Robust Nonconvex Optimization for Simulation-based Problems

Dimitris Bertsimas
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, E40-147, Cambridge, Massachusetts 02139, dbertsim@mit.edu

Omid Nohadani
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, E40-111, Cambridge, Massachusetts 02139, nohadani@mit.edu

Kwong Meng Teo
Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, kmteo@alum.mit.edu

In engineering design, an optimized solution often turns out to be suboptimal, when implementation errors are encountered. While the theory of robust convex optimization has taken significant strides over the past decade, all approaches fail if the underlying cost function is not explicitly given; it is even worse if the cost function is nonconvex. In this work, we present a robust optimization method, which is suited for problems with a nonconvex cost function as well as for problems based on simulations such as large PDE solvers, response surface, and kriging metamodels. Moreover, this technique can be employed for most real-world problems, because it operates directly on the response surface and does not assume any specific structure of the problem. We present this algorithm along with the application to an actual engineering problem in electromagnetic multiple-scattering of aperiodically arranged dielectrics, relevant to nano-photonic design. The corresponding objective function is highly nonconvex and resides in a 100-dimensional design space. Starting from an "optimized" design, we report a robust solution with a significantly lower worst case cost, while maintaining optimality. We further generalize this algorithm to address a nonconvex optimization problem under both implementation errors and parameter uncertainties.

Subject classifications: Robust optimization; Nonconvex Optimization; Robustness; Implementation errors; Data uncertainty
Area of review: Robust Optimization
History: June 2007

1. Introduction

Uncertainty is typically present in real-world applications. Information used to model a problem is often noisy, incomplete or even erroneous. In science and engineering, measurement errors are inevitable. In business applications, the cost and selling price as well as the demand of a product are, at best, expert opinions. Moreover, even if uncertainties in the model data can be ignored, solutions cannot be implemented to infinite precision, as assumed in continuous optimization. Therefore, an "optimal" solution can easily be sub-optimal or, even worse, infeasible. Traditionally, sensitivity analysis was performed to study the impact of perturbations on specific designs. While these approaches can be used to compare different designs, they do not intrinsically find one that is less sensitive, that is, they do not improve the robustness directly.

Stochastic optimization (see Birge and Louveaux 1997, Prekopa and Ruszczynski 2002) is the traditional approach to address optimization under uncertainty. The approach takes a probabilistic approach. The probability distribution of the uncertainties is estimated and incorporated into the model using

1. Chance constraints (i.e. a constraint which is violated less than $p\%$ of the time) (see Charnes and Cooper 1959),

2. Risk measures (i.e. standard deviations, value-at-risk and conditional value-at-risk) (see Markowitz 1952, Park et al. 2006, Ramakrishnan and Rao 1991, Ruszczynski and Shapiro 2004, Uryasev and Rockafellar 2001), or

3. A large number of scenarios emulating the distribution (see Mulvey and Ruszczynski 1995, Rockafellar and Wets 1991).

However, the actual distribution of the uncertainties is seldom available. Take the demand of a product over the coming week. Any specified probability distribution is, at best, an expert's opinion. Furthermore, even if the distribution is known, solving the resulting problem remains a challenge (see Dyer and Stougie 2006). For instance, a chance constraint is usually "computationally intractable" (see Nemirovski 2003).

Robust optimization is another approach towards optimization under uncertainty. Adopting a min-max approach, a robust optimal design is one with the best worst-case performance. Despite significant developments in the theory of robust optimization, particularly over the past decade, a gap remains between the robust techniques developed to date, and problems in the real-world. Current robust methods are restricted to convex problems such as linear, convex quadratic, conic-quadratic and linear discrete problems (see Ben-Tal and Nemirovski 1998, 2003, Bertsimas and Sim 2003, 2006). However, an increasing number of design problems in the real-world, besides being nonconvex, involve the use of computer-based simulations. In simulation-based applications, the relationship between the design and the outcome is not defined as functions used in mathematical programming models. Instead, that relationship is embedded within complex numerical models such as partial differential equation (PDE) solvers (see Ciarlet 2002, Cook et al. 2007), response surface, radial basis functions (see Jin et al. 2001) and kriging metamodels (see Simpson et al. 2001). Consequently, robust techniques found in the literature cannot be applied to these important practical problems.

In this paper, we propose an approach to robust optimization that is applicable to problems whose objective functions are non-convex and given by a numerical simulation driven model. Our proposed method requires only a subroutine which provides the value as well as the gradient of the objective function. Because of this generality, the proposed method is applicable to a wide range of practical problems. To show the practicability of our robust optimization technique, we applied it to an actual nonconvex application in nanophotonic design.

Moreover, the proposed robust local search is analogous to local search techniques, such as gradient descent, which entails finding descent directions and iteratively taking steps along these directions to optimize the nominal cost. The proposed robust local search iteratively takes appropriate steps along descent directions for the robust problem, in order to find robust designs. This analogy continues to hold through the iterations; the robust local search is designed to terminate at a robust local minimum, a point where no improving direction exists. We introduce descent directions and the local minimum of the robust problem; the analogies of these concepts in the optimization theory are important, well studied, and form the building blocks of powerful optimization techniques, such as steepest descent and subgradient techniques. Our proposed framework has the same potential, but for the richer robust problem.

In general, there are two common forms of perturbations: (i) *implementation errors*, which are caused in an imperfect realization of the desired decision variables, and (ii) *parameter uncertainties*, which are due to modeling errors during the problem definition, such as noise. Even though both of these perturbations have been addressed as sources of uncertainty, the case where both are simultaneously present, has not received appropriate attention. For the ease of exposition, we first introduce a robust optimization method for generic nonconvex problems, in order to minimize the worst case cost under implementation errors. We further generalize the method to the case where both implementation errors and parameter uncertainties are present.

Structure of the paper: In Section 2, we define the robust optimization problem with implementation errors and present relevant theoretical results for this problem. Here, we introduce the conditions for descent directions for the robust problem in analogy to the nominal case. In Section 3, we present the local search algorithm. We continue by demonstrating the performance of the algorithm with two application examples. In Section 4, we discuss the application of the algorithm to a problem with a polynomial objective function in order to illustrate the algorithm at work and to provide geometric intuition. In Section 5, we describe an actual electromagnetic scattering design problem with a 100-dimensional design space. This example serves as a showcase of an actual real-world problem with a large decision space. It demonstrates that the proposed robust optimization method improves the robustness significantly, while maintaining optimality of the nominal solution. In Section 6, we generalize the algorithm to the case where both implementation errors and parameter uncertainties are present, discuss the necessary modifications to the problem definition as well as to the algorithm, and present an example. Finally, Section 7 contains our conclusions.

2. The Robust Optimization Problem Under Implementation Errors

First, we define the robust optimization problem with implementation errors. This leads to the notion of the descent direction for the robust problem, which is a vector that points away from all the worst implementation errors. A robust local minimum is a solution at which no such direction exists.

2.1. Problem Definition

The cost function, possibly nonconvex, is denoted by $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$ is the design vector. $f(\mathbf{x})$ denotes the *nominal cost*, because it does not consider possible implementation errors in $\mathbf{x}$. Consequently, the *nominal optimization problem* is

$$\min_{\mathbf{x}} f(\mathbf{x}). \tag{1}$$

When implementing $\mathbf{x}$, additive implementation errors $\Delta\mathbf{x} \in \mathbb{R}^n$ may be introduced due to an imperfect realization process, resulting in an eventual implementation of $\mathbf{x} + \Delta\mathbf{x}$. $\Delta\mathbf{x}$ is assumed to reside within an uncertainty set

$$\mathcal{U} := \{\Delta\mathbf{x} \in \mathbb{R}^n \mid \|\Delta\mathbf{x}\|_2 \leq \Gamma\}. \tag{2}$$

Here, $\Gamma > 0$ is a scalar describing the size of perturbation against which the design needs to be protected. We seek a robust design $\mathbf{x}$ by minimizing the *worst case cost*

$$g(\mathbf{x}) := \max_{\Delta\mathbf{x} \in \mathcal{U}} f(\mathbf{x} + \Delta\mathbf{x}) \tag{3}$$

instead of the nominal cost $f(\mathbf{x})$. The worst case cost $g(\mathbf{x})$ is the maximum possible cost of implementing $\mathbf{x}$ due to an error $\Delta\mathbf{x} \in \mathcal{U}$. Thus, the *robust optimization problem* is given through

$$\min_{\mathbf{x}} g(\mathbf{x}) \equiv \min_{\mathbf{x}} \max_{\Delta\mathbf{x} \in \mathcal{U}} f(\mathbf{x} + \Delta\mathbf{x}). \tag{4}$$

2.2. A Geometric Perspective of the Robust Problem

When implementing a certain design $\mathbf{x} = \hat{\mathbf{x}}$, the possible realization due to implementation errors $\Delta\mathbf{x} \in \mathcal{U}$ lies in the set

$$\mathcal{N} := \{\mathbf{x} \mid \|\mathbf{x} - \hat{\mathbf{x}}\|_2 \leq \Gamma\}. \tag{5}$$

We call $\mathcal{N}$ the *neighborhood* of $\hat{\mathbf{x}}$; such a neighborhood is illustrated in Figure 1. A design $\mathbf{x}$ is a *neighbor* of $\hat{\mathbf{x}}$ if it is in $\mathcal{N}$. Therefore, the worst case cost of $\hat{\mathbf{x}}$, $g(\hat{\mathbf{x}})$, is the maximum cost attained within $\mathcal{N}$. Let $\Delta\mathbf{x}^*$ be one of the worst implementation error at $\hat{\mathbf{x}}$, $\Delta\mathbf{x}^* = \arg\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\hat{\mathbf{x}}+\Delta\mathbf{x})$. Then, $g(\hat{\mathbf{x}})$ is given by $f(\hat{\mathbf{x}}+\Delta\mathbf{x}^*)$. Since we seek to navigate away from all the worst implementation errors, we define the *set of worst implementation errors* at $\hat{\mathbf{x}}$

$$\mathcal{U}^*(\hat{\mathbf{x}}) := \left\{ \Delta\mathbf{x}^* \mid \Delta\mathbf{x}^* = \arg\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\hat{\mathbf{x}}+\Delta\mathbf{x}) \right\}. \tag{6}$$

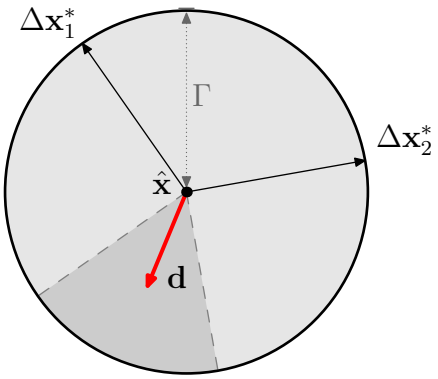

Figure 1 A two dimensional illustration of the neighborhood $\mathcal{N} = \{\mathbf{x} \mid \|\mathbf{x}-\hat{\mathbf{x}}\|_2 \le \Gamma\}$. The shaded circle contains all possible realizations when implementing $\hat{\mathbf{x}}$, when we have errors $\Delta\mathbf{x} \in \mathcal{U}$. The bold arrow $\mathbf{d}$ shows a possible descent direction pointing away from all the worst implementation errors $\Delta\mathbf{x}_i^*$, represented by thin arrows. All the descent directions lie within the cone, which is of a darker shade and demarcated by broken lines.

2.3. Descent Directions and Robust Local Minima

2.3.1. Descent Directions When solving the robust problem, it is useful to take *descent directions* which reduce the worst case cost by excluding worst errors. It is defined as:

Definition 1.
$\mathbf{d}$ is a descent direction for the robust optimization problem (4) at $\mathbf{x}$, if the directional derivative in direction $\mathbf{d}$ satisfies the following condition:

$$g'(\mathbf{x};\mathbf{d}) < 0. \tag{7}$$

The directional derivative at $\mathbf{x}$ in the direction $\mathbf{d}$ is defined as:

$$g'(\mathbf{x};\mathbf{d}) = \lim_{t\downarrow 0} \frac{g(\mathbf{x}+t\mathbf{d}) - g(\mathbf{x})}{t}. \tag{8}$$

Note, that in Problem (4), a directional derivative exists for all $\mathbf{x}$ and for all $\mathbf{d}$ (see Appendix A).

A descent direction $\mathbf{d}$ is a direction which will reduce the worst case cost if it is used to update the design $\mathbf{x}$. We seek an efficient way to find such a direction. The following theorem shows that a descent direction is equivalent to a vector pointing away from all the worst implementation errors in $\mathcal{U}$:

THEOREM 1.
Suppose that $f(\mathbf{x})$ *is continuously differentiable,* $\mathcal{U} = \{\Delta\mathbf{x} \mid \|\Delta\mathbf{x}\|_2 \leq \Gamma\}$ *where* $\Gamma > 0$, $g(\mathbf{x}) := \max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x})$ *and* $\mathcal{U}^*(\mathbf{x}) := \left\{\Delta\mathbf{x}^* \mid \Delta\mathbf{x}^* = \arg\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x})\right\}$. *Then,* $\mathbf{d}\in\mathbb{R}^n$ *is a descent direction for the worst case cost function* $g(\mathbf{x})$ *at* $\mathbf{x}=\hat{\mathbf{x}}$ *if and only if*

$$\begin{aligned}\mathbf{d}'\Delta\mathbf{x}^* &< 0,\\ \nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} &\neq \mathbf{0},\end{aligned}$$

for all $\Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}})$.

