

A Self-adaptive Differential Evolution Algorithm with Constraint Sequencing

Md. Asafuddoula, Tapabrata Ray, and Ruhul Sarker

School of Engineering and Information Technology
University of New South Wales
Northcott Drive, Canberra ACT 2600 Australia
{Md.Asaf}@student.adfa.edu.au,
{t.ray, r.sarker}@adfa.edu.au
<http://www.unsw.adfa.edu.au>

Abstract. Constrained optimization is an active area of research where attempts are being regularly made to improve the efficiency of the underlying optimization algorithms. While population based stochastic algorithms such as evolutionary algorithms, differential evolution (DE), particle swarm optimization etc. have been the popular choice as the underlying optimization scheme, adaptive strategies are usually employed to deal with constraints. Most of such approaches adopt a complete evaluation policy, i.e., all constraints and objectives corresponding to a solution is always evaluated for every solution under consideration. However, in a typical constrained optimization problem, one or more constraints are often difficult to satisfy and it might be beneficial to evaluate the constraints in a sequence. Evaluation of subsequent constraints and objective function can be skipped whenever a constraint is violated. In this paper, a self adaptive differential evolution algorithm is introduced which maintains multiple subpopulations, each of which is assigned a prescribed constraint sequence based on a ring topology. Solutions are ranked in each subpopulation and a migration scheme is employed to transfer feasible solutions to a subpopulation of feasible individuals. The performance of the proposed scheme is compared with a single sequence approach and other state of the art DE forms using the standard *g-series* test functions having inequality constraints. The results clearly indicate the potential savings in the computational cost. Apart from savings in computational cost, the paper also makes an important contribution as it provides useful physical insights on the search trajectories and their effect in various forms of constrained optimization problems.

Keywords: differential evolution, constraint sequencing, sequence sorting, self-adaptive.

1 Introduction

The performance of all population based stochastic optimization algorithms are affected by the presence of constraints. The nonlinearity, multi-modality and the feasibility associated with each constraint is likely to be different and would vary in different regions of the search space. Various forms of constraint handling schemes have been proposed

in literature. Such methods can be broadly categorized in four different types [10,2] i.e. use of penalty functions [3], repair schemes [4], use of decoders [7] and the separation of objective function and constraints [11,5,13]. More recent methods on the other hand, maintain infeasible solutions via stochastic ranking [9], ε based comparisons [14] and adaptive penalty function formulations [12]. However, in all such formulations a full evaluation policy is adopted wherein for an infeasible solution, its constraint violation measure is computed. The term constraint violation (CV) is defined as follows:

$$CV = \sum_{i=1}^p |\min(g_i, 0)| + \sum_{i=1}^q \max(|h_i - \varepsilon|, 0) \quad (1)$$

where g and h are the inequality and equality constraints and p is the number of inequality constraints and q is the number of equality constraints. In this work, we have suggested a multi-population based algorithm in the framework of DE where, the population is divided into multiple subpopulations and follow the prescribed constraint sequences. The choice of DE as the underlying optimizer in this work is based on DE's recent success in the winning entries for CEC-2011 competition problems. The performance of the algorithm is subsequently compared with four top performing algorithms include ε -DE [14], JDE [1], COPSO [15] and SaDE [6] in well known *g-series* [8] test problems for inequalities.

Furthermore, since the performance of DE is known to be affected by the choice of its user defined parameters, a self adaptive scheme is employed. The rest of the paper is organized as follows. The proposed approach with an illustrative example is presented in Section 2 while the details of the algorithm is presented in Section 3. The performance of the algorithm is presented in Section 4. The final section summarizes the findings.

