11th World Congress on Structural and Multidisciplinary Optimization
7th - 12th, June 2015, Sydney Australia

A Novel Constraint Handling Strategy for Expensive Optimization Problems

Kalyan Shankar Bhattacharjee[1], Tapabrata Ray[2]

[1] University of New South Wales, Canberra, ACT 2610, Australia, Kalyan.Bhattacharjee@.adfa.edu.au
[2] University of New South Wales, Canberra, ACT 2610, Australia, T.Ray@.adfa.edu.au

1. Abstract
Constraints are inherently present in any real world problem. In the context of multidisciplinary design optimization problems, such constraints arise out of physical laws, statutory requirements, user preferences etc. and are often computed using computationally expensive analysis e.g. FEM, CFD, CEM etc. While population based stochastic optimization algorithms are a preferred choice for the solution of such class of problems (often with the aid of approximations), they typically adopt a full evaluation policy i.e. all constraints and objective functions for all solutions are evaluated. Recent studies have highlighted the possibility of selected constraint evaluation (i.e. a subset of relevant constraints are only evaluated), although learning the sequence (or the subset) of constraints is far from trivial. In this paper, we introduce an approach for selective evaluation based on Support Vector Machine (SVM) models, wherein promising solutions are identified and evaluated based on the trade-off between need to learn and cost to learn. The performance of the proposed scheme is compared with other state-of-the-art constraint handling methods using a set of well-studied engineering design optimization problems. The aspect of selective evaluation has rarely been investigated in literature and the results clearly indicate the benefits selective evaluation which is of immense value in the context of computationally expensive optimization problems.
2. Keywords: Constraint Handling, Classifiers, Selective evaluation.

3. Introduction
Most real world optimization problems involve constraints and feasible solutions need to satisfy them. A generic constrained optimization problem can be expressed as:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \mathrm{f}(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \geq 0,\ i = 1,2,\ldots,q \\ & h_j(\mathbf{x}) = 0,\ j = 1,2,\ldots,r \end{aligned}$$

where q represents the number of inequality constraints and r denotes the number of equality constraints. The equality constraints are replaced by a pair of inequalities. The vector $\mathbf{x} = [x_1\ x_2\ \ldots x_n]$ denotes a solution represented using n design variables.

It is well known that the performance of all stochastic optimization algorithms are affected by the underlying mechanism of constraint handling. Constraint handling is an active area of research and existing methods can be classified into three categories i.e. (a) full evaluation policy with *feasibility first* principle: all constraints and objectives are evaluated for all solutions and feasible solutions are preferred over infeasible solutions (objective function value of infeasible solutions are essentially useless information) e.g. Non-dominated Sorting Genetic Algorithm(NSGA-II) [1] (b) full evaluation policy with marginally infeasible solutions preserved or stochastically preferred e.g. Infeasibility Driven Evolutionary Algorithm(IDEA) [5], Stochastic Ranking(SRES) [6], Epsilon Differential Evolution(eps-DEA) [7] (objective function values of infeasible solutions are used in ranking solutions) and (c) partial evaluation strategy e.g. *evaluate till you violate*[8] where constraints are evaluated in a sequence (DEACS) until a violation is encountered. In the context of computationally expensive optimization problems, partial evaluation policy offers the potential to reduce the number of function evaluations.

The evaluation cost can be further reduced if one can (a) screen potentially promising offsprings and (b) evaluate relevant constraints. In this paper, support vector machines (SVM) [9, 10] are used to identify promising offsprings and the relevant set of constraints. In the proposed approach, a SVM classifier is used to estimate the class label and the associated confidence about the quality of a solution. The use of a SVM classifier to identify promising solutions appeared in [11, 12]. In this study, we have used SVM ranking [10] models to predict the rank of a partially evaluated solution.

4. Proposed Approach
We adopt a generational model. The initial population is fully evaluated i.e. objective and all constraints for all the individuals are evaluated and the information is stored in an *Archive*. The first set of N parents are identified using roulette wheel selection, while the second set of N parents are identified using a random selection. A binary tournament between these N pairs of solutions result in N participating parents and offspring solutions are created

using simulated binary crossover (SBX) and polynomial mutation (PM). Offspring solutions are screened using a two-class SVM classifier, trained using the data from the *Archive* with its inputs being the variables of the optimization problem (**x**) and the output being the final rank of the solutions. Offspring solutions predicted with a class label of 1 are considered as potential solutions.