Note, that the condition $\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} \neq \mathbf{0}$, or $\hat{\mathbf{x}}+\Delta\mathbf{x}^*$ not being a unconstrained local maximum of $f(\mathbf{x})$ is equivalent to the condition $\|\Delta\mathbf{x}^*\|_2 = \Gamma$. Figure 1 illustrates a possible scenario under Theorem 1. All the descent directions $\mathbf{d}$ lie in the strict interior of a cone, the normal of the cone spanned by all the vectors $\Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}})$. Consequently, all descent directions point away from all the worst implementation errors. From $\hat{\mathbf{x}}$, the worst case cost can be strictly decreased if we take a sufficiently small step along any directions within this cone, leading to solutions that are more robust. All the worst designs, $\hat{\mathbf{x}}+\Delta\mathbf{x}^*$, would also lie outside the neighborhood of the new design.

The detailed proof of Theorem 1 is presented in Appendix B. The main ideas behind the proof are

(i) the directional derivative of the worst case cost function, $g'(\mathbf{x};\mathbf{d})$, equals the maximum value of $\mathbf{d}'\nabla_{\mathbf{x}} f(\mathbf{x}+\Delta\mathbf{x}^*)$, for all $\Delta\mathbf{x}^*$ (See Corollary 1(a)), and

(ii) the gradient at $\mathbf{x}+\Delta\mathbf{x}^*$ is parallel to $\Delta\mathbf{x}^*$, due to the Karush-Kuhn-Tucker conditions (See Proposition 3).

Therefore, in order for $g'(\mathbf{x};\mathbf{d}) < 0$, we require $\mathbf{d}'\Delta\mathbf{x}^* < 0$ and $\nabla_{\mathbf{x}} f(\mathbf{x}+\Delta\mathbf{x}^*) \neq \mathbf{0}$, for all $\Delta\mathbf{x}^*$. The intuition behind Theorem 1 is: we have to move sufficiently far away from all the designs $\hat{\mathbf{x}}+\Delta\mathbf{x}^*$ for there to be a chance to decrease the worst case cost.

2.3.2. Robust Local Minima Definition 1 for a descent direction leads naturally to the following concept of a robust local minimum:

DEFINITION 2.
$\mathbf{x}^*$ is a robust local minimum if there exists no descent directions for the robust problem at $\mathbf{x}=\mathbf{x}^*$.

Similarly, Theorem 1 easily leads the following characterization of a robust local minimum:

PROPOSITION 1 **(Robust Local Minimum)**.
Suppose that $f(\mathbf{x})$ *is continuously differentiable. Then,* $\mathbf{x}^*$ *is a robust local minimum if and only if either one of the following two conditions are satisfied:*

i. there does not exist a $\mathbf{d}\in\mathbb{R}^n$ *such that for all* $\Delta\mathbf{x}^* \in \mathcal{U}^*(\mathbf{x}^*)$,

$$\mathbf{d}'\Delta\mathbf{x}^* < 0,$$

ii. there exists a $\Delta\mathbf{x}^* \in \mathcal{U}^*(\mathbf{x}^*)$ *such that* $\nabla_{\mathbf{x}} f(\mathbf{x}+\Delta\mathbf{x}^*) = \mathbf{0}$.

Given Proposition 1, we illustrate common types of robust local minima, where either one of the two conditions are satisfied.

Convex case. If f is convex, there are no local maxima in f and therefore, $\nabla_{\mathbf{x}} f(\mathbf{x}+\Delta\mathbf{x}^*) = \mathbf{0}$ is never satisfied. The only condition for the lack of descent direction is (i) where there are no $\mathbf{d}$ satisfying the condition $\mathbf{d}'\Delta\mathbf{x}_i^* < 0$, as shown in Fig. 2(a). Furthermore, if f is convex, g is convex (see Corollary 1(b)). Thus, a robust local minimum of g is a robust global minimum of g.

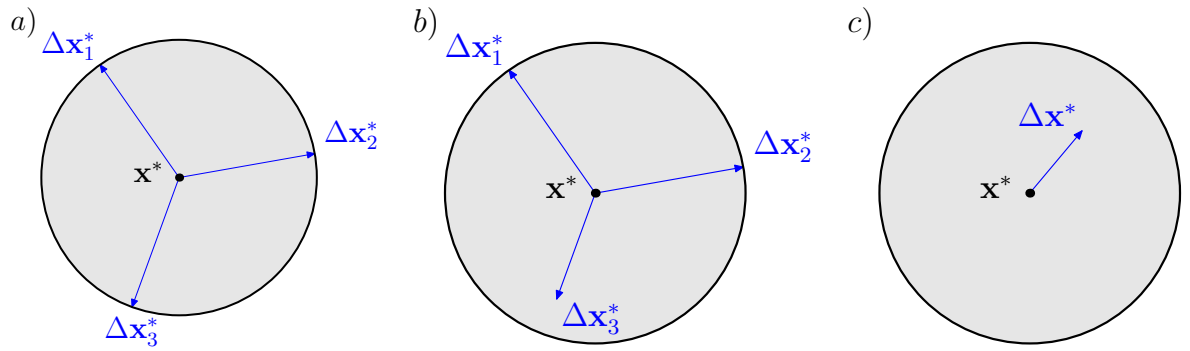

Figure 2 A two-dimensional illustration of common types of robust local minima. In (a) and (b), there are no direction pointing away from all the worst implementation errors $\Delta\mathbf{x}_i^*$, which are denoted by arrows. In (b) and (c), one of the worst implementation errors $\Delta\mathbf{x}_i^*$ lie in the strict interior of the neighborhood. Note, for convex problems, the robust local (global) minimum is of the type shown in (a).

General case. Three common types of robust local minimum can be present when f is nonconvex, as shown in Figure 2. Condition (i) in Proposition 1, that there are no direction pointing away from all the worst implementation errors $\Delta\mathbf{x}_i^*$, is satisfied by both the robust local minimum in Fig. 2(a) and Fig. 2(b). Condition (ii), that one of the worst implementation errors $\Delta\mathbf{x}_i^*$ lie in the strict interior of the neighborhood, is satisfied by Fig. 2(b) and Fig. 2(c).

Compared to the others, the "robust local minimum" of the type in Fig. 2(c) may not be as good a robust design, and can actually be a bad robust solution. For instance, we can find many such "robust local minima" near the global maximum of the nominal cost function $f(\mathbf{x})$, i.e. when $\mathbf{x}^*+\Delta\mathbf{x}^*$ is the global maximum of the nominal problem. Therefore, we seek a robust local minimum satisfying Condition (i), that there does not exist a direction pointing away from all the worst implementation errors.

The following algorithm seeks such a desired robust local minimum. We further show the convergence result in the case where f is convex.

2.4. A Local Search Algorithm for the Robust Optimization Problem

Given the set of worst implementation errors at $\hat{\mathbf{x}}$, $\mathcal{U}^*(\hat{\mathbf{x}})$, a descent direction can be found efficiently by solving the following second-order cone program (SOCP):

$$\begin{aligned} \min_{\mathbf{d},\beta}\ & \beta \\ s.t.\ & \|\mathbf{d}\|_2 \le 1 \\ & \mathbf{d}'\Delta\mathbf{x}^* \le \beta \quad \forall \Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}}) \\ & \beta \le -\epsilon, \end{aligned} \tag{9}$$

where ϵ is a small positive scalar. When Problem (9) has a feasible solution, its optimal solution, $\mathbf{d}^*$, forms the maximum possible angle $\theta_{\max}$ with all $\Delta\mathbf{x}^*$. An example is illustrated in Fig. 3. This angle is always greater than 90° due to the constraint $\beta \le -\epsilon < 0$. $\beta \le 0$ is not used in place of $\beta \le -\epsilon$, because we want to exclude the trivial solution $(\mathbf{d}^*, \beta^*) = (\mathbf{0}, 0)$. When ϵ is sufficiently small, and Problem (9) is infeasible, $\hat{\mathbf{x}}$ is a good estimate of a robust local minimum satisfying Condition (i) in Proposition 1. Note, that the constraint $\|\mathbf{d}^*\|_2 = 1$ is automatically satisfied if the problem is feasible. Such an SOCP can be solved efficiently using both commercial and noncommercial solvers.

Consequently, if we have an oracle returning $\mathcal{U}^*(\mathbf{x})$ for all $\mathbf{x}$, we can iteratively find descent directions and use them to update the current iterates, resulting in the following local search algorithm. The term $\mathbf{x}^k$ is the term being evaluated in iteration k.

Algorithm 1.

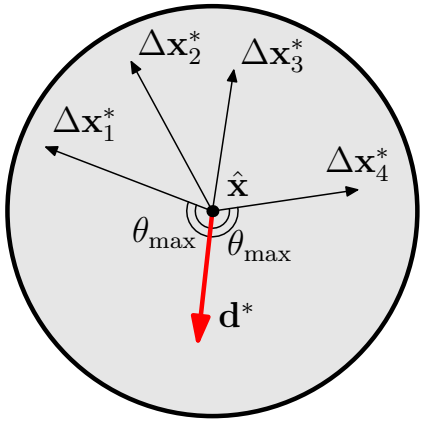

Figure 3 A two-dimensional illustration of the optimal solution of SOCP, Prob. (9), in the neighborhood of $\hat{\mathbf{x}}$. The solid arrow indicates the optimal direction $\mathbf{d}^*$ which makes the largest possible angle θ_{max} with all the vectors $\Delta\mathbf{x}^*$, $\Delta\mathbf{x}^*$ being the worst case implementation errors at $\hat{\mathbf{x}}$. The angle $\theta_{\max} = \cos^{-1}\beta^*$ and is at least 90^o due to the constraint $\beta \leq -\epsilon$, ϵ being a small positive scalar.

Step 0. Initialization: Let $\mathbf{x}^1$ be the initial decision vector arbitrarily chosen. Set $k := 1$.
Step 1. Neighborhood Exploration :
Find $\mathcal{U}^*(\mathbf{x}^k)$, set of worst implementation errors at the current iterate $\mathbf{x}^k$.
Step 2. Robust Local Move :
(i) Solve the SOCP (Problem 9), terminating if the problem is infeasible.
(ii) Set $\mathbf{x}^{k+1} := \mathbf{x}^k + t^k\mathbf{d}^*$, where $\mathbf{d}^*$ is the optimal solution to the SOCP.
(iii) Set $k := k+1$. Go to Step 1.

If $f(\mathbf{x})$ is continuously differentiable and convex, Algorithm 1 converges to the robust global minimum when appropriate step size t^k are chosen. This is reflected by the following theorem:

THEOREM 2. *Suppose that $f(\mathbf{x})$ is continuously differentiable and convex with a bounded set of minimum points. Then, Algorithm 1 converges to the global optimum of the robust optimization problem (4), when $t^k > 0$, $t^k \to 0$ as $k \to \infty$ and $\sum_{k=1}^{\infty} t^k = \infty$.*

This Theorem follows from the fact that at every iteration, $-\mathbf{d}^*$ is a subgradient of the worst cost function $g(\mathbf{x})$ at the iterate $\mathbf{x}^k$. Therefore, Algorithm 1 is a subgradient projection algorithm, and under the stated step size rule, convergence to the global minimum is assured. A detailed proof to Theorem 2 is presented in Appendix C.

2.5. Practical Implementation

Finding the set of worst implementation errors $\mathcal{U}^*(\hat{\mathbf{x}})$ equates to finding all the global maxima of the inner maximization problem

$$\max_{\|\Delta\mathbf{x}\|_2 \leq \Gamma} f(\hat{\mathbf{x}} + \Delta\mathbf{x}). \tag{10}$$

Even though there is no closed-form solution in general, it is possible to find $\Delta\mathbf{x}^*$ in instances where the problem has a small dimension and $f(\mathbf{x})$ satisfies a Lipschtiz condition (see Horst and Pardalos 1995). Furthermore, when $f(\mathbf{x})$ is a polynomial function, numerical experiments suggest that $\Delta\mathbf{x}^*$ can be found for many problems in the literature on global optimization (Henrion and Lasserre 2003). If $\Delta\mathbf{x}^*$ can be found efficiently, the descent directions can be determined. Consequently, the robust optimization problem can be solved readily using Algorithm 1.

In most real-world instances, however, we cannot expect to find $\Delta\mathbf{x}^*$. Therefore, an alternative approach is required. Fortunately, the following proposition shows that we do not need to know $\Delta\mathbf{x}^*$ exactly in order to find a descent direction.

