAs. After that, Section 3 includes a description of our proposed approach. Then, in Section 4 the experimental design, the results obtained and their corresponding discussions are shown. Finally, Section 5 summarizes our findings and presents some possible paths for future work.

2 Related work

There are several approaches reported in the specialized literature about fitness approximation models used in EAs. However, the main efforts have been centered either in unconstrained global optimization problems [2], or in multiobjective optimization problems [6, 7]. We will focus here on the approaches proposed precisely to solve the general NLP problem (with constraints):

- Runarsson [8] used a nearest-neighborhood model to solve a set of NLP problems. The results showed that, using just the information of the closest solution in the decision space (defined by the lower and upper limits of the decision variables (see Section 1) based on a set of solutions found during the search provides a more competitive performance with respect to using the average value of a set of solutions in the vicinity of the new solution to be evaluated. The overall performance obtained in this approach is highly competitive, but its main disadvantage is that it requires to store a considerable high number of solutions to obtain a better approximation of the new solutions to be generated, and this storage may be prohibitive in some cases.

- Mezura-Montes and Coello Coello [9] proposed an approach based on Differential Evolution (DE) [10] to solve the general NLP by reducing the computational cost measured by the number of evaluations. Instead of using approximation models, the authors proposed to use features related to the search engine, DE in this case, in order to avoid the evaluation of some solutions and assigning a zero fitness value (death penalty). Afterwards, these solutions are discarded. This mechanism also slowed down the convergence of the approach and, for some problems, better solutions were found. The main disadvantage of the approach is that it only works with DE.
- Won and Ray [3] compared Kriging and Cokriging surrogate models with radial-basis functions using a set of five constrained engineering design problems. They found that the results obtained by using these models were very competitive. However, these models may be more difficult to implement.

3 Our proposed approach

Motivated by the findings of Runarsson with respect to the use of information of the closest solution in the decision space, but trying to avoid keeping a large set of solutions and also aiming to provide an easy implementation, we propose a simple approach which provides competitive results, but decreasing the number of evaluations required by the EA.

Fitness inheritance (FI) was originally proposed by Smith [4]. The idea is to approximate the values of the objective function of an offspring based on the values of its parents. Smith initially proposed to compute the average of the parents' objective function values. He alternatively proposed to use the distance of each parent to its offspring in the decision space to determine the amount of contribution of each parent's objective function value to the corresponding value of the offspring. When using FI, several evaluations may be saved during the evolutionary process.

We then propose to use Smith's ideas, originally incorporated into a genetic algorithm (GA) to solve simple unconstrained optimization problems, in an Evolution Strategy designed to solve constrained optimization problems, i.e. the general NLP problem.

Evolution Strategies (ESs) were developed in Germany in 1964 to solve complex hydrodynamical problems. The researchers involved in this work were Ingo Rechenberg, Hans-Paul Schwefel and Paul Bienert [11].

The ES simulates the evolution at an individual level; thus, this approach incorporates a crossover operator, either sexual or panmictic (more than two parents), which, however, acts as a secondary operator. Mutation is the main operator and it is used with random numbers generated under a Gaussian distribution. The mutation values vary over time and are self-adaptive. The

encoding is at a phenotypic level (i.e., no encoding is required). Parent selection is performed in a purely random process (i.e., it is not based on fitness values) and the replacement process is deterministic and extinctive, based on fitness value (the worst individuals have zero probability of survival).

There are several versions of ESs. The first of them is the $(1 + 1)$-ES. This version has only one solution which is mutated to create one child. If the child is better than the parent, it will replace it. The first number between parentheses refers to the size of the parents population (one in this case), the "+" sign refers to the type of replacement (the other possible replacement is the "," replacement) and the last value refers to the number of offspring generated from the parents (also one in this case). There are other types of ESs like the $(\mu + 1)$-ES, $(1 + \lambda)$-ES, $(\mu + \lambda)$-ES and the $(\mu + \lambda)$-ES.