For any given potential solution, the probability of satisfying its i^{th} constraint is given by $Feasibility\_Index_i$. A value of 1 would indicate that it would satisfy the i^{th} constraint and 0 otherwise. Similarly, for a solution under consideration, $Rank\_Index_i$ is computed for each constraint. The term $Rank\_Index_i$ reflects the confidence that this solution is among the top 50% in a list based on the i^{th} constraint. To construct a list based on the i^{th} constraint, solutions are ordered based on the i^{th} constraint values i.e. feasible and furthest from the i^{th} constraint boundary at the top and infeasible and the furthest from the i^{th} constraint boundary at the bottom. In order to capture the local behavior, for each potential offspring solution, the classifiers (one for each constraint) are trained using k closest (in variable space) neighbors from the *Archive*. The inputs to these classifiers are the variable values and the outputs are the corresponding ranks based on that particular constraint under consideration. For a potential offspring, the constraint associated with least *Feasibility_Index* and least *Rank_Index* will be evaluated first. However, in the situation where *Feasibility_index* of a solution is same in all its constraints, the sequence of evaluation is based on the following rule: (a) if all neighboring k solutions have all the constraints violated i.e. ($Feasibility\_Index = 0$), the constraint having least *Rank_Index* is evaluated first and (b) if all neighboring k solutions have all the constraints satisfied i.e. ($Feasibility\_Index = 1$), the objective function for this solution is only evaluated since it is most likely a feasible solution. In the next step, SVM ranking model is utilized to predict rank of the potential offspring in all other constraints, where it has not been evaluated. In this ranking scheme, a regression model is created using actual ranks of all the solutions from the *Archive* based on the evaluated constraint as inputs and ranks based on other constraints or final rank as outputs. Hence, for any potential offspring solution, ranks based on all other constraints and its final rank in the population can be predicted. Rank prediction and insertion in the population are executed taking one offspring at a time instead of taking the whole set. Therefore insertion of one offspring might affect the position of the previous offsprings.

It is important for any learning based scheme to focus the region of interest and progressively improve its quality of prediction. To assist this, top solutions in a population undergo full evaluation (i.e. all constraints and objective function values for these solutions are evaluated). Please note that the final ranking of the population is based on *feasibility first* principle. The pseudo-code of the proposed approach is presented below: Classifier Guided Constraint Selection Mechanism (CGCSM).

5. Numerical Experiments

The success of a constraint handling strategy can be assessed from two different angles (a) ability of the approach to deliver the first feasible solution with minimal computational cost and (b) quality of the solution delivered for a fixed computational budget. While the second metric is largely used within the evolutionary computation community, such a metric does not solely assess the performance of constraint handling schemes. We objectively evaluate the performance of CGCSM and compare it with IDEA, NSGA-II, DEACS and SRES using 5 well studied benchmark engineering design problems: Belleville Spring [13], Helical Spring [13], Speed Reducer [14], and Step Cone Pulley [15]. The properties of the objective functions, nature of constraints, number of active constraints and the percentage of the feasible space are listed in Table 1.

Table 1: Properties of the Problems (Maximization Problem: max, Minimization Problem: min)

Problem	**n**	**Obj**	ρ **%**	*LI*	*NE*	*NI*	*NA*
Belleville Spring(min)	4	Quadratic	0.2595	7	0	0	-
Helical Spring(min)	3	Polynomial	0.0316	9	0	0	-
Speed Reducer(min)	7	Polynomial	0.0962	11	0	0	-
Step Cone Pulley(min)	5	Polynomial	0.0000	0	3	8	-

where n: number of variable, Obj: Objective function type, ρ: Percentage ratio of feasible space over entire search space, LI: number of linear inequalities, NE: number of equalities, NI: number of nonlinear inequalities and NA: number of active constraints.

