Fundamenta Informaticae 95 (2009) 1–21
DOI 10.3233/FI-2009-186
IOS Press

Low Discrepancy Initialized Particle Swarm Optimization for Solving Constrained Optimization Problems

Millie Pant, Radha Thangaraj
Department of Paper Technology, IIT Roorkee, Saharanpur, India.

Ajith Abraham*
Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence
P.O. Box 2259 Auburn, Washington 98071-2259, USA
ajith.abraham@ieee.org

Abstract. Population based metaheuristics are commonly used for global optimization problems. These techniques depend largely on the generation of initial population. A good initial population may not only result in a better fitness function value but may also help in faster convergence. Although these techniques have been popular since more than three decades very little research has been done on the initialization of the population. In this paper, we propose a modified Particle Swarm Optimization (PSO) called Improved Constraint Particle Swarm Optimization (ICPSO) algorithm for solving constrained optimization. The proposed ICPSO algorithm is initialized using quasi random Vander Corput sequence and differs from unconstrained PSO algorithm in the phase of updating the position vectors and sorting every generation solutions. The performance of ICPSO algorithm is validated on eighteen constrained benchmark problems. The numerical results show that the proposed algorithm is a quite promising for solving constraint optimization problems.

Keywords: Particle Swarm Optimization, Constrained Optimization Problems, Quasi Random, Vander Corput Sequence

*Address for correspondence: Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, P.O. Box 2259 Auburn, Washington 98071-2259, USA.

1. Introduction

Most of the problems occurring in the field technology and engineering can be formulated as global optimization problems. Mathematical models of such problems is often complex. The objective function may be multimodal having several local and global optima. Many real-world optimization problems are solved subject to sets of constraints. Such problems are called constrained optimization problems (COP). The search space in COPs consists of two kinds of solutions: feasible and infeasible. Feasible points satisfy all the constraints, while infeasible points violate at least one of them. Therefore, the final solution of an optimization problem must satisfy all constraints. A constrained optimization problem may be distinguished as a Linear Programming Problem (LPP) and Nonlinear Programming Problem (NLP). In this paper we have considered NLP problems where either the objective function or the constraints or both are nonlinear in nature. There are many traditional methods in the literature for solving NLP. However, most of the traditional methods require certain auxiliary properties (like convexity, continuity etc.) of the problem and also most of the traditional techniques are suitable for only a particular type of problem (for example Quadratic Programming Problems, Geometric Programming Problems etc). Keeping in view the limitations of traditional techniques researchers have proposed the use of stochastic optimization methods and intelligent algorithms for solving NLP which may be constrained or unconstrained. Based on the research efforts in literature, constraint handling methods have been categorized in a number of classes [2, 3, 4, 5].

In the past few decades researchers have shown significant interest in population based stochastic search techniques for dealing with global optimization problems. Some popular population based metaheuristics include Genetic Algorithms [6, 7, 8], Ant Colony Optimization [9], Particle Swarm Optimization [10], Differential Evolution [11] etcetera. The first step in all these algorithms is "*generate an initial population*"; but how the population is to be generated finds little or no mention.

The initial generation of random numbers plays an important role in the population based search techniques for optimization. The uniform distribution of points in the search domain is very likely to improve the performance of a search technique that depends largely on the generation of random number numbers. Now the question arises which method should be used to generate random numbers which will result in faster convergence of the algorithm without compromising with the quality of solution. The most commonly used method for generating the initial population is the use of an inbuilt subroutine available in most of the programming languages for generating random numbers. Though this method is very simple from the programming point of view, it is not very efficient because the computer generated numbers do not cover the search space uniformly. Despite the relevance of distribution of initial population this topic has not gained much attention in the EA community.

In the present study we have made use of quasi random Vander Corput sequence to initialize the population. Some previous instances where low discrepancy sequences have been used to improve the performance of optimization algorithms include [12, 13, 14, 15, 16]. Kimura and Matsumura [12] have used Halton sequence for initializing the Genetic Algorithms (GA) population and have shown that a real coded GA performs much better when initialized with a quasi random sequence in comparison to a GA which initialized with a population having uniform probability distribution. Instances where quasi random sequences have been used for initializing the swarm in PSO can be found in [13, 14, 15, 16]. In [14, 15, 16] authors have made use of Sobol and Faure sequences. Similarly, Nguyen et al. [13] have shown a detailed comparison of Halton Faure and Sobol sequences for initializing the swarm. In the pre-

vious studies, it has already been shown that the performance of Sobol sequence dominates the performance of Halton and Faure sequences.

The authors suggested the use of quasi random sequences for the initial generation of population in PSO and recoded its performance for unconstrained optimization. In [17], the authors used Vander Corput sequence and Sobol sequence for the initial generation of random numbers in the basic PSO and compared its performance with the PSO in which computer generated random numbers were used for initial population. The numerical results showed that the proposed quasi random sequences significantly improve the performance of basic PSO. Encouraged by the versions made for solving unconstrained optimization problems in [17], in this paper the authors propose the use of quasi random Vander Corput sequence for solving constrained optimization problems.

This paper presents an Improved Constraint Particle Swarm Optimization (ICPSO) algorithm for solving constrained optimization problems. Its initial population is generated using Vander Corput sequence and its constraint solving approach is similar to the approach used by K. Zielinski et al. [18] where a personal or neighborhood best solution $\bar{g}$ is substituted by a new solution $\bar{x}$ if:

- Both vectors are feasible, but $\bar{x}$ yield the smaller objective function value.
- $\bar{x}$ is feasible and $\bar{g}$ is not.
- Both vectors are infeasible, but $\bar{x}$ results in the lower sum of constraint violations.

The proposed ICPSO is different from the algorithm proposed in [18], as the swarm is initialized by Low discrepancy Vander Corput Sequence and above rules are applied during the updating of position vectors. In the present study we have concentrated our work on PSO which is relatively a new member to a class of population based search technique. To the best of our knowledge, no results are available on the performance of low discrepancy sequence for solving constrained optimization problems. Moreover, the proposed concept may be applied to any of the population based search technique for solving constrained optimization problems.