One feature that clearly distinguishes ESs from other EAs like GAs, is that an ES performs a self-adaptive process with the stepsize values (σ_i) of each individual for each dimension "i" of the search space. Figure 1 presents how a solution is encoded in an ES. Both, the decision variables of the problem and the stepsizes for each one of them are stored in one single solution. These stepsizes are subject to recombination and mutation because they are evolving as well.

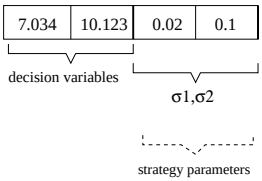

Fig. 1. Encoding of a solution in a typical Evolution Strategy. Decision variables and strategy parameters are both represented in a single solution.

In this work we use a $(\mu + \lambda) - ES$ with panmictic discrete-intermediate recombination (more than two parents are used to generate one offspring) applied to both decision variables and strategy parameters as shown in Figure 2.

Noncorrelated Gaussian mutation is implemented as follows:

$$\sigma_i' = \sigma_i \cdot exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \quad (1)$$

$$x_i' = x_i + \sigma_i' \cdot N_i(0,1) \quad (2)$$

where σ_i is the stepsize of the variable x_i, τ and τ' are interpreted as "learning rates" and are defined by Schwefel [5] as: $\tau = (\sqrt{2\sqrt{n}})^{-1}$ and $\tau' = (\sqrt{2n})^{-1}$, where n is the number of decision variables. $N_i(x, y)$ is a function that returns a real normal-distributed random number with mean x and standard deviation y. The index i indicates that this random number is generated anew for each decision variable or strategy parameter.

```
recombination
    Select randomly Parent_1 from the parent population
    FOR i=1 to n DO
        Select randomly Parent_2 from the parent population
        IF flip(0.5) THEN
            IF flip(0.5) THEN
                child_i = Parent_1_i
            ELSE
                child_i = Parent_2_i
            END IF
        ELSE
                child_i = Parent_1_i + ((Parent_2_i − Parent_1_i/2, 0)
        END IF
    END FOR
END
```

Fig. 2. Pseudocode of the recombination operator used in our approach. Parent 1 is fixed during all the process, but parent 2 is chosen anew for each decision variable. flip(P) is a function that returns TRUE with probability P.

The constraint-handling mechanism chosen is based on Deb's feasibility rules [12] used to rank solutions as to choose those that will remain in the population for the next generation. Those rules are: (1) Between 2 feasible solutions, the one with the highest fitness value wins, (2) if one solution is feasible and the other one is infeasible, the feasible solution wins and (3) if both solutions are infeasible, the one with the lowest sum of constraint violation is preferred ($\sum_{i=1}^{m} max(0, g_i(\mathbf{x}))$). The fitness inheritance mechanism is proposed as follows: In the recombination operator used and detailed in Figure 2, "$n + 1$" parents are used, where "n" is the number of decision variables. The inherited values for the objective function and the constraints for the only offspring generated in each recombination and mutation process will be calculated from this subset of parents taken from the whole population.

The Manhattan distance in the decision space is calculated between the offspring and each one of its parents by using the following expression: $\sum_{i=1}^{n} |X_{pi} - X_{hi}|$, where X_p is the parent solution, X_h is the offspring and n is the number of decision variables of the problem. The offspring will take all its values (objective function and constraints) from the closest parent in the decision space.

It is important to note that, in this approach, the set of solutions to be considered to inherit their values to the offspring is adapted depending of the dimensionality of the problem. Moreover, there is no need to store a high number of solutions because the same parents are only considered to inherit their values to the offspring.

In order to this fitness inheritance approach to perform well, two parameters were considered:

- $0 \leq IR \leq 1$: Inheritance ratio. The percentage of the set of λ offspring that will use the fitness inheritance mechanism. The remaining $1 - IR$ solutions will be evaluated in the real model of the problem.

- $0 \leq RR \leq 1$: Replacement ratio. The percentage of solutions with inherited values that will survive for the next generation.

These parameters allow the user to control the error caused by the solutions with inherited values. If several solutions with approximated values are in the population, the search may be guided by non-exact information more frequently.