The relative sizes of the feasible region (feasibility ratio) is based on random sampling of 1,000,000 random points. The results obtained using the proposed algorithm CGCSM are compared with those obtained using infeasibility driven evolutionary algorithm (IDEA) [5], non-dominated sorting genetic algorithm (NSGA-II) [1], Constraint sequencing (DEACS) [8], and Stochastic ranking (SRES) [6]. A one-to-one comparison of CGCSM with IDEA, NSGA-II, DEACS and SRES would offer insights on the actual utility of the classifier. Results, presented in Table 2, indicate the cost of evaluation till first feasible is obtained and Table 3 indicates the quality of the solution after 1000 function evaluations equivalent to evaluating 1000 solutions during the course of optimization using a full evaluation policy). As an example, for a problem involving 7 constraints, full evaluation of 1000 solutions would mean an evaluation budget of 8000. Each objective function evaluation or a constraint evaluation

Algorithm 1 CGCSM

SET: FE_{max}\{Maximum evaluation budget\}, N\{Population size\}, S_t\{Confidence associated with SVM classifier (exponentially increases from 0 to 0.8 over FE_{max})\}, *Popbin*\{Repository of ordered solutions evaluated so far (includes partial and fully evaluated solutions)\}, *Archive*\{Repository of all fully evaluated solutions\}

Initialize the population of N individuals using latin hypercube sampling
Evaluate $Pop_{1:N,g_{1:q+2r},f}$ and order them according to their final ranks (ordered based on feasibility first)
$Popbin = Pop$
Update FE, Update *Archive*
Update S_t
while ($FE \leq FE_{max}$) **do**
    Generate offspring solutions using BT, SBX and PM from $Popbin_{1:N}$
    Construct a binary SVM classifier: Top 100(1-S_t) percent solutions of the *Archive* is assigned a class label of 1
    Offspring solutions unique w.r.t *Archive* and with a predicted class label of 1 constitutes the set of C eligible offsprings ($Childpop$)
    for i = 1:C **do**
        For every member of $Childpop$, calculate the $Feasibility\_Index_{i,1:q+2r}$ and $Rank\_Index_{i,1:q+2r}$ based on its k neighbors from the *Archive*
        if $Feasibility\_Index_{i,1:q+2r} = 1$ **then**
            Evaluate $Childpop_{i,f}$
        else if $Feasibility\_Index_{i,1:q+2r} = 0$ **then**
            $[val, index]$ = min($Rank\_Index_{i,g_{1:q+2r}}$); g_{index} is the constraint to be evaluated
        else
            $[val1, list1]$ = sort($Feasibility\_Index_{i,g_{1:q+2r}}$); $[val2, list2]$ = sort($Rank\_Index_{i,g_{1:q+2r}}$)
            Find $index$, where $list1$ has a preference over $list2$ and evaluate $Childpop_{i,g_{index}}$
        end if
        Update FE
        Construct SVM ranking model with inputs being rank of solutions in g_{index} and outputs being ranks in other constraints and final rank from *Archive*
        Predict rank of $Childpop_i$ in other constraints and its final rank based on the above SVM model
        if (Final rank of $Childpop_i \leq (1 - S_t)|Popbin|$) **then**
            Evaluate $Childpop_{i,g_{1:q+2r}}$ and place it in *Popbin* based on its actual final rank
            Update FE, Update *Archive*
        else
            Place $Childpop_i$ in *Popbin* according to its predicted final rank
        end if
    end for
end while

*FE denotes the evaluation cost i.e. 1 unit for each objective and 1 unit for each constraint evaluated

incurs a cost of 1 unit. Please take note that each equality constraints are imposed as two inequalities using all the algorithms except SRES (kept same as original formulation), however total evaluation budget is kept same for all the algorithms.

6. Results and discussion

The following parameters were used in this study: population size: 40; total evaluation budget is 1000 times the total number of constraints and objective for the problem; crossover probability: 0.9; mutation probability: 0.1; distribution index for crossover: 20; distribution index of mutation: 30; confidence in the classifier varied exponentially from 0 to 0.8 and the number of neighbors (k) was set to 12. We refer the readers to [9, 10] for the details on support vector machine classifiers. In our study, the standard SVM classifier of MATLAB toolbox was used with a Gaussian Radial Basis Function kernel with default settings and Karush-Kuhn-Tucker (KKT) violation level set as 0.05. Also for all the algorithms, same initial population has been used for a fair comparison.

