Unconstrained Robust Optimization using a Descent-based Crossover Operator

Aleksi Porokka[2], Ankur Sinha[1], Pekka Malo[2], and Kalyanmoy Deb[3]

[1]Productivity and Quantitative Methods
Indian Institute of Management Ahmedabad, Ahmedabad 380015, India
{asinha@iimahd.ernet.in}

[2]Department of Information and Service Economy
Aalto University School of Business, PO Box 21220, 00076 Aalto, Finland
{firstname.lastname@aalto.fi}

[3]Department of Electrical and Computer Engineering
Michigan State University, East Lansing, MI, USA
{kdeb@egr.msu.edu}

COIN Report: 2015008

***Abstract*— Most of the practical optimization problems involve variables and parameters that are not reliable and often vary around their nominal values. If the optimization problem is solved at the nominal values without taking the uncertainty into account, it can lead to severe operational implications. In order to avoid consequences that can be detrimental for the system, one resorts to the robust optimization paradigm that attempts to optimize the "worst case" solution arising as a result of perturbations. In this paper, we propose an evolutionary algorithm for robust optimization of unconstrained problems involving uncertainty. The algorithm utilizes a novel crossover operator that identifies a cone-based descent region to produce the offsprings. This leads to a large saving in function evaluations, but still guarantees convergence on difficult multimodal problems. A number of test cases are constructed to evaluate the proposed algorithm and comparisons are drawn against two benchmark cases.**

***Index Terms*— Evolutionary algorithms, Robust Optimization, Test Problem Construction, Cone Programming.**

I. Introduction

Optimization problems are often solved with an assumption that the given problem is accurately defined and does not contain any uncertainties. However, most of the practical optimization problems in science, engineering and business are never accurately identified and always involve certain kinds of approximations, noise, measurement errors etc. Under such a scenario one often resorts to different kinds of uncertainty-based optimization approaches, for instance stochastic optimization and robust optimization [2]. A large body of literature exists on sensitivity analysis as well that aims to tackle a similar problem [16]. There are a number of definitions of robustness available in the literature [4], [19], [7]. One of the definitions aims at optimizing the "worst case" performance of a system, while the other definition aims at optimizing the "expected" performance of a system under perturbations. Both the definitions, in some sense, attempt to identify a region in the search space that is least sensitive to minor perturbations in the design variables or parameters.

The focus of this paper is to develop an evolutionary algorithm to handle robust optimization problems, wherein we would like to optimize the "worst case" solution of a problem in a given neighbourhood in the presence of uncertainties in the decision variables. The motivation behind finding the robust optimum stems from the large number of applications that not only demand a high quality optimum, but also require the solution to be tolerant to minor variations in the decision variables. Such problems are commonly encountered in the area of control systems [13], scheduling [15], [12], [10], design and manufacturing [21], [14], [20], [9], [1], [18] etc.

A number of ideas for robust optimization using evolutionary algorithms have already been suggested in the past. For instance, Bonham et al. [5] utilize a hierarchical strategy to perform a search in several high performance regions in the fitness landscape. Tsutsui and Ghosh [19] use a schema theorem for genetic algorithms to estimate robustness. Branke [6] discusses a number of methods for searching robust solutions, and points that instead of sampling fitness function several times to determine robustness, one should consider using a global approximation model. Sano and Kita [17] use the search history of the genetic algorithm to optimize noisy fitness functions that leads to savings in fitness evaluations. Researchers [11] have also considered converting the robust optimization problem into a multi-objective optimization task where the objectives are maximizing performance and robustness. In the context of single objective optimization, the recent research in the area of robust optimization using

evolutionary algorithms has been focussed mostly along similar ideas with attempts to determine appropriate sampling or meta-modelling techniques that would lead to savings in function evaluations.

A common approach to estimate robustness of a solution is to sample a finite number of points in the neighbourhood of the given point. Many of the evolutionary methods rely on localized sampling to determine robustness. However, these approaches have failed to extract sufficient information that is available from the localized sample. A sample around a point not only gives an estimate of how robust the given point is, but can also be used to determine the region in which the search should propagate to explore even better solutions. In this paper, we develop a crossover operator that creates points in more robust regions by using local information. The idea of the crossover operator is motivated by recent work on robust optimization by Bertsimas et al. [3] and an earlier study in max-min programming by [8].

The rest of the paper is structured as follows. We start with the mathematical formulation of robust optimization problems and provide insights into it based on discussions in [8], [3]. Thereafter, we introduce the crossover operator used to produce offspring based on local information. This is followed by the algorithm description and a description of test problems on which we intend to evaluate our algorithm. Then we provide the results of our algorithm on the test problems and compare it with two benchmark cases. Finally, we conclude the paper with conclusions and future extensions to this work.