2 Proposed Approach

In this work, a population is divided into $p + q$ subpopulations where p and q denotes the number of inequality and equality constraints of the problem. Each subpopulation is assigned a constraint sequence and solutions in the subpopulation are evaluated following the prescribed sequence until it encounters a constraint violation. For any solution in a given subpopulation, the objective function is only computed when it satisfies all its constraints. The solutions are sorted in its subpopulation following a constraint sequence and feasible solutions migrate to a feasible subpopulation. The sequences in each subpopulation follow a ring topology wherein in each subpopulation, solutions tend to approach feasibility from different search directions. Such an approach could be beneficial in two ways i.e. (a) less likely to be trapped at a local optima and (b) save computational cost by avoiding evaluation of infeasible solutions whenever infeasibility is detected. A simple two variable constrained optimization problem is formulated to illustrate the principle.

2.1 An Illustrative Example

To demonstrate the proposed constraint handling method, a simple two variable optimization problem ($T1$) involving three constraints is presented below. The problem has a feasible region bounded by three linear constraints. Since our focus is on handling constraints, our attempt is to show the trajectory of solutions in different subpopulations.

$$\begin{aligned}
 &\text{Minimize } f_1(x) = x_1^2 + x_2^2 + 2 * x_1 * x_2 \\
 &\text{Subject to} \\
 &\quad g_1(x) \equiv x_1 + 2 * x_2 \geq 0, \\
 &\quad g_2(x) \equiv 10 * x_1 - 8 * x_2 - 15 \geq 0, \\
 &\quad g_3(x) \equiv -10 * x_1 + 2 * x_2 - 2 \geq 0,
 \end{aligned} \tag{2}$$

In the proposed approach, three constraint sequences have been considered each of which is assigned to a subpopulation i.e. constraints (g_1, g_2, g_3) , (g_2, g_3, g_1) and (g_3, g_1, g_2) are the prescribed sequences for subpopulation, 1, 2 and 3. As an example, let us consider subpopulation 1, containing 4 solutions (S1, S2, S3, S4). The constraint violation matrix would assume a form illustrated in Table 1. Since the prescribed sequence for this subpopulation is (g_1, g_2, g_3) , the solutions are sequentially sorted to yield (S3, S2, S4, S1) where S3 is deemed the best and S1 the worst.

Table 1. An example of sequence sorting

Initial order				Final order			
	g1	g2	g3	g1	g2	g3	
S1	5	inf	inf	0	0	1	S3
S2	0	3	inf	0	3	inf	S2
S3	0	0	1	2	inf	inf	S4
S4	2	inf	inf	5	inf	inf	S1

A small population size of 30 is used to illustrate the behavior. Presented in Fig. 1 is the trajectory followed by the infeasible solutions, if a CV based scheme is employed. A CV based scheme is often used for population based stochastic algorithms. It is interesting to observe that the solutions tend to be located in a small region of the space satisfying g_2 and g_3 constraints. With such infeasible solutions, the CV based approach will be effective if the optimal feasible solution is close to such infeasible solutions and will be ineffective if the feasible solution is away from such regions.

Presented in Fig. 2, is the same trajectory of the solutions in subpopulations 1, 2 and 3. One can clearly observe that the infeasible solutions in subpopulations clearly are in different regions of the search space and solutions in each subpopulation tend to approach the feasible region from different directions. Such an approach is thus likely to locate optimal solutions as they attempt to enter the feasible search space from different directions.