PROPOSITION 2.
Suppose that $f(\mathbf{x})$ is continuously differentiable and $\|\Delta\mathbf{x}^*\|_2 = \Gamma$, for all $\Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}})$. Let $\mathcal{M} := \{\Delta\mathbf{x}_1, \ldots, \Delta\mathbf{x}_m\}$ be a collection of $\Delta\mathbf{x}_i \in \mathcal{U}$, where there exists scalars $\alpha_i \geq 0$, $i = 1, \ldots, m$ such that

$$\Delta\mathbf{x}^* = \sum_{i|\Delta\mathbf{x}_i \in \mathcal{M}} \alpha_i \Delta\mathbf{x}_i \tag{11}$$

for all $\Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}})$. Then, $\mathbf{d}$ is a descent direction for the worst case cost function $g(\mathbf{x} = \hat{\mathbf{x}})$, if

$$\mathbf{d}'\Delta\mathbf{x}_i < 0, \, \forall \Delta\mathbf{x}_i \in \mathcal{M}\,. \tag{12}$$

PROOF. Given conditions (11) and (12),

$$\mathbf{d}'\Delta\mathbf{x}^* = \sum_{i|\Delta\mathbf{x}_i \in \mathcal{M}} \alpha_i \mathbf{d}'\Delta\mathbf{x}_i < 0,$$

we have $\Delta\mathbf{x}^{*'}\mathbf{d} < 0$, for all $\Delta\mathbf{x}^*$ in set $\mathcal{U}^*(\hat{\mathbf{x}})$. Since the "sufficient" conditions in Theorem 1 are satisfied, the result follows. □

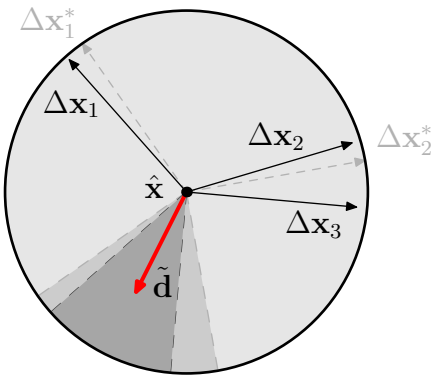

Figure 4 The solid bold arrow indicates a direction $\tilde{\mathbf{d}}$ pointing away from all the implementation errors $\Delta\mathbf{x}_j \in \mathcal{M}$, for $\mathcal{M}$ defined in Proposition 2. $\tilde{\mathbf{d}}$ is a descent direction if all the worst errors $\Delta\mathbf{x}_i^*$ lie within the cone spanned by $\Delta\mathbf{x}_j$. All the descent directions pointing away from $\Delta\mathbf{x}_j$ lie within the cone with the darkest shade, which is a subset of the cone illustrated in Fig. 1.

Proposition 2 shows that descent directions can be found without knowing the worst implementation errors $\Delta\mathbf{x}^*$ exactly. As illustrated in Fig. 4, finding a set $\mathcal{M}$ such that all the worst errors $\Delta\mathbf{x}^*$ are confined to the sector demarcated by $\Delta\mathbf{x}_i \in \mathcal{M}$ would suffice. The set $\mathcal{M}$ does not have to be unique and if it satisfies Condition (11), the cone of descent directions pointing away from $\Delta\mathbf{x}_i \in \mathcal{M}$ is a subset of the cone of directions pointing away from $\Delta\mathbf{x}^*$.

Because $\Delta\mathbf{x}^*$ usually reside among designs with nominal costs higher than the rest of the neighborhood, the following algorithm embodies a heuristic strategy to finding a more robust neighbor:

ALGORITHM 2.
Step 0. Initialization: Let $\mathbf{x}^1$ be an arbitrarily chosen initial decision vector. Set $k := 1$.
Step 1. Neighborhood Exploration :

Find $\mathcal{M}^k$, a set containing implementation errors $\Delta\mathbf{x}_i$ indicating where the highest cost is likely to occur within the neighborhood of $\mathbf{x}^k$.

Step 2. Robust Local Move :
(i) Solve a SOCP (similar to Problem 9, but with the set $\mathcal{U}^*(\mathbf{x}^k)$ replaced by set $\mathcal{M}^k$), terminating if the problem is infeasible.
(ii) Set $\mathbf{x}^{k+1} := \mathbf{x}^k + t^k\mathbf{d}^*$, where $\mathbf{d}^*$ is the optimal solution to the SOCP.
(iii) Set $k := k+1$. Go to Step 1.

This algorithm is the robust local search, to be elaborated upon in the next section.

3. Local Search Algorithm when Implementation Errors are Present

The robust local search method is an iterative algorithm with two parts in every iteration. In the first part, we explore the neighborhood of the current iterate both to estimate its worst case cost and to collect neighbors with high cost. Next, this knowledge of the neighborhood is used to make a robust local move, a step in the descent direction of the robust problem. These two parts are repeated iteratively until termination conditions are met, which is when a suitable descent direction cannot be found anymore. We now discuss these two parts in more detail.

3.1. Neighborhood Exploration

In this subsection, we describe a generic neighborhood exploration algorithm employing $n+1$ gradient ascents from different starting points within the neighborhood. When exploring the neighborhood of $\hat{\mathbf{x}}$, we are essentially trying to solve the inner maximization problem (10).

We first apply a gradient ascent with a diminishing step size. The initial step size used is $\frac{\Gamma}{5}$, decreasing with a factor of 0.99 after every step. The gradient ascent is terminated after either the neighborhood is breached or a time-limit is exceeding. Then, we use the last point that is inside the neighborhood as an initial solution to solve the following sequence of unconstrained problems using gradient ascents:

$$\max_{\Delta\mathbf{x}} \; f(\hat{\mathbf{x}}+\Delta\mathbf{x}) + \epsilon_r \ln\{\Gamma - \|\Delta\mathbf{x}\|_2\}. \tag{13}$$

The positive scalar ϵ_r is chosen so that the additional term $\epsilon_r \ln\{\Gamma - \|\Delta\mathbf{x}\|_2\}$ projects the gradient step back into the strict interior of the neighborhood, so as to ensure that the ascent stays strictly within it. A good estimate of a local maximum is found quickly this way.

Such an approach is modified from a barrier method on the inner maximization problem (10). Under the standard barrier method, one would solve a sequence of Problem (13) using gradient ascents, where ϵ_r are small positive diminishing scalars, $\epsilon_r \to 0$ as $r \to \infty$. However, empirical experiments indicate that using the standard method, the solution time required to find a local maximum is unpredictable and can be very long. Since (i) we want the time spent solving the neighborhood exploration subproblem to be predictable, and (ii) we do not have to find the local maximum exactly, as indicated by Proposition 2, the standard barrier method was not used. Our approach gives a high quality estimate of a local maximum efficiently.

The local maximum obtained using a single gradient ascent can be an inferior estimate of the global maximum when the cost function is nonconcave. Therefore, in every neighborhood exploration, we solve the inner maximization problem (10) using multiple gradient ascents, each with a different starting point. A generic neighborhood exploration algorithm is: for a n-dimensional problem, use $n+1$ gradient ascents starting from $\Delta\mathbf{x} = \mathbf{0}$ and $\Delta\mathbf{x} = sign(\frac{\partial f(\mathbf{x}=\hat{\mathbf{x}})}{\partial x_i})\frac{\Gamma}{3}\mathbf{e}_i$ for $i = 1,\ldots,n$, where $\mathbf{e}_i$ is the unit vector along the i-th coordinate.

During the neighborhood exploration in iteration k, the results of all function evaluations $(\mathbf{x}, f(\mathbf{x}))$ made during the multiple gradient ascents are recorded in a history set $\mathcal{H}^k$, together with all past histories. This history set is then used to estimate the worst case cost of $\mathbf{x}^k$, $\tilde{g}(\mathbf{x}^k)$.

3.2. Robust Local Move

In the second part of the robust local search algorithm, we update the current iterate with a local design that is more robust, based on our knowledge of the neighborhood $\mathcal{N}^k$. The new iterate is found by finding a direction and a distance to take, so that all the neighbors with high cost will be excluded from the new neighborhood. In the following, we discuss in detail how the direction and the distance can be found efficiently.

3.2.1. Finding the Direction To find the direction at $\mathbf{x}^k$ which improves $\tilde{g}(\mathbf{x}^k)$, we include all known neighbors with high cost from $\mathcal{H}^k$ in the set

$$\mathcal{M}^k := \left\{ \mathbf{x} \mid \mathbf{x} \in \mathcal{H}^k, \mathbf{x} \in \mathcal{N}^k, f(\mathbf{x}) \geq \tilde{g}(\mathbf{x}^k) - \sigma^k \right\}. \tag{14}$$

The cost factor σ^k governs the size of the set and may be changed within an iteration to ensure a feasible move. In the first iteration, σ^1 is first set to $0.2 \times \left(\tilde{g}(\mathbf{x}^1) - f(\mathbf{x}^1)\right)$. In subsequent iterations, σ^k is set using the final value of σ^{k-1}.

The problem of finding a good direction $\mathbf{d}$, which points away from bad neighbors as collected in $\mathcal{M}^k$, can be formulated as a SOCP

$$\begin{aligned} \min_{\mathbf{d},\beta} \ & \beta \\ s.t. \ & \|\mathbf{d}\|_2 \leq 1 \\ & \mathbf{d}' \left(\tfrac{\mathbf{x}_i - \mathbf{x}^k}{\|\mathbf{x}_i - \mathbf{x}^k\|_2} \right) \leq \beta \quad \forall \mathbf{x}_i \in \mathcal{M}^k \\ & \beta \leq -\epsilon, \end{aligned} \tag{15}$$

where ϵ is a small positive scalar. The discussion for the earlier SOCP (9) applies to this SOCP as well.

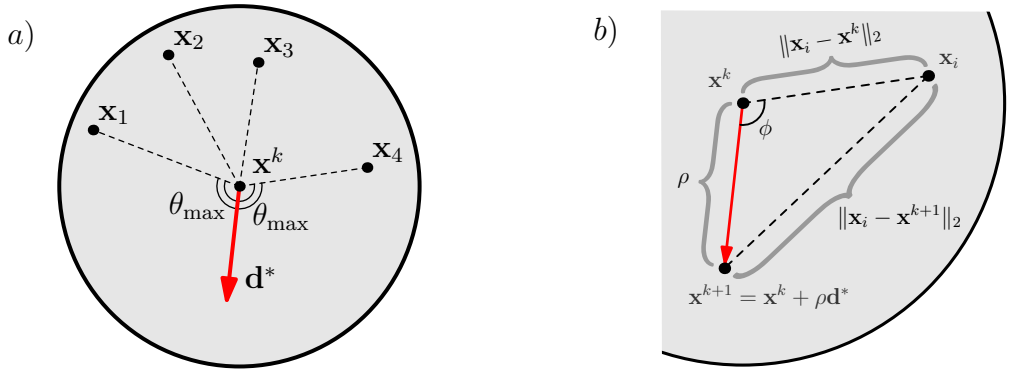

Figure 5 a) A two-dimensional illustration of the optimal solution of the SOCP, Prob. (15). Compare with Fig. 3. b) Illustration showing how the distance $\|\mathbf{x}_i - \mathbf{x}^{k+1}\|_2$ can be found by cosine rule using ρ, $\mathbf{d}^*$ and $\|\mathbf{x}_i - \mathbf{x}^k\|_2$ when $\mathbf{x}^{k+1} = \mathbf{x}^k + \rho\mathbf{d}^*$. $\cos\phi = \rho(\mathbf{x}_i - \mathbf{x}^k)'\mathbf{d}^*$.

We want to relate Problem (15) with the result in Proposition 2. Note, that $\mathbf{x}_i - \mathbf{x}^k = \Delta\mathbf{x}_i \in \mathcal{U}$ and $\|\mathbf{x}_i - \mathbf{x}^k\|$ is a positive scalar, assuming $\mathbf{x}_i \neq \mathbf{x}^k$. Therefore, the constraint $\mathbf{d}' \left(\frac{\mathbf{x}_i - \mathbf{x}^k}{\|\mathbf{x}_i - \mathbf{x}^k\|} \right) \leq \beta < 0$ maps to the condition $\mathbf{d}'\Delta\mathbf{x}_i < 0$ in Proposition 2, while the set $\mathcal{M}^k$ maps to the set $\mathcal{M}$. Comparison between Fig. 3 and Fig. 5(a) shows that we can find a descent direction pointing away from all the implementation errors with high costs. Therefore, if we have a sufficiently detailed knowledge of the neighborhood, $\mathbf{d}^*$ is a descent direction for the robust problem.

When Problem (15) is infeasible, $\mathbf{x}^k$ is surrounded by "bad" neighbors. However, since we may have been too stringent in classifying the bad neighbors, we reduce σ^k, reassemble $\mathcal{M}^k$, and solve

the updated SOCP. When reducing σ^k, we divide it by a factor of 1.05. The terminating condition is attained, when the SOCP is infeasible and σ^k is below a threshold. If $\mathbf{x}^k$ is surrounded by "bad" neighbors and σ^k is small, we presume that we have attained a robust local minimum, of the type as illustrated in Fig. 2(a). and Fig. 2(b).

3.2.2. Finding the Distance After finding the direction $\mathbf{d}^*$, we want to choose the smallest stepsize ρ^* such that every element in the set of bad neighbors $\mathcal{M}^k$ would lie at least on the boundary of the neighborhood of the new iterate, $\mathbf{x}^{k+1} = \mathbf{x}^k + \rho^*\mathbf{d}^*$. To make sure that we make meaningful progress at every iteration, we set a minimum stepsize of $\frac{\Gamma}{100}$ in the first iteration, and decreases it successively by a factor of 0.99.

Figure 5(b) illustrates how $\|\mathbf{x}_i - \mathbf{x}^{k+1}\|_2$ can be evaluated when $\mathbf{x}^{k+1} = \mathbf{x}^k + \rho\mathbf{d}^*$ since

$$\|\mathbf{x}_i - \mathbf{x}^{k+1}\|_2^2 = \rho^2 + \|\mathbf{x}_i - \mathbf{x}^k\|_2^2 - 2\rho(\mathbf{x}_i - \mathbf{x}^k)'\mathbf{d}^*.$$

Consequently,

$$\begin{aligned} \rho^* = \arg\min_{\rho}\ & \rho \\ s.t.\ & \rho \geq \mathbf{d}^{*'}(\mathbf{x}_i - \mathbf{x}^k) + \sqrt{(\mathbf{d}^{*'}(\mathbf{x}_i - \mathbf{x}^k))^2 - \|\mathbf{x}_i - \mathbf{x}^k\|_2^2 + \Gamma^2}, \quad \forall \mathbf{x}_i \in \mathcal{M}^k. \end{aligned} \tag{16}$$

Note, that this problem can be solved with $|\mathcal{M}^k|$ function evaluations without resorting to a formal optimization procedure.