In the first and last generation all solutions are evaluated with the original model (i.e. $IR = 0$) as to start from exact information and also to report the best solution found so far.

The replacement process (i.e. to select the μ solutions from the $\mu + \lambda$ which will survive for the next generation) of the ES with the FI mechanism is designed in such a way that only a percentage of solutions with inherited values will be in the next generation. Then, the process looks to decrease the error generated by the solutions with non-exact values.

The complete pseudocode of our proposed Evolution Strategy with Fitness Inheritance is detailed in Figure 3. The initial population is generated with random values for each decision variables between its lower and upper bounds. All stepsize values are initialized as follows: $\sigma_i(0) = \left(\frac{x_i^u - x_i^l}{\sqrt{n}}\right)$ where $x_i^u - x_i^l$ are the upper and lower bounds of the decision variable $i, i = 1, \ldots, n$.

```
Generate an initial population of size μ
Evaluate each solution in the population with the original model
   FOR G = 1 TO Max_Generations DO
      FOR i=1 TO λ DO
      Generate one offspring by using recombination and mutation
      (Figure 2 and Equations 1 and 2)
→        If flip(IR) AND G < Max_Generations THEN
→           The offspring inherits its objective function and constraints values
→           from its closest parent
         ELSE
            The offspring is evaluated in the original model
         END IF
      END FOR
      Split the μ + λ solutions in two groups (solutions with inherited values
      and solutions evaluated in the original model) and rank each group
      based on Deb's feasibility rules.
      FOR i=1 TO μ DO
→        IF flip(RR) THEN
→           Select to survive the best individual from the group of solutions
→           with inherited values.
→           Delete this solution from its group.
         ELSE
            Select to survive the best individual from the group of solutions
            evaluated with the original model.
            Delete this solution from its group.
         END IF
      END FOR
   END FOR
```

Fig. 3. Pseudocode of the $(\mu+\lambda)$-ES with fitness inheritance. *flip(p) returns* 1 *with probability p*. Steps marked with → are those where the fitness inheritance approach is included.

4 Experiments and results

In the experimental design, we used 13 well-known benchmark problems found in the specialized literature [13]. A summary of the main features per test problem is presented in Table 1 and the complete expressions are included in an Appendix at the end of this document. In every experiment, 30 independent runs were performed.

Two experiments were executed: (1) To compare the ES with the fitness inheritance mechanism considering that this version will perform less evaluations with respect to the original ES without using fitness inheritance and (2) to compare the ES with the fitness inheritance mechanism but now adjusting it to perform the same number of evaluations that the original ES without using fitness inheritance. The goal of the first experiment is to analyze the capabilities of the FI approach to decrease the number of evaluations without affecting the performance of the original approach. The second experiment aims to analyze the behavior of the FI mechanism in similar conditions with respect to the original version of the algorithm.

The following nomenclature was used in the results reported: **IR-RR-FIES**. Where **IR** is the inheritance ratio, **RR** is the survival ratio. Finally, **FIES** means: Fitness Inheritance Evolution Strategy.

For the first version of the ES without fitness inheritance the parameters used were as follows: $(100 + 300)$-ES with **0-0-FIES**, i.e. $IR = 0\%$ and $RR = 0\%$, Max_Generations $= 800$ (240,000 evaluations). For the versions of the ES with fitness inheritance the following parameters were used: (100+300)-ES with **30-50-FIES** i.e. $IR = 30\%$ and $RR = 50\%$, Max_Generations $= 800$ (167,000 evaluations). For the second version of the ES without fitness inheritance the parameters used were as follows: $(100+200)$-ES with **0-0-FIES**, i.e. $IR = 0\%$ and $RR = 0\%$, Max_Generations $= 850$ (170,000 evaluations). The statistical results obtained from the set of 30 independent runs performed per ES version per problem are presented in Table 2.