II. Robust Optimization and Descent Direction

Robustness issues commonly arise due to two forms of uncertainties, namely uncertainty in decision variables and uncertainty in parameters. While decision variable uncertainty stems from inaccurate realizations of the decision variables, the parameter uncertainty stems from lack of precision in specifying the models. In this paper, we focus on decision variable uncertainty. However, the ideas can easily be extended to incorporate robustness issues arising from parameter uncertainty as well.

Consider an unconstrained mathematical optimization problem with decision vector $x \in \mathbb{R}^n$ and objective function $f(x) \in \mathbb{R}$ defined below,

$$\min_x f(x).$$

The robust counterpart of the above optimization problem can be written as

$$\min_x \max_{\Delta x \in \mathcal{B}_\delta} f(x + \Delta x),$$

where Δx represents the implementation errors around x. The implementation errors Δx belong to the uncertainty set $\mathcal{B}_\delta := \{\Delta x \in \mathbb{R}^n : ||\Delta x||_2 \leq \delta\}$, which causes a precise decision vector x to be actually realized in its δ-neighbourhood given as $\mathcal{X}_\delta(x) := \{y : ||y - x||_2 \leq \delta\}$. Let the worst implementation error be denoted as Δx^*, then the set $\mathcal{B}^*_\delta(x) := \{\Delta x^* \in \mathbb{R}^n : \Delta x^* \in \arg\max_{\Delta x \in \mathcal{B}_\delta} f(x + \Delta x)\}$ denotes all the possible worst implementation errors around x. If a function $f_r(x)$ represents the robust function value of $f(x)$ as point x, then one can define $f_r(.)$ as follows

$$f_r(x) = \max_{\Delta x \in \mathcal{B}_\delta} f(x + \Delta x).$$

The above definition and symbols can be easily extended for constrained problems as well. Consider a constrained mathematical optimization problem with decision vector $x \in \mathbb{R}^n$, objective function $f(x) \in \mathbb{R}$ and inequality constraints $g_i(x) \in \mathbb{R} \; \forall \; i \in I$ defined below,

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g_i(x) \leq 0 \; \forall \; i \in I. \end{aligned}$$

The robust counterpart of the above constrained optimization problem can then be written as

$$\begin{aligned} &\min_x \max_{\Delta x \in \mathcal{B}_\delta} f(x + \Delta x) \\ &\text{s.t. } \max_{\Delta x \in \mathcal{B}_\delta} g_i(x + \Delta x) \leq 0 \; \forall \; i \in I, \end{aligned}$$

where all the symbols have a similar interpretation as in the unconstrained case. Function $f_r(.)$ in constrained robust optimization case can also be defined in a similar manner.

The notations used in the unconstrained and constrained robust optimization formulations have been shown geometrically through Figures 1 and 2, respectively. Note that in Figure 1 the worst implementation errors are given at the worst function value, while in the context of constrained robust optimization the worst implementation errors may be defined in a slightly different manner. If there exist infeasible points in the neighbourhood of x then the worst implementation errors correspond to the maximally infeasible point(s)[1] in the neighbourhood.

Since the focus of this paper is on unconstrained robust optimization, our further discussions will be limited to unconstrained problems. In our future study, we intend to extend this paper to include constrained robust optimization problems as well.

Next, we will briefly introduce ideas related to unconstrained robust optimization that will be used for the construction of a crossover operator in the next section. A detailed discussion of some of these ideas can be also be found in [3], [8].

The unconstrained robust optimization formulation is a special case of bilevel optimization problem and can be written as

$$\begin{aligned} &\min_x f(x + \Delta x) \\ &\text{s.t. } \Delta x \in \arg\max_{\Delta x} \{f(x + \Delta x) : ||\Delta x||_2 \leq \delta\}. \end{aligned}$$

If $f(x)$ is continuously differentiable with respect to x, and Δx^* is a lower level optimum then it satisfies the necessary Karush-Kuhn-Tucker (KKT) condition

$$\nabla f(x + \Delta x^*) - 2\mu \Delta x^* = 0, \tag{1}$$

[1]If the constraint violation at a given point is defined as the sum of all constraint violations, then a point with maximum constraint violation in the neighbourhood of x is referred as the maximally infeasible point.

where μ is a non-negative KKT multiplier. The KKT condition shows us an interesting relationship for the gradient at $x+\Delta x^*$. Given that our aim is to find a direction d that leads to a descent for the robust function f_r, we will next motivate how the KKT condition can be helpful in identifying such directions.

