2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Solving Problems With a Mix of Hard and Soft Constraints Using Modified Infeasibility Driven Evolutionary Algorithm (IDEA-M)

Hemant Kumar Singh, Md. Asafuddoula and Tapabrata Ray

***Abstract*—Most optimization problems in the field of engineering design involve constraints. These constraints are often due to statutory requirements (e.g. safety, physical laws, user requirements/functionality) and/or limits imposed on time and resources. Population based stochastic optimization algorithms are a preferred choice for solving design optimization problems due to their ability to deal with nonlinear *black-box* functions. Having a good constraint handling technique embedded within the algorithm is imperative for its good performance. With the final aim of achieving feasible optimum solutions, *feasibility first* techniques, i.e., those which prefer feasible solutions over infeasible, have been commonly used in the past. However, in recent studies more emphasis has been laid on intelligent use of infeasible solutions (instead of their indiscreet rejection) during the course of optimization; particularly because optimum solutions often lie on the constraint boundary. The preservation of *good* infeasible solutions in the population is likely to improve the convergence in constricted or disconnected feasible regions. In addition, it provides a set of marginally infeasible solutions for trade-off considerations. However, in the case of a problem consisting of a mix of *hard* (non-negotiable) and *soft* (negotiable) constraints, such trade-off solutions are practically useful if they violate the *soft* constraints only. In this paper, previously introduced Infeasibility Driven Evolutionary Algorithm (IDEA) is modified to deliver solutions which strictly satisfy the *hard* constraints and offer tradeoff solutions with respect to the *soft* constraints. The performance of the algorithm is demonstrated on three benchmark problems.**

I. INTRODUCTION

In real life, one often encounters problems in which one or more objectives have to be optimized simultaneously, subject to a set of constraints. In recent years, population-based metaheuristic optimization algorithms (such as Evolutionary Algorithms / EAs) have been largely preferred for solving optimization problems, particularly multi-objective problems, as they do not have prerequisites on mathematical properties of objective functions and can provide Pareto-optimal set of solutions in a single run. For constrained optimization problems, these algorithms need to be augmented by constraint handling mechanisms, the quality of which affects the performance of the algorithm significantly. A detailed review of various constraint handling techniques used with EAs is presented in [1], [2], [3].

Some of the most commonly used constraint-handling techniques are listed below.

- **Penalty function-based methods:** Penalty function methods are one of the most commonly adopted forms of constraint handling. In this approach, the fitness of infeasible solutions is degraded using a weighted sum of constraint violations. Variants of the penalty function based approach include static penalty [4], [5], dynamic penalty [6], annealing penalty [7], adaptive penalty [8] and death penalty [9]. Implementations of most of these schemes require additional parameters. The choice of these parameters is often not trivial, and the result of the optimization process is known to be highly sensitive to these parameters.
- **Dominance-based approaches:** "A dominance based" constraint handling technique implies that while performing a Pareto-dominance ranking of solutions, the constraints (or a quantity calculated based on them) are also considered as objectives. Ray *et al.* [10] developed an EA based on the non-dominance of solutions in both the objective and the constraint space. Ho and Shimizu [11] converted the objective function value and the constraint violation into numerical values of the same order of magnitude. Often, a single-objective constraint problem is solved by converting it into a multi-objective problem by adding constraints as objectives. A comparison of performance of various multi-objective evolutionary algorithms (MOEAs) on constrained optimization (single-objective non-linear problems) using concepts of Pareto-dominance can be found in [12]. For multi-objective problems, Vieira *et al.* [13] used constraints as additional objectives. In above approaches, convergence issues may arise if number of constraints is high, as Pareto-dominance sorting does not work well for high number of objectives. There is also a risk of generating solutions with excellent objective function values but poor constraint satisfaction.
- **Maintaining infeasible solutions:** A few researchers have proposed maintaining a proportion of infeasible solutions in the population during the course of evolution. For single-objective optimization, Coello Coello [14] proposed splitting the population into various sub-populations, each of which use either the objective or one of the constraints as the fitness function. Hamida and Schoenauer [15] developed an Adaptive Segregational Algorithm (ASCHEA) in which the proportion of feasible solutions in the population is controlled using an adaptive penalty. This approach used a single penalty coefficient for all constraints and was later extended [16] to incorporate a separate penalty coefficient for each constraint. Hinterding and Michalewicz [17] proposed another approach (CONGA) for constraint handling

Hemant Kumar Singh, Md. Asafuddoula and Tapabrata Ray are with School of Engineering and Information Technology, University of New South Wales Canberra, Australia. email: {h.singh, md.asaf, t.ray}@adfa.edu.au

