A COMPARISON OF PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION

Vu Truong Vu

Ho Chi Minh City University of Transport, Faculty of Civil Engineering
No.2, D3 Street, Ward 25, Binh Thanh District, HCMC, Viet Nam
vutruongvu@gmail.com

ABSTRACT

Two modern optimization methods including Particle Swarm Optimization and Differential Evolution are compared on twelve constrained nonlinear test functions. Generally, the results show that Differential Evolution is better than Particle Swarm Optimization in terms of high-quality solutions, running time and robustness.

KEYWORDS

Particle Swarm Optimization, Differential Evolution

1. INTRODUCTION

As early as 1975, Holland [1] published the primary book of Genetic Algorithms, one of the most attractive modern optimization methods so far. Other modern methods have been introduced since then. Some well-known methods are listed chronologically here. Kirkpatrick *et al.* [2] with Simulated Annealing in 1983, Glover [3] with Tabu Search in 1986, Dorigo *et al.* [4] with Ant System in 1991 and later version Ant Colony Optimization [5] in 1999, Kennedy and Eberhart [6, 7] with Particle Swarm Optimization in 1995, Storn and Price [8] with Differential Evolution in 1995. It should be noted that all modern optimization techniques in this study belong to stochastic class of optimization and have some common characteristics such as being based on nature, using probabilistic rules, and only using objective function information. The methods have proven their usefulness in real problems in which the objective functions and constraints are not continuous and non-differentiable.

The "No Free Lunch" theorem [9] states that no algorithm is perfect or in another words, each method has its own advantages and disadvantages. As a result, each method is only effective in certain problems. However this does not mean that all comparisons are unnecessary. At most, a comparison provides insight into the algorithms. At least, it shows which type of problem is suitable for which algorithm. From this light, one can select the most suitable each algorithm for a particular situation. Two methods Swarm Optimization and Differential Evolution which are originally applied to continuous variables will be compared in this study.

2. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO), proposed by Kennedy and Eberhart in 1995 [6, 7, 10], simulates the behaviour of bird flocks searching for targets. During searching, each individual

DOI: 10.5121/ijsc.2012.3302

continuously adjusts its position/movement according to its own experience (its best position so far) and its neighbourhood experience (the best position obtained by its neighbourhood). This simulation was introduced to optimization of nonlinear, multi-dimensional functions with the similarity given in Table 1.

In the original version of PSO [6, 7], each particle position and velocity are updated in the form:

$$v_{ij}^{k+1} = v_{ij}^{k} + cr_{1j}^{k}(p_{ij}^{k} - x_{ij}^{k}) + cr_{2j}^{k}(p_{gj}^{k} - x_{ij}^{k}) \quad (1)$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \quad (2)$$

where i = 1, 2, . . ., N, and N is the size of the swarm
j = 1, 2, . . ., D, and D is the number of dimensions of the problem
k = 1, 2, . . ., Iter, and Iter is the maximum iteration number
x_{ij}^{k} is the position of particle i, dimension j, at iteration k. The position is also parameter of the problem
v_{ij}^{k} is the velocity (position change in a unit time step) of particle i, dimension j, at iteration k
c is an acceleration constant, c = 2
r_1, r_2 are independent random numbers, uniformly distributed in (0, 1)
p_{ij}^{k} is the personal best position of particle i, dimension j, at iteration k
p_{gj}^{k} is the neighbourhood's best position, dimension j, at iteration k

There are two models of PSO, one with a global neighbourhood and another with a local neighbourhood. In the global model, a particle has information of its own and of the entire population, whereas in the local model, a particle has information of its own and its nearest neighbours. The global model, which is faster than the local model [11], is described in the algorithm for a minimization problem as shown in Fig. 1. Based on the basic Eq. (1), many variants of PSO are developed. Each part in the right-hand side of Eq. (1) has its own meaning.

- The first term, v_{ij}, is a momentum. An attempt to remove this value from the algorithm leads to ineffectiveness at finding global optima, as Kennedy *et al.* [7] pointed out.

- The second term is the cognitive component that relates to individual experience. This component represents the distance from current particle position to its best position so far. The cognitive component represents the natural tendency of individuals to return to environments where they experienced their best performance [12].

- The third term is the social component that shows the collaboration among particles. It represents the distance from a current particle position to the best position of the whole swarm (in global model) or its neighbourhood (in local model). This component represents the tendency of individuals to follow the success of other individuals [12].

Two versions of PSO will be studied in the paper. They are Inertia weight variant introduced by Shi *et al.* [13, 14], and Constriction factor variant proposed by Clerc [15], Clerc *et al.* [16].

Table 1 The similarity between bird flock's properties and optimization problem in Particle Swarm Optimization

Bird flock's properties	**Optimization problem**
Particle (or Individual)	Variable
Swarm (or Population)	Set of variables
Neighbourhood	Subset/Set of variables
3 dimensions	D dimensions
Particle position	Variable value
Particle velocity	Variable increment
Search target	Objective function

2.1. Inertia weight variant (PSO-1)

Shi *et al.* [13] introduced a modified PSO in which the Eq. (1) is changed into

$$v_{ij}^{k+1} = \omega v_{ij}^{k} + c_1 r_{1j}^{k}(p_{ij}^{k} - x_{ij}^{k}) + c_2 r_{2j}^{k}(p_{gj}^{k} - x_{ij}^{k}) \quad (3)$$

where, ω is the inertia weight; c_1, c_2 are two positive constants, called cognitive and social parameters respectively.

The velocity in Eq. (3) is limited by a maximum velocity, v_{max}. This limit reduces the excessive step size in Eq. (2). The maximum velocity is computed as

$$v_{max} = c_v(x_{max} - x_{min}) \quad (4)$$

where c_v is a maximum velocity coefficient, x_{max} and x_{min} are the maximum and minimum positions of a particle.

The recommended choice in [7] for constants c_1 and c_2 is 2 since on average it makes the weights for cognitive and social parts to be 1.