The structure of the paper is as follows: in Section 2, we give a brief definition of low discrepancy sequences and Vander Corput sequence. In Section 3, we explain the Particle Swarm Optimization Algorithm, in Section 4, the proposed ICPSO algorithm is given. Section 5 deals with experimental settings and test problems, Section 6 gives the numerical results and discussion and finally the paper conclude with Section 7.

2. Quasi Random Vander Corput Sequence

2.1. Low discrepancy or Quasi Random Sequences

The most common practice of generating random numbers is the one using an inbuilt subroutine (available in most of the programming languages), which uses a uniform probability distribution to generate random numbers. This method is not very proficient as it has been shown that uniform pseudorandom number sequences have discrepancy of order $(\log(\log N))^{1/2}$ and thus do not achieve the lowest possible discrepancy. Subsequently, researchers have proposed an alternative way of generating 'quasirandom' numbers through the use of low discrepancy sequences. Their discrepancies have been shown to be optimal, of order $(\log N)^s/N$ [19], [20]. Quasirandom sequences, on the other hand are more useful for global optimization, because of the variation of random numbers that are produced in each iteration.

Many of the relevant low discrepancy sequences are linked to the Van der Corput sequence introduced initially for dimension $s = 1$ and base $b = 2$ [21]. The Van der Corput discovery inspired other quasi random sequences like Halton [22], Faure, Sobol [23, 24], etc. However, it has been reported that Halton and Faure sequences do not work too well when the search space has large dimensions. Keeping this fact in mind we decided to scrutinize the performance of PSO using Van der Corput sequence along with Sobol sequence (which is said be superior than other low discrepancy sequences according to the previous studies) for swarm initialization and tested them for solving global optimization problems in large dimension search spaces.

2.2. Van der Corput Sequence

A Van der Corput sequence is a low-discrepancy sequence over the unit interval first published in 1935 by the Dutch mathematician J. G. Van der Corput. It is a digital $(0, 1)$-sequence, which exists for all bases $b \geq 2$. It is defined by the *radical inverse function* $\varphi_b : N_0 \rightarrow [0, 1)$. If $n \in N_0$ has the b-adic expansion

$$n = \sum_{j=0}^{T} a_j b^{j-1} \tag{1}$$

with $a_j \in \{0, \ldots, b-1\}$, and $T = \lfloor \log_b n \rfloor$ then φ_b is defined as

$$\varphi_b(n) = \sum_{j=0}^{T} \frac{a_j}{b^j} \tag{2}$$

In other words, the jth b-adic digit of n becomes the jth b-adic digit of $\varphi_b(n)$ behind the decimal point. The Van der Corput sequence in base b is then defined as $(\varphi_b(n))_{n\geq 0}$.

The elements of the Van der Corput sequence (in any base) form a dense set in the unit interval: for any real number in $[0, 1]$ there exists a sub sequence of the Van der Corput sequence that converges towards that number. They are also uniformly distributed over the unit interval. Figures 2 and 3 depict the initial 500 points generated by using the inbuilt subroutine and by using Vander Corput sequence respectively. The figures clearly show that the initial points generated by using quasi random sequences cover the search space more evenly in comparison to the pseudo random numbers generated by using computer subroutine. The distribution of sample points in space using computer generated pseudo random numbers and sample points generated using quasi random Vander Corput sequence are shown in Fig 1 and 2 respectively. From these figures it can be easily seen that the sample points generated by quasi random sequence are far more uniformly distributed in comparison to the computer generated pseudo random sample points generated by using inbuilt subroutine.

3. Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is a relatively newer addition to a class of population based search technique for solving numerical optimization problems. Its mechanism is inspired from the complex social behavior shown by the natural species like flock of birds, school of fish and even crowd of human beings. The particles or members of the swarm fly through a multidimensional search space looking for

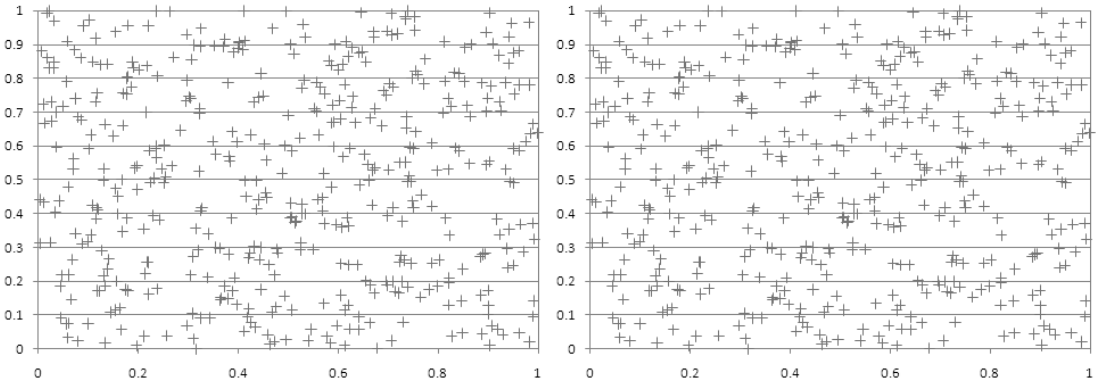

Figure 1. Sample points generated using: Fig.1 a pseudo random sequence, Fig.2 Vander Corput sequence