For convenience, let us assume that the function is continuously twice differentiable. Consider that one takes an infinitesimally small step αd with $\alpha > 0$ from a point x. If this direction is a descent direction, then

$$f_r(x + \alpha d) - f_r(x) \leq 0. \tag{2}$$

Let $x_1 = x$ and $x_2 = x+\alpha d$. Then the worst implementation around x_1 is $\Delta x_1^* = y^*$, and around x_2 is $\Delta x_2^* = y^* + \delta y^*$, where Δx_1^* and $\Delta x_2^* \in \mathcal{B}_\delta$ and δy^* is infinitesimal. Then Equation 2 can be written as

$$\begin{aligned} &f(x_2 + \Delta x_2^*) - f(x_1 + \Delta x_1^*) \leq 0 \\ \Rightarrow &f(x + \alpha d + y^* + \delta y^*) - f(x + y^*) \leq 0 \end{aligned}$$

Replacing the first term with its first order Taylor's expansion about $x + y^* + \delta y^*$ and by rearranging the terms, we get

$$\begin{aligned} &f(x + y^* + \delta y^*) - f(x + y^*) + \\ &\nabla^T f(x + y^* + \delta y^*)(\alpha d) + |\alpha d|\varepsilon(\alpha d) \leq 0, \end{aligned}$$

where $\alpha \to 0$ and function $\varepsilon(\alpha d) \to 0$. Here, the difference $f(x + y^* + \delta y^*) - f(x + y^*)$ is always non-negative, because in the neighbourhood $\mathcal{B}_\delta$ around x, y^* is the worst implementation error. As a result, we have that

$$\nabla^T f(x + y^* + \delta y^*)(\alpha d) + |\alpha d|\varepsilon(\alpha d) \leq 0.$$

If f is twice continuously differentiable, we can again apply the first-order Taylor's expansion on ∇f about $x + y^*$. By inserting the expansion and dividing both sides by $\alpha > 0$, we obtain

$$\begin{aligned} &\nabla^T f(x + y^*)d + (\delta y^\star)^T \nabla^2 f(x + y^\star)d + \\ &|\delta y^\star|\varepsilon(\delta y^\star)d + |d|\varepsilon(\alpha d) \leq 0. \end{aligned}$$

Taking the limits $\alpha \to 0$ and $\delta y^\star \to 0$, simplifies the result to

$$\nabla^T f(x + y^*)d \leq 0.$$

From Equation 2, we know that $\nabla f(x + \Delta x^*) = 2\mu\Delta x^*$, therefore we can write

$$d^T \Delta x^* \leq 0.$$

The above relationship suggests that if one knows the worst implementation about x, then choosing a direction d such that it points away from the worst implementation would lead to an improvement in the robust value of a function. The geometrical interpretations of the notations used in the above proof have been shown in Figure 3.

If instead of one there are multiple worst implementation errors present about x, the descent direction lies in a region that points away from all the worst implementation errors. This has been shown graphically through Figures 4 and 5 for two and three dimensional robust optimization problems respectively. We will utilize this idea in the next section to generate a crossover operator for robust optimization that creates offspring in the descent region, which is a cone. However, finding all the worst implementation errors around a point x can be a very difficult task. Therefore, we use sampling to obtain an approximate descent cone.

III. Descent-based Crossover Operator for Robust Optimization

Consider a point x around which we sample p points, $x + \Delta x_i;\ i \in \{1, \ldots, p\}$. From these p points we choose at most the worst q points ($q \leq p$) that are worse than x and find an approximate descent cone based on the chosen points. The higher the sample size, the better is the approximation for the descent cone. Below we discuss the descent-based crossover operator that creates offspring in this approximated descent cone. The proposed crossover operator requires two parents (x_1 and x_2) and produces two offspring (c_1 and c_2).

1) Sample p points around x_1 and x_2, such that the new points lie the neighbourhoods $\mathcal{X}_\delta(x_1)$ and $\mathcal{X}_\delta(x_2)$ respectively.

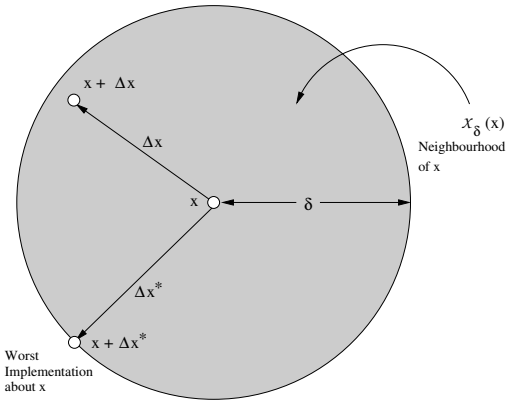

Fig. 1. A geometrical interpretation of the symbols used in unconstrained robust optimization formulation.

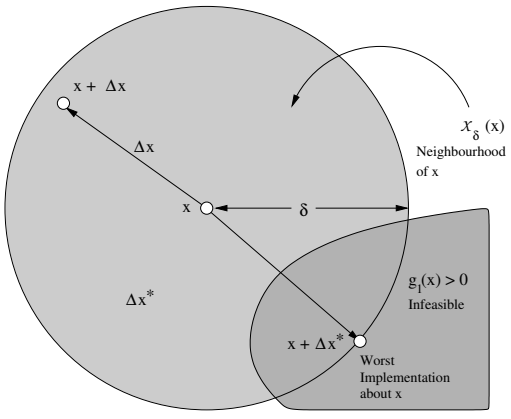

Fig. 2. A geometrical interpretation of the symbols used in constrained robust optimization formulation.