Coefficient ω is used to contain the explosion as particles velocities and positions move toward a boundary. While a large inertia weight ω favours a global search, a small value supports a local search. A suitable selection of the inertia weight can provide a balance between global and local exploration abilities [14]. Playing such a role, naturally its value should be large in early phase of search in order to reduce omitting potential targets. Once a search area is found, small value of ω may be suitable for refining the search. As proposed in [14], ω decreases linearly from 0.9 to 0.4 combined with the limit of the velocity v_{max}.

2.2. Constriction factor variant (PSO-2)

Instead of using the inertia weight and velocity clamping, Clerc [15], Clerc *et al.* [16] introduced a constriction factor χ. In this case, the Eq. (1) becomes

$$v_{ij}^{k+1} = \chi(v_{ij}^{k} + c_1 r_{1j}^{k}(p_{ij}^{k} - x_{ij}^{k}) + c_2 r_{2j}^{k}(p_{gj}^{k} - x_{ij}^{k})) \quad (5)$$

where $\chi = \dfrac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$, $\varphi = c_1 + c_2$, and $\varphi > 4$

This method can prevent explosion and more importantly induces particles to converge on a steady state. Unlike the inertia weight variant, this method does not need a velocity limit [16], although the use of it may improve performance, for example, in [17] and [18]. Comparing inertia weights and constriction factors can be seen in [17] where the conclusion is in favour of the constriction factor variant.

2.3. Suggestions on choice of parameters

Some authors investigated the optimal choice of parameters. Eberhart *et al.* [19] showed that population size N is problem-dependent. Sizes of 20-50 are most common. This size is usually smaller than that of other evolutionary algorithms. Carlisle *et al.* [18] showed that for general problems, the optimal value of φ is 4.1 along with $c_1 = 2.8$, $c_2 = 1.3$, population N = 30 and maximum velocity v_{max} is set to zero at maximum position x_{max}.

For each particle x_i, i = 1, N
    Initialize random positions and velocities on D dimensions.
    Evaluate objective function, $f(x_i)$
    Assign $p_i = x_i$ and $f(p_i) = f(x_i)$
    If $f(x_i) < f(p_g)$ then
        $f(p_g) = f(x_i)$
        $p_g = x_i$
    End If
End For
Repeat
    For each particle x_i, i = 1, N
        Update particle velocity and position using
            $v_{ij}^{k+1} = v_{ij}^{k} + cr_{1j}^{k}(p_{ij}^{k} - x_{ij}^{k}) + cr_{2j}^{k}(p_{gj}^{k} - x_{ij}^{k})$, j = 1, D
            $x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1}$
        Evaluate objective function, $f(x_i)$
        Compare $f(x_i)$ with particle's previous best value, $f(p_i)$
        If $f(x_i) < f(p_i)$ then
            $f(p_i) = f(x_i)$
            $p_i = x_i$
        End If
        Compare $f(x_i)$ with neighbourhood's previous best, $f(p_g)$
        If $f(x_i) < f(p_g)$ then
            $f(p_g) = f(x_i)$
            $p_g = x_i$
        End If
    End For
Until maximum iteration or minimum criterion is met.

Fig. 1 Particle Swarm Optimization algorithm for minimization problem.

3. DIFFERENTIAL EVOLUTION

Differential Evolution (DE), invented by Storn and Price [8, 20-22], is based on the idea of employing information within the vector population to alter the search space. This operation is called "mutation". DE also borrows the idea from Evolution Strategy such as "population", "crossover" in which components are mixed, and "selection" in which the best is reserved for the next generation.

Let us consider a population of D-dimensional vectors $x_{i,G}$ i = 1, 2,…, N as a population for each generation G, $v_{i,G+1}$ as a mutant vector in generation (G + 1) and $u_{i,G+1}$ as a trial vector in generation (G + 1). Three operators in DE are described as follows.

- Mutation:

$$v_{i,G+1} = x_{r1,G} + FM(x_{r2,G} - x_{r3,G}) \quad i = 1, 2,\ldots, N \tag{6}$$

where r_1, r_2, r_3 are random numbers in [1, N], integer, mutually different, and different from the running index i; FM is mutation constant in [0, 2]], i.e. $0 \le FM \le 2$.

- Crossover:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1},\ldots, u_{Di,G+1}) \tag{7}$$

where $$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (r \le CR) \text{ or } j = k \\ x_{ji,G} & \text{if } (r > CR) \text{ and } j \ne k \end{cases}, \quad j = 1, 2, \ldots D$$

CR is the crossover constant in [0, 1], r is random number in (0, 1)), i.e. $0 < r < 1$, k is a random integer number in [1, D] which ensures that $u_{i,G+1}$ gets at least one component from $v_{i,G+1}$.

- Selection:

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \tag{8}$$

where $f(x_{i,G+1})$ is the objective function.

For this variant of DE, the notation *DE/rand/1/bin* (DE-1, for brevity) is introduced where *rand* denotes a randomly chosen population vector for mutation, *1* is the number of difference vectors used, and *bin* is the crossover scheme due to independent binomial experiments. Algorithm of this variant for a minimization problem is shown in Fig. 2.

Another variant proved to be useful is *DE/best/2/bin* (DE-2, for brevity), where *best* means the vector of lowest objective function to be mutated, *2* is the number of difference vectors used [22]. In this variant, Eq. (6) becomes

$$v_{i,G+1} = x_{best,G} + FM(x_{r1,G} + x_{r2,G} - x_{r3,G} - x_{r4,G}) \tag{9}$$

According to [23], variant DE-2 seems to be better than DE-1 because DE-2 can generate more different trial vectors than DE-1. For example, with a population of six individuals, 360 different

trial vectors can be generated by DE-2 but only 30 for DE-1. On the other hand, using the *best* vector in DE-2 makes the population tend to the local minimum.

3.1. Suggestion on choice of parameters

There are three parameters in DE with the range suggested as follows.

- Population size N: N is at least 4 in DE-1 or at least 5 in DE-2 to make sure there are sufficient materials for mutation. It ranges between 5*D and 10*D in [20], or between 1.5*D and 2*D in [24], or from 3*D to 8*D in [23], from 1.5*D to 3.8*D in optimization of four nonlinear constrained functions [25].