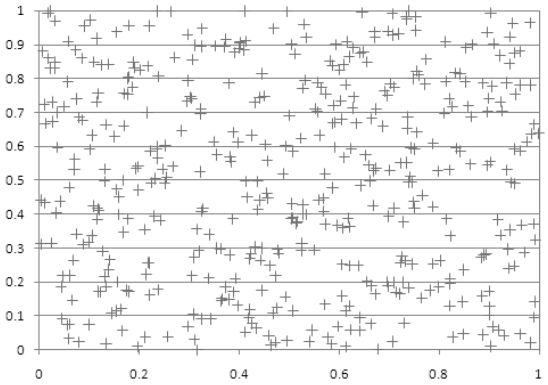

Figure 2. Sample points generated using a pseudo random sequence

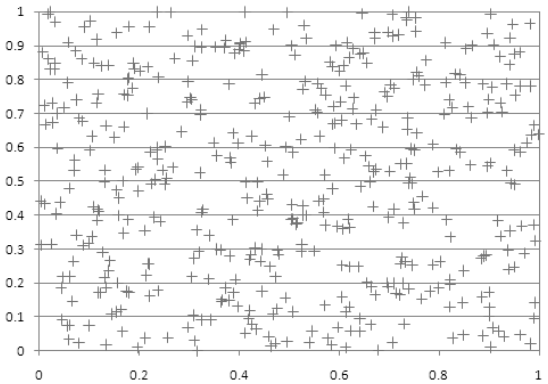

Figure 3. Sample points generated using Vander Corput sequence

a potential solution. Each particle adjusts its position in the search space from time to time as per its own experience and also as per the position of its neighbors (or colleagues).

For a D-dimensional search space the position of the ith particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$. Each particle maintains a memory of its previous best position $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ and a velocity $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ along each dimension. At each iteration, the P vector of the particle with best fitness in the local neighborhood, designated g, and the P vector of the current particle are combined to adjust the velocity along each dimension and a new position of the particle is determined using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector are given by:

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (3)$$

$$x_{id} = x_{id} + v_{id} \quad (4)$$

The first part of equation (3) represents the inertia of the previous velocity, the second part is the cognition part and it tells us about the personal thinking of the particle, the third part represents the cooperation among particles and is therefore named as the social component [25]. Acceleration constants c_1, c_2 [26] and inertia weight ω [27] are predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range of $[0, 1]$.

4. Proposed ICPSO Algorithm

The proposed algorithm ICPSO is a simple algorithm for solving constraint optimization problems, it is easy to implement. It is initialized using Vander Corput sequence and it differs from unconstrained PSO in the phase of updating the position vectors and sorting every generation solutions. The proposed ICPSO algorithm uses the low discrepancy Vander Corput Sequence for initializing the population and uses the following three selection criteria after calculating the new particle

1. If the new particle and the previous particle are feasible then select the best one
2. If both the particles are infeasible then select the one having smaller constraint violation
3. If one is feasible and the other one is infeasible then select the feasible one.

Also at the end of every iteration, the particles are sorted by using the three criteria:

1. Sort feasible solutions in front of infeasible solutions
2. Sort feasible solutions according to their fitness function values
3. Sort infeasible solutions according to their constraint violations.

The computational steps of ICPSO algorithm is given Algorithm 4:

5. Benchmark problems, Experimental Settings

5.1. Benchmark problems

The general NLP is given by nonlinear objective function f, which is to be minimized/maximized with respect to the design variables $\bar{x} = (x_1, x_2, \ldots, x_n)$ and the nonlinear inequality and equality constraints. The mathematical models of the problems considered in the paper are of the type:

Algorithm 1 ICPSO Algorithm

```
Step 1 Initialize the population (X_i) using low discrepancy Vander
    Corput Sequence
Step 2 For all particles
 Evaluate the objective function
    Calculate the constraint violation
    End for
Step 3 While stopping criterion is not satisfied
    Do
    w linearly decreases from 0.9 to 0.4
 For all particles
     Calculate velocity vector using eqn (3)
   Calculate a new particle (NX) using eqn (4)
     i.e.  NX = X_i^t + V_i^{t+1}
     If (NX and X_i^t are feasible) Then
        If (f(NX) < f(X_i^t) Then X_i^{t+1} = NX
        Else X_i^{t+1} = X_i^t
        End if
     End if
     If (NX and X_i^t are infeasible) Then
If (constraint violate(NX) < constraint violate(X_i^t)
Then X_i^{t+1} = NX
        Else X_i^{t+1} = X_i^t
        End if
     End if
   If (NX is feasible and X_i^t is infeasible)
   Then X_i^{t+1} = NX
   Else X_i^{t+1} = X_i^t
   End if
     If (f(X_i^{t+1}) < f(Pi^t)) P_i^{t+1} = X_i^{t+1}
 If (f(P_i^{t+1}) < f(Pg^t)) Pg^{t+1} = P_i^{t+1}
 End if
 End if
 End for
 Sort the particles using the three sorting rules
 Go to next generation
Step 4 End while
```

Minimize/Maximize $f(\bar{x})$
Subject to:

$$g_j(\bar{x}) \leq 0,\ j = 1, \ldots, p \quad (5)$$
$$h_k(\bar{x}) = 0,\ k = 1, \ldots, q \quad (6)$$

$x_{i\min} \leq x_i \leq x_{i\max}$ $(i = 1, \ldots, n)$, where p and q are the number of inequality and equality constraints respectively.

A set of 18 constrained benchmark problems is considered to evaluate the performance of the proposed ICPSO. All the problems are nonlinear in nature i.e. either the objective function or the constraints or both have a nonlinear term in it. The mathematical models of the problems along with the optimal solution are given in Appendix A.

5.2. Experimental settings

A total of 25 runs for each experimental setting are conducted and the average fitness of the best solutions throughout the run is recorded. The population size is taken as 50. A linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters $c_1 = c_2 = 2.0$ and r_1, r_2 as uniformly distributed random numbers between 0 and 1. The proposed ICPSO algorithm is compared with two more variations of PSO.

6. Comparison Criteria, Results and Discussion

6.1. Comparison Criteria

We used several criteria to measure the performance of the proposed ICPSO algorithm and to compare it with other versions of PSO. In Tables 1 – 4, we recorded the performance the proposed ICPSO in terms of best worst and average fitness function value along with the standard deviation (Std) while increasing the NFE (number of function evaluations) to three different values 5×10^3, 5×10^4, 5×10^5.