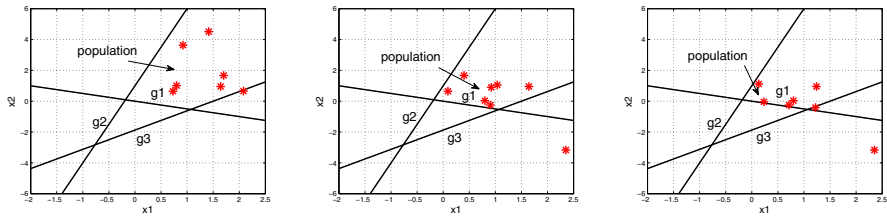
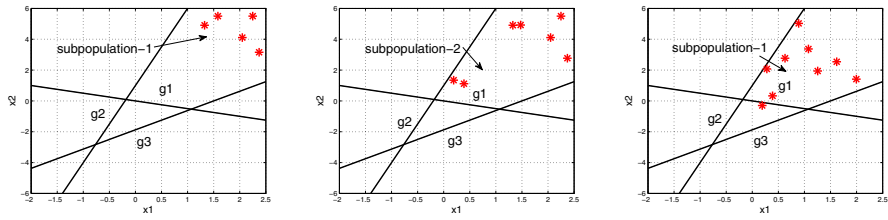
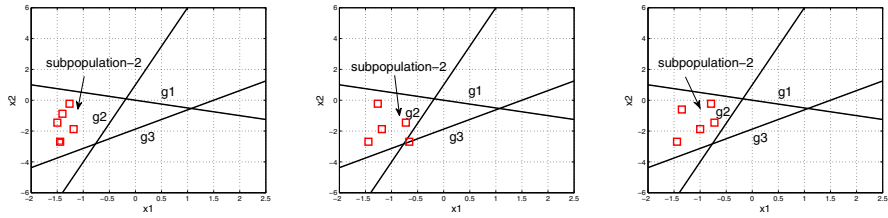


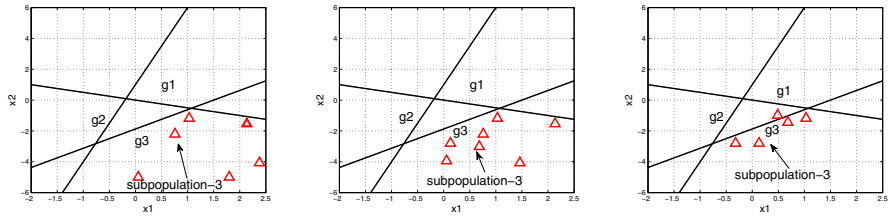
Fig. 1. Progress plots for test problem $T1$ using DE algorithm with constraint violation (DEACV) at generation 5, 10 and 20



(a) Progress of subpopulation-1



(b) Progress of subpopulation-2



(c) Progress of subpopulation-3

Fig. 2. Progress plots for test problem $T1$ of multiple subpopulations using DE algorithm with constraint sequencing (DEACS) at generation 5, 10 and 20

3 Self-adaptive Differential Evolution Algorithm with Constraint Sequencing (DEACS)

The pseudocode of the proposed algorithm is presented below. A population of N individuals and an archive of $N * 2$ are initialized. For each individual in the population the parameters regarding the crossover rate (CR) and mutation factor (F) are assigned randomly in the bounded of $[0,1]$. The population is divided into $p + q$ subpopulations with a prescribed constraint sequence for each subpopulation. In each subpopulation the solutions are ranked using a sequence sort. In order to create a new candidate solution, first parent is selected from the subpopulation itself, the second parent is selected from the entire population and the third parent is selected from the archive.

Algorithm 1. DEACS

SET: NT_{max} {Total number of constraints and objective function evaluation}, M {Size of elite archive}, N {Size of population}, $FeasibleSet = \{\}$, $Evalcount = 0$

```

1:  $Pop = initialize()$ ;  $Archive = initialize()$ 
2: Assign a random  $CR$  and  $F$  values from  $[0, 1]$  for all the individuals
3: Distribute individuals to  $(p + q + 1)$  subpopulations
4: Assign constraint sequences to subpopulations
5: Evaluate solutions in the subpopulations;  $Update(Evalcount)$ 
6: Rank solutions in each subpopulation using sequence sort
7: Migrate feasible solutions from all the subpopulations to  $FeasibleSet$ 
8: while  $((Evalcount \leq NT_{max}))$  do
9:   for  $i=1:p+q+1$  do
10:    if  $!isempty(Subpop_i)$  then
11:      for  $j=1:size(Subpop_i)$  do
12:         $p_1 = i, p_2 = rand(M), p_3 = rand(N), p_1 \neq p_2 \neq p_3$ 
13:         $O = DE_{evolve}(p_1, p_2, p_3)$ 
14:         $Evaluate(O); Update(Evalcount)$ 
15:         $Temp = Merge(O, Subpop_i)$ 
16:         $Rank = SequenceSort(Temp)$ 
17:        Select best individual from  $Temp$  and replace  $Subpop_i$ 
18:      end for
19:    end if
20:  end for
21:  Migrate feasible solutions from all the subpopulations to  $FeasibleSet$ 
22: end while