- Mutation constant FM: It is normally between 0.4 and 1, with 0.5 as a good initial choice. If premature convergence occurs, N and/or FM should be increased [20]. Larger FM increases the probability of escaping a local minimum but for FM > 1 the convergence decreases. A good initial choice is 0.6 [23]. Choice of 0.9 in optimization of four nonlinear constrained functions [25] is reasonable.

- Crossover constant CR: The larger the value of CR, the faster and riskier the convergence. A first good choice is 0.9 or 1 to quickly see the possible solution [20], a choice of 0.1 will slow down convergence but be more thorough. According to [23], CR varies form 0.3 to 0.9. A suitable value of 0.9 in optimization of four nonlinear constrained functions is given in [25].

4. CONSTRAINT HANDLING TECHNIQUES

The two optimization methods mentioned above were originally developed for unconstrained optimization problems. In order to apply them to constrained optimization problems, a typical form of most real-world problems, constraint-handling techniques are needed. Comprehensive overviews of the most popular constraint-handling techniques currently used with heuristic methods can be found in Michalewicz and Fogel [26], Coello [27]. In the paper, a method that makes a clear distinction between feasible solutions and infeasible solutions will be adopted.

In this method, the comparison of two solutions follows the rule suggested by Jimenez *et al.* [28]: (i) for two feasible solutions, the one with better objective function value is chosen, (ii) for one feasible solution and one infeasible solution, the feasible solution is chosen, and (iii) for two infeasible solutions, the one with smaller constraint violation is chosen. This method requires no additional coefficient and gives a natural approach to the feasible zone from trials in the infeasible zone. Pulido *et al.* [29], Zavala *et al.* [30] applied the similar rule to PSO. Lampinen [25] and Montes *et al.* [31] also applied this rule to DE.

```
For i = 1, N
        For j = 1, D
                Initialize random values x_ji,G1
        End For
        Evaluate objective function, f(x_i,G1)
End For
Repeat
        For i = 1, N
                Generate 3 integer random numbers r1 ≠ r2 ≠ r3 ≠ i∈ [1,
N]
                Generate integer random number k∈ [1, D]
                For j = 1, D
                        Generate random number r∈ (0,1)
                        If ((r ≤ CR) or (j = D)) then
                                u_ki,G2 = x_jr1,G1 + FM(x_jr2,G1 − x_jr3,G1)
                        Else
                                u_ki,G2 = x_ji,G1
                        End If
                        k = (k + 1) modulo D
                End For
                Evaluate objective function, f(u_i,G2)
                If f(u_i,G2) < f(x_i,G1) then
                        x_ji,G2 = u_ji,G1
                Else
                        x_ji,G2 = x_ji,G1
                End If
        End For
        For i = 1, N
                For j = 1, D
                        Update x_ji,G1 = x_ji,G2
                End For
        End For
Until maximum iteration or minimum criterion is met.
```

Fig. 2 Differential Evolution algorithm (DE-1) for minimization problem.

5. SETTING NUMERICAL EXPERIMENTS

5.1. The experimentation goals and performance measures

The experimentation goals should be considered when assessing the effectiveness of algorithms and comparing them. Barr [32] suggested the following criteria:

- Accurate: identifying higher-quality solutions than other approaches.
- Fast: producing high-quality solutions quicker than other approaches.
- Robust: less sensitive to differences in problem characteristics and choices of the initial variables than other approaches.
- Simple: easy to implement (less number of code lines and parameters).

5.2. Test functions

A constrained optimization problem has the following type:

$$\begin{aligned} &\text{Minimise} && f(x) \qquad x \in R^n \\ &\text{Subject to} && g_j(x) \geq 0,\ j = 1, \ldots, q \\ & && h_k(x) = 0,\ k = 1, \ldots, p \\ & && x_i^l \leq x_i \leq x_i^u,\ i = 1, \ldots, n \end{aligned} \tag{10}$$

where x is a vector of size n, f(x) is the objective function, $g_j(x)$ is the j^{th} inequality constraint, $h_k(x)$ is the k^{th} equality constraints, and $[x_i^l, x_i^u]$ are the lower and upper bounds of the variable x_i.

Twelve test functions have been taken randomly from Hock and Schittkowski [33], Himmelblau [34]. Information on them is shown in Table 2 and details of functions can be seen in the appendix. Test functions are grouped in four objective function types: linear, quadratic, polynomial and general type. Each type in turn has three constraint types: quadratic, polynomial and general type. The published optimal or best-known solutions, x_{min}, of all test examples were found with constraint violations which are below a tolerance, ε, of 10^{-7}. For compatibility with published solutions, the tolerance will be the same, i.e., $\varepsilon = 10^{-7}$.

Table 2 Information of test functions

Function	Objective function type	Constraint type	Inequality constraints	Equality constraints	No. of variables	Best-known solution, f_{min}
f1	Linear	Quadratic	6	0	8	7049.330923
f2	Linear	Polynomial	2	0	4	727.67937
f3	Linear	General	2	0	3	0.5181632741
f4	Quadratic	Quadratic	6	0	5	-30665.53867
f5	Quadratic	Polynomial	1	0	2	1.0
f6	Quadratic	General	8	3	10	-1768.80696
f7	Polynomial	Quadratic	2	0	2	-6961.81381
f8	Polynomial	Polynomial	1	1	4	17.0140173
f9	Polynomial	General	2	3	4	5126.4981
f10	General	Quadratic	3	0	2	-7.804226324
f11	General	Polynomial	0	3	5	0.0539498478
f12	General	General	0	3	10	-47.76109026

5.3. Setting parameters of methods

Parameters are based on suggestions of each method as mentioned above. Combinations of parameters are also tried to find the best one. Each function will be tested with 50 independent runs. The number of trials for each run is fixed, $I_{max} = 3x10^6$. Parameters for variants of PSO and DE are used as follows.