3.2.3. Checking the Direction Knowing that we aim to take the update direction $\mathbf{d}^*$ and a stepsize ρ^*, we update the set of bad neighbors with the set

$$\mathcal{M}^k_{updated} := \left\{ \mathbf{x} \mid \mathbf{x} \in \mathcal{H}^k, \|\mathbf{x} - \mathbf{x}^k\|_2 \leq \Gamma + \rho^*, f(\mathbf{x}) \geq \tilde{g}(\mathbf{x}^k) - \sigma^k \right\}. \tag{17}$$

This set will include all the known neighbors lying slightly beyond the neighborhood, and with a cost higher than $\tilde{g}(\mathbf{x}^k) - \sigma^k$.

We check whether the desired direction $\mathbf{d}^*$ is still a descent direction pointing away from all the members in set $\mathcal{M}^k_{updated}$. If it is, we accept the update step $(\mathbf{d}^*, \rho^*)$ and proceed with the next iteration. If $\mathbf{d}^*$ is not a descent direction for the new set, we repeat the robust local move by solving the SOCP (15) but with $\mathcal{M}^k_{updated}$ in place of $\mathcal{M}^k$. Again, the value σ^k might be decreased in order to find a feasible direction. Consequently, within an iteration, the robust local move might be attempted several times. From computational experience, this additional check becomes more important as we get closer to a robust local minimum.

4. Application Example I - Robust Polynomial Optimization Problem

4.1. Problem Description

For the first problem, we chose a polynomial problem. Having only two dimensions, we can illustrate the cost surface over the domain of interest to develop intuition into the algorithm. Consider the nonconvex polynomial function

$$\begin{aligned} f_{poly}(x,y) = {} & 2x^6 - 12.2x^5 + 21.2x^4 + 6.2x - 6.4x^3 - 4.7x^2 + y^6 - 11y^5 + 43.3y^4 - 10y - 74.8y^3 \\ & + 56.9y^2 - 4.1xy - 0.1y^2x^2 + 0.4y^2x + 0.4x^2y. \end{aligned}$$

Given implementation errors $\Delta = (\Delta x, \Delta y)$ where $\|\Delta\|_2 \leq 0.5$, the robust optimization problem is

$$\min_{x,y} g_{poly}(x,y) \equiv \min_{x,y} \max_{\|\Delta\|_2 \leq 0.5} f_{poly}(x + \Delta x, y + \Delta y). \tag{18}$$

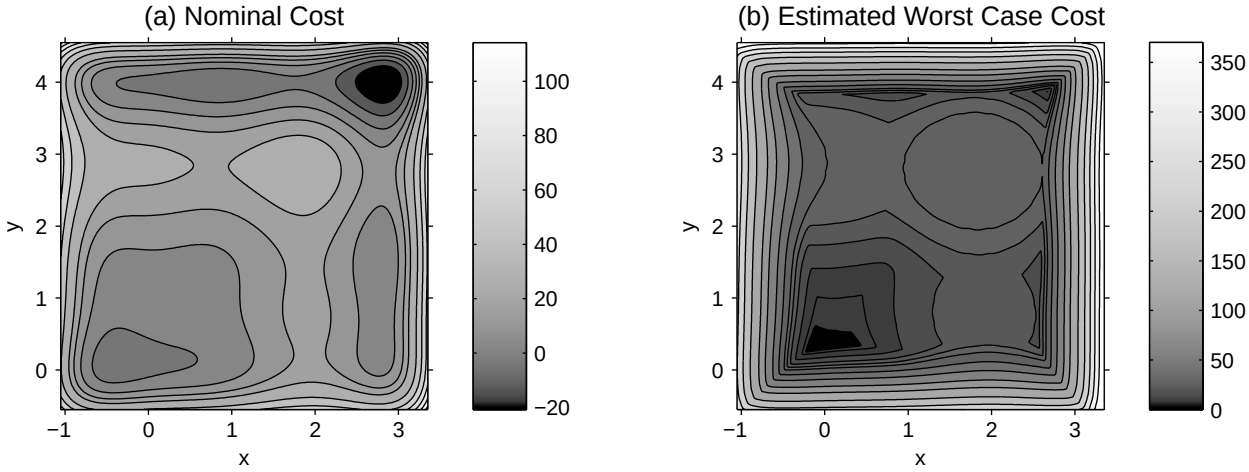

Figure 6 Contour plot of nominal cost function $f_{poly}(x, y)$ and the estimated worst case cost function $g_{poly}(x, y)$ in Application Example I.

Note that even though this problem has only two dimensions, it is already a difficult problem. Recently, relaxation methods have been applied successfully to solve polynomial optimization problems (Henrion and Lasserre 2003). Applying the same technique to Problem (18), however, leads to polynomial semidefinite programs (SDP), where the entries of the semidefinite constraint are made up of multivariate polynomials. Solving a problem approximately involves converting it into a substantially larger SDP, the size of which increases very rapidly with the size of the original problem, the maximum degree of the polynomials involved, and the number of variables. This prevents polynomial SDPs from being used widely in practice (see Kojima 2003). Therefore, we applied the local search algorithm on Problem (18).

4.2. Computation Results

Figure 6(a) shows a contour plot of the nominal cost of $f_{poly}(x, y)$. It has multiple local minima and a global minimum at $(x^*, y^*) = (2.8, 4.0)$, where $f(x^*, y^*) = -20.8$. The global minimum is found using the Gloptipoly software as discussed in Reference (Henrion and Lasserre 2003) and verified using multiple gradient descents. The worst case cost function $g_{poly}(x, y)$, estimated by evaluating discrete neighbors using data in Fig. 6(a), is shown in Fig. 6(b). Fig. 6(b) suggests that $g_{poly}(x, y)$ has multiple local minima.

We applied the robust local search algorithm in this problem using 2 initial design (x, y), A and B; terminating when the SOCP (See Prob. (15)) remains infeasible when σ^k is decreased below the threshold of 0.001. Referring to Fig. 7, Point A is a local minimum of the nominal problem, while B is arbitrarily chosen. Fig. 7(a) and Fig. 7(c) show that the algorithm converges to the same robust local minimum from both starting points. However, depending on the problem, this observation cannot be generalized. Figure 7(b) shows that the worst case cost of A is much higher than its nominal cost, and clearly a local minimum to the nominal problem need not be a robust local minimum. The algorithm decreases the worst case cost significantly while increasing the nominal cost slightly. A much lower number of iterations is required when starting from point A when compared to starting from point B. As seen in Fig. 7(d), both the nominal and the worst case costs decrease as the iteration count increases when starting from point B. While the decrease in worst case costs is not monotonic for both instances, the overall decrease in the worst case cost is significant.

Figure 8 shows the distribution of the bad neighbors upon termination. At termination, these neighbors lie on the boundary of the uncertainty set. Note, that there is no good direction to move

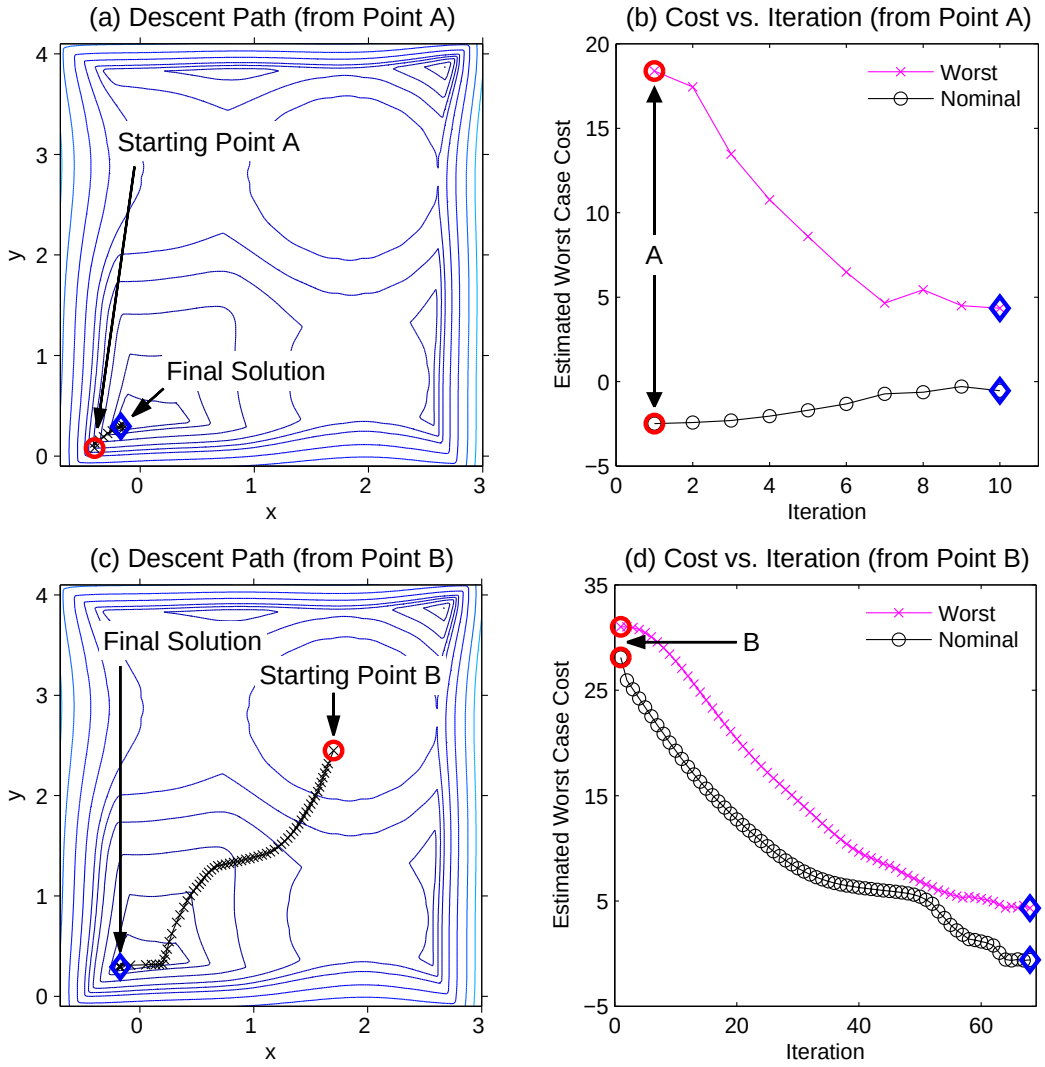

Figure 7 Performance of the robust local search algorithm in Application Example I from 2 different starting points A and B. The circle marker and the diamond marker denote the starting point and the final solution, respectively. (a) The contour plot showing the estimated surface of the worst case cost, $g_{poly}(x, y)$. The descent path taken to converge at the robust solution is shown; point A is a local minimum of the nominal function. (b) From starting point A, the algorithm reduces the worst case cost significantly while increasing the nominal cost slightly. (c) From an arbitrarily chosen starting point B, the algorithm converged at the same robust solution as starting point A. (d) Starting from point B, both the worst case cost and the nominal cost are decreased significantly under the algorithm.

the robust design away from these bad neighbors, so as to lower the disc any further. The bad neighbors form the support of the discs. Compare these figures with Fig. 2(a) where Condition (i) of Proposition 1 was met, indicating the arrival at a robust local minimum. The surface plot of the nominal cost function in Fig. 8 further confirms that the terminating solutions are close to a true robust local minimum.

5. Application Example II - Electromagnetic Scattering Design Problem

The search for attractive and novel materials in controlling and manipulating electromagnetic field propagation has identified a plethora of unique characteristics in photonic crystals (PCs). Their novel functionalities are based on diffraction phenomena, which require periodic structures. Upon

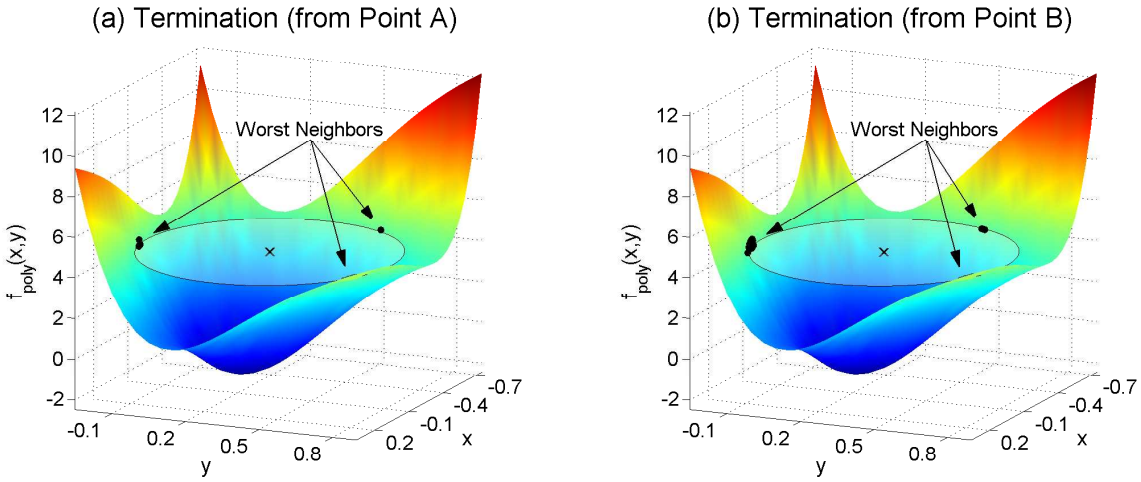

Figure 8 Surface plot shows the cost surface of the nominal function $f_{poly}(x, y)$. The same robust local minimum, denoted by the cross, is found from both starting points A and B. Point A is a local minimum of the nominal function, while point B is arbitrarily chosen. The worst neighbors are indicated by black dots. At termination, these neighbors lie on the boundary of the uncertainty set, which is denoted by the transparent discs. At the robust local minimum, with the worst neighbors forming the "supports", both discs cannot be lowered any further. Compare these figures with Fig. 2(a) where the condition of a robust local minimum is met

breaking the spatial symmetry, new degrees of freedom are revealed which allow for additional functionality and, possibly, for higher levels of control. More recently, unbiased optimization schemes were performed on the spatial distribution (aperiodic) of a large number of identical dielectric cylinders (see Gheorma et al. 2004, Seliger et al. 2006) While these works demonstrate the advantage of optimization, the robustness of the solutions still remains an open issue. In this section, we apply the robust optimization method to electromagnetic scattering problems with large degrees of freedom, and report on novel results when this technique is applied to optimization of aperiodic dielectric structures.