```

* $Evalcount$ denotes the sum of objective and all individual constraint evaluations

In the DE evolve process, a binomial crossover [6] has been used with the crossover and mutation parameters (i.e. CR and F) from the first parent (i.e. base parent). The new candidate solution is then evaluated and compared with the base parent via sequence sort. If the candidate solution replaces the parent solution, the crossover and mutation parameters of the base parent are retained by the candidate solution else new parameters are randomly assigned to the base parent and candidate solution is moved to archive. In the event a feasible solution is uncovered in any subpopulation, it migrates to the

feasible subpopulation. In the event all the infeasible subpopulations are empty, three parents are chosen from the feasible subpopulation.

4 Numerical Experiments

The performance of the algorithm is evaluated using the widely used benchmark problems Table 2 [8]. The selected test problems involve quadratic, nonlinear, cubic, and polynomial objective functions with a number of inequality constraints. For comparison, we have selected five algorithms including the proposed DEACS. These five algorithms are listed as follows: The proposed algorithm (DEACS), ϵ -DE [14], JDE [1], COPSO [15], SaDE [6].

Table 2. Summary of test problems

Problem	D	Search Range	f type	No. of Inequality	Feasibility (p)
g01	13	$[0, 1]^9 [0, 100]^3 [0, 1]^{(D-12)}$	quadratic	6	0.0111%
g02	20	$[0, 10]^D$	nonlinear	2	99.971%
g04	5	$[78, 102][33, 45][27, 45]^{(D-2)}$	quadratic	6	52.1230%
g06	2	$[13, 100][0, 100]$	cubic	2	0.0066%
g07	10	$[-10, 10]^D$	quadratic	8	0.0003%
g08	2	$[0, 10]^D$	nonlinear	2	0.8560%
g09	7	$[-10, 10]^D$	polynomial	4	0.5121%
g10	8	—	linear	6	0.0010%
g12	3	$[0, 10]^D$	quadratic	1	4.7713%
g18	9	$[-10, 10]^8 [0, 20]^{(D-8)}$	quadratic	13	0.000%
g24	2	$[0, 3][0, 4]$	linear	2	79.6556%

4.1 Experimental Setup

A population size of 50 is used for the proposed algorithm which is the same used in previous studies. Results of all the problems are computed using 25 independent runs. An archive size of 100 is used for the proposed algorithm across all problems. A self-adaptive approach is used to determine the set of parameters (i.e. CR and F). We also include the results of two other adaptive DE algorithms for a fair comparison.

4.2 Computational Complexity

The algorithm complexity is computed for the selected 11 test problems of CEC-2006 according to [8]. Table 3 shows the complexity of the algorithm by $T1$, $T2$ and $(T2 - T1)/T1$ where, $T1$ is the average computing time of 10000 evaluations for all the problems and $T2$ is the average complete computing time for the algorithm with 10000 evaluations for all the problems. We have used Matlab 7.8 to implement the algorithm and the system configuration is as follows: Intel(R) Core(TM)2 Duo 3.00 GHZ, 3.49 GB of RAM, Windows XP Professional Version 2002.

Table 3. Computational Complexity

T1	T2	(T2-T1)/T1
4.8602	5.3359	0.0979

4.3 Results Comparison

One can observe from Table 4 that the proposed algorithm required significantly less number of constraint and objective function evaluations for problems g01, g04, g07, g09, g10, g12 and g18 as compared to the method proposed by Takahama and Sakai [14]. When compared with ε -DE, the proposed algorithm is better in 7 out of 11 test problems using $1e-4$ as the measure of merit.