Regarding the comparison of the first experiment **0-0-FIES** (240,000 evaluations) and **30-50-FIES** (167,000 evaluations) we observed the following: In functions g01, g04, g06, g08 and g12 both approaches reached the best known solution consistently. On the other hand, in functions g02, g03, g05, g07, g10 and g13 **0-0-FIES** obtained an overall better performance than **30-50-FIES**. However, the differences in the obtained results are not as high as expected. These results suggest that, there is a small performance decrease when the fitness inheritance is applied (as expected because several solutions have an inexact value to guide the search). However, considerable savings in the number of evaluations (about 30%) are also achieved.

For the second experiment (both compared approaches performing the same number of evaluations $\approx 170,000$) the following was observed: In functions g03, g07, g10 and g13, **30-50-FIES** obtained a "better" best result with respect to **0-0-FIES**. With respect to the worst value found, **30-50-FIES** obtained "better" results in problems g03, g05, g07 and g11, and **0-0-FIES** was

Table 1. Main features for each benchmark problem used in the experiments. ρ is the estimated size of the feasible region with respect to the whole search space [13], n is the number of decision variables, LI is the number of linear inequality constraints , NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints.

Problem	n	Type of function	ρ	LI	NI	LE	NE
g01	13	quadratic	0.0003%	9	0	0	0
g02	20	nonlinear	99.9973%	0	2	0	0
g03	10	nonlinear	0.0026%	0	0	0	1
g04	5	quadratic	27.0079%	0	6	0	0
g05	4	nonlinear	0.0000%	2	0	0	3
g06	2	nonlinear	0.0057%	0	2	0	0
g07	10	quadratic	0.0000%	3	5	0	0
g08	2	nonlinear	0.8581%	0	2	0	0
g09	7	nonlinear	0.5199%	0	4	0	0
g10	8	linear	0.0020%	3	3	0	0
g11	2	quadratic	0.0973%	0	0	0	1
g12	3	quadratic	4.7697%	0	1	0	0
g13	5	nonlinear	0.0000%	0	0	0	3

better in problem g10, and both approaches provided a "similar" worst result in problem g13. Finally, regarding the mean and standard deviation values, **30-50-FIES** provided better results in problems g02, g07 and g11. The results in this second experiment suggest that the fitness inheritance mechanism, which indeed introduces some error in the values which guide the search, is able to promote the exploration of other regions of the search space as to obtain either "better" results or a more consistent behavior to reach the vicinity of the best known solution. This behavior was observed in problems with a very small feasible region (g03, g05, g07, g10, g11 and g13) where some of them have nonlinear equality constraints. In this type of problems it is very common that the search is strongly biased by the first feasible solution found and premature convergence inside the feasible region may occur. The incorporation of solutions which may be infeasible, but based on its closeness to a feasible solution will be considered feasible (due to the inheritance process) seems to allow the evolutionary search to explore in a different way the search space and to approach the feasible region from different regions. These results are far from being conclusive and more detailed experiments are necessary to validate the aforementioned discussion. Nonetheless, the results obtained show that fitness inheritance is a valid option to be considered in this type of ES in order to save evaluations without considerably affecting the good performance of the approach.

5 Conclusions and future work

An approach to incorporate a fitness inheritance mechanism to an Evolution Strategy to solve the general NLP problem was proposed. The approach is based on a panmictic recombination operator, where the closest one from the set of parents (in the decision space) is chosen to inherit all their values to the offspring. Two experiments were designed to evaluate (1) the capabilities

Table 2. Statistical results obtained with the three ES versions: one with fitness inheritance **30-50-FIES** (167,000 evaluations) and two without fitness inheritance **(0-0-FIES** (240,000 evaluations) and **0-0-FIES** (170,000 evaluations). A result in **boldface** indicates either a better result or that the best know solution was reached.