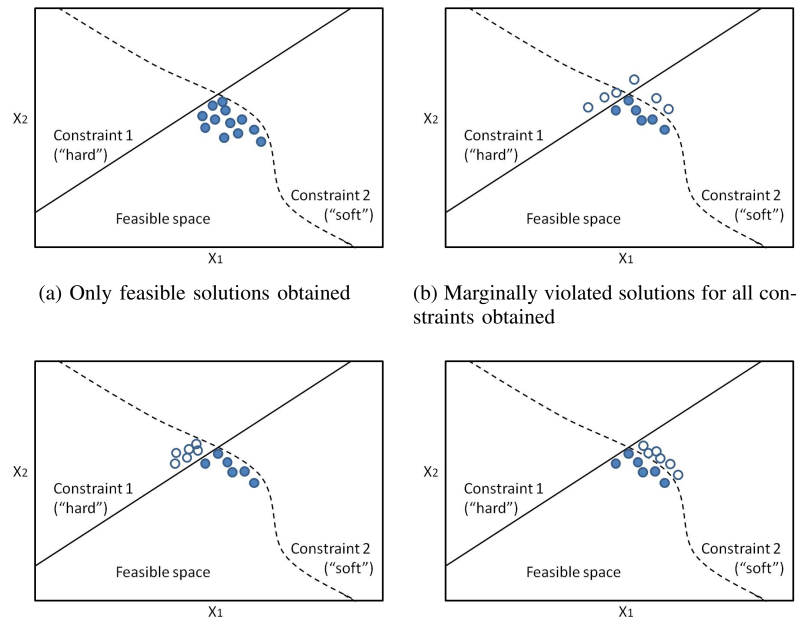
using effective parent matching in which mating is done between two infeasible solutions satisfying different constraints in order to create children which will satisfy all constraints. Mezura-Montes and Coello Coello [18] suggested a Simple Multimembered Evolutionary Strategy (SMES) in which the "best" infeasible solution determined by its objective function value is allowed to be copied into the next generation. In [19], ϵ-level control is used in order to deal effectively with equality constraints. In an attempt to simultaneously generate solutions to unconstrained and constrained optimization formulations of a multi-objective problem, Isaacs, Ray and Smith [20] introduced a Constraint Handling Evolutionary Algorithm (CHEA). In CHEA, some of the infeasible solutions are preserved during the search. The infeasible solutions in the population are ranked using the original objectives along with an additional objective, the number of constraint violations. The incorporation of search through the infeasible space improves the efficiency of the algorithm. However, CHEA does not have any provisions for quantifying the amount of constraint violation and the infeasible solutions obtained are not suitable for *trade-off* studies. Trade-off studies imply searching for a possibility of deriving benefits in the objective value(s) by marginal compromise on the constraints. To overcome this drawback and focus the search on constraint boundaries with an aim of achieving good quality feasible as well as *marginally* infeasible solutions, Infeasibility Driven Evolutionary Algorithm (IDEA) was proposed in [21], [22]. Instead of using number of violated constraints as an objective, IDEA uses a constraint violation measure based on the relative ranks of the solutions with respect to each constraint violation; thus delivering good quality feasible solutions as well as marginally infeasible solutions for trade-off considerations. This algorithm is further extended later in this paper.

- **Other constraint-handling methods:** These include special representation schemes for maintaining feasibility [23], [24], repair algorithms [25], [26], handling constraints and objectives separately [27], and incorporation of heuristic rules such as linear ranking [28] and binary tournament [29] to compare individuals in the population. The main drawbacks of these approaches include the need to develop problem-specific repair mechanisms, the unavailability of a feasible starting point, and early loss of diversity.

In the methods discussed above, there is usually no differentiation made between the constraints based on how critical it is to satisfy them in the final design. The methods which operate on *feasibility first* principles effectively consider that all constraints are *hard* constraints and must be satisfied in the final design. On the other hand, the methods that prefer or preserve infeasible solutions during the search do so considering all of them are *soft* constraints. However, it is not uncommon to encounter situations in which there could be a mix of hard and soft constraints. In engineering design problems, hard constraints are often a result of statutory requirements e.g. stresses should not exceed prescribed values, metacentric height should be more than certain value for a vessel, etc. Violating these constraints would imply critical failure of design. On the other hand, soft constraints refer to those whose infraction does not compromise the purpose of the design; for example cost, geometry, time schedule constraints can be flexible in many cases. In such problems, the trade-off solutions obtained may be practically useful only if they satisfy all hard constraints and marginally violate the soft constraints only. For example, consider the case shown in Figure 1, in which Constraint 1 is hard and Constraint 2 is soft. If a feasibility first EA (e.g. NSGA-II) is used, the most likely solutions obtained will be contained in the feasible region, as shown in Figure1(a). If infeasible solutions are preserved during the search (such as in IDEA), situations shown in Figures 1(b) and 1(c) may arise. In 1(b) the infeasible solutions include those that violate either of the constraints or both. Although in this case the solutions which satisfy the hard constraint may be filtered out, a preference strategy incorporated within the search process could deliver more diverse tradeoff set. In 1(c) only the solutions violating the hard constraint are obtained, which are of no practical use, whereas in 1(d), the desirable set of solutions is shown, which satisfy the hard constraint and marginally violate soft constraints.

Very limited research has been reported in numerical constrained optimization domain which differentiates between hard and soft constraints during the search. Most of them deal with specific applications. In [30], [31] a stochastic algorithm was developed to deal with MAX-SAT problems. In [32] industrial columns was optimized while considering hard and soft constraints on the manipulated variables. In the context of combinatorial optimization, hard and soft constraints were considered for creating nurse schedules for a hospital in [33]. In all these applications, the preferences for hard and soft constraints were imposed using weighting parameters on the constraints (high weights for hard and low weights for soft constraints). This introduces a number of user-defined parameters to be set, the choice of which can affect the performance significantly. At the same time, appropriate normalization methods have to be employed in order to effectively enforce the preferences.

In this paper, the earlier proposed algorithm IDEA is modified in order to deal with problems involving hard and soft constraints. Instead of using weighting factors, the algorithm segregates the solutions satisfying the hard constraints and ranks them. Rest of the paper is organized as follows. The proposed algorithm, Infeasibility Driven Evolutionary Algorithm for Mixed constraints (IDEA-M) is discussed in Section II. The performance of the algorithm on three benchmark problems is reported in Section III. Finally, a summary of the findings paper is presented in Section IV.