Table 4. Comparison of function evaluations is used by the proposed method, DEACS with other best known algorithms with an accuracy level of $(f(x) - f^*(x)) \leq 0.0001$ and feasible solution, the lowest evaluation in total of function and constraints are in bold face

Proposed Method, DEACS		Function eval.					
Prob.	Best		Median		Worst		
	#func.	#const.	#func.	#const.	#func.	#const.	
g01	6,847	10,537	11,699	16,787	16,613	22,163	
g02	49,176	71,438	105,373	152,686	108,302	154,151	
g04	3,691	4,857	4,234	5,592	4,777	6,110	
g06	2,426	6,057	2,626	7,820	4,162	9,214	
g07	12,652	26,790	17,891	37,084	47,587	87,334	
g08	138	900	557	1,537	812	1,960	
g09	6,608	9,291	7,418	10,547	8,109	11,931	
g10	13,849	32,364	20,954	46,230	38,609	77,123	
g12	450	450	1,000	1,000	1,200	1,200	
g18	5,674	13,206	7,554	18,240	34,839	57,641	
g24	1,584	2,454	1,867	2,948	2,143	3,559	
ε -DE		Function eval.					
Prob.	Best		Median		Worst		
	#func.	#const.	#func.	#const.	#func.	#const.	
g01	18,594	18,594	19,502	19,502	19,971	19,971	
g02	108,303	108,303	114,347	114,347	129,255	129,255	
g04	12,771	12,771	13,719	13,719	14,466	14,466	
g06	5,037	5,037	5,733	5,733	6,243	6,243	
g07	60,873	60,873	67,946	67,946	75,569	75,569	
g08	621	621	881	881	1,173	1,173	
g09	19,234	19,234	21,080	21,080	21,987	21,987	
g10	87,848	87,848	92,807	92,807	107,794	107,794	
g12	2,901	2,901	4,269	4,269	5,620	5,620	
g18	46,856	46,856	57,910	57,910	60,108	60,108	
g24	1,959	1,959	2,451	2,451	2,739	2,739	

The proposed algorithm is also compared against other adaptive DEs (i.e. JDE and SaDE). Table 5 shows the comparison among three different algorithms. Here, the proposed algorithm also shows better result when compared with COPSO, JDE and SaDE for most of the problems. Table 6 shows a one to one comparison with ε -DE including the obtained results. Successful run is shown within 0.1% of the best-known optimum and is feasible.

4.4 Performance Comparison

The same set of problems are used to compare the performance of constraint sequencing. The proposed algorithm is applied with its two variants i.e. with and without constraint sequencing (full evaluation policy). One can see from Fig. 3, that the mean of total number of function evaluations (i.e. summation of number of constraints function evaluations and the number of objective function evaluations) is less when constraint sequencing is used.

Table 5. Comparison of function evaluations is used by the proposed method, DEACS with other best known algorithms with an accuracy level of $(f(x) - f^*(x)) \leq 0.0001$ and feasible solution