5.1. Problem Description

The incoming electromagnetic field couples in its lowest mode to the perfectly conducting metallic wave-guide. Figure 9(a) sketches the horizontal set-up. In the vertical direction, the domain is bound by two perfectly conducting plates, which are separated by less than $1/2$ the wave length, in order to warrant a two-dimensional wave propagation. Identical dielectric cylinders are placed in the domain between the plates. The sides of the domain are open in the forward direction. In order to account for a finite total energy and to warrant a realistic decay of the field at infinity, the open sides are modeled by perfectly matching layers. (see Kingsland et al. 2006, Berenger 1996) The objective of the optimization is to determine the position of the cylinders such that the forward electromagnetic power matches the shape of a desired power distribution, as shown in Fig. 9(b).

As in the experimental measurements, the frequency is fixed to $f = 37.5$ GHz. (see Seliger et al. 2006) Furthermore, the dielectric scatterers are nonmagnetic and lossless. Therefore, stationary solutions of the Maxwell equations are given through the two-dimensional Helmholtz equations, taking the boundary conditions into account. This means, that only the z-component of the electric field E_z can propagate in the domain. The magnitude of E_z in the domain is given through the partial differential equation (PDE)

$$(\partial_x(\mu_{r_y}^{-1}\partial_x) + \partial_y(\mu_{r_x}^{-1}\partial_y))E_z - \omega_0^2\mu_0\epsilon_0\epsilon_{r_z}E_z = 0 \tag{19}$$

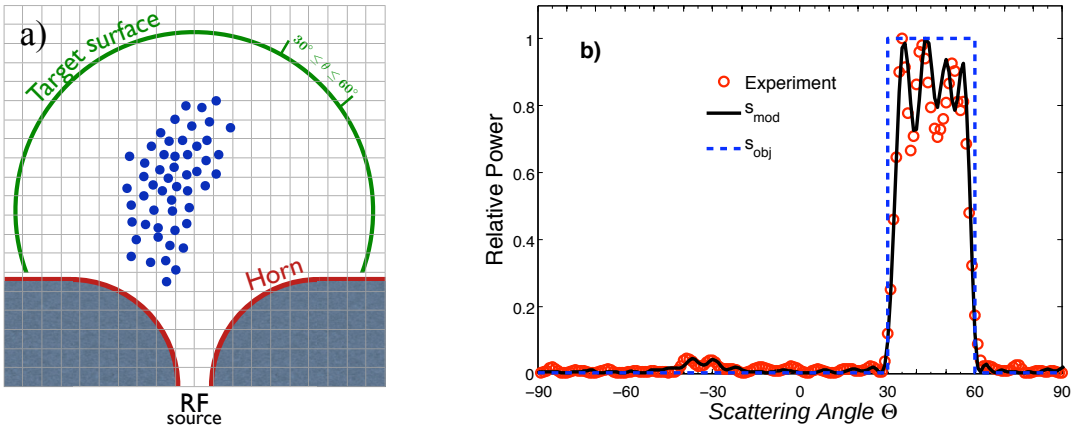

Figure 9 (a) schematic setup: the RF-source couples to the wave guide. Blue circles sketch the positions of scattering cylinders for a desired top-hat power profile. The unshaded grid depicts the domain. (b) Comparison between experimental data (circles) (see Seliger et al. 2006) and modeled predictions.

with μ_r the relative and μ_0 the vacuum permeability. ϵ_r denotes the relative and ϵ_0 the vacuum permittivity. Equation (19) is numerically determined using an evenly meshed square-grid (x_i, y_i). The resulting finite-difference PDE approximates the field $E_{z,i,j}$ everywhere inside the domain including the dielectric scatterers. The imposed boundary conditions (Dirichlet condition for the metallic horn and perfectly matching layers) are satisfied. This linear equation system is solved by ordering the values of $E_{z,i,j}$ of the PDE into a column vector. Hence, the finite-difference PDE can be rewritten as

$$\mathbf{L} \cdot \mathbf{E}_z = \mathbf{b}\,, \tag{20}$$

where $\mathbf{L}$ denotes the finite-difference matrix, which is complex-valued and sparse. $\mathbf{E}_z$ describes the complex-valued electric field, that is to be computed and $\mathbf{b}$ contains the boundary conditions. With this, the magnitude of the field at any point of the domain can be determined by solving the linear system of Eq. (20).

The power at any point on the target surface $(x(\theta), y(\theta))$ for an incident angle θ is computed through interpolation using the nearest four mesh points and their standard Gaussian weights $\mathbf{W}(\theta)$ with respect to $(x(\theta), y(\theta))$ as

$$s_{\mathbf{mod}}(\theta) = \frac{\mathbf{W}(\theta)}{2} \cdot \mathbf{diag}(\mathbf{E_z}) \cdot \mathbf{E_z}\,. \tag{21}$$

In the numerical implementation, we exploited the sparsity of $\mathbf{L}$, which improved the efficiency of the algorithm significantly. In fact, the solution of a realistic forward problem ($\sim 70,000 \times 70,000$ matrix), including 50 dielectric scatterers requires about 0.7 second on a commercially available Intel Xeon 3.4 GHz. Since the size of $\mathbf{L}$ determines the size of the problem, the computational efficiency of our implementation is independent of the number of scattering cylinders.

To verify this finite-difference technique for the power along the target surface (radius $= 60$ mm from the domain center), we compared our simulations with experimental measurements from Seliger et al. 2006 for the same optimal arrangement of 50 dielectric scatterers ($\epsilon_r = 2.05$ and 3.175 ± 0.025 diameter). Figure 9(b) illustrates the good agreement between experimental and model data on a linear scale for an objective top-hat function.

In the optimization problem, the design vector $\mathbf{x} \in \mathbb{R}^{100}$ describes the positions of the 50 cylinders. For a given $\mathbf{x}$ in the domain, the power profile s_{mod} over discretized angles on the target surface, θ_k, is computed. We can thus evaluate the objective function

$$f_{EM}(\mathbf{x}) = \sum_{k=1}^{m} |s_{mod}(\theta_k) - s_{obj}(\theta_k)|^2 . \tag{22}$$

Note, that $f(\mathbf{x})$ is not a direct function of $\mathbf{x}$ and not convex in $\mathbf{x}$. Furthermore, using adjoint technique, our implementation provides the cost function gradient $\nabla_{\mathbf{x}} f_{EM}(\mathbf{x})$ at no additional computational expense. We refer interested readers to Reference (Bertsimas, Nohadani, and Teo 2007) for a more thorough discussion of the physical problem.

Because of the underlying Helmholtz equation, the model scales with frequency and can be extended to nanophotonic designs. While degradation due to implementation errors is already significant in laboratory experiments today, it will be amplified under nanoscale implementations. Therefore, there is a need to find designs that are robust against implementation errors. Thus, the robust optimization problem is defined as

$$\min_{\mathbf{x}\in\mathcal{X}} g_{EM}(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{X}} \max_{\Delta\mathbf{x}\in\mathcal{U}} f_{EM}(\mathbf{x}+\Delta\mathbf{x}).$$

In this setting, $\Delta\mathbf{x}$ represents displacement errors of the scattering cylinders.

5.2. Computation Results

We first construct the uncertainty set $\mathcal{U}$ to include most of the errors expected. In laboratory experiments, the implementation errors $\Delta\mathbf{x}$ are observed to have a standard deviation of $40\mu m$ (Levi 2006). Therefore, to define an uncertainty set incorporating 99% of the perturbations (i.e., $P(\Delta\mathbf{x} \in \mathcal{U} = 99\%)$), we define

$$\mathcal{U} = \{\Delta\mathbf{x} \,|\, \|\Delta\mathbf{x}\|_2 \leq \Gamma = 550\mu m\}, \tag{23}$$

where Δx_i is assumed to be independently and normally distributed with mean 0 and standard deviation $40\mu m$.

The standard procedure used to address Problem (5.1) is to find an optimal design minimizing Eqn. (22). Subsequently, the sensitivity of the optimal design to implementation errors will be assessed through random simulations. However, because the problem is highly nonconvex and of high dimension, there is, to the best of our knowledge, no approach to find a design that is less sensitive to implementation errors. Nevertheless, a design that minimizes $f_{EM}(\mathbf{x})$ locally is readily found by applying a gradient-related descent algorithm. We applied the robust local search technique using such a local minimum as the starting point.

Figure 10 shows that the robust local algorithm finds a final design $\mathbf{x}^{65}$ with a worst case cost that is 8% lower than that of $\mathbf{x}^1$. Throughout the iterations, the nominal cost remains practically the same. The worst case cost of $\mathbf{x}^{65}$ was estimated with 110000 neighbors in its neighborhood. Note, however, that these 110000 neighbors were not evaluated a single neighborhood exploration. Instead, more than 95% of them were performed in the iterations leading up to the iteration 65. The estimated worst case cost at each iteration also shows a downward trend: as the iteration count increases, the knowledge about problem grows and more robust designs are discovered.

Since we can only estimate the worst case cost, there is always a chance for late discoveries of worst implementation errors. Therefore, the decrease of the estimated worst case cost may not be monotonic. Finally, note that the initial design $\mathbf{x}^1$ may already has inherent robustness to implementation errors because it is a local minimum, i.e. with all factors being equal, a design with a low nominal cost will have a low worst case cost.

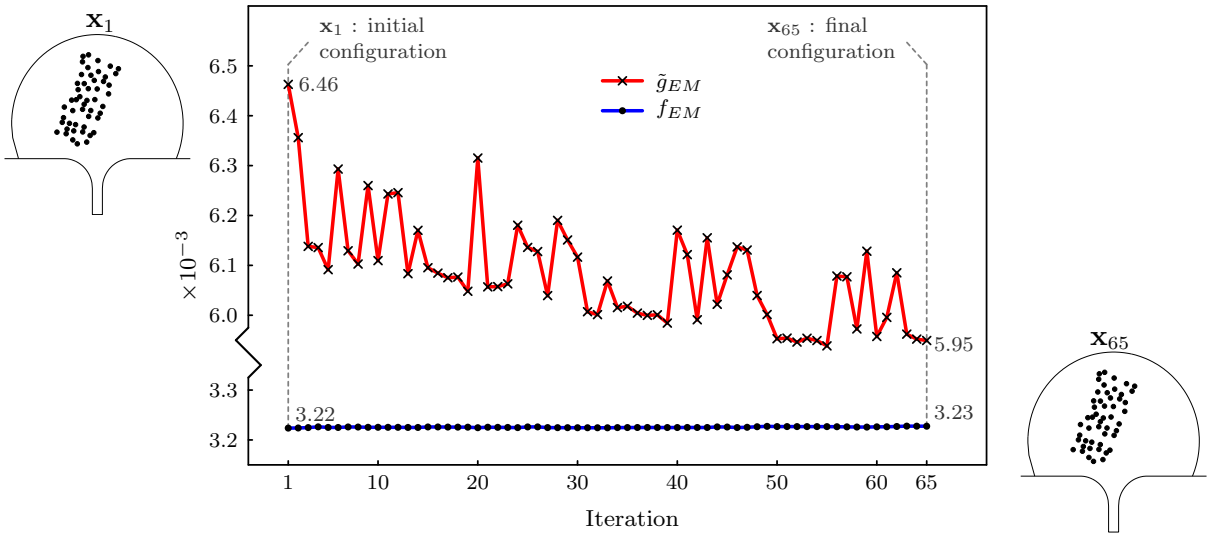

Figure 10 Performance of the robust local search algorithm in Application Example II. The initial cylinder configuration, $\mathbf{x}_1$, and final configuration, $\mathbf{x}_{65}$, are shown outside the line plot. While the differences between the two configurations seem negligible, the worst case cost of $\mathbf{x}_{65}$ is 8% lower than that of $\mathbf{x}_1$. The nominal costs of the two configurations are practically the same.

We also observed that the neighborhood exploration strategy is more efficient in assessing the worst case cost when compared to random sampling as is the standard today in perturbation analysis. For example, when estimating $\tilde{g}_{EM}(\mathbf{x}^1)$, the best estimate attained by random sampling after 30000 function evaluations using the numerical solver is 96% of the estimate obtained by our algorithm using only 3000 function evaluations. This is not surprising since finding the optimal solution to the inner optimization problem by random searches is usually inferior to applying gradient-related algorithms using multiple starting points.

6. Generalized Method for Problems with Both Implementation Errors and Parameter Uncertainties

In addition to implementation errors, uncertainties can reside in problem coefficients. These coefficients often cannot be defined exactly, because of either insufficient knowledge or the presence of noise. In this section, we generalize the robust local search algorithm to include considerations for such parameter uncertainties.