(a) Only feasible solutions obtained

(b) Marginally violated solutions for all constraints obtained

(c) Marginally violated solutions for hard constraint obtained

(d) Marginally violated solutions for soft constraint obtained

Fig. 1: Final population of solutions using an Evolutionary Algorithm. The optimum lies at the intersection point of constraint boundaries

II. Infeasibility Driven Evolutionary Algorithm for Mixed constraints (IDEA-M)

The proposed algorithm is a modification of Infeasibility Driven Evolutionary Algorithm (IDEA) proposed earlier [22], [34]. IDEA aimed at delivering: (a) the set of optimal solutions (best objective values for single-objective and Pareto fronts for multi-objective problems); (b) a few *marginally* infeasible solutions for trade-off studies; and (c) an improvement in the rate of convergence by effectively utilizing the infeasible solutions during the search. The modified algorithm, IDEA-M presented in this section aims to retain the above goals, while focusing on obtaining infeasible solutions which only violate the soft constraints.

Infeasibility Driven Evolutionary Algorithm [22] differs from the conventional EAs significantly in the terms of ranking and selection of the solutions. IDEA ranks solutions based on the original objectives along with additional objective representing constraint violation measure (CVM). Thereafter, the best ranked infeasible solutions are explicitly preserved in the population by assigning them higher ranks compared to the feasible solutions in the overall ranking of the population. Consequently the search proceeds through both feasible and infeasible regions, resulting in greater rate of convergence towards optimal solution(s). In IDEA, *all* the constraints are treated as soft/negotiable, which means that the additional objective CVM is represents overall rank sum considering violations of all objectives. However, in IDEA-M, this ranking process is modified to handle the solutions that violate hard constraints differently from others. The modified ranking process, and main steps involved in IDEA-M algorithm are described in the following subsections. The pseudo code of IDEA-M is shown in Algorithm 1.

Algorithm 1 Infeasibility Driven Evolutionary Algorithm for Mixed constraints (IDEA-M)

Require: N {Population Size}
Require: $N_G > 1$ {Number of Generations}
Require: $0 < \alpha < 1$ {Proportion of infeasible solutions}

```
N_inf = α * N
N_f = N − N_inf
pop_1 = Initialize()
Evaluate(pop_1)
for i = 2 to N_G do
    childpop_{i−1} = Evolve(pop_{i−1})
    Evaluate(childpop_{i−1})
    (S_f, S_inf) = Split(pop_{i−1} + childpop_{i−1})
    (S_{inf−hs}, S_{inf−hv}) = Split(S_inf)
    Rank(S_f) {non-dominance+crowding sort}
    Rank(S_{inf−hs}) {non-dominance+crowding sort}
    Rank(S_{inf−hv}) {sort based on CVM_h}
    S_inf = S_{inf−hs}+S_{inf−hv}
    pop_i = S_inf(1, N_inf) + S_f(1, N_f)
end for
```

1) Problem reformulation: A generic k-objective optimization problem with m inequality constraints can be posed as shown in Equation 1.

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}) \\ \text{Subject to} \quad & g_i(\mathbf{x}) \geq 0, \quad i = 1, \ldots, m \end{aligned} \tag{1}$$

In IDEA/IDEA-M, to focus the search near constraint boundaries, the first step is to reformulate the original problem as an unconstrained ($k+1$)-objective problem as shown in Equation 2

$$\begin{aligned} \text{Minimize} \quad & f'_1(\mathbf{x}) = f_1(\mathbf{x}), \ldots, f'_k(\mathbf{x}) = f_k(\mathbf{x}) \\ & f'_{k+1}(\mathbf{x}) = CVM \end{aligned} \tag{2}$$

The additional objective, CVM, is based on the amount of relative constraint violations among the population members. Consider one of the constraints (g_i). All solutions in the population are sorted in ascending order based on the value of the constraint violation for g_i. The solutions that do not violate the constraint g_i are assigned a constraint violation value of 0. The rest of the solutions are assigned constraint violation for the constraint g_i based on the sorted list, starting with rank 1 for the solution with least constraint violation. Solutions with the same value of constraint violation get the same rank. This ranking procedure is repeated for all constraints. The CVM for each solution is then calculated as the sum of the ranks (based on their constraint violations) obtained for all the constraints. The process of calculating CVM for a set of 10 solutions is illustrated for a problem with 3 constraints (C_1, C_2 and C_3) in Table I. A corresponding quantity, CVM_h is also calculated following exactly the same process, but only considering the set of hard constraints. This quantity is also used in IDEA-M ranking process as will be further discussed in Section II-3. In this example, constraint C_1 is assumed to be hard.

TABLE I: Calculation of Constraint Violation Measure for all constraints (CVM) and for hard constraints only (CVM_h). The solutions with $CVM_h = 0$ satisfy hard constraint(s)

	Violations			Relative ranks				
Individual	C_1	C_2	C_3	C_1	C_2	C_3	CVM	CVM_h
1	3.50	90.60	8.09	3	8	7	18	3
2	5.76	7.80	6.70	4	6	5	15	4
3	0.00	3.40	7.10	0	4	6	10	0
4	1.25	0.00	0.69	1	0	1	2	1
5	13.75	90.10	5.87	6	7	4	17	6
6	100.70	2.34	3.20	7	3	2	12	7
7	0.00	5.09	4.76	0	5	3	8	0
8	1.90	0.00	0.00	2	0	0	2	2
9	0.00	0.56	0.00	0	1	0	1	0
10	8.90	2.30	9.80	5	2	8	15	5

2) Evolution: In IDEA, the parent solutions are selected through binary tournament. For crossover, Simulated Binary Crossover (SBX) [35] operator is applied variable by variable. For mutation, a polynomial mutation [36] operator is used.