In Tables 5 and 6, the performance of ICPSO is compared with two other variants of PSO for solving constrained optimization problems. The comparison criteria for all the algorithms taken in the present study are given as:

Feasible Run: A run during which at least one feasible solution is found in Max NFE.
Successful Run: A run during which the algorithm finds a feasible solution x satisfying $(f(x) - f(x^*)) \leq 0.0001$.
Feasible Rate = (# of feasible runs) / total runs
Success Rate = (# of successful runs) / total runs
Success Performance = mean (FEs for successful runs) $\times$x (# of total runs) / (# of successful runs)

6.2. Results and Discussion

From Tables 1 – 4, we can see that the performance of the proposed ICPSO improves with the increase in the number of function evaluations. This is quite an expected out come. However it can be seen that 5×10^4 NFE is sufficient for reaching a good optimum solution which lies in the vicinity of the true

Table 1. Fitness function values achieved when $NFE = 5 \times 10^3$, $NFE = 5 \times 10^4$ and $NFE = 5 \times 10^5$ for problems f01 – f05

FES		f01	f02	f03	f04	f05
5 x 10^3	Best	-12.7810	0.412234	-0.5123	-30665.5314	5126.2298
	Worst	-10.3994	0.354648	-0.2144	-30665.3480	5189.3433
	Mean	-11.3257	0.363072	-0.4231	-30665.3712	5165.7069
	Std	0.77603	0.021727	0.0393	0.228162	56.177
5 x 10^4	Best	-15	0.803138	-0.7181	-30665.5386	5126.4967
	Worst	-15	0.784856	-0.3990	-30665.5386	5126.4967
	Mean	-15	0.793258	-0.6495	-30665.5386	5126.4967
	Std	9.3e-09	0.00976	0.1294	1.02e-12	5.53e-05
5 x 10^5	Best	-15	0.803618	-0.8324	-30665.5386	5126.4967
	Worst	-15	0.794661	-0.4751	-30665.5386	5126.4967
	Mean	-15	0.803113	-0.7563	-30665.5386	5126.4967
	Std	1.5e-11	0.009781	0.0245	0.0000	2.40e-12

Table 2. Fitness function values achieved when $NFE = 5 \times 10^3$, $NFE = 5 \times 10^4$ and $NFE = 5 \times 10^5$ for problems f06 – f10

NFE		f06	f07	f08	f09	f10
5 x 10^3	Best	-6961.8127	25.5805	-0.095826	680.6481	8207.3551
	Worst	-6939.9306	28.9778	-0.095826	681.1337	8399.2033
	Mean	-6958.7191	27.6499	-0.095826	680.7835	8344.4623
	Std	10.1347	0.9488	2.77e-18	0.1498	2.734
5 x 10^4	Best	-6961.8138	24.3062	-0.095826	680.6303	7049.2533
	Worst	-6961.8138	24.3118	-0.095826	681.0767	7049.2738
	Mean	-6961.8138	24.4006	-0.095826	680.6683	7049.2697
	Std	9.09e-13	0.01911	4.80e-19	0.089227	0.01456
5 x 10^5	Best	-6961.8138	24.3062	-0.095826	680.6301	7049.2480
	Worst	-6961.8138	24.3246	-0.095826	680.6435	7049.2480
	Mean	-6961.8138	24.3073	-0.095826	680.6329	7049.2480
	Std	1.98e-15	0.00402	0.0000	0.0525	1.81e-13

Table 3. Fitness function values achieved when $NFE = 5 \times 10^3$, $NFE = 5 \times 10^4$ and $NFE = 5 \times 10^5$ for problems f11 – f15

NFE		f11	f12	f13	f14	f15
5 x 10^3	Best	0.7499	-1	0.4923	-44.4379	961.7302
	Worst	0.8539	-1	0.9997	-39.8987	962.0497
	Mean	0.8102	-1	0.8807	-42.1293	961.7565
	Std	0.0733	0.0000	0.1940	1.3022	1.9369
5 x 10^4	Best	0.7499	-1	0.3212	-47.6380	961.7150
	Worst	0.7499	-1	0.6389	-45.7222	962.6006
	Mean	0.7499	-1	0.4783	-46.2218	962.2491
	Std	2.22e-16	0.0000	0.1067	1.0495	0.8847
5 x 10^5	Best	0.7499	-1	0.0531	-47.7648	961.7150
	Worst	0.7499	-1	0.434	-47.7648	961.7150
	Mean	0.7499	-1	0.32736	-47.7648	961.7150
	Std	2.22e-16	0.0000	0.171017	4.71e-15	4.42e-13

Table 4. Fitness function values achieved when $NFE = 5 \times 10^3$, $NFE = 5 \times 10^4$ and $NFE = 5 \times 10^5$ for problems f16 – f18

NFE		f16	f17	f18
5 x 10^3	Best	-1.9015	8967.5800	-0.6485
	Worst	-1.8991	11028.714	-0.4833
	Mean	-1.9001	9311.915	-0.5261
	Std	0.00251	7.618	0.05928
5 x 10^4	Best	-1.9051	8868.7455	-0.8657
	Worst	-1.9051	10903.986	-0.8644
	Mean	-1.9051	9070.5204	-0.8650
	Std	7.90e-16	4.8995	0.00066
5 x 10^5	Best	-1.9051	8853.5338	-0.8660
	Worst	-1.9051	8853.5338	-0.8660
	Mean	-1.9051	8853.5338	-0.8660
	Std	8.05e-17	1.00e-12	1.06e-16

Table 5. Comparison Results: NFE to achieve the fixed accuracy level ($(f(x) - f(x^*)) \leq 0.0001$), success rate, Feasible Rate and Success Performance for problems f01 – f10