6.1. Problem Definition

Let $f(\mathbf{x}, \bar{\mathbf{p}})$ be the *nominal cost* of design vector $\mathbf{x}$, where $\bar{\mathbf{p}}$ is an estimation of the true problem coefficient $\mathbf{p}$. For example, for the case $f(\mathbf{x}, \bar{\mathbf{p}}) = 4x_1^3 + x_2^2 + 2x_1^2 x_2$, $\mathbf{x} = \binom{x_1}{x_2}$ and $\bar{\mathbf{p}} = \binom{4}{1}{2}$. Since $\bar{\mathbf{p}}$ is an estimation, the true coefficient $\mathbf{p}$ can instead be $\bar{\mathbf{p}} + \Delta\mathbf{p}$, $\Delta\mathbf{p}$ being the parameter uncertainties. Often, the *nominal optimization problem*

$$\min_{\mathbf{x}} \; f(\mathbf{x}, \bar{\mathbf{p}}), \tag{24}$$

is solved, ignoring the presence of uncertainties.

We consider Problem (24), where both $\Delta\mathbf{p} \in \mathbb{R}^m$ and implementation errors $\Delta\mathbf{x} \in \mathbb{R}^n$ are present, while further assuming $\Delta\mathbf{z} = \binom{\Delta\mathbf{x}}{\Delta\mathbf{p}}$ lies within the uncertainty set

$$\mathcal{U} = \left\{ \; \Delta\mathbf{z} \in \mathbb{R}^{n+m} \;\; | \;\; \|\Delta\mathbf{z}\|_2 \; \leq \; \Gamma \; \right\}. \tag{25}$$

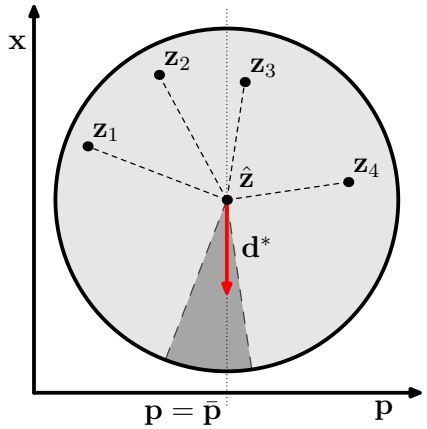

Figure 11 A two-dimensional illustration of Prob. (28), and equivalently Prob. (27). Both implementation errors and uncertain parameters are present. Given a design $\hat{\mathbf{x}}$, the possible realizations lie in the neighborhood $\mathcal{N}$, as defined in Eqn. (29). $\mathcal{N}$ lies in the space $\mathbf{z} = (\mathbf{x}, \mathbf{p})$. The shaded cone contains vectors pointing away from the bad neighbors, $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{p}_i)$, while the vertical dotted denotes the intersection of hyperplanes $\mathbf{p} = \bar{\mathbf{p}}$. For $\mathbf{d}^* = (\mathbf{d}_x^*, \mathbf{d}_p^*)$ to be a feasible descent direction, it must lie in the intersection between the both the cone and the hyperplanes, i.e. $\mathbf{d}_p^* = \mathbf{0}$.

As in Eqn. (2), $\Gamma > 0$ is a scalar describing the size of perturbations. We seek a robust design $\mathbf{x}$ by minimizing the worst case cost given a perturbation in $\mathcal{U}$,

$$g(\mathbf{x}) := \max_{\Delta\mathbf{z} \in \mathcal{U}} \; f\left(\mathbf{x} + \Delta\mathbf{x}, \bar{\mathbf{p}} + \Delta\mathbf{p}\right). \tag{26}$$

The *generalized robust optimization problem* is consequently

$$\min_{\mathbf{x}} \; g(\mathbf{x}) \equiv \min_{\mathbf{x}} \max_{\Delta\mathbf{z} \in \mathcal{U}} \; f(\mathbf{x} + \Delta\mathbf{x}, \bar{\mathbf{p}} + \Delta\mathbf{p}). \tag{27}$$

6.2. Basic Idea Behind Generalization

To generalize the robust local search to consider parameter uncertainties, note that Problem (27) is equivalent to the problem

$$\begin{aligned} \min_{\mathbf{z}} \; \max_{\Delta\mathbf{z}} \; & f(\mathbf{z} + \Delta\mathbf{z}) \\ s.t. \; & \mathbf{p} = \bar{\mathbf{p}}, \end{aligned} \tag{28}$$

where $\mathbf{z} = \binom{\mathbf{x}}{\mathbf{p}}$. This formulation is similar to Problem (4), the robust problem with implementation errors only, but with some decision variables fixed; the feasible region is the intersection of the hyperplanes $p_i = \bar{p}_i$, $i = 1, \ldots, m$.

The geometric perspective is updated to capture these equality constraints and presented in Fig. 11. Thus, the necessary modifications to the local search algorithm are:

i. Neighborhood Exploration : Given a design $\hat{\mathbf{x}}$, or equivalently $\hat{\mathbf{z}} = \binom{\hat{\mathbf{x}}}{\bar{\mathbf{p}}}$, the neighborhood is

$$\mathcal{N} := \left\{\mathbf{z} \mid \|\mathbf{z} - \hat{\mathbf{z}}\|_2 \leq \Gamma\right\} = \left\{\binom{\mathbf{x}}{\mathbf{p}} \mid \left\|\begin{smallmatrix}\mathbf{x} - \hat{\mathbf{x}} \\ \mathbf{p} - \bar{\mathbf{p}}\end{smallmatrix}\right\|_2 \leq \Gamma\right\}. \tag{29}$$

ii. Robust Local Move : Ensure that every iterate satisfies $\mathbf{p} = \bar{\mathbf{p}}$.

6.3. Generalized Local Search Algorithm

For ease of exposition, we shall only highlight the key differences to the local search algorithm previously discussed in Section 3.

6.3.1. Neighborhood Exploration The implementation is similar to that in Section 3.1. However, $n+m+1$ gradient ascents are used instead, since the neighborhood $\mathcal{N}$ now lies in the space $\mathbf{z} = \binom{\mathbf{x}}{\mathbf{p}}$ (see Fig. 11), and the inner maximization problem is now

$$\max_{\Delta\mathbf{z}\in\mathcal{U}} f(\hat{\mathbf{z}}+\Delta\mathbf{z}) \equiv \max_{\Delta\mathbf{z}\in\mathcal{U}} f(\hat{\mathbf{x}}+\Delta\mathbf{x}, \bar{\mathbf{p}}+\Delta\mathbf{p}). \tag{30}$$

The $n+m+1$ sequences start from $\Delta\mathbf{z}=\mathbf{0}$, $\Delta\mathbf{z} = sign(\frac{\partial f(\mathbf{x}=\hat{\mathbf{x}})}{\partial x_i})\frac{\Gamma}{3}\mathbf{e}_i$ for $i=1,\ldots,n$ and $\Delta\mathbf{z} = sign(\frac{\partial f(\mathbf{p}=\bar{\mathbf{p}})}{\partial p_{i-n}})\frac{\Gamma}{3}\mathbf{e}_i$ for $i=n+1,\ldots,n+m$.

6.3.2. Robust Local Move At every iterate, the condition $\mathbf{p}=\bar{\mathbf{p}}$ is satisfied by ensuring that the descent direction $\mathbf{d}^* = \binom{\mathbf{d}_x^*}{\mathbf{d}_p^*}$ fulfills the condition $\mathbf{d}_p^* = \mathbf{0}$ (see Fig. 11). Referring to the robust local move discussed in Section 3.2, we solve the modified SOCP:

$$\begin{aligned}
&\min_{\mathbf{d},\beta}\ \beta \\
&s.t.\ \|\mathbf{d}\|_2 \le 1 \\
&\quad \mathbf{d}'\begin{pmatrix}\mathbf{x}_i-\mathbf{x}^k\\ \mathbf{p}_i-\bar{\mathbf{p}}\end{pmatrix} \le \beta\left\|\begin{pmatrix}\mathbf{x}_i-\mathbf{x}^k\\ \mathbf{p}_i-\bar{\mathbf{p}}\end{pmatrix}\right\|_2 \ \forall \binom{\mathbf{x}_i}{\mathbf{p}_i} \in \mathcal{M}^k \\
&\quad \mathbf{d}_p = \mathbf{0} \\
&\quad \beta \le -\epsilon,
\end{aligned} \tag{31}$$

which reduces to

$$\begin{aligned}
&\min_{\mathbf{d}_x,\beta}\ \beta \\
&s.t.\ \|\mathbf{d}_x\|_2 \le 1 \\
&\quad \mathbf{d}_x'(\mathbf{x}_i-\mathbf{x}^k) \le \beta\left\|\begin{pmatrix}\mathbf{x}_i-\mathbf{x}^k\\ \mathbf{p}_i-\bar{\mathbf{p}}\end{pmatrix}\right\|_2 \ \forall \binom{\mathbf{x}_i}{\mathbf{p}_i} \in \mathcal{M}^k \\
&\quad \beta \le -\epsilon.
\end{aligned} \tag{32}$$

6.4. Application Example III - Revisiting Application Example I

6.4.1. Problem Description To illustrate the performance of the generalized robust local search algorithm, we revisit Application Example I from Section 4 where polynomial objective function is

$$\begin{aligned}
f_{poly}(x,y) &= 2x^6 - 12.2x^5 + 21.2x^4 + 6.2x - 6.4x^3 - 4.7x^2 + y^6 - 11y^5 + 43.3y^4 - 10y - 74.8y^3 \\
&\quad +56.9y^2 - 4.1xy - 0.1y^2x^2 + 0.4y^2x + 0.4x^2y \\
&= \sum_{\substack{r>0,\ s>0\\ r+s\le 6}} c_{rs}x^r y^s.
\end{aligned}$$

In addition to implementation errors as previously described, there is uncertainty in each of the 16 coefficients of the objective function. Consequently, the objective function with uncertain parameters is

$$\tilde{f}_{poly}(x,y) = \sum_{\substack{r>0,\ s>0\\ r+s\le 6}} c_{rs}(1+0.05\Delta p_{rs})x^r y^s,$$

where $\Delta\mathbf{p}$ is the vector of uncertain parameters; the robust optimization problem is

$$\min_{x,y} g_{poly}(x,y) \equiv \min_{x,y} \max_{\|\Delta\|_2\le 0.5} \tilde{f}_{poly}(x+\Delta x, y+\Delta y),$$

where $\Delta = \begin{pmatrix}\Delta x\\ \Delta y\\ \Delta\mathbf{p}\end{pmatrix}$.

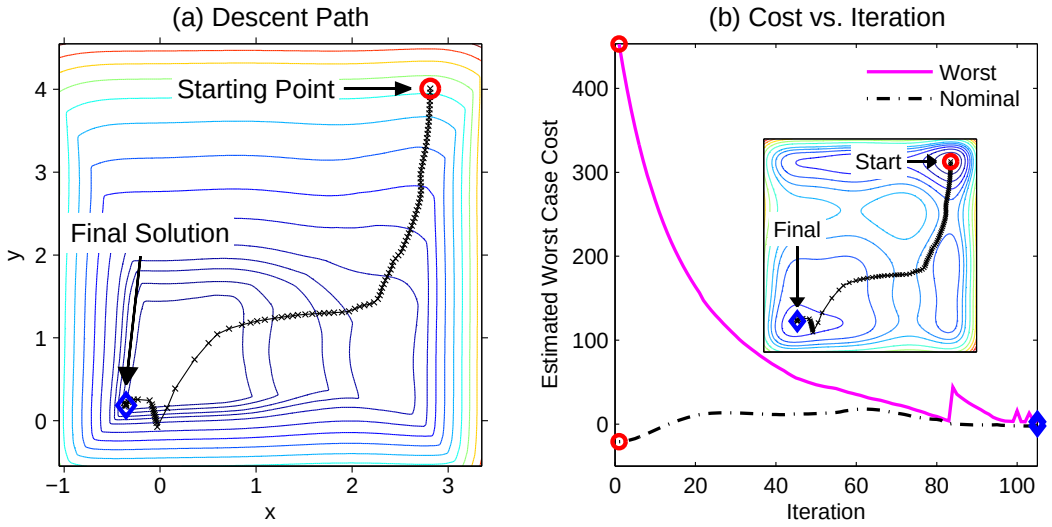

Figure 12 Performance of the generalized robust local search algorithm in Application Example III. (a) Path taken on the estimated worst case cost surface $g_{poly}(x,y)$. Algorithm converges to region with low worst case cost. (b) The worst cost is decreased significantly; while the nominal cost increased slightly. Inset shows the nominal cost surface $f_{poly}(x,y)$, indicating that the robust search moves from the global minimum of the nominal function to the vicinity of another local minimum.

6.4.2. Computation Results Observations on the nominal cost surface has been discussed in Application Example I. Given both implementation errors and parameter uncertainties, the estimated cost surface of $g_{poly}(x,y)$ is shown in Fig. 12(a). This estimation is done computationally through simulations using 1000 joint perturbations in all the uncertainties. Fig. 12(a) suggests that $g_{poly}(x,y)$ has local minima, or possibly a unique local minimum, in the vicinity of $(x,y)=(0,0.5)$.

We applied the generalized robust local search algorithm on this problem starting from the global minimum of the nominal cost function $(x,y)=(2.8,4.0)$. Figure 12(b) shows the performance of the algorithm. Although the initial design has a nominal cost of -20.8, it has a large worst case cost of 450. The algorithm finds a robust design with a significantly lower worst case cost. Initially, the worst case cost decreases monotonically with increasing iteration count, but fluctuates when close to convergence. On the other hand, the nominal cost increases initially, and decreases later with increasing iteration count.

Figure 12(a) shows that the robust search finds the region where the robust local minimum is expected to reside. The inset in Fig. 12(b) shows the path of the robust search "escaping" the global minimum. Because the local search operates on the worst cost surface in Fig. 12(a) and not the nominal cost surface in the inset of Figure 12(b), such an "escape" is possible.