3) Ranking and reduction: The main modification in the proposed IDEA-M compared to IDEA is the mechanism of elite preservation. The difference is pictorially illustrated in Figure 2. In IDEA, the combined (parent and offspring) population is divided into a feasible set (S_f) and an infeasible set (S_{inf}). *All* solutions in the feasible and the infeasible sets are separately ranked using non-dominated sorting and crowding distance sorting of $k+1$ objectives. For the feasible solutions, non-dominated sorting using $k+1$ objectives is equivalent to non-dominated sorting using the original k objectives, as the additional objective value (which is based on the constraint violations) for feasible solutions is always 0. The next step is to choose the solutions that form the population for the next generation. A user-defined parameter α is used to identify the proportion of the infeasible solutions to be retained in the population. The numbers N_f ($= (1-\alpha) \times N$) and N_{inf} ($= \alpha \times N$) denote the number of feasible and infeasible solutions in the population respectively, where N is the population size. If the infeasible set S_{inf} has more than N_{inf} solutions, the first N_{inf} solutions are selected based on their ranking; otherwise all the solutions from S_{inf} are selected. The rest of the solutions are selected from the feasible set S_f, provided there are at least N_f feasible solutions. If S_f has fewer solutions, all the feasible solutions are selected and the rest are filled with remaining infeasible solutions from S_{inf}. The solutions are ranked from 1 to N in the order they are selected. Hence, the infeasible solutions selected first (at most N_{inf}) receive a higher rank than the feasible solutions.

In IDEA-M, the approach is modified as follows. After division into feasible and infeasible solutions, S_f is ranked the same way as above, but S_{inf} is further split into set of solutions that satisfy hard constraints (S_{inf-hs}) and those which violate (any of) the hard constraints (S_{inf-hv}). Evidently, for all the solutions in S_{inf-hs}, the quantity $CVM_h = 0$. For the set of solutions S_{inf-hs} only, the ranking is done using (non-dominance sorting + crowding distance) of original objectives and CVM. On the other hand, the infeasible solutions which form the set S_{inf-hv} are separately sorted (in ascending order) only based on their CVM_h values (*i.e.*, non-dominance and crowding distance sorting are not performed for these solutions). Thereafter, the solutions are recombined to get sorted S_{inf}, in which solution set S_{inf-hs} placed above the solution set S_{inf-hv}. The rationale behind this ordering is that solutions that satisfy the hard constraints are of more more practical importance than those which violate them and for these solutions, the drive is towards getting better tradeoffs in objectives and soft constraints. On the other hand, the solutions that violate the hard constraints are sorted based on CVM_h in order to promote solutions with lower hard constraint violation and eventually drive them to satisfy the hard constraints. Thereafter, the selection of next generation population is done the same way as described above for IDEA.

III. Numerical Experiments

Three constrained problems are considered for studies presented in this section. For each problem, results obtained using different constraints as hard/soft are discussed. For all experiments, a population size of 200 is evolved for 500 generations. The proportion of infeasible solutions α is set to 0.2, while the crossover probability, crossover index mutation probability and polynomial mutation index are set as 0.9, 10, 0.1 and 20 respectively. Thirty independent runs are done for each problem.

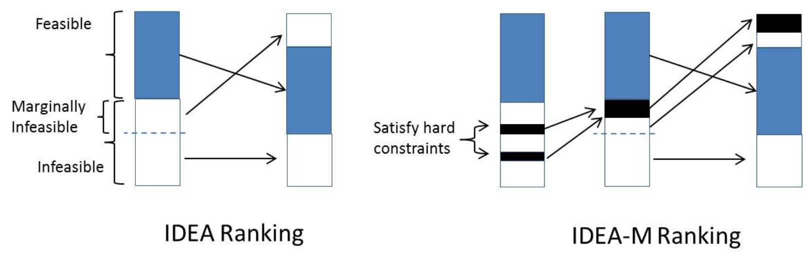


Fig. 2: Difference in ranking process between IDEA and IDEA-M

A. g6 problem

First example studied here is g6 problem from the constrained g-series test suite [37]. The problem contains two constraints, both of which are active at the optimum solution. Three different variations of the problem are solved, first considering both constraints as soft, second considering only g_1 as soft and third considering g_2 as soft.

In Figure 3, the solutions obtained for the above three cases are shown in the variable space in the vicinity of the optimum $(14.095, 0.84296)$. The narrow region between the two constraints for $x_1 \geq 14.095$, $x_2 \geq 0.84296$ comprise the feasible space. For each case, two graphs are presented, one to show behavior for a typical run, and other to ascertain the behavior over multiple runs. For the run shown in Figure 3(a), it is seen that the population contains a mix of solutions that violate g_1 or g_2. The multiple run plot in Figure 3(d) further shows solutions that violated both constraints simultaneously. Figures 3(b) and 3(c) shows solutions which satisfy hard constraint (g_2 and g_1 in respective cases) and show marginal violations in the soft constraints (g_1 and g_2 respectively). Figures 3(e) and 3(f) show the consistency in this behavior across multiple runs.