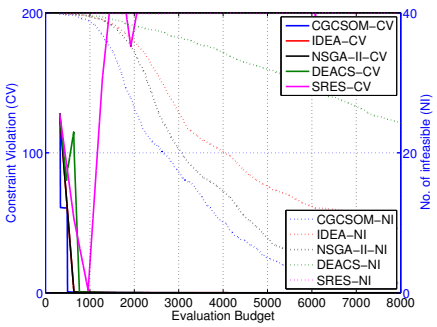
6.1. Performance on Problems

Convergence plots of mean sum of constraint violations (CV) and average number of infeasible individuals (NI) over the evaluation budget are shown in Figure 1a, Figure 2a, and Figure 3a for Belleville Spring, Speed Reducer, and Step Cone Pulley respectively. While, for the same problems the convergence plots of the mean objective function value (Obj) and average number of feasible individuals (NF) versus evaluation budget are shown in Figure 1b, Figure 2b, Figure 3b. In the context of partial evaluation policy (CGCSM and DEACS), solutions are evaluated offline to obtain the sum of constraint violation and objective function value for best individual in each generation. Hence, a fill up cost of 1 unit was assumed (i.e. to account for the case when a partially filled population is delivered).

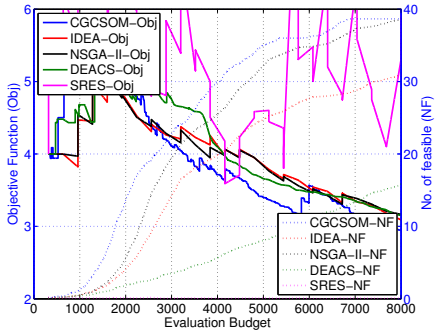
The best, mean, worst, median and standard deviation measures of the best solution across 30 independent runs obtained using *CGCSM, IDEA, NSGA-II, DEACS and SRES* are presented in Table 3.

The observations from the results can be summarized as follows:

(a) In Belleville Spring problem, Figure 1 indicates that CGCSM has the highest convergence rate when compared with other algorithms both in the contexts of mean sum of constraint violations and mean objective function value.

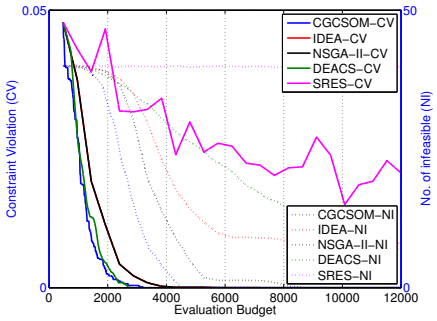


(a) Mean sum of CV and no. of infeasible individuals versus cost: Belleville Spring

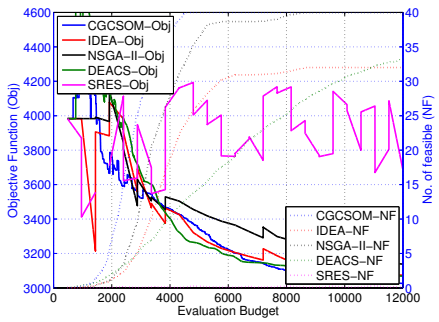


(b) Mean Obj and no. of feasible individuals versus cost: Belleville Spring

Figure 1: Convergence plot: Belleville Spring

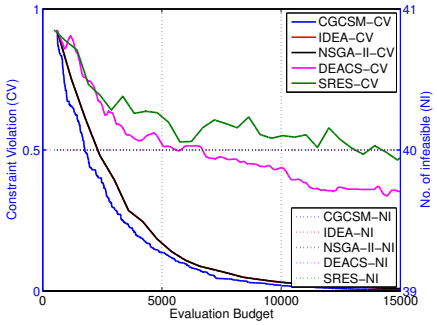


(a) Mean sum of CV and no. of infeasible individuals versus cost: Speed Reducer

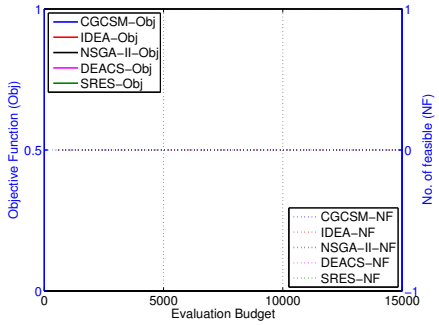


(b) Mean Obj and no. of feasible individuals versus cost: Speed Reducer

Figure 2: Convergence plot: Speed Reducer



(a) Mean sum of CV and no. of infeasible individuals versus cost: Step Cone Pulley