COPSO Function eval.						
Prob.	Best		Median		Worst	
	#func.	#const.	#func.	#const.	#func.	#const.
g01	80,776	80,776	90,343	90,343	96,669	96,669
g02	87,419	87,419	93,359	93,359	99,654	99,654
g04	3,147	3,147	103,308	103,308	110,915	110,915
g06	95,944	95,944	109,765	109,765	130,293	130,293
g07	114,709	114,709	138,767	138,767	208,751	208,751
g08	2,270	2,270	4,282	4,282	5,433	5,433
g09	94,593	94,593	103,857	103,857	119,718	119,718
g10	109,243	109,243	135,735	135,735	193,426	193,426
g12	482	482	6,158	6,158	9,928	9,928
g18	97,157	97,157	107,690	107,690	124,217	124,217
g24	11,081	11,081	18,278	18,278	633,378	633,378
JDE Function eval.						
Prob.	Best		Median		Worst	
	#func.	#const.	#func.	#const.	#func.	#const.
g01	46,559	46,559	50,354	50,354	56,968	56,968
g02	101,201	101,201	138,102	138,102	173,964	173,964
g04	38,288	38,288	40,958	40,958	42,880	42,880
g06	26,830	26,830	29,844	29,844	31,299	31,299
g07	114,899	114,899	126,637	126,637	141,847	141,847
g08	1,567	1,567	3,564	3,564	4,485	4,485
g09	49,118	49,118	55,515	55,515	58,230	58,230
g10	139,095	139,095	144,247	144,247	165,498	165,498
g12	1,820	1,820	6,684	6,684	9,693	9,693
g18	91,049	91,049	101,076	101,076	142,674	142,674
g24	7,587	7,587	10,354	10,354	11,550	11,550
SaDE Function eval.						
Prob.	Best		Median		Worst	
	#func.	#const.	#func.	#const.	#func.	#const.
g01	25,115	25,115	25,115	25,115	25,115	25,115
g02	76,915	76,915	128,970	128,970	—	—
g04	25,107	25,107	25,107	25,107	25,113	25,113
g06	12,546	12,546	14,404	14,404	18,347	18,347
g07	25,195	25,195	101,240	101,240	422,860	422,860
g08	782	782	1,272	1,272	1,775	1,775
g09	12,960	12,960	16,787	16,787	33,166	33,166
g10	26,000	26,000	52,000	52,000	153,000	153,000
g12	463	463	1717	1717	2,576	2,576
g18	26,000	26,000	26,000	26,000	—	—
g24	4,280	4,280	4,843	4,843	5,657	5,657

Table 6. Comparison of function evaluations is used by the proposed method, DEACS with an well-known algorithm ε -DE. The results are achieved with an accuracy level of $(f(x) - f^*(x)) \leq 0.0001$ and feasible solution

Proposed method, DEACS									
Prob.	Optimum	f^*	Function eval.						Succ.
			Best	Median	Worst	#func.	#const.	#func.	
g01	-15.00000	-15	-15	-15	-15	6,847	10,537	11,699	100%
g02	-0.803619	-0.803619	-0.803619	-0.803619	-0.778310	49,176	71,438	16,787	100%
g04	-30665.538671	-30665.538671	-30665.538671	-30665.538671	-30665.538671	3,691	4,857	4,234	100%
g06	-6961.813876	-6961.813876	-6961.813876	-6961.813876	-6961.813876	2,426	6,057	2,626	100%
g07	24.306209	24.306209	24.306209	24.306209	24.306209	12,652	26,790	17,891	100%
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	138	900	557	100%
g09	680.630057	680.630057	680.630057	680.630057	680.630057	6,608	9,291	7,418	100%
g10	7049.248021	7049.248021	7049.248021	7049.248021	7049.248021	13,849	32,364	20,954	100%
g12	-1	-1	-1	-1	-1	450	450	1,000	100%
g18	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	5,674	13,206	7,554	100%
g24	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013	1,584	2,454	1,867	100%
ε -DE									
g01	-15.00000	-15	-15	-15	-15	6,847	10,537	11,699	100%
g02	-0.803619	-0.803619	-0.803619	-0.803619	-0.778310	49,176	71,438	16,787	100%
g04	-30665.538671	-30665.538671	-30665.538671	-30665.538671	-30665.538671	3,691	4,857	4,234	100%
g06	-6961.813876	-6961.813876	-6961.813876	-6961.813876	-6961.813876	2,426	6,057	2,626	100%
g07	24.306209	24.306209	24.306209	24.306209	24.306209	12,652	26,790	17,891	100%
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	138	900	557	100%
g09	680.630057	680.630057	680.630057	680.630057	680.630057	6,608	9,291	7,418	100%
g10	7049.248021	7049.248021	7049.248021	7049.248021	7049.248021	13,849	32,364	20,954	100%
g12	-1	-1	-1	-1	-1	450	450	1,000	100%
g18	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	5,674	13,206	7,554	100%
g24	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013	1,584	2,454	1,867	100%