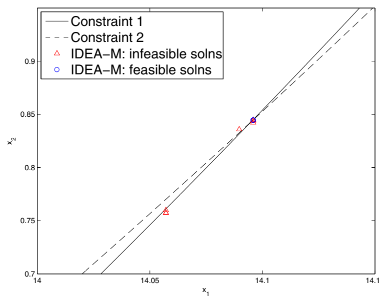
The numerical results obtained using IDEA-M are summarized in Table II. In terms of the quality of feasible solutions obtained, the statistics for all three cases are very close, with median results for the cases with one soft constraint being marginally better. The table also lists some of the infeasible solutions obtained during the experiments. Once again, for Case 1, there is a mix of solutions with either g_1 or g_2 violated. For Case 2, all solutions satisfy g_2, with marginal violation in g_1. The advantage of focusing the search on soft constraint boundary can be seen from the tradeoff solutions obtained. For example, a solution with a violation of 0.15817 in g_1 for Case 2 gives an objective value of -7124.0. By comparison, a solution for Case 1 has a violation of 0.35043 in g_1 with a worse objective value of -7035.8. Focusing the search on the soft constraint boundary increases the probability of getting better tradeoff solutions such as these. Similarly, for Case 3, there exists a solution with violation of 0.0047218 in g_2, which has almost the same objective value as a solution with much higher violation of 0.042193 in g_2 for Case 1 (-6964.4 v/s -6966.0).

B. CTP8 problem

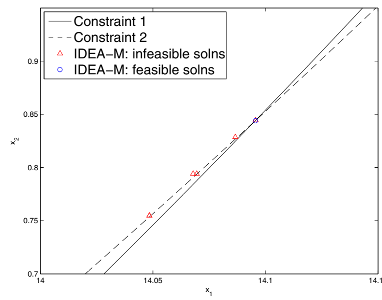
Next, bi-objective constrained CTP8 problem from the CTP test suite [38] is studied. The CTP8 problem has two constraints, which intersect each other forming alternate feasible and infeasible regions in the objective space, resulting in a disconnected (three segments) Pareto front as shown in Figure 4. Once again, three cases are studied. For a typical run, Figure 4(a) shows the distribution of final population for Case 1. The feasible solutions lie on the Pareto front, whereas the infeasible solutions are present in either g_1 and/or g_2 violated region. In Case 2 (Figure 4(b)), all solutions satisfy the constraint g_2 strictly, while for Case 3 (Figure 4(c)), all solutions satisfy g_1 strictly. The behavior is consistent among multiple runs, as seen from Figures 4(d)-(f). In terms of hypervolume, the median results are very close to each other for all three cases, as seen from Table III. The reference point taken for the calculation is the maximum of f_1 and f_2 obtained across all runs, and the code available from [39] is used for estimating hypervolumes.

C. Pressure vessel design

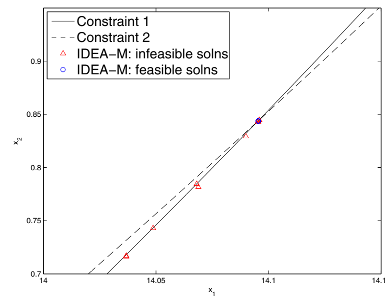
Lastly, an engineering optimization problem of pressure vessel design [40] is studied, which involves 4 variables and 3 constraints, with minimization of total manufacturing cost as the objective. Four different cases are considered; Case 1 with all constraints treated as soft, and Cases 2, 3, 4 with g_1, g_2 and g_3 soft respectively. The results obtained are presented in Table IV. In terms of the statistics for the feasible solutions, it is seen that Cases 2 and 4 clearly outperform Cases 1 and 3. Thus, it is clear than depending on the problem landscape, ensuring feasibility in certain constraints can drive the population towards the optimal regions quicker as compared to considering maintaining infeasible solutions for all constraints. Some of the previously reported best solutions include 6288.7445 [40] and 6119.97 [41], whereas the best solution found for Case 2 here is 5888.05 with $\mathbf{x}^* = \{12.4696, 6.16741, 40.3807, 199.152\}$. Some of the sample infeasible solutions obtained during the runs are also shown in Table IV, which respect the hard constraints as prescribed. Again, it can be seen that some of the solutions with better tradeoffs with respect to specifically prescribed constraints are obtained. For example, for one of the solutions in Case 2, the objective value is reduced to 5523.7, with 0.050855 units violation in g_1 only.



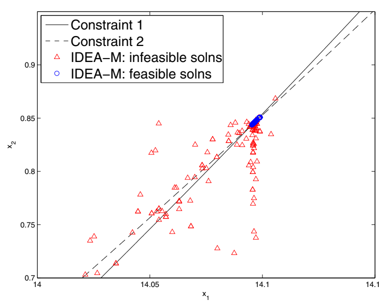
(a) Case 1: Both constraints treated as soft (typical run)



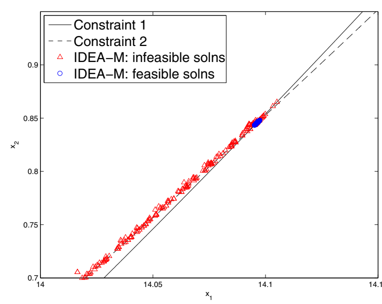
(b) Case 2: Constraint 1 treated as soft (typical run)



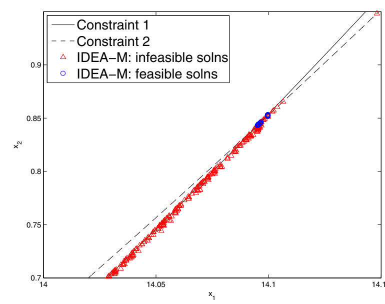
(c) Case 3: Constraint 2 treated as soft (typical run)



