INFORMS Journal on Computing
Articles in Advance, pp. 1–15
ISSN 1091-9856 | EISSN 1526-5528

DOI 10.1287/ijoc.1090.0319

Nonconvex Robust Optimization for Problems with Constraints

Dimitris Bertsimas, Omid Nohadani, Kwong Meng Teo
Operations Research Center and Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 {dbertsim@mit.edu, nohadani@mit.edu, kwongmeng@alum.mit.edu}

We propose a new robust optimization method for problems with objective functions that may be computed via numerical simulations and incorporate constraints that need to be feasible under perturbations. The proposed method iteratively moves along descent directions for the robust problem with nonconvex constraints and terminates at a robust local minimum. We generalize the algorithm further to model parameter uncertainties. We demonstrate the practicability of the method in a test application on a nonconvex problem with a polynomial cost function as well as in a real-world application to the optimization problem of intensity-modulated radiation therapy for cancer treatment. The method significantly improves the robustness for both designs.

Key words: optimization; robust optimization; nonconvex optimization; constraints
History: Accepted by John Hooker, Area Editor for Constraint Programming and Optimization; received August 2007; revised September 2008, September 2008; accepted December 2008. Published online in *Articles in Advance*.

1. Introduction

In recent years, there has been considerable literature on robust optimization, which has primarily focused on convex optimization problems whose objective functions and constraints were given explicitly and had specific structure (linear, convex quadratic, conic-quadratic, and semidefinite) (Ben-Tal and Nemirovski 1998, 2003; Bertsimas and Sim 2003, 2006). In an earlier paper, we proposed a local search method for solving unconstrained robust optimization problems, whose objective functions are given via numerical simulation and may be nonconvex; see Bertsimas et al. (2009).

In this paper, we extend our approach to solve constrained robust optimization problems, assuming that cost and constraints as well as their gradients are provided. We also consider how the efficiency of the algorithm can be improved if some constraints are convex. We first consider problems with only implementation errors and then extend our approach to admit cases with implementation and parameter uncertainties.

The rest of the paper is structured as follows: In §2 a brief review on unconstrained robust nonconvex optimization along with the necessary definitions are provided. In §3, the robust local search, as we proposed in Bertsimas et al. (2009), is generalized to handle constrained optimization problems with implementation errors. We also explore how the efficiency of the algorithm can be improved if some of the constraints are convex. In §4, we further generalize the algorithm to admit problems with implementation and parameter uncertainties. In §5, we discuss an application involving a polynomial cost function to develop intuition. We show that the robust local search can be more efficient when the simplicity of constraints are exploited. In §6 we report on an application in an actual health-care problem in intensity-modulated radiation therapy for cancer treatment. This problem has 85 decision variables and is highly nonconvex.

2. Review on Robust Nonconvex Optimization

In this section, we review the robust nonconvex optimization for problems with implementation errors, as we introduced in Bertsimas et al. (2009, 2007). We discuss the notion of the descent direction for the robust problem, which is a vector that points away from all the worst implementation errors. Consequently, a robust local minimum is a solution at which no such direction can be found.

2.1. Problem Definition

The nominal cost function, possibly nonconvex, is denoted by $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$ is the design vector. The *nominal optimization problem* is

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

In general, there are two common forms of perturbations: (1) *implementation errors*, which are caused in an imperfect realization of the desired decision variables $\mathbf{x}$; and (2) *parameter uncertainties*, which are due to modeling errors during the problem definition, such as noise. Note that our discussion on parameter errors

in §4 also extends to other sources of errors, such as deviations between a computer simulation and the underlying model (e.g., numerical noise) or the difference between the computer model and the metamodel, as discussed by Stinstra and den Hertog (2007). For ease of exposition, we first introduce a robust optimization method for implementation errors only, as they may occur during the fabrication process.