Function eval.									
Prob.	Optimum	f^*	Function eval.						Succ.
			Best	Median	Worst	#func.	#const.	#func.	
g01	-15.00000	-15	-15	-15	-15	18,594	18,594	19,502	100%
g02	-0.803619	-0.803619	-0.803619	-0.803619	-0.803619	108,303	108,303	114,347	100%
g04	-30665.538671	-30665.538671	-30665.538671	-30665.538671	-30665.538671	12,771	12,771	13,719	100%
g06	-6961.813876	-6961.813876	-6961.813876	-6961.813876	-6961.813876	5,037	5,037	5,733	100%
g07	24.306209	24.306209	24.306209	24.306209	24.306209	60,873	60,873	67,946	100%
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	621	621	881	100%
g09	680.630057	680.630057	680.630057	680.630057	680.630057	19,234	19,234	21,080	100%
g10	7049.248021	7049.248021	7049.248021	7049.248021	7049.248021	87,848	87,848	92,807	100%
g12	-1	-1	-1	-1	-1	2,901	2,901	4,269	100%
g18	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	46,856	46,856	57,910	100%
g24	-5.508013	-5.508013	-5.508013	-5.508013	-5.508013	1,959	1,959	2,451	100%

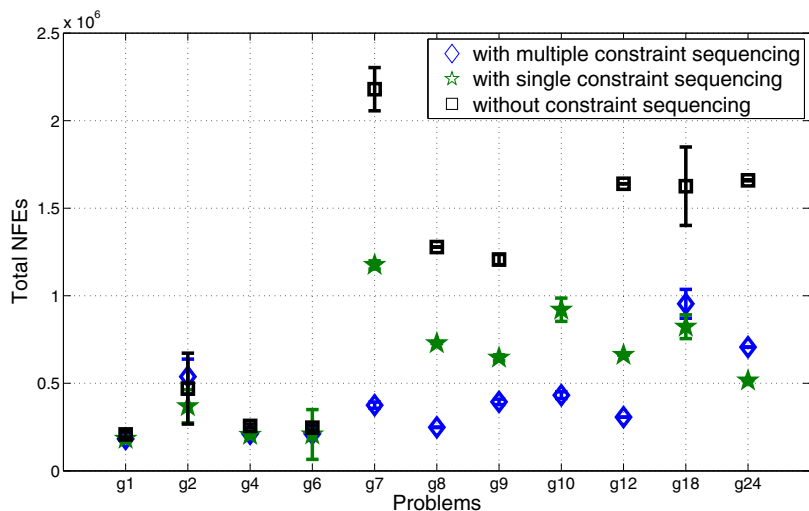


Fig. 3. A comparison of total number of function evaluations without constraint sequencing , with a single constraint sequencing and multiple constraint sequencing

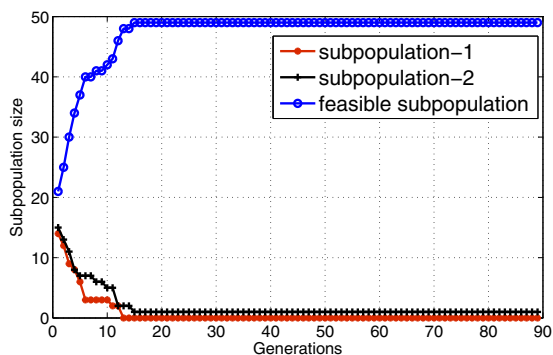


Fig.4. An example of population migration from infeasible subpopulations to feasible subpopulation

In an another experiment, the comparison is made between the use of a single constraint sequence and multiple constraint sequences following a ring topology. For single constraint sequence, a predefined sequence is assigned to all the subpopulations whereas for the later, different constraint sequences are assigned to each subpopulation. The results in figure 4 illustrate the benefits of multiple sequencing across the subpopulations.