(d) Case 1: Both constraints treated as soft (multiple runs)



(e) Case 2: Constraint 1 treated as soft (multiple runs)



(f) Case 3: Constraint 2 treated as soft (multiple runs)

Fig. 3: Final populations obtained for g6 for a typical and multiple (30) runs

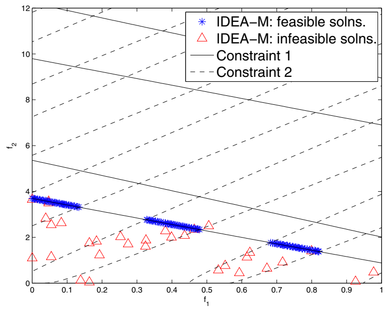
TABLE II: Results obtained for g6 problem using IDEA-M

	Feasible solutions (statistics for 30 runs)		A few Sample infeasible solutions obtained (Negative value implies violated constraint)				
			x_1	x_2	f	g_1	g_2
Case 1: Both constraints soft	Median:	-6959.86	14.096	0.84439	-6960.2	0.0018778	-0.00036388
	Best:	-6961.30	14.046	0.77824	-7035.8	-0.35043	0.25201
	Mean:	-6958.78	14.096	0.84439	-6960.2	0.0018872	-0.00037333
	Worst:	-6952.95	14.096	0.83916	-6966.0	0.043498	-0.042193
	Std.:	2.06391	14.085	0.82845	-6978.3	-0.060416	0.040472
Case 2: Constraint 1 (g_1) soft	Median:	-6959.88	14.068	0.79267	-7018.7	-0.071339	0.01716
	Best:	-6961.41	14.069	0.79287	-7018.4	-0.052075	0.00020966
	Mean:	-6959.55	14.095	0.84339	-6961.3	-0.0068064	0.0064545
	Worst:	-6956.37	14.020	0.7001	-7124.0	-0.15817	0.0073188
	Std.:	1.47356	14.037	0.73328	-7086.1	-0.1225	0.0070838
Case 3: Constraint 2 (g_2) soft	Median:	-6960.92	14.068	0.77978	-7033	0.034831	-0.089278
	Best:	-6961.69	14.094	0.84069	-6964.4	0.0029734	-0.0047218
	Mean:	-6960.33	14.067	0.78227	-7030.2	0.0010702	-0.056921
	Worst:	-6951.60	14.085	0.81796	-6989.9	0.034712	-0.053826
	Std.:	1.89789	14.045	0.7335	-7085.5	0.0092845	-0.10992

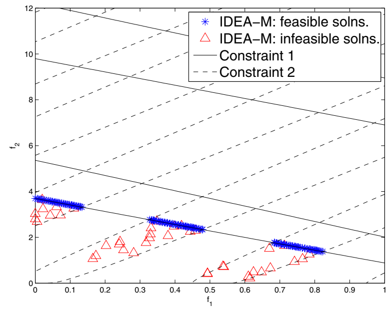
We end this section with a remark relating to generic mixed constraint problems, especially ones with high number of constraints. While for the given problems the prescribed constraint conditions (hard/soft) were met, some of the required combinations may not be achievable for certain problems. For example, if two constraints have high correlation, reduction in violation of one would also drive the other constraint towards feasibility, and hence if one of them is prescribed as hard whereas other as soft, an infeasible solution with such a combination may not be achieved.

IV. Summary

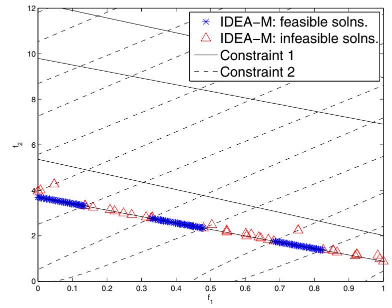
In this paper, a modification to earlier reported Infeasibility Driven Evolutionary Algorithm (IDEA) is proposed to deal with optimization problems involving a mix of *hard* and *soft* constraints. The proposed algorithm (IDEA-M) aims to deliver a set of solutions which satisfy hard constraints, and achieve tradeoffs with respect to the soft constraints. This is achieved by altering the ranking process used in IDEA. Three problems have been studied to highlight the benefits of the proposed approach. The results obtained using IDEA-M are consistent with the behavior prescribed in terms of hard and soft constraints. In addition, significant benefits are also seen in terms of convergence and the best feasible solutions obtained for some cases.



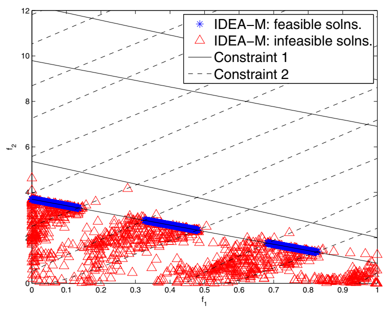
(a) Case 1: Both constraints treated as soft (typical run)



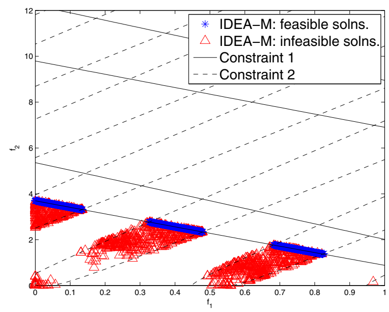
(b) Case 2: Constraint 1 treated as soft (typical run)