(b) Mean Obj and no. of feasible individuals versus cost: Step Cone Pulley

Figure 3: Convergence plot: Step Cone Pulley

This problem has relatively larger feasibility ration when compared to others. One can also observe from Table 3 that CGCSM outperforms other algorithms in the terms of the median optimal solution delivered. However, SRES is the best in terms of its ability to deliver the first feasible solution (Table 2 and Figures 1b). (b) CGCSM is able to deliver its first feasible solution at minimum cost for the Helical Spring design problem (Table 2). The design space for this problem is highly constrained. However quality of median optimal solution delivered by DEACS is best closely followed by CGCSM (Table 3). (c) Speed Reducer is also a highly constrained design problem which has highest number of linear inequalities among the problems discussed. The spread of the solutions is within 0.6% of the average objective value for DEACS, while it is 1.1% for CGCSM and much higher for others. This indicates better quality of convergence for the algorithms having partial evaluation policy in this problem. CGCSM performs better in terms of convergence (both mean CV and mean objective function value) than others. (d) Step Cone Pulley is an interesting problem as it has 8 non-linear inequality and 3 equality constraints which makes the problem highly constrained and difficult to solve. From Figure 3b, it can be observed that none of the algorithms could deliver feasible solutions within the fixed computational budget. However Figure 3a indicates better convergence of CGCSM and the final median violation values is also the lowest for CGCSM.

Table 2: Statistics of evaluation cost till first feasible for Problems

Problems	Algorithms	Best	Mean	Median	Worst	Std	Success
Belleville Spring	CGSCM	160.0000	1997.3793	1616.0000	5994.0000	1609.0422	29.0000
	IDEA	160.0000	2280.0000	1528.0000	6496.0000	1767.8565	29.0000
	NSGA-II	160.0000	2280.0000	1528.0000	6496.0000	1767.8565	29.0000
	DEACS	160.0000	1367.5000	1228.0000	3201.0000	792.5307	30.0000
	SRES	160.0000	1967.3600	1048.0000	7664.0000	2130.4546	25.0000
Helical Spring	CGSCM	920.0000	2314.8571	2130.0000	4780.0000	991.7325	21.0000
	IDEA	1450.0000	3317.0833	2730.0000	7610.0000	1583.5691	24.0000
	NSGA-II	1450.0000	3317.0833	2730.0000	7610.0000	1583.5691	24.0000
	DEACS	927.0000	3027.7000	2974.0000	6305.0000	1250.3888	30.0000
	SRES	530.0000	4978.4615	4740.0000	8760.0000	2762.9267	13.0000
Speed Reducer	CGSCM	336.0000	1622.0000	1428.0000	3204.0000	697.0129	30.0000
	IDEA	336.0000	2294.4000	2100.0000	6816.0000	1267.1713	30.0000
	NSGA-II	336.0000	2294.4000	2100.0000	6816.0000	1267.1713	30.0000
	DEACS	336.0000	1246.0333	1224.0000	2550.0000	601.8644	30.0000
	SRES	336.0000	4971.8571	5106.0000	9984.0000	2692.2567	28.0000
Step Cone Pulley	CGSCM	NaN	NaN	NaN	NaN	NaN	0.0000
	IDEA	NaN	NaN	NaN	NaN	NaN	0.0000
	NSGA-II	NaN	NaN	NaN	NaN	NaN	0.0000
	DEACS	NaN	NaN	NaN	NaN	NaN	0.0000
	SRES	NaN	NaN	NaN	NaN	NaN	0.0000