5 Conclusion

In this paper, a novel constraint handling scheme has been introduced within a framework of a self adaptive DE using a scheme of constraint sequencing. The paper presents

the idea of constraint sequencing and using a simple example illustrates why such a scheme is likely to improve the efficiency of the search. The study also provides answers to important questions, i.e. when is such a scheme beneficial. The performance of the algorithm is subsequently assessed on 11 well known constrained single objective optimization benchmarks. The results clearly indicate that the approach is efficient when compared with the state of the art algorithms and their underlying schemes for constraint handling. The scheme is generic and can be embedded within other forms of population based stochastic optimization algorithms. The approach is likely to provide significant computational benefits for problems when each of such constraint or objective function evaluation is computationally expensive and can be computed independently. The approach is also efficient for problems where the CV function is highly multi-modal where the proposed scheme is less likely to be trapped in local optima. Since the efficiency of the algorithm stems from handling constraints, the approach is likely to be ineffective for some problems with high feasibility ratio as observed for problems g2 and g24.

References

1. Brest, J.: Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: 2006 IEEE Congress on Evolutionary Computation (CEC 2006), July 16-21, pp. 215–222. IEEE Press, Vancouver (2006)
2. Coello, C.A.C.: 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* 191(11-12), 1245–1287 (2002)
3. Coit, D.W., Smith, A.E.: Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering* 30(4), 895–904 (1996)
4. FitzGerald, A., O'Donoghue, D.P.: Genetic Repair for Optimization under Constraints Inspired by *Arabidopsis Thaliana*. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 399–408. Springer, Heidelberg (2008)
5. Hinterding, R., Michalewicz, Z.: Your brains and my beauty: Parent matching for constrained optimisation. In: Proceedings of the 5th International Conference on Evolutionary Computation, Anchorage, Alaska, pp. 810–815 (May 1998)
6. Huang, V.L., Qin, A.K., Suganthan, P.N.: Self-adaptative differential evolution algorithm for constrained real-parameter optimization. In: 2006 IEEE Congress on Evolutionary Computation (CEC 2006), pp. 324–331. IEEE Press, Vancouver (2006)
7. Koziel, S., Michalewicz, Z.: A Decoder-Based Evolutionary Algorithm for Constrained Parameter Optimization Problems. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 231–240. Springer, Heidelberg (1998)
8. Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C.A., Deb, K.: Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical Report, Nanyang Technological University, Singapore (December 2005), <http://www.ntu.edu.sg/home/EPNsugan/>
9. Liu, J., Fan, Z., Goodman, E.: Srde: An improved differential evolution based on stochastic ranking. In: 2009 ACM SIGEVO Summit on Genetic and Evolutionary Computation (GEC 2009), June 12-14, pp. 345–352. ACM Press, Shanghai (2009)
10. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation* 4(1), 1–32 (1996)

11. Powell, D., Skolnick, M.M.: Using genetic algorithms in engineering design optimization with non-linear constraints. In: Forrest, S. (ed.) *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 1993)*, pp. 424–431. Morgan Kaufmann Publishers, San Mateo (1993)
12. Qu, B.Y., Suganthan, P.N.: Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods. *Engineering Optimization* 43(4), 403–416 (2011)
13. Schoenauer, M., Xanthakis, S.: Constrained GA optimization. In: Forrest, S. (ed.) *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA 1993)*, pp. 573–580. Morgan Kauffman Publishers, San Mateo (1993)
14. Takahama, T., Sakai, S.: Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In: *2010 IEEE Congress on Evolutionary Computation (CEC 2010)*, July 18–23, pp. 1680–1688. IEEE Press, Barcelona (2010)
15. Zavala, A.E.M., Aguirre, A.H., Diharce, E.R.V.: Continuous Constrained Optimization with Dynamic Tolerance Using the COPSO Algorithm. In: Mezura-Montes, E. (ed.) *Constraint-Handling in Evolutionary Optimization*. SCI, vol. 198, pp. 1–23. Springer, Heidelberg (2009)