Population, N	30
Maximum iteration, Iter 10^5	(→ No. of maximum trial I_{max} = N x Iter = $3x10^6$)
For variant PSO-1	$c_1 = 2$, $c_2 = 2$, ω decreases linearly from 0.9 to 0.4, $c_v = 0.5$
For variant PSO-2	There are 9 pairs of (c_1, c_2) which satisfy the relation $c_1+c_2 = 4.1$ to be used for determination of the best combination. They are (c_1, c_2) = (0.3, 3.8), (1.0, 3.1), (1.3, 2.8), (1.7, 2.4), (2.05, 2.05), (2.4, 1.7), (2.8, 1.3), (3.1, 1.0), (3.8, 0.3).

For variants DE-1, DE-2	There are 6 pairs of (CR, FM) to be used for determination of the best combination, including (CR, FM) = (0.1, 0.5), (0.1, 1.0), (0.5, 0.5), (0.5, 1.0), (1.0, 0.5), (1.0, 1.0)

6. RESULTS

6.1. Quality of solutions

- As shown in Table 3, PSO-1 obtained 2 solutions better than best-known solutions while the corresponding values of PSO-2, DE-1, DE-2 are 3, 5 and 7 respectively. Generally, the solution quality decreases in the order: DE-2, DE-1, PSO-2 and PSO-1. Therefore, two variants DE-2 and PSO-2 represented for the two methods will be discussed next.
- For PSO-2, combinations with $c_1 \geq c_2$ give better solutions than those with $c_1 < c_2$. For DE-2, in overall the combination (CR = 1.0, FM = 0.5) is the best.
- For all test functions, the best solutions of DE are always better than or as good as those of PSO. In addition, other best results of DE are approximate to best-known solutions, except the function f6. Reason of discrepancy is still unknown and this makes the use of DE and PSO as global optimization methods with some carefulness. Best solutions of methods are listed in Table 4. Details of result are given in Table 5 and Table 6.

6.2. Running time

Besides the quality of solution which is the most important factor for evaluation of an algorithm, running time only has meaning if the solution quality is good enough. In this light, the results in Table 7 show that running time of DE-2 is faster than that of PSO-2 from 1.1 to 4.4 times depending on problems, except for function f10.

6.3. Robustness

- From Table 7, it is seen that the number of runs which found best solutions of DE-2 are always higher than or equal to that of PSO-2. The rate of feasible trials is the ratio between the number of feasible trials and the total trials. In general, the rate of feasible trials of DE-2 is higher than that of PSO-2. Besides, number of runs in DE-2 which could not find any feasible solution is the least.
- Again, from Table 5 and Table 6, combinations with $c_1 \geq c_2$ give better rate of feasible trials than combinations with $c_1 < c_2$ in PSO-2; and the combination (CR = 1.0, FM = 0.5) is the best for the rate of feasible trials in DE-2.

6.4. Simplicity

DE-2 and PSO-2 have the same 4 parameters. For method complexity, there are about 40 code lines for PSO-2 and 70 code lines for DE-2 in the main loop.

7. CONCLUSION

Generally, DE-2 is better than PSO-2 in terms of solution quality, running time and chance of reaching the best solutions in a variety of problems. In the other hand, number of code lines of PSO-2 is the least. In the author's viewpoint, the best method for the twelve test functions is DE-2. It is also worth noting that this conclusion is based on results of test functions which cover a

variety of types such as linear, quadratic, polynomial and general functions; with given parameters and the constraint handling techniques associated with the algorithms.

Table 3 Minimum values of objective function after 50 testing runs
(**Bold** values mean that the solutions are equal to or better than best-known solutions)

Func.	Minimum value of objective functions				
	PSO-1	**PSO-2**	**DE-1**	**DE-2**	**Best-known**
f1	7469.04471	7053.817430	**7049.246945**	**7049.246506**	7049.330923
f2	727.674508	**727.674485**	**727.674443**	**727.674443**	727.679370
f3	0.5181667914	0.5436562994	**0.5181631954**	**0.5181631954**	0.5181632741
f4	**-30665.53902**	**-30665.53902**	**-30665.53902**	**-30665.53902**	-30665.53867
f5	**0.9907383666**	**0.9907383666**	**0.9907383666**	**0.9907383666**	1.0000
f6	-1221.54117	-1477.62925	-1278.07878	-1690.00300	-1768.80696
f7	-6961.81111	-6961.81111	-6961.81111	-6961.81111	-6961.81381
f8	23.8848784	19.0024993	20.2730530	**17.0140172**	17.0140173
f9	5126.6299	5126.9172	5126.5157	5126.4983	5126.4981
f10	-7.802786736	-7.802786736	-7.802786736	-7.802786736	-7.804226324
f11	0.4561284690	0.0645899015	0.4524708138	**0.0539498394**	0.0539498478
f12	-42.67380277	-47.37065154	-47.67061450	-47.44572377	-47.76109026

Table 4 Best solutions, x_{min}, of two methods

Function	PSO-2	DE-2
f1	(634.3674968, 1250.0134128, 5169.4365273, 186.4438325, 293.2225136, 213.5562164, 293.2213678, 393.2225258)	(579.4606149, 1359.8756712, 5109.9102202, 182.0305765, 295.6035661, 217.9694724, 286.4270592, 395.6035784)
f2	(193.4090071, 179.5210305, 185.0854729, 168.6589740)	(193.4077513, 179.5462924, 185.0160256, 168.7043735)
f3	(0.0, 0.9999998, 2.7182814)	(0.1841265, 1.2021678, 3.3273220)
f4	(78.0, 33.0, 29.9952492, 45.0, 36.7758311)	(78.0, 33.0, 29.9952492, 45.0, 36.7758311)
f5	(1.0046415, 0.0)	(1.0046415, 0.0)
f6	(1746.9672310, 15995.7276777, 63.2719138, 3071.2845436, 2000.0, 87.3578034, 94.0349684, 10.3011250, 2.5096923, 150.6019638)	(1648.4425610, 14406.1129388, 57.2832896, 2935.8537982, 1933.2991566, 90.7273299, 95.0, 9.9120298, 1.5640451, 153.5349454)
f7	(14.0950011, 0.8429632)	(14.0950011, 0.8429632)
f8	(1.0, 3.4347215, 5.0, 1.4841454)	(1.0, 4.7429996, 3.8211499, 1.3794082)
f9	(688.8741362, 1016.5435397, 0.1125169, -0.3992564)	(679.9453373, 1026.0671532, 0.1188763, -0.3962335)
f10	(13.5501432, 51.6599703)	(13.5501432, 51.6599705)
f11	(-1.7130059, 1.5909154, 1.7775005, 0.5445980, 1.0385103)	(-1.7171435, 1.5957097, 1.8272457, 0.7636430, 0.7636430)
f12	(-96.6741714, -98.6387642, -1.5556892, -6.5901656, -0.6945218, -100.0, -99.5173022, -99.9970261, -41.9674309, -15.1797737)	(-3.2354681, -8.0191455, -0.0202026, -82.8016103, -0.6931470, -41.2589767, -92.5399330, -4.7868082, -5.1446518, -17.5464207)