Problem & Best known solution	Statistical results			
	Stats	30-50-FIES	0-0-FIES (240,000)	0-0-FIES (170,000)
g01	Best	**-15.000**	**-15.000**	**-15.000**
-15	Mean	**-15.000**	**-15.000**	**-15.000**
	Worst	**-15.000**	**-15.000**	**-15.000**
	St. Dev.	**0**	**0**	**0**
g02	Best	0.803534	**0.803595**	0.803569
0.803619	Mean	**0.792789**	0.787545	0.784593
	Worst	0.744716	**0.755566**	0.754253
	St. Dev.	0.012727	**0.01184**	0.013125
g03	Best	0.95351	**0.98147**	0.914961
1	Mean	0.799909	**0.886853**	0.804792
	Worst	0.643692	**0.738482**	0.605035
	St. Dev.	0.087753	**0.06431**	0.08445
g04	Best	**-30665.539**	**-30665.539**	**-30665.539**
-30665.539	Mean	**-30665.539**	**-30665.539**	**-30665.539**
	Worst	**-30665.539**	**-30665.539**	**-30665.539**
	St. Dev.	**0**	**0**	**0**
g05	Best	5142.870	5126.610	**5122.240**
5126.498	Mean	5177.359	5204.006	**5162.288**
	Worst	**5229.140**	5386.690	5391.810
	St. Dev.	**30.5066**	114.6465	76.25770
g06	Best	**-6961.814**	**-6961.814**	**-6961.814**
-6961.814	Mean	**-6961.814**	**-6961.814**	**-6961.814**
	Worst	**-6961.814**	**-6961.814**	**-6961.814**
	St. Dev.	**0**	**0**	**0**
g07	Best	24.347	**24.328**	24.378
24.306	Mean	**24.458**	24.462	24.484
	Worst	24.707	**24.646**	24.833
	St. Dev.	0.0878	**0.07206**	0.10997
g08	Best	**0.095826**	**0.095826**	**0.095826**
0.095825	Mean	**0.095826**	**0.095826**	**0.095826**
	Worst	**0.095826**	**0.095826**	**0.095826**
	St. Dev.	**0**	**0**	**0**
g09	Best	**680.63**	**680.63**	**680.63**
680.63	Mean	**680.642**	**680.642**	**680.642**
	Worst	680.678	680.678	**680.667**
	St. Dev.	0.00987	0.00987	**0.007427**
g10	Best	7058.1	**7052.22**	7063.72
7049.25	Mean	7273.402	7261.021	**7252.095**
	Worst	7674.44	**7488.69**	7608.37
	St. Dev.	120.559	**102.93005**	128.2823
g11	Best	**0.75**	**0.75**	**0.75**
0.75	Mean	**0.75**	0.7504	0.7544
	Worst	**0.76**	**0.76**	0.79
	St. Dev.	**0.0014**	0.0022	0.01
g12	Best	**1.000**	**1.000**	**1.000**
1	Mean	**1.000**	**1.000**	**1.000**
	Worst	**1.000**	**1.000**	**1.000**
	St. Dev.	**0**	**0**	**0**
g13	Best	0.497647	**0.464606**	0.992215
0.053949	Mean	0.99617813	**0.92083767**	0.99611781
	Worst	0.999926	**0.998043**	0.999926
	St. Dev.	0.2465	0.17461	**0.00227**

of the proposed inheritance approach to save evaluations without affecting the overall performance of the original algorithm and (2) the behavior of the fitness inheritance in similar conditions (number of evaluations) with respect to the original algorithm. The results obtained showed that the inheritance mechanism is able to decrease the number of evaluations required by the original approach (about 30%) without considerably affecting its good performance. Furthermore, it was initially analyzed that the fitness inheritance mechanism is able to promote further exploration of the search space in some problems, most of them with a small feasible region and with nonlinear inequality constraints, as to obtain "better" results. However, this last finding

requires further experimentation and analysis, which in fact is part of the future work, besides a more careful study of the effect of the IR and RR parameters in the behavior of the evolutionary search.

Acknowledgement. The first and third authors gratefully acknowledge support from CONACyT through projects No. 52048-Y and No. 45683-Y respectively.