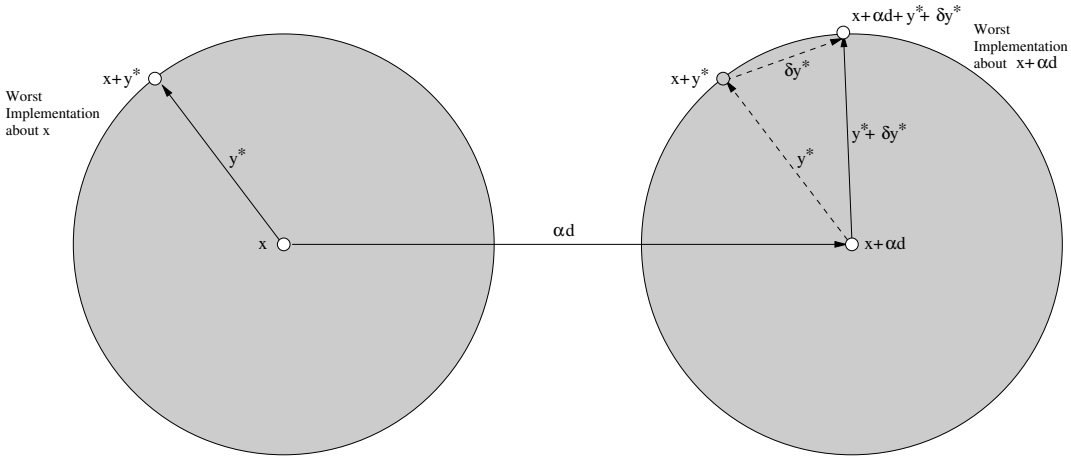

Fig. 3. A geometrical representation of notations used in the gradient descent approach.

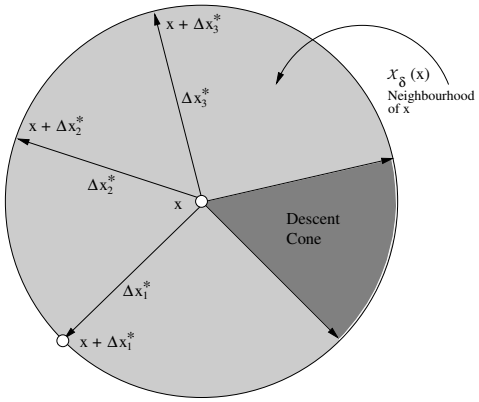

Fig. 4. Descent cone in the presence of multiple worst implementation errors around a point x.

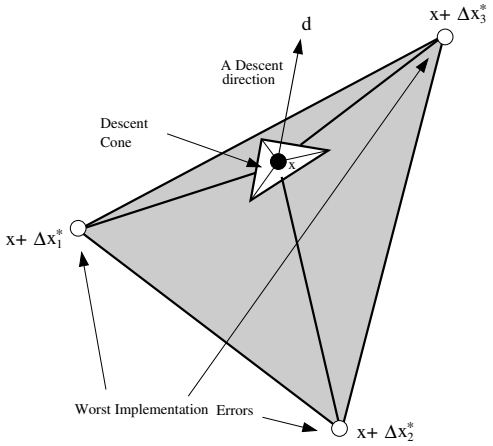

Fig. 5. Descent cone in higher dimensions in the presence of multiple worst implementation errors around a point x.

2) Find at the most q worst implementation errors around each parent x_1 and x_2.
3) Identify a descent cone for each parent. The cones corresponding to x_1 and x_2 are composed of multiple planes, say r_1 and r_2 in number respectively.
4) Find the unit vectors $\hat{u}_{1j}, j \in \{1, \ldots, r_1\}$ and $\hat{u}_{2j}, j \in \{1, \ldots, r_2\}$ that are perpendicular to the planes[2] forming the descent cones corresponding to the parents x_1 and x_2 respectively.
5) Take a weighted sum of the unit vectors $\hat{u}_{1j}, j \in \{1, \ldots, r_1\}$ and $\hat{u}_{2j}, j \in \{1, \ldots, r_2\}$ to find directions d_1 and d_2 as follows

$$d_i = \sum_{j=1}^{r_i} w_{ij} \hat{u}_{ij},$$

where $0 \leq w_{ij} \leq 1$. Refer to Section III-A for a discussion on choosing the weights.

[2]Note that the perpendicular vector for a plane in this case is negative of the corresponding implementation error.