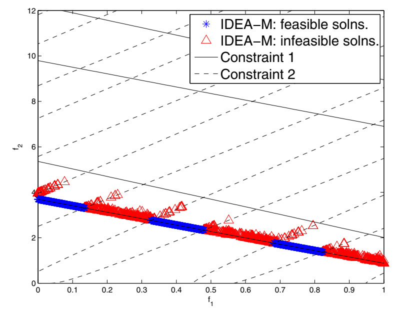
(c) Case 3: Constraint 2 treated as soft (typical run)



(d) Case 1: Both constraints treated as soft (multiple runs)



(e) Case 2: Constraint 1 treated as soft (multiple runs)



(f) Case 3: Constraint 2 treated as soft (multiple runs)

Fig. 4: Final populations obtained for CTP8 using IDEA-M for a typical and multiple (30) runs

TABLE III: Hypervolume metric results for CTP8 problem (over 30 runs)

	Case 1: Constraint 1 & 2 soft		Case 2: Constraint 1 soft		Case 3: Constraint 2 soft	
	Mean	Std.	Mean	Std.	Mean	Std.
CTP8	0.279	0.0122219	0.275	0.0115333	0.280	0.0138508

TABLE IV: Results obtained for pressure vessel problem using IDEA-M

	Feasible solutions (statistics for 30 runs)		A few Sample infeasible solutions obtained (Negative value implies violated constraint)							
			x_1	x_2	x_3	x_4	f	g_1	g_2	g_3
Case 1: All constraints soft	Median:	6123.73	0.8125	0.4375	42.09	176.74	6060.7	0.00015618	0.035958	-0.11457
	Best:	6060.7	0.6875	0.4375	31.428	176.24	3697	0.080938	0.13768	-6.1908e+05
	Mean:	6258.49	0.125	0.0625	10	23.425	33.597	-0.068	-0.0329	-1.2845e+06
	Worst:	6826.74	0.0625	0.0625	10	19.578	19.746	-0.1305	-0.0329	-1.2857e+06
	Std.:	189.975	0.625	0.4375	30.301	176.74	3250.9	0.040193	0.14843	-6.6968e+05
Case 2: Constraint 1 (g_1) soft	Median:	5928.55	11.656	6.1674	40.381	199.15	5523.7	-0.050855	0.00023345	0.019355
	Best:	5888.05	11.123	6.1674	40.381	199.15	5289	-0.084172	0.00023257	0.0086405
	Mean:	5950.87	10.357	6.1674	40.381	199.15	4957.4	-0.13204	0.0002332	0.0021459
	Worst:	6314.82	5.7579	6.1675	40.381	199.15	3104.3	-0.41948	0.00023604	0.011776
	Std.:	85.3576	8.6635	6.1674	40.381	199.15	4247.6	-0.23788	0.00023317	0.38278
Case 3: Constraint 2 (g_2) soft	Median:	6233.33	18.757	1	60.719	30.952	3571.2	0.00043487	-0.51676	182.71
	Best:	5993.23	15.912	7.8651	51.527	86.683	6367.8	5.2608e-06	-4.0648e-09	90.523
	Mean:	6289.61	18.757	3.474	60.718	30.958	4585.1	0.00046394	-0.36212	201.11
	Worst:	6788.87	18.757	1.6108	60.728	30.952	3822.1	0.00025497	-0.47867	712.27
	Std.:	222.125	18.756	3.1968	60.717	30.953	4470.9	0.00043265	-0.37944	63.095
Case 4: Constraint 3 (g_3) soft	Median:	5988.77	14.814	7.3225	47.972	115.3	6188.9	2.7461e-05	7.8201e-07	-3.7184e-05
	Best:	5988.77	14.814	7.3225	47.972	108.08	5969.7	2.7475e-05	7.6034e-07	-52190
	Mean:	6009.13	11.111	5.4171	34.789	184.99	4125.4	0.022997	0.0066822	-4.1628e+05
	Worst:	6217.01	10.425	5.5907	32.458	200	3829.3	0.025126	0.039766	-4.9082e+05
	Std.:	90.803	10.448	5.2606	31.362	157.62	3062.1	0.047708	0.029597	-6.7977e+05

References

[1] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[2] Z. Michalewicz, "A Survey of Constraint Handling Techniques in Evolutionary Computation Methods," in *Proceedings of the 4th Annual Conference on Evolutionary Programming*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds. Cambridge, Massachusetts: The MIT Press, 1995, pp. 135–155.

[3] E. Mezura-Montes, Ed., *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence. Springer-Verlag Berlin Heidelberg, 2009, vol. 198.

[4] A. Kuri-Morales and C. V. Quezada, "A Universal Eclectic Genetic Algorithm for Constrained Optimization," in *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98*. Aachen, Germany: Verlag Mainz, September 1998, pp. 518–522.

[5] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained Optimization via Genetic Algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[6] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs," in *Proceedings of the first IEEE Conference on Evolutionary Computation*, D. Fogel, Ed., Orlando, Florida, 1994, pp. 579–584.

[7] Z. Michalewicz, "Genetic Algorithms, Numerical Optimization, and Constraints," in *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, L. J. Eshelman, Ed., University of Pittsburgh. San Mateo, California: Morgan Kaufmann Publishers, July 1995, pp. 151–158.

[8] J. C. Bean and A. B. Hadj-Alouane, "A Dual Genetic Algorithm for Bounded Integer Programs," Department of Industrial and Operations Engineering, The University of Michigan, Tech. Rep. TR 92-53, 1992.

[9] F. Hoffmeister and J. Sprave, "Problem-independent handling of constraints by use of metric penalty functions," in *Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, L. J. Fogel, P. J. Angeline, and T. Bäck, Eds. San Diego, California: The MIT Press, February 1996, pp. 289–294.