Table 3: Statistics for Problems

Problems	Algorithms	Feasibility	Best	Mean	Median	Worst	Std	Success
Belleville Spring	CGSCM	Feasible	2.2507	2.9440	2.6626	6.7106	0.9822	29.0000
		Infeasible	0.0000	0.0007	0.0000	0.0205	0.0037	1.0000
	IDEA	Feasible	2.4369	3.0897	2.8994	4.9255	0.5972	29.0000
		Infeasible	0.0000	0.0006	0.0000	0.0167	0.0030	1.0000
	NSGA-II	Feasible	2.2319	3.1649	2.9879	6.4377	0.8304	29.0000
		Infeasible	0.0000	0.0006	0.0000	0.0167	0.0030	1.0000
	DEACS	Feasible	2.4165	3.1518	3.1642	4.0260	0.4003	30.0000
		Infeasible	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	SRES	Feasible	5.9192	5.9192	5.9192	5.9192	0.0000	1.0000
		Infeasible	0.0000	232.2841	1.0874	4625.5502	853.9785	29.0000
Helical Spring	CGSCM	Feasible	2.8223	4.0102	3.1356	6.9907	1.4696	21.0000
		Infeasible	0.0000	0.0659	0.0000	0.7494	0.1596	9.0000
	IDEA	Feasible	2.7684	3.9751	3.3809	7.1769	1.2899	24.0000
		Infeasible	0.0000	0.0438	0.0000	0.5380	0.1218	6.0000
	NSGA-II	Feasible	2.8068	4.1526	3.4674	7.3881	1.5122	24.0000
		Infeasible	0.0000	0.0438	0.0000	0.5380	0.1218	6.0000
	DEACS	Feasible	2.7434	2.9800	2.9937	3.1405	0.1057	30.0000
		Infeasible	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	SRES	Feasible	3.7105	3.7105	3.7105	3.7105	0.0000	1.0000
		Infeasible	0.0000	0.5466	0.5863	1.4689	0.3452	29.0000
Speed Reducer	CGSCM	Feasible	3000.0118	3024.7515	3014.0820	3182.4136	33.0295	30.0000
		Infeasible	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	IDEA	Feasible	3005.2093	3073.7129	3033.5242	3578.8939	116.8753	30.0000
		Infeasible	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	NSGA-II	Feasible	3007.6698	3138.1983	3042.9740	4152.7143	249.6745	30.0000
		Infeasible	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	DEACS	Feasible	3025.9020	3068.4133	3062.5951	3121.7496	18.5588	30.0000
		Infeasible	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	SRES	Feasible	3259.1542	3684.9077	3684.9077	4110.6611	425.7534	2.0000
		Infeasible	0.0000	0.0207	0.0184	0.0636	0.0175	28.0000
Step Cone Pulley	CGSCM	Feasible	NaN	NaN	NaN	NaN	NaN	0.0000
		Infeasible	0.0005	0.0046	0.0040	0.0118	0.0028	30.0000
	IDEA	Feasible	NaN	NaN	NaN	NaN	NaN	0.0000
		Infeasible	0.0018	0.0083	0.0066	0.0304	0.0065	30.0000
	NSGA-II	Feasible	NaN	NaN	NaN	NaN	NaN	0.0000
		Infeasible	0.0018	0.0083	0.0066	0.0304	0.0065	30.0000
	DEACS	Feasible	NaN	NaN	NaN	NaN	NaN	0.0000
		Infeasible	0.0811	0.3527	0.3624	0.7589	0.1719	30.0000
	SRES	Feasible	NaN	NaN	NaN	NaN	NaN	0.0000
		Infeasible	0.2134	0.5110	0.4698	1.0447	0.1959	30.0000

Since only promising solutions are evaluated, use of classifiers would reduce the computational cost. However, the classifiers need to learn and their assessment needs to be reliable. The process of learning requires information from the evaluated solutions. Although use of poorly trained classifiers would save computational cost, the search outcome may not be satisfactory. To achieve this balance, the confidence associated with the classifier is varied from 0 to 0.8 exponentially during the course of search. In case of high confidence associated with the classifier, very few solutions would be evaluated and in turn the classifier would not have the opportunity to learn from diverse solutions. These observations seems to favor CGCSM for problems having highly constrained search space. In the context of evaluation cost associated with first feasible identification, performances of CGCSM and DEACS are nearly similar. However, in terms of convergence (both mean CV and mean objective function value) CGCSM performs better in 3 out of 4 problems.