6) The offspring c_1 and c_2 are then created as follows

$$c_i = x_i + \left|\mathcal{N}(0, ||x_2 - x_1||_2)\right| d_i,$$

where $\left|\mathcal{N}(0, ||x_2 - x_1||_2)\right|$ represents a half-normal distribution with 0 mean and standard deviation $||x_2 - x_1||_2$.

The above steps have also been explained through Figure 6, where a crossover between x_1 and x_2 leads to offspring c_1 and c_2 along descent directions d_1 and d_2 respectively.

A. Choosing weights for unit vectors

The weights for the unit vectors $\hat{u}_{1j}, j \in \{1, \ldots, r_1\}$ and $\hat{u}_{2j}, j \in \{1, \ldots, r_2\}$ are all positive. By choosing positive weights we ensure that the directions d_1 and d_2 always lie within the descent cone. In this paper, we assign the weights to each unit vector based on the function value at the implementation error that corresponds to the respective plane. Note that each unit vector is associated to a plane, and each plane is associated to an implementation error. If a unit vector $\hat{u}$, corresponds to an implementation error Δx about

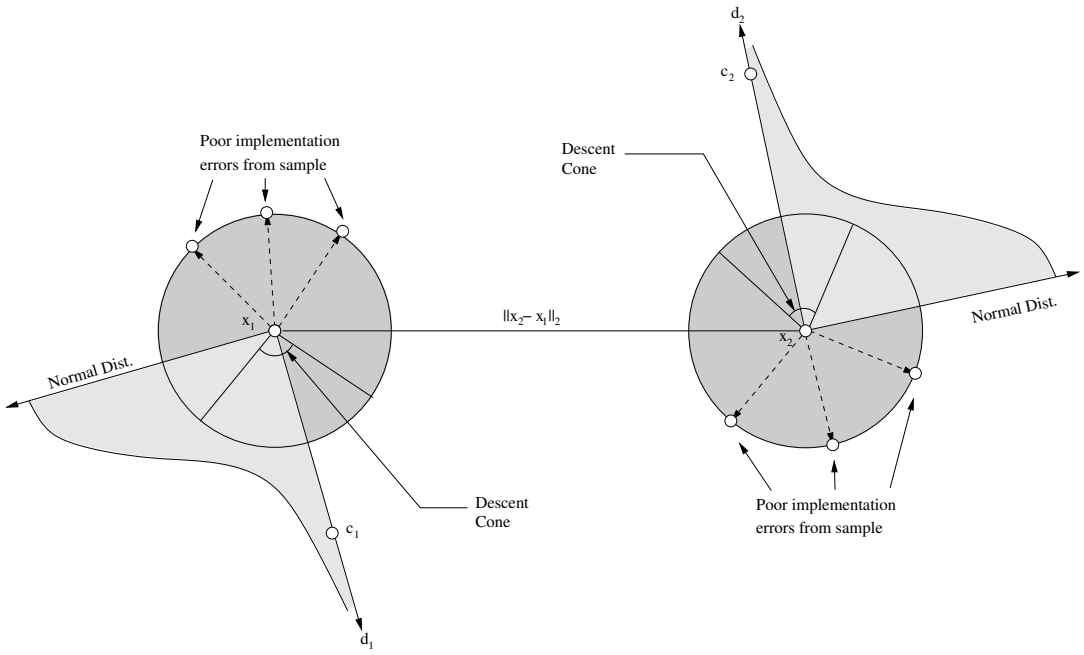

Fig. 6. Crossover operation between parents x_1 and x_2 producing offspring c_1 and c_2.

x, then we compute its weight as follows:

$$w = \frac{1}{1 + f(x + \Delta x^w) - f(x + \Delta x)},$$

where Δx^w is the worst implementation error found around x. Such an assignment ensures that the poorer an implementation error, the higher is the weight associated with it.

IV. Algorithm description

Below we provide a step-by-step procedure for the robust optimizer that utilizes the descent-based crossover operator. We refer to the algorithm as RO-Descent. The steps are similar to that of a standard real-coded genetic algorithm.

Step 1 *Algorithm initialization*. Create a random population of P individuals and compute the function value for all the individuals.

Step 2 *Robust function value computation*. For each population member (say x_i) initialize an empty set S_i. Sample p points that lie inside the δ-neighbourhood of x_i and add it to S_i. Also check for other population members that are in the δ-neighbourhood of x_i and append it to the set S_i. Compute the function value of the newly sampled points and choose at most q worst points from the set S_i that are also worse than x_i. Assign robust function value to x_i as the function value of the worst point in S_i that is also worse than x_i. If there is no worse point than x_i then the robust function value for x_i is same as the function value at x_i. Compute the robust function value for all the members using this operation.

Step 3 *Parent selection*. Select two parents using tournament selection.

Step 4 *Offspring production*. Produce two offspring using the descent-based crossover and polynomial mutation.