[10] T. Ray, K. Tai, and K. Seow, "Multiobjective design optimization by an evolutionary algorithm," *Engineering Optimization*, vol. 33, no. 4, pp. 399–424, 2001.

[11] P. Y. Ho and K. Shimizu, "Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme," *Information Sciences*, vol. 177, pp. 2985–3004, 2007.

[12] E. Mezura-Montes and C. A. Coello Coello, "Constrained Optimization via Multiobjective Evolutionary Algorithms," in *Multiobjective Problem Solving from Nature: From Concepts to Applications*, ser. Natural Computing Series, J. Knowles, D. Corne, and K. Deb, Eds. Springer-Verlag, 2008, pp. 53–75.

[13] D. A. G. Vieira, R. L. S. Adriano, J. A. Vasconcelos, and L. Krahenbuhl, "Treating constraints as objectives in multiobjective optimization problems using niched pareto genetic algorithm," *IEEE Transactions on Magnetics*, vol. 40, no. 2, March 2004.

[14] C. A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil engineering and environmental systems*, vol. 17, no. 4, pp. 319–346, 2000.

[15] S. B. Hamida and M. Schoenauer, "An adaptive algorithm for constrained optimization problems," in *Proceedings of Parallel Problem Solving from Nature (PPSN-VI), Lecture notes in Computer Science 1917*. Springer Berlin/Heidelberg, 2000, pp. 529–538.

[16] ——, "ASCHEA: new results using adaptive segregational constraint handling," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, May 2002, pp. 884–889.

[17] R. Hinterding and Z. Michalewicz, "Your brains and my beauty: parent matching for constrained optimisation," in *Proceedings of IEEE Conference on Evolutionary Computaion (CEC)*, May 1998, pp. 810–815.

[18] E. Mezura-Montes and C. Coello Coello, "A simple multimembered evolution strategy to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, Feb. 2005.

[19] T. Takahama and S. Sakai, "Constrained optimization by the constrained differential evolution with gradient-based mutation and feasible elites," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, 2006, pp. 1–8.

[20] A. Isaacs, T. Ray, and W. Smith, "Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, 2008, pp. 2780–2787.

[21] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, "Infeasibility driven evolutionary algorithm for constrained optimization," in *Constraint-Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence, E. Mezura-Montes, Ed. Springer Berlin Heidelberg, 2009, vol. 198, pp. 145–165.

[22] H. K. Singh, A. Isaacs, T. Ray, and W. Smith, "Infeasibility Driven Evolutionary Algorithm (IDEA) for Engineering Design Optimization," in *Proceedings of 21st Australiasian Joint Conference on Artificial Intelligence (AI)*, 2008, pp. 104–115.

[23] L. Davis, Ed., *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[24] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[25] J. Xiao, Z. Michalewicz, and K. Trojanowski, "Adaptive Evolutionary Planner/Navigator for Mobile Robots," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 18–28, 1997.

[26] Z. Michalewicz and J. Xiao, "Evaluation of Paths in Evolutionary Planner/Navigator," in *Proceedings of International Workshop on Biologically Inspired Evolutionary Systems*, Tokyo, Japan, May 1995, pp. 45–52.

[27] P. D. Surry and N. J. Radcliffe, "The COMOGA method: Constrained optimisation by multi-objective genetic algorithms," *Control and Cybernetics*, vol. 26, no. 3, 1997.

[28] D. Powell and M. M. Skolnick, "Using genetic algorithms in engineering design optimization with non-linear constraints," in *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA)*, S. Forrest, Ed. San Mateo, California: Morgan Kaufmann Publishers, July 1993, pp. 424–431.

[29] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, 2000.

[30] Y. Jiang, H. Kautz, and B. Selman, "Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT," in *1st International Joint Workshop on Artificial Intelligence and Operations Research*. Citeseer, 1995.

[31] H. Kautz, B. Selman, and Y. Jiang, "A general stochastic approach to solving problems with hard and soft constraints," *The Satisfiability Problem: Theory and Applications*, vol. 17, pp. 573–586, 1997.

[32] U. Volk, D.-W. Kniese, R. Hahn, R. Haber, and U. Schmitz, "Optimized multivariable predictive control of an industrial distillation column considering hard and soft constraints," *Control engineering practice*, vol. 13, no. 7, pp. 913–927, 2005.

[33] I. Berrada, J. A. Ferland, and P. Michelon, "A multi-objective approach to nurse scheduling with both hard and soft constraints," *Socio-Economic Planning Sciences*, vol. 30, no. 3, pp. 183–193, 1996.

[34] T. Ray, H. K. Singh, A. Isaacs, and W. Smith, "Infeasibility driven evolutionary algorithm for constrained optimization," in *Constraint Handling in Evolutionary Optimization*, ser. Studies in Computational Intelligence, E. Mezura-Montes, Ed. Springer, 2009, pp. 145–165.

[35] K. Deb and S. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[36] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

[37] Z. Michalewicz and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[38] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons Pvt. Ltd., 2001.

[39] Y. Cao. Hypervolume indicator. [Online]. Available: http://www.mathworks.com.au/matlabcentral/fileexchange/19651-hypervolume-indicator

[40] C. A. Coello Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[41] A. Isaacs, "Development of optimization methods to solve computationally expensive problems," Ph.D. dissertation, University of New South Wales, Australian Defence Force Academy (UNSW@ADFA), Canberra, Australia, 2009.