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 a design $\mathbf{x}+\Delta\mathbf{x}$. Here, $\Delta\mathbf{x}$ is assumed to reside within an uncertainty set

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

Note that $\Gamma > 0$ is a scalar describing the size of perturbation against which the design needs to be protected. Although our approach applies to other norms $\|\Delta\mathbf{x}\|_p \le \Gamma$ in (2) (p being a positive integer, including $p=\infty$), we present the case of $p=2$. We seek a robust design $\mathbf{x}$ by minimizing the *worst-case cost*

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

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

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

In other words, the robust optimization method seeks to minimize the worst-case cost. 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

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

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

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

2.2. Robust Local Search Algorithm

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

$$\begin{aligned} \min_{\mathbf{d},\beta}\ & \beta \\ \text{s.t.}\ & \|\mathbf{d}\|_2 \le 1, \\ & \mathbf{d}'\Delta\mathbf{x}^* \le \beta \quad \forall\, \Delta\mathbf{x}^* \in \mathcal{U}^*(\hat{\mathbf{x}}), \\ & \beta \le -\epsilon, \end{aligned} \tag{7}$$

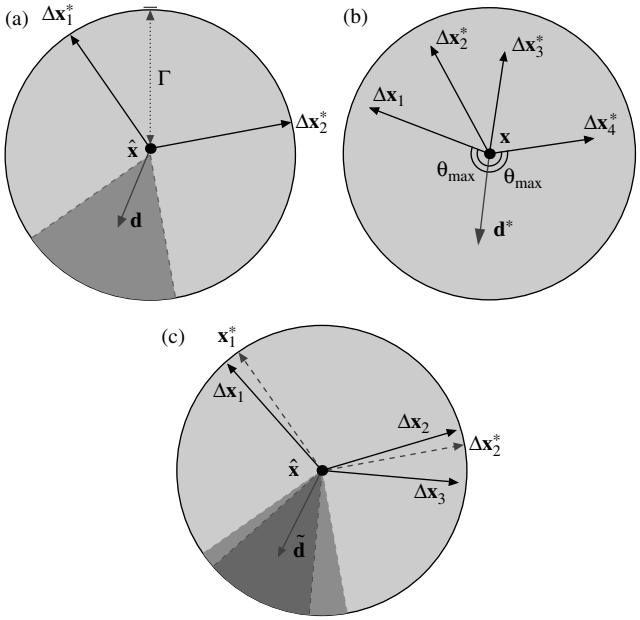

Figure 1 A Two-Dimensional Illustration of the Neighborhood; For a Design $\hat{\mathbf{x}}$, All Possible Implementation Errors $\Delta\mathbf{x} \in \mathcal{U}$ Are Contained in the Shaded Circle

Notes. (a) The bold arrow $\mathbf{d}$ shows a possible descent direction and thin arrows $\Delta\mathbf{x}_i^*$ represent worst errors. (b) The solid arrow indicates the optimal direction $\mathbf{d}^*$ that makes the largest possible angle $\theta_{\max} = \cos^{-1}\beta^* \ge 90°$ with all $\Delta\mathbf{x}^*$. (c) Without knowing all $\Delta\mathbf{x}^*$, the direction $\tilde{\mathbf{d}}$ points away from all $\Delta\mathbf{x}_j \in \mathcal{M} = \{\Delta\mathbf{x}_1, \Delta\mathbf{x}_2, \Delta\mathbf{x}_3\}$, when all $\mathbf{x}_i^*$ lie within the cone spanned by $\Delta\mathbf{x}_j$.

where ϵ is a small positive scalar. A feasible solution to problem (7), $\mathbf{d}^*$, forms the maximum possible angle $\theta_{\max}$ with all $\Delta\mathbf{x}^*$. An example is illustrated in Figure 1(b). This angle is always greater than 90° because of the constraint that $\beta \le -\epsilon < 0$. When ϵ is sufficiently small, and problem (7) is infeasible, $\hat{\mathbf{x}}$ is a good estimate of a robust local minimum. Note that the constraint $\|\mathbf{d}^*\|_2 = 1$ is automatically satisfied if the problem is feasible. Such a SOCP can be solved efficiently using both commercial and noncommercial solvers.

Consequently, if we have an oracle returning $\mathcal{U}^*(\mathbf{x})$, we can iteratively find descent directions and use them to update the current iterates. In most real-world instances, however, we cannot expect to find $\Delta\mathbf{x}^*$. Therefore, an alternative approach is required. We argue in Bertsimas et al. (2009) that descent directions can be found without knowing the worst implementation errors $\Delta\mathbf{x}^*$ exactly. As illustrated in Figure 1(c), 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. If this set satisfies condition

$$\Delta\mathbf{x}^* = \sum_{i\,|\,\Delta\mathbf{x}_i\in\mathcal{M}} \alpha_i \Delta\mathbf{x}_i, \tag{8}$$

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 summarizes a heuristic strategy for the robust local search.