Table 5 Results of PSO-2 with different combinations (c_1, c_2)

Func.	Measures	$c_1 = 0.3$ $c_2 = 3.8$	$c_1 = 1$ $c_2 = 3.1$	$c_1 = 1.3$ $c_2 = 2.8$	$c_1 = 1.7$ $c_2 = 2.4$	$c_1 = 2.05$ $c_2 = 2.05$	$c_1 = 2.4$ $c_2 = 1.7$	$c_1 = 2.8$ $c_2 = 1.3$	$c_1 = 3.1$ $c_2 = 1.0$	$c_1 = 3.8$ $c_2 = 0.3$
1	Minimum	9016.665	7132.133	7916.664	7117.271	7099.304	7098.460	7053.817	7094.391	7082.338
	Feasible rate (%)	25.53	48.62	56.46	57.92	67.12	60.53	63.93	49.67	22.28
	Unsuccessful runs	31	15	13	10	3	2	0	0	0
2	Minimum	727.681	727.686	727.695	727.692	727.677	727.680	727.675	727.674	727.675
	Feasible rate (%)	85.05	73.92	64.84	57.17	52.52	46.11	43.51	41.68	41.54
	Unsuccessful runs	0	0	0	0	0	0	0	0	0
3	Minimum	0.544	0.544	0.544	0.544	0.544	0.544	0.544	0.544	0.544
	Feasible rate (%)	34.24	30.53	39.84	46.86	50.79	52.9	63.16	65.65	75.95
	Unsuccessful runs	30	32	27	23	21	20	14	11	3
4	Minimum	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
	Feasible rate (%)	77.42	74.83	74.54	74.14	75.57	72.26	73.7	71.2	62.78
	Unsuccessful runs	0	0	0	0	0	0	0	0	0
5	Minimum	0.991	0.991	0.991	0.991	0.991	4	0.991	0.991	0.991
	Feasible rate (%)	99.98	99.99	99.99	99.99	99.99	100.00	99.98	99.99	99.97
	Unsuccessful runs	0	0	0	0	0	0	0	0	0
6	Minimum	-1232.67	-996.52	-949.43	-1221.29	-1124.04	-1218.89	-1439.50	-1477.63	-936.28
	Feasible rate (%)	0.25	0.63	1.66	0.88	2.30	2.71	5.50	6.49	0.22
	Unsuccessful runs	47	45	39	41	31	30	17	11	47
7	Minimum	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811
	Feasible rate (%)	7.464	3.03	8.552	16.09	20.64	7.86	25.81	32.52	67.34
	Unsuccessful runs	45	48	45	40	37	45	34	30	5
8	Minimum	21.452	21.614	19.002	21.320	20.987	20.722	21.724	26.277	23.890
	Feasible rate (%)	4.09	3.75	4.07	4.21	4.40	4.54	4.61	4.36	3.31
	Unsuccessful runs	0	0	0	0	0	0	0	0	0
9	Minimum	5131.798	5132.247	5132.699	5127.920	5127.124	5129.886	5130.157	5127.295	5126.917
	Feasible rate (%)	15.77	24.11	38.01	42.73	46.84	49.28	48.55	32.5	9.25
	Unsuccessful runs	40	35	25	22	19	15	13	11	18
10	Minimum	-7.803	-7.803	-7.803	-7.803	-7.803	-7.803	-7.803	-7.803	-7.803
	Feasible rate (%)	94.98	67.64	66.84	69.16	68.74	63.75	64.39	59.44	51.73
	Unsuccessful runs	0	0	0	0	0	0	0	0	0
11	Minimum	0.46	0.12	0.08	0.49	0.12	0.06	0.10	0.31	0.68
	Feasible rate (%)	6.10	11.03	10.59	11.70	14.81	15.79	13.02	13.73	0.61
	Unsuccessful runs	24	7	7	4	0	1	0	0	42
12	Minimum	-47.369	-45.201	-47.369	-45.201	-47.369	-47.371	-47.371	-47.370	-47.371
	Feasible rate (%)	21.62	26.71	23.36	26.84	37.2	39.67	37.07	43.43	19.19
	Unsuccessful runs	35	33	32	27	19	15	9	2	0

Table 6 Results of DE-2 with different combinations (CR, FM)