Efficiency of the Neighborhood Exploration: In the local search algorithm, $n+1$ gradient ascents are carried out when the perturbations has n dimensions (See Section 3.1). Clearly, if the cost function is less nonlinear over the neighborhood, less gradient ascents will suffice in finding the bad neighbors; the converse is true as well. Therefore, for a particular problem, one can investigate empirically the tradeoff between the depth of neighborhood search (i.e., number of gradient ascents) and the overall run-time required for robust optimization.

For this example, we investigate this tradeoff with (i) the standard $n+1$ gradient ascents, (ii) $10+1$, and (iii) $3+1$ gradient ascents, in every iteration. Note, that dimension of the perturbation, n is 18: 2 for implementation errors and 16 for parameter uncertainties. Case (i) has been discussed in Section 3.1 and serve as the benchmark. In case (ii), 1 ascent starts from $\mathbf{z}^k$, while the remaining 10

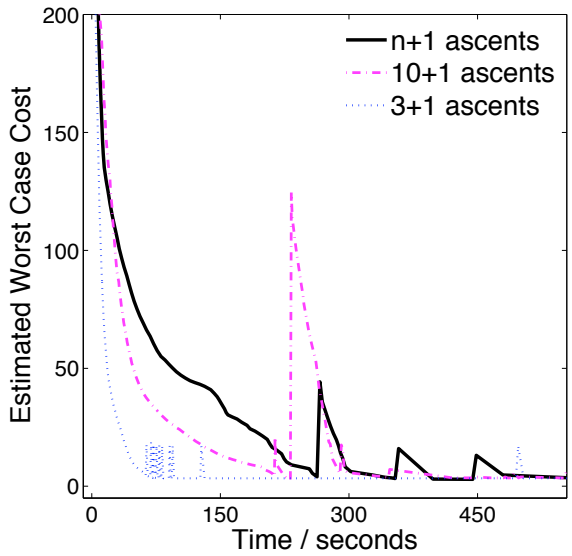

Figure 13 Performance under different number of gradient ascents during the neighborhood exploration. In all instances, the worst case cost is lowered significantly. While the decrease is fastest when only $3+1$ gradient ascents are used, the terminating conditions were not attained. The instance with $10+1$ gradient ascents took the shortest time to attain convergence.

start from $\mathbf{z}^k + sign\left(\frac{\partial f(\mathbf{z}^k)}{\partial z_i}\right)\frac{\Gamma}{3}\mathbf{e}_i$, where i denotes coordinates with the 10 largest partial derivatives $\left|\frac{\partial f(\mathbf{z}^k)}{\partial z_i}\right|$. This strategy is similarly applied in case (iii).

As shown in Fig. 13, the worst case cost is lowered in all three cases. Because of the smaller number of gradient ascents per iteration, the decrease in worst case cost is the fastest in case (iii). However, the algorithm fails to converge long after terminating conditions have been attained in the other two cases. In this example, case (ii) took the shortest time, taking 550 seconds to converge compared to 600 seconds in case (i). The results seem to indicate that depending on the problem, the efficiency of the algorithm might be improved by using a smaller number of gradient ascents, but if too few gradient ascents are used, the terminating conditions might not be attained.

7. Conclusions

We have presented a new robust optimization technique that can be applied to nonconvex problems when both implementation errors and parameter uncertainties are present. Because the technique assumes only the capability of function and gradient evaluations, few assumptions are required on the problem structure. Consequently, the presented robust local search technique is generic and suitable for most real-world problems, including computer-based simulations, response surface and kriging metamodels that are often used in the industry today.

This robust local search algorithm operates by acquiring knowledge of the cost surface $f(\mathbf{x}, \mathbf{p})$ in a domain close to a given design $\hat{\mathbf{x}}$ thoroughly but efficiently. With this information, the algorithm recommends an updated design with a lower estimated worst case cost by excluding neighbors with high cost from the neighborhood of the updated design. Applied iteratively, the algorithm discovers more robust designs until termination criteria is reached and a robust local minimum is confirmed. The termination criteria of the algorithm is an optimality condition for robust optimization problems.

The effectiveness of the method was demonstrated through the application to (i) a nonconvex polynomial problem, and (ii) an actual electromagnetic scattering design problem with a nonconvex objective function and a 100 dimensional design space. In the polynomial problem, robust local

minima are found from a number of different initial solutions. For the engineering problem, we started from a local minimum to the nominal problem, the problem formulated without considerations for uncertainties. From this local minimum, the robust local search found a robust local minimum with the same nominal cost, but with a worst case cost that is 8% lower.

Appendix A: Continuous Minimax Problem

A *continuous minimax problem* is the problem

$$\min_{\mathbf{x}} \max_{\mathbf{y}\in\mathcal{C}} \phi(\mathbf{x},\mathbf{y}) \equiv \min_{\mathbf{x}} \psi(\mathbf{x}) \tag{33}$$

where ϕ is a real-valued function, $\mathbf{x}$ is the decision vector, $\mathbf{y}$ denotes the uncertain variables and $\mathcal{C}$ is a closed compact set. ψ is the max-function. Refer to Rustem and Howe 2002 for more discussions about the continuous minimax problem.

The robust optimization problem (4) is a special instance of the continuous minimax problem, as can be seen through making the substitutions: $\mathbf{y}=\Delta\mathbf{x}$, $\mathcal{C}=\mathcal{U}$, $\phi(\mathbf{x},\mathbf{y})=f(\mathbf{x}+\mathbf{y})=f(\mathbf{x}+\Delta\mathbf{x})$ and $\psi=\mathbf{g}$. Thus, Prob. (4) shares properties of the minimax problem; the following theorem captured the relevant properties:

THEOREM 3 **(Danskin's Min-Max Theorem)**.
Let $\mathcal{C}\subset\mathbb{R}^m$ be a compact set, $\phi:\mathbb{R}^n\times\mathcal{C}\mapsto\mathbb{R}$ be continuously differentiable in $\mathbf{x}$, and $\psi:\mathbb{R}^n\mapsto\mathbb{R}$ be the max-function $\psi(\mathbf{x}):=\max_{\mathbf{y}\in\mathcal{C}}\ \phi(\mathbf{x},\mathbf{y})$.

(a) Then, $\psi(\mathbf{x})$ is directionally differentiable with directional derivatives

$$\psi'(\mathbf{x};\mathbf{d}) = \max_{\mathbf{y}\in\mathcal{C}^*(x)} \mathbf{d}'\nabla_x\phi(\mathbf{x},\mathbf{y}),$$

where $\mathcal{C}^*(\mathbf{x})$ is the set of maximizing points

$$\mathcal{C}^*(\mathbf{x}) = \left\{\mathbf{y}^* \mid \phi(\mathbf{x},\mathbf{y}^*)=\max_{\mathbf{y}\in\mathcal{C}}\phi(\mathbf{x},\mathbf{y})\right\}.$$

(b) If $\phi(\mathbf{x},\mathbf{y})$ is convex in $\mathbf{x}$, $\phi(\cdot,\mathbf{y})$ is differentiable for all $\mathbf{y}\in\mathcal{C}$ and $\nabla_{\mathbf{x}}\phi(\mathbf{x},\cdot)$ is continuous on $\mathcal{C}$ for each $\mathbf{x}$, then $\psi(\mathbf{x})$ is convex in $\mathbf{x}$ and $\forall\mathbf{x}$,

$$\partial\psi(\mathbf{x}) = conv\left\{\nabla_{\mathbf{x}}\phi(\mathbf{x},\mathbf{y}) \mid \mathbf{y}\in\mathcal{C}^*(x)\right\} \tag{34}$$

where $\partial\psi(\mathbf{x})$ is the subdifferential of the convex function $\psi(\mathbf{x})$ at $\mathbf{x}$

$$\partial\psi(\mathbf{x}) = \{\mathbf{z} \mid \psi(\bar{\mathbf{x}})\geq\psi(\mathbf{x})+\mathbf{z}'(\bar{\mathbf{x}}-\mathbf{x}),\forall\bar{\mathbf{x}}\}$$

and conv denotes the convex hull.

For a proof of Theorem 3, see (Danskin 1966, Danskin 1967).

Appendix B: Proof of Theorem 1

Before proving Theorem 1, observe the following results:

PROPOSITION 3.
Suppose that $f(\mathbf{x})$ is continuously differentiable in $\mathbf{x}$, $\mathcal{U}=\{\Delta\mathbf{x} \mid \|\Delta\mathbf{x}\|_2\leq\Gamma\}$ where $\Gamma>0$ and $\mathcal{U}^*(\mathbf{x}):=\left\{\Delta\mathbf{x}^* \mid \Delta\mathbf{x}^*=\arg\max_{\Delta\mathbf{x}\in\mathcal{U}}\ f(\mathbf{x}+\Delta\mathbf{x})\right\}$. Then, for any $\hat{\mathbf{x}}$ and $\Delta\mathbf{x}^*\in\mathcal{U}^*(\mathbf{x}=\hat{\mathbf{x}})$,

$$\nabla_{\mathbf{x}}f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} = k\Delta\mathbf{x}^*$$

where $k\geq 0$.

In words, the gradient at $\mathbf{x} = \hat{\mathbf{x}} + \Delta\mathbf{x}^*$ is parallel to the vector $\Delta\mathbf{x}^*$.

PROOF. Since $\Delta\mathbf{x}^*$ is a maximizer of the problem $\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\hat{\mathbf{x}}+\Delta\mathbf{x})$ and a regular point, because of the Karush-Kuhn-Tucker necessary conditions, there exists a scalar $\mu \geq 0$ such that

$$-\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} + \mu\nabla_{\Delta\mathbf{x}}(\Delta\mathbf{x}'\Delta\mathbf{x} - \Gamma)|_{\Delta\mathbf{x}=\Delta\mathbf{x}^*} = 0.$$

This is equivalent to the condition

$$\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} = 2\mu\Delta\mathbf{x}^*.$$

The result follows by choosing $k = 2\mu$. □

In this context, a feasible vector is said to be a regular point if all the active inequality constraints are linearly independent, or if all the inequality constraints are inactive. Since there is only one constraint in the problem $\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\hat{\mathbf{x}}+\Delta\mathbf{x})$ which is either active or not, $\Delta\mathbf{x}^*$ is always a regular point. Furthermore, note that where $\|\Delta\mathbf{x}^*\|_2 < \Gamma$, $\hat{\mathbf{x}} + \Delta\mathbf{x}^*$ is an unconstrained local maximum of f and it follows that $\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} = \mathbf{0}$ and $k = 0$. Using Proposition 3, the following corollary can be extended from Theorem 3:

COROLLARY 1.
Suppose that $f(\mathbf{x})$ is continuously differentiable, $\mathcal{U} = \{\Delta\mathbf{x} \mid \|\Delta\mathbf{x}\|_2 \leq \Gamma\}$ where $\Gamma > 0$, $g(\mathbf{x}) := \max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x})$ and $\mathcal{U}^*(\mathbf{x}) := \left\{\Delta\mathbf{x}^* \mid \Delta\mathbf{x}^* = \arg\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x})\right\}$.

(a) Then, $g(\mathbf{x})$ is directionally differentiable and its directional derivatives $g'(\mathbf{x};\mathbf{d})$ are given by

$$g'(\mathbf{x};\mathbf{d}) = \max_{\Delta\mathbf{x}\in\mathcal{U}^*(x)} f'(\mathbf{x}+\Delta\mathbf{x};\mathbf{d}).$$

(b) If $f(\mathbf{x})$ is convex in $\mathbf{x}$, then $g(\mathbf{x})$ is convex in $\mathbf{x}$ and $\forall\mathbf{x}$,

$$\partial g(\mathbf{x}) = conv\left\{\Delta\mathbf{x} \mid \Delta\mathbf{x} \in \mathcal{U}^*(x)\right\}.$$

PROOF. Referring to the notation in Theorem 3, if we let $\mathbf{y} = \Delta\mathbf{x}$, $\mathcal{C} = \mathcal{U}$, $\mathcal{C}^* = \mathcal{U}^*$, $\phi(\mathbf{x},\mathbf{y}) = f(\mathbf{x},\Delta\mathbf{x}) = f(\mathbf{x}+\Delta\mathbf{x})$, then $\psi(\mathbf{x}) = g(\mathbf{x})$. Because all the conditions in Theorem 3 are satisfied, it follows that

(a) $g(\mathbf{x})$ is directionally differentiable with

$$\begin{aligned} g'(\mathbf{x};\mathbf{d}) &= \max_{\Delta\mathbf{x}\in\mathcal{U}^*(x)} \mathbf{d}'\nabla_{\mathbf{x}} f(\mathbf{x}+\Delta\mathbf{x}) \\ &= \max_{\Delta\mathbf{x}\in\mathcal{U}^*(x)} f'(\mathbf{x}+\Delta\mathbf{x};\mathbf{d}). \end{aligned}$$

(b) $g(\mathbf{x})$ is convex in $\mathbf{x}$ and $\forall\mathbf{x}$,

$$\begin{aligned} \partial g(\mathbf{x}) &= conv\left\{\nabla_{\mathbf{x}} f(\mathbf{x},\Delta\mathbf{x}) \mid \Delta\mathbf{x} \in \mathcal{U}^*(x)\right\} \\ &= conv\left\{\Delta\mathbf{x} \mid \Delta\mathbf{x} \in \mathcal{U}^*(x)\right\}. \end{aligned}$$

The last equality is due to Proposition 3. □

We shall now prove Theorem 1:

THEOREM 1
Suppose that $f(\mathbf{x})$ is continuously differentiable, $\mathcal{U} = \{\Delta\mathbf{x} \mid \|\Delta\mathbf{x}\|_2 \leq \Gamma\}$ where $\Gamma > 0$, $g(\mathbf{x}) := \max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x})$ and $\mathcal{U}^*(\mathbf{x}) := \left\{\Delta\mathbf{x}^* \mid \Delta\mathbf{x}^* = \arg\max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x})\right\}$. Then, $\mathbf{d} \in \mathbb{R}^n$ is a descent direction for the worst case cost function $g(\mathbf{x})$ at $\mathbf{x} = \hat{\mathbf{x}}$ if and only if for all $\Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}})$

$$\mathbf{d}'\Delta\mathbf{x}^* < 0$$

and $\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} \neq \mathbf{0}$.