Step 5 *Fitness value update*. Run the same computations as in Step 2 for the offspring and compute the robust function values.

Step 6 *Population update*. Pool two random population members with the offspring. Replace the two population members with the two best members from the pool.

Step 7 *Termination check*. Stop the algorithm if the termination conditions hold. Otherwise, go back to Step 3.

A. Termination Condition

In this paper, we use an termination condition based on the movement of the elite individual. If the location of the elite individual has not changed by more than 10^{-3} in Euclidean distance during 200 generations, then we terminate the algorithm.

B. Parameters

The algorithm utilizes a number of parameters that have been adjusted as follows in our computations. Some of the parameters depend on the number of dimensions (N) in the test problem being solved. The values have been adjusted based on a parametric study.

1) Population size: $30N$
2) Probability of crossover: 0.9
3) Probability of mutation: 0.1
4) Number of sample points (p): $2N^2 + 2$
5) Maximum number of worst points chosen (q): N

V. Test problems

In this section, we suggest a framework to create unconstrained multi-modal test problems that are scalable with respect to variables. In the later part of this paper, we utilize these test problems to evaluate the RO-Descent approach.

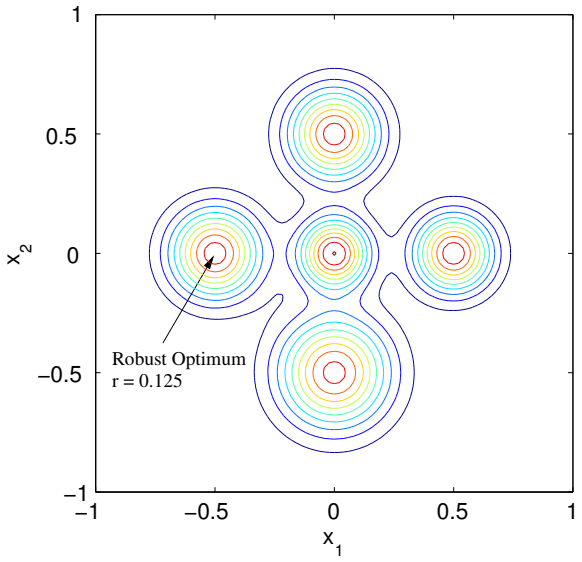

Fig. 7. Test problem for robust optimization with single variable.

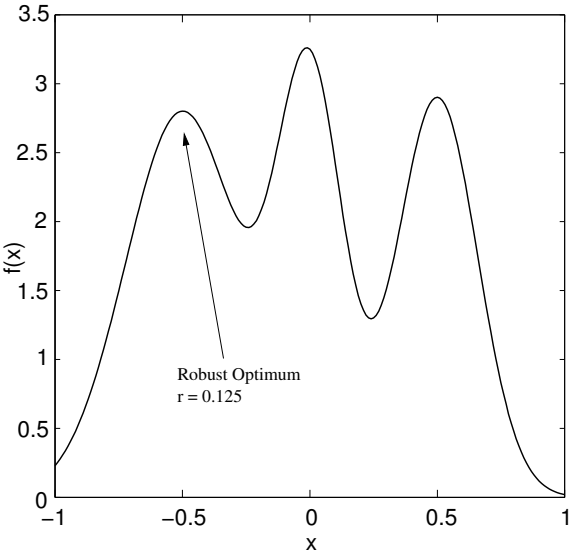

Fig. 8. Test problem for robust optimization with two variables.

Ideally, test problems for robust optimization should contain multiple peaks, and one of the peaks should be the robust optimum based on the chosen radius of the uncertainty. This would enable a user to test the accuracy of her algorithm in finding the correct solution (peak) given some radius. We would also like to have the test problems scalable with respect to the number of variables.

We have devised a method for constructing N-dimensional robust optimization test problems that meet the aforementioned properties. The method relies on creating a mixture distribution, which can be expressed as a combination of k different N-dimensional normal probability density functions $\mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$, where $\mu_\mathbf{i} \in \mathcal{R}^N$ and $\mathbf{\Sigma}_i \in \mathcal{R}^{N\times N}$, for $i = 1, ..., k$. The mixture distribution has a landscape with as many peaks as there are normal distributions, provided that the means μ_i are not too close to each other.

The usage of the mixture distribution enables us to modify the test problems in three ways. First, the placement of the normal distributions can be altered by changing their means. Second, the landscape of the test problems can be modified using different covariance matrices for the normal distributions. Finally, by expressing the mixture distribution as a weighted combination $\sum_{i=1}^{N} w_i \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$ (each normal distribution i is associated with a weight $w_i > 0$), we can change the differences in height between the normal distributions. In the case of weights that do not sum up to 1, the mixture distribution is not a probability distribution anymore. However, this is not an issue for our purposes.