7. Summary and Conclusions

Real life optimization problems often involve objective and constraint functions that are evaluated using computationally expensive numerical simulations e.g. computational fluid dynamics (CFD), finite element methods (FEM) etc. In order to solve such classes of problems, surrogate assisted optimization (SAO) methods are typically used, wherein computationally cheap and less accurate surrogates/approximation models of objectives/constraints are used during the course of search. In this paper, we explore an alternative path i.e. one where promising solutions are identified using support vector machine (SVM) based models. The key difference being, SVM models are used to identify promising solutions without explicitly attempting to approximate objective and constraint functions. Furthermore, for every promising solution, the approach identifies the constraints that are most likely to be violated and evaluates them first. In the event the constraints and objectives are evaluated using independent computationally expensive analysis (e.g. multi-disciplinary optimization), such an approach would only evaluate relevant constraints and/or objectives that are necessary to ascertain the rank of the solutions. The search behavior of CGSCM is compared with the following: NSGA-II (algorithm adopts a full evaluation policy), IDEA(algorithm maintains selected infeasible solutions), DEACS (algorithm evaluates selected set of constraints) and finally SRES (algorithm stochastically prefers infeasible solutions). The performance of the algorithm is further objectively assessed using a number of constrained engineering design optimization problems with limited computational budget. The rate of convergence of CGCSM is better for most of the problems and the final set of results are clearly better on all problems studied in this paper. We hope that this study would prompt design of efficient algorithms that selectively evaluate solutions and in particular selected set of constraints on the fly i.e. based on the trade-off between *need to learn/evaluate* and *cost to learn*.

8. 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*, 6 (2), 182-197, 2002.

[2] D. W. Coit, and A. E. Smith, Penalty guided genetic search for reliability design optimization, *Computers and Industrial Engineering*, Special Issue on Genetic Algorithms, 30 (4), 895–904, 1996.

[3] S. Koziel and Z. Michalewicz, A decoder-based evolutionary algorithm for constrained parameter optimization problems, *Proceedings of the 5th Parallel Problem Solving from Nature – PPSN V*, Lecture Notes in Computer Science, T. Bäck, A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, (Eds.), Springer, Heidelberg, Germany, 5199, 231-240, 1998.

[4] A. Fitzgerald, and D. P. O'Donoghue, Genetic repair for optimization under constraints inspired by arabidopsis thaliana, *Parallel Problem Solving from Nature–PPSN X*, Lecture Notes in Computer Science, G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, (Eds.), Springer, Dortmund, Germany, 5199, 399-408, 2008.

[5] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, Infeasibility driven evolutionary algorithm for constrained optimization, *Constraint-handling in evolutionary optimization*, Springer, 145-165, 2009.

[6] T. P. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization, *IEEE Transactions on Evolutionary Computation*, 4 (3), 284-294, 2000.

[7] T. Takahama and S. Sakai, Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites, *IEEE Congress on Evolutionary Computation (CEC)*, 1-8, 2006.

[8] M. Asafuddoula, T. Ray, and R. Sarker, Evaluate till you violate: A differential evolution algorithm based on partial evaluation of the constraint set, *IEEE Symposium on Differential Evolution (SDE)*, 31-37, 2013.

[9] J. A. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least squares support vector machines*, 4, World Scientific, 2002.

[10] O. Chapelle and S. S. Keerthi, Efficient algorithms for ranking with SVMs, *Information Retrieval*, 13 (3), 201-215, 2010.

[11] I. Loshchilov, M. Schoenauer, and M. Sebag, A mono surrogate for multiobjective optimization, *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 471-478, ACM, 2010.

[12] S. Bandaru, A. Ng, and K. Deb, On the performance of classification algorithms for learning pareto-dominance relations, *IEEE Congress on Evolutionary Computation (CEC)*, 1139-1146, 2014.

[13] J. N. Siddall, *Optimal engineering design: principles and applications*, CRC Press, 1982.

[14] J. Golinski, Optimal synthesis problems solved by means of nonlinear programming and random methods, *Journal of Mechanisms*, 5 (3), 287-309, 1970.

[15] S. S. Rao, and S. S. Rao, Engineering optimization: theory and practice, *John Wiley & Sons*, 2009.