Problem	Algo	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Perf.
f01	ICPSO	25250	55250	29796	100	100	29796
	[17]	25273	346801	76195	100	52	146530
	[27]	95100	106900	101532	100	100	101532
f02	ICPSO	81800	135750	115850	100	100	115850
	[17]	-	-	-	100	0	-
	[27]	180000	327900	231193	100	56	412844.3878
f03	ICPSO	-	-	-	100	0	-
	[17]	-	-	-	100	0	-
	[27]	450100	454000	450644	100	100	450644
f04	ICPSO	7750	12650	9568	100	100	9568
	[17]	15363	25776	20546	100	100	20546
	[27]	74300	85000	79876	100	100	79876
f05	ICPSO	13350	65400	19286	100	100	19286
	[17]	94156	482411	364218	100	16	2276363
	[27]	450100	457200	452256	100	100	452256
f06	ICPSO	7300	9600	8252	100	100	8252
	[17]	16794	22274	20043	100	100	20043
	[27]	47800	61100	56508	100	100	56508
f07	ICPSO	29050	57800	40046	100	100	40046
	[17]	315906	338659	327283	100	8	4091031
	[27]	198600	444100	352592	100	96	367282.9861
f08	ICPSO	1050	1350	1158	100	100	1158
	[17]	1395	3921	2360	100	100	2360
	[27]	2800	8400	6124	100	100	6124
f09	ICPSO	10450	29550	16248	100	100	16248
	[17]	45342	84152	58129	100	100	58129
	[27]	77000	129000	97544	100	100	97544
f010	ICPSO	66050	84900	75920	100	100	75920
	[17]	290367	486655	426560	100	32	1332999
	[27]	398000	475600	452575	100	16	2828593.75

Table 6. Comparison Results: NFE to achieve the fixed accuracy level ($(f(x) - f(x^*)) \leq 0.0001$), success rate, Feasible Rate and Success Performance for problems f11 – f18

Problem Problem	Algo Algo	Best Best	Worst Worst	Mean Mean	Feasible Rate (%)	Success Rate (%)	Success Perf.
f11	ICPSO	1650	24250	13630	100	100	13630
	[17]	5475	21795	16386	100	100	16386
	[27]	450100	450100	450100	100	100	450100
f12	ICPSO	850	1100	976	100	100	976
	[17]	1409	9289	4893	100	100	4893
	[27]	3300	10900	8088	100	100	8088
f13	ICPSO	88700	111100	102512	100	16	640700
	[17]	-	-	-	100	0	-
	[27]	450100	453200	450420	100	100	450420
f14	ICPSO	21250	339550	50614	100	100	50614
	[17]	-	-	-	100	0	-
	[27]	-	-	-	100	0	-
f15	ICPSO	7400	128100	54306	100	100	54306
	[17]	17857	348138	176827	100	80	221033
	[27]	450100	450100	450100	100	100	450100
f16	ICPSO	7100	10650	8732	100	100	8732
	[17]	24907	51924	33335	100	100	33335
	[27]	43400	53900	49040	100	100	49040
f17	ICPSO	256800	463350	408506	96	72	567369
	[17]	-	-	-	100	0	-
	[27]	-	-	-	100	0	-
f18	ICPSO	53600	89900	71694	100	100	71694
	[17]	85571	455907	191220	100	80	239026
	[27]	120800	394900	214322	100	92	232958.4121

optimum value, under the present parameter settings. Also, we can see that except for problem number 5 (g05), where the standard deviation (std) is 56.177, the std for all the remaining problems is quite low. This shows the consistency of the proposed ICPSO algorithm. The superior performance of ICPSO is more visible from Tables 5 and 6 where the results are recorded after fixing the accuracy at 0.0001. In these tables we can see that the proposed ICPSO gave a better or at par performance with the other two algorithms. We will now take the comparison criteria one-by-one and discuss them briefly. The first criterion is that of a feasible run. A run is said to be feasible if at least one feasible solution is obtained in maximum number of function evaluations. According to this criterion all the algorithm gave 100% feasible rate for all the test problems except ICPSO which gave 96% feasible rate for test problem f17. However, if we observe the second criterion which is of successful run and is recorded when the algorithm finds a feasible solution satisfying the given accuracy (=0.0001) it can be seen that the proposed ICPSO outperforms the other algorithms in all the test cases including f17. In f17, the

percentage of success rate for ICPSO is 72, whereas the other algorithms were not able to reach the prescribed accuracy in any of the run. The third criterion is that of the success performance which depends on the feasibility rate and success rate, as described in the previous subsection. Here also ICPSO gave a better performance in comparison to the other two algorithms taken for comparison.

7. Conclusion

In the present study a low discrepancy Vander Corput sequence initialized particle swarm optimization called ICPSO is proposed for solving constrained optimization. Besides the initialization process, the proposed algorithm differs from the basic PSO in the updating of position vectors and sorting of every generation solutions. The proposed technique for solving constrained optimization problems, though used for PSO in this paper, can be applied to any other population based search technique with minor modifications. The empirical analysis of the proposed ICPSO algorithm on 18 constrained benchmark problems and its comparison with other algorithms show that the proposed algorithm is quite promising for solving constrained problems. In the present study we have used the Vander Corput sequence however any other low discrepancy sequences like Sobol or Halton may also be used to initialize the population. We are continuing the further study of the algorithm and are using it for solving constrained real life problems taken from various fields of Science and Engineering.

References

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, 2002.

[2] A. P. Engelbrecht, “Fundamentals of Computational Swarm Intelligence”, England : John Wiley & Sons Ltd., 2005.

[3] G. Coath, S. K. Halgamuge, “A Comparison of Constraint- Handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization Problems”, In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 4, pp. 2419 - 2425, 2003.

[4] S. Koziel, Z. Michalewicz, “Evolutionary Algorithms, Homomorphus Mappings, and Constrained Optimization”, Evolutionary Computation, Vol. 7(1), pp. 19 - 44, 1999.

[5] Z. Michalewicz, “A Survey of Constraint Handling Techniques in Evolutionary Computation Methods”, In Proc. of the Fourth Annual Conf. on Evolutionary Programming, pp. 135 - 155, 1995.

[6] Y. Li, M.Gen, “Non-linear mixed integer programming problems using genetic algorithm and penalty function”, In Proc. of 1996 IEEE Int. Conf. on SMC, pp. 2677 - 2682, 1996.

[7] Y. Takao, M. Gen, T. Takeaki, Y. Li, “A method for interval 0-1 number non-linear programming problems using genetic algorithm”, Computers and Industrial Engineering, Vol. 29, pp. 531 - 535, 1995.