In the test problems used in this paper, an N-dimensional test problem is a mixture distribution of $2N + 1$ different N-dimensional normal distributions. One of the distributions is always placed at the origin. Two distributions are placed on each of the axes symmetrically around the origin: one on the negative axis and the other on the positive axis. The distance from the origin is a parameter that the user can adjust. If the mean of the independent normal distributions are appropriately placed, the resulting problem has $2N + 1$ peaks, out of which one is the robust optimum based on the chosen radius.

The value of the distance parameter affects the location of robust optimum for the test problem, as the function value at each point depends on the placement of different normal distributions. Therefore, if all the normal distribution are far apart so that they do not interfere with each other, then the robust optimum can be found at one of the means of the normal distributions. On the other hand, if this is not the case, then the optimum may not correspond to one of the means. For the test problems used in this paper, the function landscape does not consist of isolated peaks, but instead it is the result of a combination of multiple distributions that interfere with each other. We have provided a plot for 2 and 3 dimensional test problems in Figures 7 and 8 respectively.

In our study we have used the test problems with dimensions 1-5 and 10. In addition to a normal distribution at the center, other independent normal distributions have been placed at a distance of 0.5 units from the center on the positive and negative sides of each axis. Because of interactions between the distributions, the location of the exact optimum is observed to be different from the location of the means of the independent normal distributions. The standard deviation for the peak at the center is kept the lowest, which makes it the highest peak. Therefore, a deterministic algorithm should always converge to the center peak. However, based on the radius of the neighbourhood, the robust optimum may lie on any of the peaks. We have chosen the radius to be 0.125 for each $N-$dimensional problem, that makes the peak close to $((0)^{N-1}, -0.5)$ as the robust optimum. The location of the robust optimum to the accuracy of three decimal places have been reported in Table I.

TABLE I

SUMMARY OF THE TEST PROBLEMS WHEN THE RADIUS OF THE $\delta-$NEIGHBOURHOOD IS 0.125.

Test Problem	Approximate Solution	Dim.	No. of peaks
TP1	(-0.496)	1	3
TP2	(-0.000, -0.498)	2	5
TP3	(-0.003, 0.002, -0.498)	3	7
TP4	(0.001, -0.007, -0.005, -0.496)	4	9
TP5	(0.003, 0.002, 0.004, 0.006, -0.496)	5	11

VI. RESULTS

In this section, we present the results obtained from the RO-Descent algorithm. The algorithm has been evaluated on test problems with dimensions 1 to 10 in terms of the number of function evaluations and accuracy. A comparative study has been performed with two benchmark cases. The first benchmark case helps us evaluate the gains generated from using a descent based crossover, while the second benchmark case helps us evaluate the gains generated as compared to a brute force nested strategy. The two benchmark cases used in this study have been described below:

B1: This approach is exactly similar to our algorithm RO-Descent, except that it does not utilize the information given by the descent cone, but instead takes a step in a random direction from the parent to produce offspring. The crossover operator otherwise works similar to the proposed crossover operators.

B2: This is a brute force approach where we perform a lower level optimization to find the exact robust function value at each point. No descent information is used in the crossover operator and the robust function value at each point is computed using local search with an evolutionary algorithm.

We start by reporting the results for RO-Descent, which are shown in Table II. The algorithm was executed on each test problem 21 times, and the best, median and worst number of function evaluations were recorded. The accuracy was then computed in terms of in how many occasions did the algorithm converge to the correct peak containing the robust optimum. As can be observed in the table, the number of function evaluations increases with the dimensions as the test problems become more difficult and highly multi-modal. The accuracy falls with increasing number of dimensions, but the approach is able to find the right peak with a good percentage even for the 10 dimensional problem that contains 21 peaks.

To find out the advantage of utilizing the descent direction in the crossover operator, we compare our approach with Benchmark 1. The results can be seen in Table III. A comparison with the median function evaluations is also included in the table. The results clearly demonstrate that when the descent direction is used, the accuracy increases and the number of function evaluations decrease. Also, we are not even able to solve the 10 dimensional problem when using the modified operator, which once again suggests that the descent-based crossover is useful.

Next, we have evaluated our approach against Benchmark 2, which is a nested strategy. All the simulations have once again been performed 21 times and the final results are presented in Table IV. It can be seen that the required number of function evaluations for the nested approach is much larger than for RO-Descent approach. This suggests that finding the exact robust function value at each point during the optimization run would prove to be computationally very heavy. This is already well-known and has motivated past efforts on utilizing appropriate sampling techniques to estimate the robust function value.

TABLE II

FUNCTION EVALUATIONS AND ACCURACY FROM 21 RUNS OF RO-DESCENT.

Test Problem	Best	Median	Worst	Accuracy (%)
TP1	1915	2107	4120	95.2
TP2	4620	5324	6490	100.0
TP3	10143	12295	17031	95.2
TP4	21175	26775	32515	85.7
TP5	33443	50933	61851	90.5
TP10	195286	364994	536529	76.2

TABLE III

FUNCTION EVALUATIONS AND ACCURACY FROM 21 RUNS OF RANDOM DIRECTION CROSSOVER.