Algorithm 1.

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 $\mathbf{x}^k$. For this we conduct multiple gradient ascent sequences. The results of all function evaluations $(\mathbf{x}, f(\mathbf{x}))$ are recorded in a history set $\mathcal{H}^k$, combined with all past histories. The set $\mathcal{M}^k$ includes elements of $\mathcal{H}^k$, which are within the neighborhood and have highest costs.

Step 2. *Robust Local Move*:

(i) Solve a SOCP (similar to Problem (7), but with the set $\mathcal{U}^*(\mathbf{x}^k)$ replaced by set $\mathcal{M}^k$); terminate 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.

Bertsimas et al. (2009) provide a detailed discussion on the actual implementation. Next, we generalize this robust local search algorithm to problems with constraints.

3. Constrained Problem Under Implementation Errors

3.1. Problem Definition

Consider the nominal optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h_j(\mathbf{x}) \le 0 \quad \forall j, \end{aligned} \tag{9}$$

where the objective function and the constraints may be nonconvex. To find a design that is robust against implementation errors $\Delta\mathbf{x}$, we formulate the robust problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_{\Delta\mathbf{x}\in\mathcal{U}} f(\mathbf{x}+\Delta\mathbf{x}) \\ \text{s.t.} \quad & \max_{\Delta\mathbf{x}\in\mathcal{U}} h_j(\mathbf{x}+\Delta\mathbf{x}) \le 0 \quad \forall j, \end{aligned} \tag{10}$$

where the uncertainty set $\mathcal{U}$ is given by

$$\mathcal{U} := \{\Delta\mathbf{x} \in \mathbb{R}^n \mid \|\Delta\mathbf{x}\|_2 \le \Gamma\}. \tag{11}$$

A design is robust if, and only if, no constraints are violated for any errors in $\mathcal{U}$. Of all the robust designs, we seek one with the lowest worst-case cost $g(\mathbf{x})$. When a design $\hat{\mathbf{x}}$ is implemented with errors in $\mathcal{U}$, the realized design falls within the neighborhood

$$\mathcal{N} := \{\mathbf{x} \mid \|\mathbf{x}-\hat{\mathbf{x}}\|_2 \le \Gamma\}. \tag{12}$$

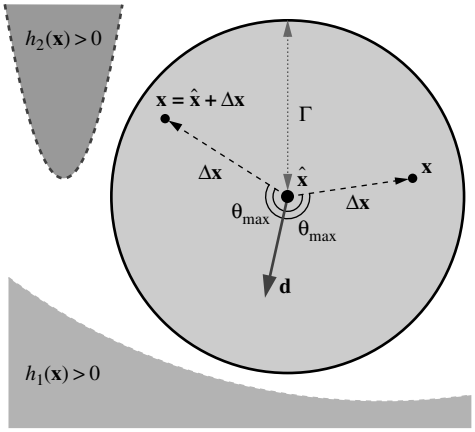

Figure 2 A Two-Dimensional Illustration of the Neighborhood $\mathcal{N}$ in the Design Space x

Notes. The shaded regions contain designs violating the constraints $h_j(\mathbf{x}) > 0$. Note that h_1 is a convex constraint but not h_2.

Figure 2 illustrates the neighborhood $\mathcal{N}$ of a design $\hat{\mathbf{x}}$ along with the constraints. $\hat{\mathbf{x}}$ is robust if, and only if, none of its neighbors violates any constraints. Equivalently, there is no overlap between the neighborhood of $\hat{\mathbf{x}}$ and the shaded regions $h_j(\mathbf{x}) > 0$ in Figure 2.

3.2. Robust Local Search for Problems with Constraints

When constraints do not come into play in the vicinity of the neighborhood of $\hat{\mathbf{x}}$, the worst-case cost can be reduced iteratively, using the robust local search algorithm for the unconstrained problem as discussed in §2. The additional procedures for the robust local search algorithm that are required when constraints are present are as follows:

(1) *Neighborhood search*. To determine if there are neighbors violating constraint h_j, the constraint maximization problem

$$\max_{\Delta\mathbf{x}\in\mathcal{U}} h_j(\hat{\mathbf{x}}+\Delta\mathbf{x}) \tag{13}$$

is solved using multiple gradient ascents from different starting designs. Gradient ascents are used because problem (13) is not a convex optimization problem, in general. We shall consider in §3.3 the case where h_j is an explicitly given convex function, and consequently, problem (13) can be solved using more efficient techniques. If a neighbor has a constraint value exceeding zero, for any constraint, it is recorded in a history set $\mathcal{Y}$.