Function	Measures	CR = 0.1 FM = 0.5	CR = 0.1 FM = 1.0	CR = 0.5 FM = 0.5	CR = 0.5 FM = 1.0	CR = 1.0 FM = 0.5	CR = 1.0 FM = 1.0
1	Minimum	7081.075	7100.780	7049.247	7250.973	7049.247	7347.184
	Feasible rate (%)	19.00	18.54	5.90	14.47	36.03	35.56
	Unsuccessful runs	0	0	0	0	0	11
2	Minimum	727.674	727.675	727.674	727.674	727.674	727.674
	Feasible rate (%)	46.15	45.75	41.20	40.17	24.62	18.70
	Unsuccessful runs	0	0	0	0	0	0
3	Minimum	0.518	0.518	0.518	0.544	0.518	0.544
	Feasible rate (%)	65.38	69.50	73.15	61.74	65.20	27.51
	Unsuccessful runs	3	14	3	19	0	36
4	Minimum	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
	Feasible rate (%)	92.66	94.14	98.28	99.31	93.60	98.84
	Unsuccessful runs	0	0	0	0	0	0
5	Minimum	0.991	1.004	0.991	1.102	0.991	0.991
	Feasible rate (%)	99.99	99.99	99.99	100.00	99.98	99.99
	Unsuccessful runs	0	0	0	0	0	0
6	Minimum	n/a	n/a	-1690.000	n/a	-1677.520	n/a
	Feasible rate (%)	0.00	0.00	0.17	0.00	32.71	0.00
	Unsuccessful runs	50	50	11	50	13	50
7	Minimum	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811	-6961.811
	Feasible rate (%)	74.48	30.16	77.64	21.51	73.30	19.83
	Unsuccessful runs	8	34	5	38	5	39
8	Minimum	n/a	n/a	19.200	17.150	17.010	18.700
	Feasible rate (%)	0.00	0.00	4.48	60.81	52.06	81.75
	Unsuccessful runs	50	50	22	2	0	3
9	Minimum	5126.505	n/a	5126.510	5126.780	5126.500	5126.520
	Feasible rate (%)	0.01	0.00	5.24	52.77	31.76	65.35
	Unsuccessful runs	3	50	0	4	0	12
10	Minimum	-7.803	-7.803	-7.803	-7.803	-7.803	-7.803
	Feasible rate (%)	86.33	91.59	97.77	95.65	99.43	98.53
	Unsuccessful runs	0	0	0	0	0	0
11	Minimum	n/a	n/a	0.440	0.860	0.050	0.390
	Feasible rate (%)	0.00	0.00	2.35	6.10	45.38	12.89
	Unsuccessful runs	50	50	2	40	0	40
12	Minimum	-47.381	-47.427	-47.446	-47.373	-47.380	-47.375
	Feasible rate (%)	1.48	0.01	4.86	1.90	34.89	49.36
	Unsuccessful runs	24	46	0	10	10	18

Table 7 Data after 50 testing runs, each test has $3x10^6$ trials
(**Bold** values mean that the solutions are equal to or better than best-known solutions)

Func.	Running time per test (s)			Number of runs which found best solutions		Rate of feasible trials (%)		Number of runs which could not find any feasible solution	
	PSO-2 (t_1)	DE-2 (t_2)	t_1/t_2	PSO-2	DE-2	PSO-2	DE-2	PSO-2	DE-2
f1	21.439	4.924	4.4	1	**1**	67.115	36.033	0	0
f2	5.292	4.111	1.3	**1**	**48**	85.047	46.150	0	0
f3	14.107	4.392	3.2	47	**12**	75.954	73.155	3	0
f4	17.461	7.453	2.3	**50**	**50**	77.725	99.307	0	0
f5	15.333	5.647	2.7	**2**	**13**	99.997	99.995	0	0
f6	16.312	7.945	2.1	1	1	6.489	32.712	11	11
f7	12.673	8.872	1.4	43	45	67.338	77.638	5	5
f8	8.356	7.289	1.1	1	**4**	4.612	81.748	0	0
f9	15.419	7.067	2.2	1	4	49.283	65.354	11	0
f10	15.587	22.051	0.7	1	1	94.983	99.428	0	0
f11	12.456	11.483	1.1	1	**1**	15.794	45.375	0	0
f12	36.630	18.408	2.0	1	1	43.428	49.358	0	0

REFERENCES

[1] Holland, J. H., (1975), Adaptation in natural and artificial systems, University of Michigan Press Ann Arbor.

[2] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., (1983), "Optimization by Simulated Annealing," Science, Vol. 220, No. 4598, pp. 671-680.

[3] Glover, F., (1986), "Future paths for integer programming and links to artificial intelligence," Computers and Operations Research, Vol. 13, No. 5, pp. 533-549.

[4] Dorigo, M., Maniezzo, V., and Colorni, A., (1991), "Positive feedback as a search strategy," Technical Report 91-016, Dipartimento di Elettronica, Politecnico de Milano, Italy.

[5] Dorigo, M., and Caro, G. D., (1999), "The ant colony optimization meta-heuristic," New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, eds., McGraw-Hill, Maidenhead, UK.

[6] Eberhart, R., and Kennedy, J., (1995), "A new optimizer using particle swarm theory," Proc. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, Piscataway, NJ: IEEE Service Center, pp. 39-43.

[7] Kennedy, J., and Eberhart, R., (1995), "Particle swarm optimization," Proc. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942-1948.

[8] Storn, R., and Price, K., (1995), "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report TR-95-012, International Computer Science Institute, Berkeley.

[9] Wolpert, D. H., and Macready, W. G., (1997), "No free lunch theorems for optimization," IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 67-82.

[10] Kennedy, J., Eberhart, R. C., and Shi., Y., (2001), Swarm intelligence, Morgan Kaufmann Publishers, San Francisco.

[11] Hu, X., and Russell, C. E., (2002), "Solving constrained nonlinear optimization problems with particle swarm optimization," Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002) Orlando, USA.

[12] Van Den Bergh, F., and Engelbrecht, A. P., (2006), "A study of particle swarm optimization particle trajectories," Information Sciences, Vol. 176, No. 8, pp. 937-971.