[8] J. F. Tang, D.Wang, et al., “A hybrid genetic algorithm for a type of nonlinear programming problem”, Computer Math. Applic, Vol. 36(5), pp. 11 - 21, 1998.

[9] M. Dorigo, V. Maniezzo, A. Colori, “Ant system optimization by a colony of cooperating agents”, IEEE Trans. on system, Man, and Cybernetics, Vol. 26(1), pp. 28 - 41, 1996.

[10] J. Kennedy, R. Eberhart, “Particle Swarm Optimization”, IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: pp. 1942-1948, 1995.

[11] R. Storn and K. Price, “Differential Evolution - a simple and efficient Heuristic for global optimization over continuous spaces”, Journal Global Optimization. 11, 1997, pp. 341 - 359.

[12] S. Kimura and K. Matsumura, “Genetic Algorithms using low discrepancy sequences”, in proc of GEECO 2005, pp. 1341 - 1346.

[13] Nguyen X. H., Nguyen Q. Uy., R. I. Mckay and P. M. Tuan, “Initializing PSO with Randomized Low-Discrepancy Sequences: The Comparative Results”, In Proc. of IEEE Congress on Evolutionary Algorithms, 2007, pp. 1985 - 1992.

[14] K.E. Parsopoulos and M.N. Vrahatis, “Particle Swarm Optimization in noisy and continuously changing environments”, in Proceedings of International Conference on Artificial Intelligence and soft computing, 2002, pp. 289-294.

[15] R. Brits and A.P. Engelbrecht and F. van den Bergh, “A niching Particle Swarm Optimizater”, In proceedings of the fourth Asia Pacific Conference on Simulated Evolution and learning, 2002, pp 692 - 696.

[16] R. Brits and A.P. Engelbrecht and F. van den Bergh, “Solving systems of unconstrained Equations using Particle Swarm Optimization”, In proceedings of the IEEE Conference on Systems, Man and Cybernetics, Vol. 3pp. 102 - 107, 2002.

[17] Millie Pant, Radha Thangaraj and Ajith Abraham “Improved Particle Swarm Optimization with Low-discrepancy Sequences”, IEEE Cong. on Evolutionary Computation (CEC’08), Hong Kong, 2008.

[18] Karin Zielinski and Rainer Laur : “Constrained Single-Objective Optimization Using Particle Swarm Optimization”, IEEE Congress on Evolutionary Computation, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006, pp. 443 - 450

[19] E. J. Gentle, “RandomNumber Generation and Monte Carlo Methods”, Springer-Verlog, 1998.

[20] D. E. Knuth, “The Art of Computer Programming”, Semi numerical Algorithms, Vol. 2, Addison-Wesley, 1998.

[21] J. G. van der Corput, Verteilungsfunktionen. Proc. Ned. Akad. v. Wet., 38: pp. 813-821, 1935.

[22] J. Halton, “On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals,” Numerische Mathematik, Vol. 2, 1960, pp. 84 - 90.

[23] I. M. Sobol’, “On the distribution of points in a cube and the approximate evaluation of integrals,” USSR Computational Mathematics and Mathematical Physics, Vol. 7, 1967, pp. 86 - 112.

[24] I. M. Sobol’, “Uniformly distributed sequences with an additional uniform property,” USSR Computational Mathematics and Mathematical Physics, Vol. 16, 1976, pp. 236 - 242.

[25] J. Kennedy, “The Particle Swarm: Social Adaptation of Knowledge”, IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ, 1997, pp. 303 - 308.

[26] R. C. Eberhart and Y. Shi, “Particle Swarm Optimization: developments, Applications and Resources”, IEEE Int. Conference on Evolutionary Computation, 2001, pp. 81 - 86.

[27] Y. H. Shi and R. C. Eberhart, “A Modified Particle Swarm Optimizer”, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998, pp. 69 - 73.

[28] Angel E. Mu?noz-Zavala, Arturo Hern?andez-Aguirre, Enrique R. Villa-Diharce and Salvador Botello-Rionda: ”PESO+ for Constrained Optimization” IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006, pp. 231 - 238.

A. Appendix

1. F01:

   Minimize $f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$

   Subject to:

$$\begin{aligned} g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\ g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\ g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\ g_4(x) &= -8x_1 + x_{10} \leq 0 \\ g_5(x) &= -8x_2 + x_{11} \leq 0 \\ g_6(x) &= -8x_3 + x_{12} \leq 0 \\ g_7(x) &= -2x_4 - x_5 + x_{10} \leq 0 \\ g_8(x) &= -2x_6 - x_7 + x_{11} \leq 0 \\ g_9(x) &= -2x_8 - x_9 + x_{12} \leq 0 \end{aligned}$$

$$0 \leq x_i \leq 1\ (i = 1, 2, \ldots, 9), 0 \leq x_i \leq 100, (i = 10, 11, 12)\ 0 \leq x_{13} \leq 1$$

   The optimum value is $f(x^*) = -15$ at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$

   Constraints $g_1, g_2, g_3, g_7, g_8, g_9$ are active.

2. F02:

   Maximize

$$f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sum_{i=1}^{n} i x_i^2} \right|$$

   Subject to:

$$\begin{aligned} g_1(x) &= 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\ g_2(x) &= \sum_{i=1}^{n} x_i - 7.5n \leq 0 \end{aligned}$$

$$0 \leq x_i \leq 10\ (i = 1, 2, \ldots, n),\ n = 20$$

   The optimum value is unknown. The known best value is $f(x^*) = 0.803619$

   Constraint g_1 is active.

3. F03:

   Minimize

$$f(x) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$$

Subject to:

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

$$0 \leq x_i \leq 10 \ (i = 1, 2, \ldots, n)$$

The optimum value is $f(x^*) = -1$ at $x^* = (1/\sqrt{n})$, $n = 10$.

4. F04

Minimize $f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$

Subject to:

$$\begin{aligned} g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \\ g_2(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \\ g_3(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \end{aligned}$$

$$0 \leq g_1(x) \leq 92$$

$$90 \leq g_2(x) \leq 110$$

$$20 \leq g_3(x) \leq 25$$

$78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45 \ (i = 3, 4, 5)$.