Test Problem	Best	Median (RO-Descent Savings)	Worst	Accuracy (%)
TP1	1950	2140 (1.0)	4590	81.0
TP2	4609	7238 (1.4)	9339	85.7
TP3	13293	23121 (1.9)	26880	61.9
TP4	20755	44170 (1.6)	66570	52.4
TP5	39008	62752 (1.2)	134673	52.4

TABLE IV

FUNCTION EVALUATIONS FROM 21 RUNS OF NESTED APPROACH.

Test Problem	Best	Median (RO-Descent Savings)	Worst
TP1	177984	177984 (84.5)	224048
TP2	194964	238852 (44.9)	533200
TP3	212544	557700 (45.4)	1174128
TP4	230724	736144 (27.5)	2127384
TP5	249504	847968 (16.6)	2607948

VII. CONCLUSIONS

In this paper, we have proposed a descent-based crossover operator for unconstrained robust optimization. We have also introduced a simple evidence using Taylor's expansion that suggests that moving away from the worst implementation error around a point leads to an improvement in the robust function value. The study is motivated by past efforts in the area of mathematical programming [3], [8]. An evolutionary algorithm designed with this crossover operator is found to solve a number of multi-modal test problems with high accuracy. Our comparative study suggests a large gain in function evaluations as well as accuracy is directly associated to the descent-based crossover strategy. As a future study, we

intend to extend this idea for constrained robust optimization problems and also multi-objective robust optimization problems. Moreover, an intelligent sampling technique needs to be implemented that would lead to a further improvement in the performance of the method.

REFERENCES

[1] DK Anthony and AJ Keane. Robust-optimal design of a lightweight space structure using a genetic algorithm. *AIAA journal*, 41(8):1601–1604, 2003.

[2] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[3] Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo. Robust optimization for unconstrained simulation-based problems. *Operations Research*, 58(1):161–178, 2010.

[4] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33):3190–3218, 2007.

[5] Christopher R Bonham and Ian C Parmee. An investigation of exploration and exploitation within cluster oriented genetic algorithms (cogas). In *GECCO*, pages 1491–1497, 1999.

[6] Jürgen Branke. Creating robust solutions by means of evolutionary algorithms. In *Parallel Problem Solving from NatureâĂŤPPSN V*, pages 119–128. Springer, 1998.

[7] Jürgen Branke. Efficient evolutionary algorithms for searching robust solutions. In *Evolutionary Design and Manufacture*, pages 275–285. Springer, 2000.

[8] John M Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664, 1966.

[9] Horst Greiner. Robust optical coating design with evolutionary strategies. *Applied Optics*, 35(28):5477–5483, 1996.

[10] Mikkel T Jensen. Generating robust and flexible job shop schedules using genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(3):275–288, 2003.

[11] Yaochu Jin and Bernhard Sendhoff. Trade-off between performance and robustness: an evolutionary multiobjective approach. In *Evolutionary Multi-Criterion Optimization*, pages 237–251. Springer, 2003.

[12] V JORGE LEON, S DAVID WU, and Robert H Storer. Robustness measures and robust scheduling for job shops. *IIE transactions*, 26(5):32–43, 1994.

[13] Christopher I Marrison and Robert F Stengel. Robust control system design using random search and genetic algorithms. *Automatic Control, IEEE Transactions on*, 42(6):835–839, 1997.

[14] Tapabrata Ray and Her Mann Tsai. A parallel hybrid optimization algorithm for robust airfoil design. *AIAA Paper*, 905:5–8, 2004.

[15] Colin R Reeves. A genetic algorithm approach to stochastic flowshop sequencing. In *Proceedings of the IEE Colloquium on Genetic Algorithms for Control and Systems Engineering, volume Digest*, number 1992/106, pages 131–134, 1885.

[16] Andrea Saltelli, Karen Chan, E Marian Scott, et al. *Sensitivity analysis*, volume 134. Wiley New York, 2000.

[17] Yasuhito Sano and Hajime Kita. Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 360–365. IEEE, 2002.

[18] Adrian Thompson. Evolutionary techniques for fault tolerance. 1996.

[19] Shigeyoshi Tsutsui and Ashish Ghosh. Genetic algorithms with a robust solution searching scheme. *Evolutionary Computation, IEEE Transactions on*, 1(3):201–208, 1997.

[20] Dirk Wiesmann, Ulrich Hammel, and Thomas Back. Robust design of multilayer optical coatings by means of evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 2(4):162–167, 1998.

[21] Y Yamaguchi and T Arima. Aerodynamic optimization for the transonic compressor stator blade. In *Optimization in Industry*, pages 163–172. Springer, 2002.