PROOF. From Corollary 1, for a given $\hat{\mathbf{x}}$

$$\begin{aligned} g'(\hat{\mathbf{x}};\mathbf{d}) &= \max_{\Delta\mathbf{x}^*\in\mathcal{U}^*(\hat{\mathbf{x}})} f'(\hat{\mathbf{x}}+\Delta\mathbf{x};\mathbf{d}) \\ &= \max_{\Delta\mathbf{x}^*\in\mathcal{U}^*(\hat{\mathbf{x}})} \mathbf{d}'\nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} \\ &= \max_{\Delta\mathbf{x}^*\in\mathcal{U}^*(\hat{\mathbf{x}})} k\mathbf{d}'\Delta\mathbf{x}^*. \end{aligned}$$

The last equality follows from Proposition 3. $k \geq 0$ but may be different for each $\Delta\mathbf{x}^*$. Therefore, for $\mathbf{d}$ to be a descent direction,

$$\max_{\Delta\mathbf{x}^*\in\mathcal{U}^*(\hat{\mathbf{x}})} k\mathbf{d}'\Delta\mathbf{x}^* < 0. \tag{35}$$

Eqn. 35 is satisfied if and only if for all $\Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}})$,

$$\begin{aligned} \mathbf{d}'\Delta\mathbf{x}^* &< 0, \\ \nabla_{\mathbf{x}} f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}+\Delta\mathbf{x}^*} &\neq 0, \quad \text{for } k \neq 0. \end{aligned}$$

□

Appendix C: Proof of Theorem 2

PROPOSITION 4. *Let $\mathcal{G} := \{\Delta\mathbf{x}_1, \ldots, \Delta\mathbf{x}_m\}$ and let $(\mathbf{d}^*, \beta^*)$ be the optimal solution to a feasible SOCP*

$$\begin{aligned} \min_{\mathbf{d},\beta}\ & \beta \\ s.t.\ & \|\mathbf{d}\|_2 \leq 1, \\ & \mathbf{d}'\Delta\mathbf{x}_i \leq \beta, \ \ \forall \Delta\mathbf{x}_i \in \mathcal{G}, \\ & \beta \leq -\epsilon, \end{aligned}$$

where ϵ is a small positive scalar. Then, $-\mathbf{d}^*$ lies in conv $\mathcal{G}$.

PROOF. We show that if $-\mathbf{d}^* \notin conv\ \mathcal{G}$, $\mathbf{d}^*$ is not the optimal solution to the SOCP because a better solution can be found. Note, that for $(\mathbf{d}^*, \beta^*)$ to be an optimal solution, $\|\mathbf{d}^*\|_2 = 1$, $\beta^* < 0$ and $\mathbf{d}^{*'}\Delta\mathbf{x}_i < 0$, $\forall \Delta\mathbf{x}_i \in \mathcal{G}$.

Assume, for contradiction, that $-\mathbf{d}^* \notin conv\ \mathcal{G}$. By the separating hyperplane theorem, there exists a $\mathbf{c}$ such that $\mathbf{c}'\Delta\mathbf{x}_i \geq 0$, $\forall \Delta\mathbf{x}_i \in \mathcal{G}$ and $\mathbf{c}'(-\mathbf{d}^*) < 0$. Without any loss of generality, let $\|\mathbf{c}\|_2 = 1$, and let $\mathbf{c}'\mathbf{d}^* = \mu$. Note, that $0 < \mu < 1$, strictly less than 1 because $|\mathbf{c}| = |\mathbf{d}^*| = 1$ and $\mathbf{c} \neq \mathbf{d}^*$. The two vectors cannot be the same since $\mathbf{c}'\Delta\mathbf{x}_i \geq 0$ while $\mathbf{d}^{*'}\Delta\mathbf{x}_i < 0$.

Given such a vector $\mathbf{c}$, we can find a solution better than $\mathbf{d}^*$ for the SOCP, which is a contradiction. Consider the vector $\mathbf{q} = \frac{\lambda\mathbf{d}^* - \mathbf{c}}{\|\lambda\mathbf{d}^* - \mathbf{c}\|_2}$. $\|\mathbf{q}\|_2 = 1$, and for every $\Delta\mathbf{x}_i \in \mathcal{G}$, we have

$$\begin{aligned} \mathbf{q}'\Delta\mathbf{x}_i &= \tfrac{\lambda\mathbf{d}^{*'}\Delta\mathbf{x}_i - \mathbf{c}'\Delta\mathbf{x}_i}{\|\lambda\mathbf{d}^* - \mathbf{c}\|_2} \\ &= \tfrac{\lambda\mathbf{d}^{*'}\Delta\mathbf{x}_i - \mathbf{c}'\Delta\mathbf{x}_i}{\lambda + 1 - 2\lambda\mu} \\ &\leq \tfrac{\lambda\beta^* - \mathbf{c}'\Delta\mathbf{x}_i}{\lambda + 1 - 2\lambda\mu} \qquad \text{since } \mathbf{d}^{*'}\Delta\mathbf{x}_i \leq \beta^* \\ &\leq \tfrac{\lambda\beta^*}{\lambda + 1 - 2\lambda\mu} \qquad \text{since } \mathbf{c}'\Delta\mathbf{x}_i \geq 0. \end{aligned}$$

We can ensure $\frac{\lambda}{\lambda+1-2\lambda\mu} < 1$ by choosing λ such that $\begin{cases} \frac{1}{2\mu} < \lambda, & \text{if } 0 < \mu \leq \frac{1}{2} \\ \frac{1}{2\mu} < \lambda < \frac{1}{2\mu - 1}, & \text{if } \frac{1}{2} < \mu < 1 \end{cases}$. Therefore, $\mathbf{q}'\Delta\mathbf{x}_i < \beta^*$. Let $\bar{\beta} = \max_i \mathbf{q}'\Delta\mathbf{x}_i$, so $\bar{\beta} < \beta^*$. We have arrived at a contradiction since $(\mathbf{q}, \bar{\beta})$ is a feasible solution in the SOCP and it is strictly better than $(\mathbf{d}^*, \beta^*)$ since $\bar{\beta} < \beta^*$. □

Given Proposition 4, we prove the convergence result:

THEOREM 2
Suppose that $f(\mathbf{x})$ is continuously differentiable and convex with a bounded set of minimum points. Then, when the stepsize t^k are chosen such that $t^k > 0$, $t^k \to 0$ as $k \to \infty$ and $\sum_{k=1}^{\infty} t^k = \infty$, Algorithm 1 converges to the global optimum of the robust optimization problem (4).

PROOF. We show that applying the algorithm on the robust optimization problem (4) is equivalent to applying a subgradient optimization algorithm on a convex problem.

From Corollary 1(b), Prob. (4) is a convex problem with subgradients if $f(\mathbf{x})$ is convex. Next, $-\mathbf{d}^*$ is a subgradient at every iteration because:

- $-\mathbf{d}^*$ lies in the convex hull spanned by the vectors $\Delta\mathbf{x}^* \in \mathcal{U}^*(\mathbf{x}^k)$ (see Prop. 4), and
- this convex hull is the subdifferential of $g(\mathbf{x})$ at $\mathbf{x}^k$ (see Corollary 1(b)).

Since a subgradient step is taken at every iteration, the algorithm is equivalent to the following subgradient optimization algorithm:

Step 0. Initialization: Let $\mathbf{x}^k$ be an arbitrary decision vector, set $k = 1$.
Step 1. Find subgradient $\mathbf{s}^k$ of $\mathbf{x}^k$. Terminate if no such subgradient exist.
Step 2. Set $\mathbf{x}^{k+1} := \mathbf{x}^k - t^k\mathbf{s}^k$.
Step 3. Set $k := k + 1$. Go to Step 1.

From Theorem 31 in (see Shor 1998), this subgradient algorithm converges to the global minimum of the convex problem under the stepsize rules: $t^k > 0$, $t^k \to 0$ as $k \to 0$ and $\sum_{k=1}^{\infty} t^k = \infty$. The proof is now complete. □

Acknowledgments
We would like to thank D. Healy and A. Levi for encouragement and fruitful discussions. We also acknowledge C. Wang, R. Mancera, and P. Seliger for providing the initial structure of the numerical implementation used in the electromagnetic scattering design problem. This work is supported by DARPA - N666001-05-1-6030.

References

Ben-Tal, A., A. Nemirovski. 1998. Robust convex optimization. *Mathematics of Operations Research* **23** 769–805.

Ben-Tal, A., A. Nemirovski. 2003. Robust optimization — methodology and applications. *Mathematical Programming* **92**(3) 453–480.

Berenger, J. P. 1996. Three-dimensional perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics* **127** 363.

Bertsimas, D., O. Nohadani, K. M. Teo. 2007. Robust optimization in electromagnetic scattering problems. *Journal of Applied Physics* **101**(7) 074507.

Bertsimas, D., M. Sim. 2003. Robust discrete optimization and network flows. *Mathematical Programming* **98**(1–3) 49–71.

Bertsimas, D., M. Sim. 2006. Tractable approximations to robust conic optimization problems. *Mathematical Programming* **107**(1) 5–36.

Birge, J. R., F. Louveaux. 1997. *Introduction to stochastic programming*. Springer-Verlag, New York.

Charnes, A., W. W. Cooper. 1959. Chance-constrained programming. *Management Science* **6**(1) 73–79.

Ciarlet, P. G. 2002. *Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Cook, R. D., D. S. Malkus, M. E. Plesha, R. J. Witt. 2007. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons.

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

Danskin, J. M. 1967. *The theory of max-min and its application to weapons allocation problems*. Econometrics and Operations Research, Springer-Verlag, New York.

Dyer, M., L. Stougie. 2006. Computational complexity of stochastic programming problems. *Mathematical Programming* **106**(3) 423–432.

Gheorma, I. L., S. Haas, A. F. J. Levi. 2004. Aperiodic nanophotonic design. *Journal of Applied Physics* **95** 1420.

Henrion, D., J. B. Lasserre. 2003. Gloptipoly: Global optimization over polynomials with matlab and sedumi. *ACM Transactions on Mathematical Software* **29**(2) 165–194.

Horst, R., P. M. Pardalos. 1995. *Handbook of Global Optimization*. Kluwer Academic Publishers, The Netherlands.

Jin, R., W. Chen, T. W. Simpson. 2001. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization* **23**(1) 1–13.

Kingsland, D. M., J. Gong, J. L. Volakis, J. F. Lee. 2006. Performance of an anisotropic artificial absorber for truncating finite-element meshes. *IEEE Transactions on Antennas Propagation* **44** 975.

Kojima, M. 2003. Sums of squares relaxations of polynomial semidefinite programs, Research Report B-397, Tokyo Institute of Technology.

Levi, A. F. J. 2006. Private communications.

Markowitz, H. 1952. Portfolio selection. *The Journal of Finance* **7**(1) 77–91.

Mulvey, J. M., A. Ruszczynski. 1995. A new scenario decomposition method for large-scale stochastic optimization. *Operations Research* **43**(3) 477–490.

Nemirovski, A. 2003. On tractable approximations of randomly perturbed convex constraints. *Proceedings. 42nd IEEE Conference on Decision and Control* **3** 2419– 2422.

Park, G. J., T. H. Lee, K. H. Lee, K. H. Hwang. 2006. Robust design — an overview. *American Institute of Aeronautics and Astronautics Journal* **44** 181–191.

Prekopa, A., A. Ruszczynski, eds. 2002. *Special Issue on Stochastic Programming*. Taylor & Francis Group, London, etc. Optimization Methods and Software (OMS), Volume 17, Number 3.

Ramakrishnan, B., S. S. Rao. 1991. A robust optimization approach using taguchi's loss function for solving nonlinear optimization problems. *Advances in Design Automation* **32**(1) 241–248.

Rockafellar, R. T., R. J. B. Wets. 1991. Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.* **16**(1) 119–147.

Rustem, B., M. Howe. 2002. *Algorithms for Worst-case Design and Applications to Risk Management*. Princeton University Press.

Ruszczynski, A., A. Shapiro. 2004. Optimization of risk measures. Risk and insurance, EconWPA.

Seliger, P., M. Mahvash, C. Wang, A. F. J. Levi. 2006. Optimization of aperiodic dielectric structures. *Journal of Applied Physics* **100** 4310.

Shor, N. Z. 1998. *Nondifferentiable Optimization and Polynomial Problems*. Econometrics and Operations Research, Kluwer Academic Publishers.

Simpson, T. W., J. D. Poplinski, P. N. Koch, J. K. Allen. 2001. Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers* **17**(2) 129–150.

Uryasev, S., R. T. Rockafellar. 2001. Conditional value-at-risk: optimization approach. *Stochastic optimization: algorithms and applications (Gainesville, FL, 2000)*, *Appl. Optim.*, vol. 54. Kluwer Acad. Publ., Dordrecht, 411–435.