Appendix

The details of the benchmark functions used are the following:

- **g01**:
  Minimize: $f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$
  subject to:
  $g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$
  $g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$
  $g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$
  $g_4(\mathbf{x}) = -8x_1 + x_{10} \leq 0$
  $g_5(\mathbf{x}) = -8x_2 + x_{11} \leq 0$
  $g_6(\mathbf{x}) = -8x_3 + x_{12} \leq 0$
  $g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10} \leq 0$
  $g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11} \leq 0$
  $g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12} \leq 0$

  where the bounds are $0 \leq x_i \leq 1$ $(i = 1, \ldots, 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ and $0 \leq x_{13} \leq 1$. The global optimum is located at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ where $f(x^*) = -15$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

- **g02**:
  Maximize: $f(\mathbf{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \right|$
  subject to:
  $g_1(\mathbf{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$
  $g_2(\mathbf{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$

  where $n = 20$ and $0 \leq x_i \leq 10$ $(i = 1, \ldots, n)$. The global maximum is unknown; the best reported solution is: $f(x^*) = 0.803619$. Constraint g_1 is close to being active ($g_1 = -10^{-8}$).

- **g03**:
  Maximize: $f(\mathbf{x}) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$
  subject to:
  $h(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$

  where $n = 10$ and $0 \leq x_i \leq 1$ $(i = 1, \ldots, n)$. The global maximum is located at $x_i^* = 1/\sqrt{n}$ $(i = 1, \ldots, n)$ where $f(x^*) = 1$.

- **g04**:
  Minimize: $f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$
  subject to:
  $g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$
  $g_2(\mathbf{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$
  $g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$
  $g_4(\mathbf{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$
  $g_5(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$
  $g_6(\mathbf{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$

  where: $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$. The global optimum is located at $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ where $f(x^*) = -30665.539$. Constraints g_1 and g_6 are active.

- **g05**
  Minimize: $f(\mathbf{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

  subject to:
  $g_1(\mathbf{x}) = -x_4 + x_3 - 0.55 \leq 0$
  $g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \leq 0$
  $h_3(\mathbf{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$
  $h_4(\mathbf{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$
  $h_5(\mathbf{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$, and $-0.55 \leq x_4 \leq 0.55$. The best known solution is $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ where $f(x^*) = 5126.4981$.

- **g06**
  Minimize: $f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$
  subject to:
  $g_1(\mathbf{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$
  $g_2(\mathbf{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$

  where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The global optimum is located at $x^* = (14.095, 0.84296)$ where $f(x^*) = -6961.81388$. Both constraints are active.
- **g07**
  Minimize: $f(\mathbf{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$
  subject to:
  $g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$
  $g_2(\mathbf{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$
  $g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$
  $g_4(\mathbf{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$
  $g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$
  $g_6(\mathbf{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \leq 0$
  $g_7(\mathbf{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$
  $g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$

  where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. The global optimum is located at $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ where $f(x^*) = 24.3062091$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.
- **g08**
  Maximize: $f(\mathbf{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$
  subject to:
  $g_1(\mathbf{x}) = x_1^2 - x_2 + 1 \leq 0$
  $g_2(\mathbf{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

  where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The global optimum is located at $x^* = (1.2279713, 4.2453733)$ where $f(x^*) = 0.095825$.
- **g09**
  Minimize: $f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7$
  subject to:

  $g_1(\mathbf{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$
  $g_2(\mathbf{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$
  $g_3(\mathbf{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$
  $g_4(\mathbf{x}) = 4x_1^2 + x_2^2 - 3x_1 x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$

  where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 7)$. The global optimum is located at $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ where $f(x^*) = 680.6300573$. Two constraints are active (g_1 and g_4).
- **g10**
  Minimize: $f(\mathbf{x}) = x_1 + x_2 + x_3$
  subject to: $g_1(\mathbf{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$
  $g_2(\mathbf{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$
  $g_3(\mathbf{x}) = -1 + 0.01(x_8 - x_5) \leq 0$
  $g_4(\mathbf{x}) = -x_1 x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$
  $g_5(\mathbf{x}) = -x_2 x_7 + 1250x_5 + x_2 x_4 - 1250x_4 \leq 0$
  $g_6(\mathbf{x}) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500x_5 \leq 0$