[13] Shi, Y., and Eberhart, R., (1998), "A modified particle swarm optimizer," Proc. Proceedings of the IEEE Conference on Evolutionary Computation, pp. 69-73.
[14] Shi, Y., and Eberhart, R. C., (1998), "Parameter selection in particle swarm optimization," Proc. Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming,, V. W. Porto, N. Saravanan, D. E. Waagen, and A. E. Eiben, eds., Springer-Verlag, NewYork, pp. 591-600.
[15] Clerc, M., (1999), "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," Proc. Proceedings of the 1999 Congress on Evolutionary Computation, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, eds., pp. 1951-1957.
[16] Clerc, M., and Kennedy, J., (2002), "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, pp. 58-73.
[17] Eberhart, R. C., and Shi, Y., (2000), "Comparing inertia weights and constriction factors in particle swarm optimization," Proc. Proceedings of the IEEE Conference on Evolutionary Computation, San Diego, CA., pp. 84-88.
[18] Carlisle, A., and Dozier, G., (2001), "An off-the-shelf PSO," Proc. Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN, pp. 1-6.
[19] Eberhart, R. C., and Shi, Y., (2001), "Particle swarm optimization: Developments, applications and resources," Proc. Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, pp. 81-86.
[20] Storn, R., and Price, K., (1997), "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," Journal of Global Optimization, Vol. 11, No. 4, pp. 341-359.
[21] Storn, R., and Price, K., (1996), "Minimizing the real functions of the ICEC'96 contest by differential evolution," Proc. Proceedings of the IEEE Conference on Evolutionary Computation, Nagoya, Japan, pp. 842-844.
[22] Price, K. V., (1996), "Differential evolution: a fast and simple numerical optimizer," Proc. Biennial Conference of the North American Fuzzy Information Processing Society - NAFIPS,, M. Smith, M. Lee, J. Keller, and J. Yen, eds., IEEE Press, New York, pp. 524-527.
[23] Gamperle, R., Muller, S. D., and Koumoutsakos, P., (2002), "A parameter study for differential evolution," Proc. Advances in Intelligent Systems, Fuzzy Ststems, Evolutionary Computation,, A. Grmela, and N. E. Mastorakis, eds., WSEAS Press, pp. 293-298.
[24] Mayer, D. G., Kinghorn, B. P., and Archer, A. A., (2005), "Differential evolution - An easy and efficient evolutionary algorithm for model optimisation," Agricultural Systems, Vol. 83, No. 3, pp. 315-328.
[25] Lampinen, J., (2001), "Solving problems subject to multiple nonlinear constraints by the differential evolution," Proc. Proceedings of MENDEL 2001, 7th International Conference on Soft Computing, R. Matousek, and P. Osmera, eds., Brno, Czech Republic, pp. 50-57.
[26] Michalewicz, Z., and Fogel, D. B., (2000), How to solve it : modern heuristics, Springer, Berlin.
[27] Coello Coello, C. A., (2002), "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," Computer Methods in Applied Mechanics and Engineering, Vol. 191, No. 11-12, pp. 1245-1287.
[28] Jimenez, F., Verdegay, J. L., and Gomez-Skarmeta, A. F., (1999), "Evolutionary techniques for constrained multiobjective optimization problems," Proc. Proceedings of the workshop on multi-criterion optimization using evolutionary methods held at genetic and evolutionary computation conference (GECCO-1999), pp. 115-116.
[29] Pulido, G. T., and Coello, C. A. C., (2004), "A constraint-handling mechanism for particle swarm optimization," Proc. Proceedings of the 2004 Congress on Evolutionary Computation, Piscataway, NJ, pp. 1396-1403.
[30] Zavala, A. E. M., Aguirre, A. H., and Villa Diharce, E. R., (2005), "Constrained optimization via Particle Evolutionary Swarm Optimization algorithm (PESO)," Proc. GECCO 2005 - Genetic and Evolutionary Computation Conference, Washington, DC, pp. 209-216.
[31] Mezura-Montes, E., Coello, C. A. C., and Tun-Morales, E. I., (2004), "Simple feasibility rules and differential evolution for constrained optimization," Proc. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), pp. 707-716.
[32] Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G. C., and Stewart, W. R., (1995), "Designing and reporting on computational experiments with heuristic methods," Journal of Heuristics, Vol. 1, No. 1, pp. 9-32.

[33] Hock, W., and Schittkowski, K., (1981), Test Examples for Nonlinear Programming Codes, Lecture notes in economics and mathematical systems, Springer-Verlag New York, Secaucus.
[34] Himmelblau, D. M., (1972), Applied Nonlinear Programming, McGraw-Hill Book Company, New York.

Appendix: 12 test functions

f1
Objective function: $f(x) = x_1 + x_2 + x_3$
Constraints:
$1 - 0.0025(x_4 + x_6) \geq 0$
$1 - 0.0025(x_5 + x_7 - x_4) \geq 0$
$1 - 0.01(x_8 - x_5) \geq 0$
$x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 \geq 0$
$x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geq 0$
$x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geq 0$
$100 \leq x_1 \leq 10000$
$1000 \leq x_i \leq 10000$, i = 2, 3
$10 \leq x_i \leq 1000$, i = 4,…, 8
Solution:
$f(x_{min}) = 7049.330923$
$x_{min} = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$

f2
Objective function: $f(x) = 1 + x_1 + x_2 + x_3 + x_4$
Constraints:
$0.0401 - 4/x_1 - 2.25/x_2 - 1/x_3 - 0.25/x_4 \geq 0$
$0.010085 - 0.16/x_1 - 0.36/x_2 - 0.64/x_3 - 0.64/x_4 \geq 0$
$0.001 \leq x_i \leq 100000(5 - i)$, i = 1,…, 4
Solution:
$f(x_{min}) = 727.67937$
$x_{min} = (193.4071, 179.5475, 185.0186, 168.7062)$

f3
Objective function: $f(x) = 0.2x_3 - 0.8x_1$
Constraints:
$x_2 - \exp(x_1) \geq 0$
$x_3 - \exp(x_2) \geq 0$
$0 \leq x_1 \leq 100$
$0 \leq x_2 \leq 100$
$0 \leq x_3 \leq 10$
Solution:
$f(x_{min}) = 0.5181632741$
$x_{min} = (0.1841264879, 1.202167873, 3.327322322)$

f4
Objective function: $f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$
Constraints:
$92 \geq 85.334407 + 0.0056858x2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \geq 0$
$20 \geq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 90 \geq 0$
$5 \geq 9.300961 + 0.0047026x3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 20 \geq 0$
$78 \leq x_1 \leq 102$
$33 \leq x_2 \leq 45$
$27 \leq x_i \leq 45$, i = 3, 4, 5