The optimum value is $f(x^*) = -30665.539$ at
$x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$

5. F05

Minimize $f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

Subject to:

$$\begin{aligned} g_1(x) &= -x_4 + x_3 - 0.55 \leq 0 \\ g_2(x) &= -x_3 + x_4 - 0.55 \leq 0 \\ h_3(x) &= 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\ h_4(x) &= 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\ h_5(x) &= 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0 \end{aligned}$$

$0 \leq x_i \leq 1200 \ (i = 1, 2)$, $-0.55 \leq x_2 \leq 0.55 \ (i = 3, 4)$.

The optimum value is $f(x^*) = 5126.4981$ at $x^* = (679.9463, 1026.067, 0.1188764, -0.3962336)$.

6. F06

Minimize $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

Subject to:

$$\begin{aligned} g_1(x) &= -(x_1-5)^2-(x_2-5)^2+100 \le 0 \\ g_2(x) &= (x_1-6)^2+(x_2-5)^2-82.81 \le 0 \end{aligned}$$

$13 \le x_1 \le 100$, $0 \le x_2 \le 100$

The optimum value is $f(x^*) = -6961.81388$ at $x^* = (14.095, 0.84296)$.

7. F07:

Minimize $f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-5)^2$
$+2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45$

Subject to:

$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0 g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0 g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0 g_4(x) = 3(x_1-2)^2 + 4(x_2-3)^2 + 2x_3^2 - 7x_4 - 120 \le 0 g_5(x) = 5x_1^2 + 8x_2 + (x_3-6)^2 - 2x_4 - 40 \le 0 g_6(x) = x_1^2 + 2(x_2-2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0 g_7(x) = 0.5(x_1-8)^2 + 2(x_2-4)^2 + 3x_5^2 - x_6 - 30 \le 0 g_8(x) = -3x_1 + 6x_2 + 12(x_9-8)^2 - 7x_{10} \le 0 - 10 \le x_i \le 10 (i = 1, 2, \ldots, 10)$

The optimum value is $f(x^*) = 24.3062091$ at $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$

Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

8. F08:

Maximize $f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$

Subject to:

$g_1(x) = x_1^2 - x_2 + 1 \le 0 g_2(x) = 1 - x_1 + (x_2-4)^2 \le 0$

$0 \le x_i \le 10$ $(i = 1, 2)$.

The optimum value is $f(x^*) = 0.095825$at $x^* = (1.2279713, 4.2453733)$.

9. F09:

Minimize $f(x) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$

Subject to:

$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0 g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0 g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0 g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0 - 10 \le x_i \le 10 (i = 1, 2, \ldots, 7)$

The optimum value is $f(x^*) = 680.6300573$at $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.624487, 1.038131, 1.5942270)$.

10. F10:

    Minimize $f(x) = x_1 + x_2 + x_3$

    Subject to:

    $g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0 g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0 g_4(x) = -x_1 x_6 + 833.33252 x_4 + 100x1 - 83333.333 \leq 0 g_5(x) = -x_2 x_7 + 1250x_5 + x_2 x_4 - 1250x_4 \leq 0 g_6(x) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500x_5 \leq 0$

    $-100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i = 2, 3)$, $10 \leq x_i \leq 1000 (i = 4, \ldots, 8)$.

    The optimum value is $f(x^*) = 7049.25$at

    $x^* = (579.19, 1360.13, 5109.5979, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$.

11. F11:

    Minimize $f(x) = x_1^2 + (x_2 - 1)^2$

    Subject to:

    $h_1(x) = x_2 - x_1^2 = 0 - 1 \leq x_i \leq 1 \ (i = 1, 2)$

    The optimum value is $f(x^*) = 0.75$at $x^* = (\pm 1/\sqrt{2}, 1/2)$.

12. F12:

    Minimize $f(x) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$

    Subject to:

    $g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

    $0 \leq x_i \leq 10$, $i = 1, 2, 3$, $p, q, r = 1, 2, \ldots, 9$

    The optimum value is $f(x^*) = -1$at $x^* = (5, 5, 5)$.

13. F13:

    Minimize $f(x) = e^{x_1 x_2 x_3 x_4 x_5}$

    Subject to:

    $h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 h_2(x) = x_2 x_3 - 5x_4 x_5 = 0 h_3(x) = x_1^3 + x_2^3 + 1 = 0$

    $-2.3 \leq x_i \leq 2.3$, $i = 1, 2$

    $-3.2 \leq x_i \leq 3.2$, $i = 3, 4, 5$

    The optimum value is $f(x^*) = 0.0539$at $x^* = (-1.7171, 1.5957, 1.8272, -0.7636, -0.7636)$.

14. F14:

    Minimize $f(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$

    Subject to:

    $h_1(x) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0 h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$

$0 < x_i \leq 10$, $i = 1, \ldots, 10$

Where $c_1 = -6.089, c_2 = -17.164, c_3 = -34.054, c_4 = -5.914, c_5 = -24.721, c_6 = -14.986, c_7 = -24.1, c_8 = -10.708, c_9 = -26.662, c_{10} = -22.179$

The optimum value is $f(x^*) = -47.7648$at $x^* = (0.04066, 0.14772, 0.78320, 0.00141, 0.48529, 0.00069, 0.02740, 0.017950, 0.03732, 0.09688)$

15. F15:

Minimize $f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$

Subject to:

$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0 h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$

$0 \leq x_i \leq 10$, $i = 1, 2, 3$

The optimum value is $f(x^*) = 961.7150$at $x^* = (3.5121, 0.2169, 3.5521)$

16. F16:

Minimize $f(x) = 0.00011y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$

Subject to:

$g_1 = \frac{0.28}{0.72}y_5 - y_4 \leq 0$, $g_2 = x_3 - 1.5x_2 \leq 0$,

$g_3 = 3496\frac{y_2}{c_{12}} - 21 \leq 0$, $g_4 = 110.6 + y_1 - \frac{62{,}212}{c_{17}} \leq 0$, $g_5 = y_1 - 405.23 \leq 0$, $g_6 = 213.1 - y_1 \leq 0$