  where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$, $(i = 2, 3)$, $10 \leq x_i \leq 1000$, $(i = 4, \ldots, 8)$. The global optimum is located at $x^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$, where $f(x^*) = 7049.25$. g_1, g_2 and g_3 are active.
- **g11**
  Minimize: $f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2$
  subject to:
  $h(\mathbf{x}) = x_2 - x_1^2 = 0$

  where: $-1 \leq x_1 \leq 1$, $-1 \leq x_2 \leq 1$. The global optimum is located at $x^* = (\pm 1/\sqrt{2}, 1/2)$ where $f(x^*) = 0.75$.
- **g12**
  Maximize: $f(\mathbf{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$
  subject to:
  $g_1(\mathbf{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

  where $0 \leq x_i \leq 10$ $(i = 1, 2, 3)$ and $p, q, r = 1, 2, \ldots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such the above

inequality (5) holds. The global optimum is located at $x^* = (5, 5, 5)$ where $f(x^*) = 1$.

- **g13**
  Minimize: $f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5}$
  subject to:
  $g_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$
  $g_2(\mathbf{x}) = x_2 x_3 - 5 x_4 x_5 = 0$
  $g_3(\mathbf{x}) = x_1^3 + x_2^3 + 1 = 0$

  where $-2.3 \leq x_i \leq 2.3$ $(i = 1, 2)$ and $-3.2 \leq x_i \leq 3.2$ $(i = 3, 4, 5)$. The global optimum is located at $x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ where $f(x^*) = 0.0539498$.

References

1. Michalewicz, Z. and Fogel, D. B. (2004) *How to Solve It: Modern Heuristics*, 2nd edition. Springer, Berlin, Germany.
2. Jin, Y. (2005) A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, **9(1)**, 3–12.
3. Won, K.-S. and Ray, T. (2004) Performance of kriging and cokriging based Surrogate Models within the Unified Framework for Surrogate Assisted Optimization. *Proceedings of the IEEE Congress on Evolutionary Computation 2004*, Piscataway, New Jersey, June, pp. 1577–1585. IEEE Service Center.
4. Smith, R. E., Dike, B. A., and Stegmann, S. A. (1995) Fitness Inheritance in Genetic Algorithms. *SAC '95: Proceedings of the 1995 ACM Symposium on Applied Computing*, Nashville, Tennessee, USA, pp. 345–350. ACM Press.
5. Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York.
6. Reyes-Sierra, M. and Coello Coello, C. A. (2005) Fitness Inheritance in Multi-Objective Particle Swarm Optimization. *2005 IEEE Swarm Intelligence Symposium (SIS'05)*, Pasadena, California, USA, June, pp. 116–123. IEEE Press.
7. Voutchkov, I. and Keane, A. (2006) Multiobjective Optimization Using Surrogates. In Parmee, I. (ed.), *Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture (ACDM'2006)*, Bristol, UK, April, pp. 167–175. The Institute for People-centred Computation.
8. Runarsson, T. P. (2004) Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. *Proceedings of 8th Parallel Problem Solving From Nature*, September, pp. 401–410. UK, Springer. LNCS Vol. 3242.
9. Mezura-Montes, E. and Coello Coello, C. A. (2005) Saving Evaluations in Differential Evolution for Constrained Optimization. *Sixth Mexican International Conference on Computer Science (ENC'05)*, September, pp. 274–281. IEEE Computer Society Press.
10. Price, K. V., Storn, R. M., and Lampinen, J. A. (2005) *Differential Evolution. A Practical Approach to Global Optimization*. Springer, Berlin.
11. Schwefel, H.-P. (1995) *Evolution and Optimum Seeking*. Wiley, New York.
12. Deb, K. (2000) An Efficient Constraint Handling Method for Genetic Algorithms. *Comp. Methods in Applied Mechanics and Engineering*, **186(2-4)**, 311–338.
13. Michalewicz, Z. and Schoenauer, M. (1996) Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, **4(1)**, 1–32.