Solution:
$f(x_{min}) = -30665.53867$
$x_{min} = (78, 33, 29.99526, 45, 36.77581)$

f5
Objective function: $f(x) = (x_1 - 2)^2 + x_2^2$
Constraints:
$(1 - x_1)^3 - x_2 \geq 0$
$0 \leq x_1 \leq 1000$
$0 \leq x_2 \leq 1000$
Solution:
$f(x_{min}) = 1$
$x_{min} = (1, 0)$

f6
Objective function: $f(x) = 5.04x_1 + 0.035x_2 + 10x_3 + 3.36x_5 - 0.063x_4x_7$
Constraints:
$g1(x) = 35.82 - 0.222x_{10} - 0.9x_9 \geq 0$
$g2(x) = -133 + 3x_7 - 0.99x_{10} \geq 0$
$g3(x) = -g1(x) + x_9(1/0.9 - 0.9) \geq 0$
$g4(x) = -g2(x) + (1/0.99 - 0.99)x_{10} \geq 0$
$g5(x) = 1.12x_1 + 0.13167x_1x_8 - 0.00667x_1x_8^2 - 0.99x_4 \geq 0$
$g6(x) = 57.425 + 1.098x_8 - 0.038x_8^2 + 0.325x_6 - 0.99x_7 \geq 0$
$g7(x) = -g5(x) + (1/0.99 - 0.99)x_4 \geq 0$
$g8(x) = -g6(x) + (1/0.99 - 0.99)x_7 \geq 0$
$g9(x) = 1.22x_4 - x_1 - x_5 = 0$
$g10(x) = 98000x_3/(x_4x_9 + 1000x_3) - x_6 = 0$
$g11(x) = (x_2 + x_5)/x_1 - x_8 = 0$
$0.00001 \leq x_1 \leq 2000$
$0.00001 \leq x_2 \leq 16000$
$0.00001 \leq x_3 \leq 120$
$0.00001 \leq x_4 \leq 5000$
$0.00001 \leq x_5 \leq 2000$
$85 \leq x_6 \leq 93$
$90 \leq x_7 \leq 95$
$3 \leq x_8 \leq 12$
$1.2 \leq x_9 \leq 4$
$145 \leq x_{10} \leq 162$
Solution:
$f(x_{min}) = -1768.80696$
$x_{min} = (1698.096, 15818.73, 54.10228, 3031.226, 2000, 90.11537, 95, 10.49336, 1.561636, 153.53535)$

f7
Objective function: $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$
Constraints:
$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0$
$-(x_2 - 5)^2 - (x_1 - 6)^2 + 82.81 \geq 0$
$13 \leq x_1 \leq 100$
$0 \leq x_2 \leq 100$
Solution:
$f(x_{min}) = -6961.81381$
$x_{min} = (14.095, 0.84296079)$

f8
Objective function: $f(x) = x_1x_4(x_1 + x_2 + x_3) + x_3$
Constraints:

$x_1x_2x_3x_4 - 25 \geq 0$
$x_1^2 + x_2^2 + x_3^2 + x_4^2 - 40 = 0$
$1 \leq x_i \leq 5$, i = 1,…, 4
Solution:
$f(x_{min}) = 17.0140173$
$x_{min} = (1, 4.7429994, 3.8211503, 1.3794082)$

f9
Objective function: $f(x) = 3x_1 + 1E\text{-}6x_1^3 + 2x_2 + 2/3E\text{-}6x_2^3$
Constraints:
$x_4 - x_3 + 0.55 \geq 0$
$x_3 - x_4 + 0.55 \geq 0$
$1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$
$1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$
$1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$
$0 \leq x_i \leq 1200$, i = 1, 2
$-0.55 \leq x_i \leq 0.55$, i = 3, 4
Solution:
$f(x_{min}) = 5126.4981$
$x_{min} = (679.9453, 1026.067, 0.1188764, -0.3962336)$

f10
Objective function: $f(x) = -75.196 + 3.8112x_1 + 0.0020567x_1^3 - 1.0345E\text{-}5x_1^4 + 6.8306x_2 - 0.030234x_1x_2 + 1.28134E\text{-}3x_2x_1^2 + 2.266E\text{-}7x_1^4x_2 - 0.25645x_2^2 + 0.0034604x_2^3 - 1.3514E\text{-}5x_2^4 + 28.106/(x_2 + 1) + 5.2375E\text{-}6x_1^2x_2^2 + 6.3E\text{-}8x_1^3x_2^2 - 7E\text{-}10x_1^3x_2^3 - 3.405E\text{-}4x_1x_2^2 + 1.6638E\text{-}6x_1x_2^3 + 2.8673\exp(0.0005x_1x_2) - 3.5256E\text{-}5x_1^3x_2 - 0.12694x_1^2$
Constraints:
$x_1x_2 - 700 \geq 0$
$x_2 - x_1^2/125 \geq 0$
$(x_2 - 50)^2 - 5(x_1 - 55) \geq 0$
$0 \leq x_1 \leq 75$
$0 \leq x_2 \leq 65$
Solution:
$f(x_{min}) = -7.804226324$
$x_{min} = (13.55010424, 51.66018129)$

f11
Objective function: $f(x) = \exp(x_1x_2x_3x_4x_5)$
Constraints:
$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$
$x_2x_3 - 5x_4x_5 = 0$
$x_1^3 + x_2^3 + 1 = 0$
$-2.3 \leq x_i \leq 2.3$, i = 1, 2
$-3.2 \leq x_i \leq 3.2$, i = 3, 4, 5
Solution:
$f(x_{min}) = 0.0539498478$
$x_{min} = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$

f12
Objective function:

$$f(x) = \sum_{j=1}^{10} \exp(x_j)(c_j + x_j - \ln(\sum_{k=1}^{10} \exp(x_k)))$$

$c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.100$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$
Constraints:
$\exp(x_1) + 2\exp(x_2) + 2\exp(x_3) + \exp(x_6) + \exp(x_{10}) - 2 = 0$

$\exp(x_4) + 2\exp(x_5) + \exp(x_6) + \exp(x_7) - 1 = 0$
$\exp(x_3) + \exp(x_7) + \exp(x_8) + 2\exp(x_9) + \exp(x_{10}) - 1 = 0$
$-100 \le x_i \le 100, i = 1,\ldots, 10$
Solution:
$f(x_{min}) = -47.76109026$
x_{min} = (-3.201212, -1.912060, -0.2444413, -6.537489, -0.7231524, -7.267738, -3.596711, -4.017769, -3.287462, -2.335582)