$g_7 = y_2 - 1053.6667 \leq 0$, $g_8 = 17.505 - y_2 \leq 0$

$g_9 = y_3 - 35.03 \leq 0$, $g_{10} = 11.275 - y_3 \leq 0$, $g_{11} = y_4 - 665.585 \leq 0$, $g_{12} = 214.228 - y_4 \leq 0$, $g_{13} = y_5 - 584.463 \leq 0$,$g_{14} = 7.458 - y_5 \leq 0$,

$g_{15} = y_6 - 265.916 \leq 0$, $g_{16} = 0.961 - y_6 \leq 0$, $g_{17} = y_7 - 7.046 \leq 0$, $g_{18} = 1.612 - y_7 \leq 0$, $g_{19} = y_8 - 0.222 \leq 0$, $g_{20} = 0.146 - y_8 \leq 0$, $g_{21} = y_9 - 273.366 \leq 0$, $g_{22} = 107.99 - y_9 \leq 0$, $g_{23} = y_{10} - 1286.105 \leq 0$, $g_{24} = 922.693 - y_{10} \leq 0$, $g_{25} = y_{11} - 1444.046 \leq 0$, $g_{26} = 926.832 - y_{11} \leq 0$, $g_{27} = y_{12} - 537.141 \leq 0$, $g_{28} = 18.766 - y_{12} \leq 0$, $g_{29} = y_{13} - 3247.039 \leq 0$, $g_{30} = 1072.163 - y_{13} \leq 0$, $g_{31} = y_{14} - 26844.086 \leq 0$,

$g_{32} = 8961.448 - y_{14} \leq 0$, $g_{33} = y_{15} - 0.386 \leq 0$, $g_{34} = 0.063 - y_{15} \leq 0$, $g_{35} = y_{16} - 140{,}000 \leq 0$,

$g_{36} = 71{,}084.33 - y_{16} \leq 0$, $g_{37} = y_{17} - 12{,}146{,}108 \leq 0$, $g_{38} = 2{,}802{,}713 - y_{17} \leq 0$,

$704.4148 \leq x_1 \leq 906.3855$,$68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$,

$25 \leq x_5 \leq 84.1988$.

Calculations:

$y_1 = x_2 + x_3 + 41.6$, $c_1 = 0.024x_4 - 4.62$, $y_2 = \frac{12.5}{c_1} + 12$,

$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$, $c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$,

$y_3 = \frac{c_2}{c_3}$, $y_4 = 19y_3$, $c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$,

$c_5 = 100x_2$, $c_6 = x_1 - y_3 - y_4$, $c_7 = 0.95 - \frac{c_4}{c_5}$, $y_5 = c_6c_7$, $y_6 = x_1 - y_5 - y_4 - y_3$

$c_8 = (y_5 + y_4)0.995$, $y_7 = \frac{c_8}{y_1}$, $y_8 = \frac{c_8}{3798}$, $c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$

$y_9 = \frac{96.82}{c_9} + 0.321y_1$, $y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$, $y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$

$c_{10} = \frac{12.3}{752.3}$, $c_{11} = (1.75y_2)(0.995x_1)$, $c_{12} = 0.995y_{10} + 1998$, $y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$

$y_{13} = c_{12} - 1.75y_2$, $y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}$

$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$, $y_{15} = \frac{y_{13}}{c_{13}}$

$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$, $c_{14} = 2324y_{10} - 28740000y_2$

$y_{17} = 14130,000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$, $c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$

$c_{16} = 1.104 - 0.72y_{15}$, $c_{17} = y_9 + x_5$.

17. F17:

Minimize $f(x) = f_1(x_1) + f_2(x_2)$

Constraints:

$$f_1(x_1) = \begin{cases} 31x_1 & 0 \le x_1 \le 300 \\ 30x_1 & 300 \le x_1 \le 400 \end{cases} \quad f_2(x_2) = \begin{cases} 28x_2 & 0 \le x_2 \le 100 \\ 29x_2 & 100 \le x_2 \le 200 \\ 30x_2 & 200 \le x_2 \le 1000 \end{cases}$$

$x_1 = 300 - \frac{x_3 x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588)$

$x_2 = -\frac{x_3 x_4}{131.078}\cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588)$

$x_5 = -\frac{x_3 x_4}{131.078}\sin(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\sin(1.47588)200 - \frac{x_3 x_4}{131.078}\sin(1.48477 - x_6) +$
$+ \frac{0.90798x_3^2}{131.078}\sin(1.47588) = 0$

$$0 \le x_1 \le 400$$
$$0 \le x_2 \le 1000$$
$$340 \le x_3 \le 420$$
$$340 \le x_4 \le 420$$
$$-1000 \le x_5 \le 1000$$
$$0 \le x_6 \le 0.5236$$

18. F18:

Minimize $f(x) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$

Subject to:

$g_1(x) = x_3^2 + x_4^2 - 1 \le 0 g_2(x) = x_9^2 - 1 \le 0 g_3(x) = x_5^2 + x_6^2 - 1 \le 0 g_4(x) = x_1^2 + (x_2 - x_9)^2 - 1 \le 0 g_5(x) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \le 0 g_6(x) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \le 0 g_7(x) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \le 0$

$$g_8(x) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \le 0$$

$$g_9(x) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$$
$$g_{10}(x) = x_2x_3 - x_1x_5 \leq 0$$
$$g_{11}(x) = -x_3x_9 \leq 0$$
$$g_{12}(x) = x_5x_9 \leq 0$$
$$g_{13}(x) = x_6x_7 - x_5x_8 \leq 0$$

$-10 \leq x_i \leq 10, i = 1, \ldots, 8, 0 \leq x_9 \leq 10$

The optimum value is $f(x^*) = -0.8660$at $x^* = (-0.65777, -0.15341, 0.32341, -0.94625, -0.65777, -0.75321, 0.32341, -0.34646, 0.59979)$