(2) *Check feasibility under perturbations*. If $\hat{\mathbf{x}}$ has neighbors in the history set $\mathcal{Y}$, then it is not feasible under perturbations. Otherwise, the algorithm treats $\hat{\mathbf{x}}$ as feasible under perturbations.

(3a) *Robust local move if $\hat{\mathbf{x}}$ is not feasible under perturbations*. Because constraint violations are more important than cost considerations, and because we want the

algorithm to operate within the feasible region of the robust problem, nominal cost is ignored when neighbors violating constraints are encountered. To ensure that the new neighborhood does not contain neighbors in $\mathcal{Y}$, an update step along a direction $\mathbf{d}^*_{\text{feas}}$ is taken. This is illustrated in Figure 3(a). Here, $\mathbf{d}^*_{\text{feas}}$ makes the largest possible angle with all the vectors $\mathbf{y}_i - \hat{\mathbf{x}}$. Such a $\mathbf{d}^*_{\text{feas}}$ can be found by solving the SOCP:

$$\begin{aligned} \min_{\mathbf{d},\,\beta}\ & \beta \\ \text{s.t.}\ & \|\mathbf{d}\|_2 \le 1, \\ & \mathbf{d}'\left(\frac{\mathbf{y}_i - \hat{\mathbf{x}}}{\|\mathbf{y}_i - \hat{\mathbf{x}}\|_2}\right) \le \beta \quad \forall\, \mathbf{y}_i \in \mathcal{Y}, \\ & \beta \le -\epsilon. \end{aligned} \tag{14}$$

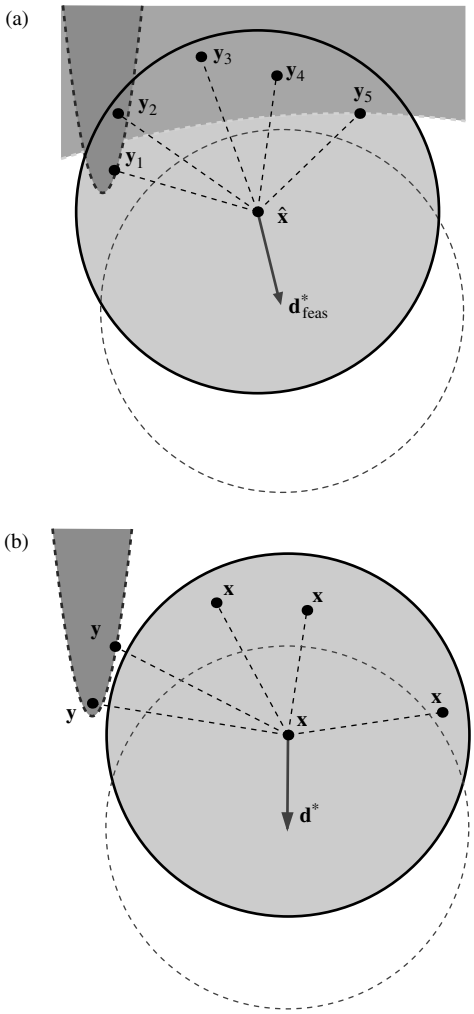

Figure 3 A Two-Dimensional Illustration of the Robust Local Move

Notes. (a) When $\hat{\mathbf{x}}$ is nonrobust, the upper shaded regions contain constraint-violating designs, including infeasible neighbors $\mathbf{y}_i$. Vector $\mathbf{d}^*_{\text{feas}}$ points away from all $\mathbf{y}_i$. (b) When $\hat{\mathbf{x}}$ is robust, $\mathbf{x}_i$ denotes a bad neighbor with high nominal cost, while $\mathbf{y}_i$ denotes an infeasible neighbor lying just outside the neighborhood. The circle with the broken circumference denotes the updated neighborhood.

As shown in Figure 3(a), a sufficiently large step along $\mathbf{d}^*_{\text{feas}}$ yields a robust design.

(3b) *Robust local move if $\hat{\mathbf{x}}$ is feasible under perturbations*. When $\hat{\mathbf{x}}$ is feasible under perturbations, the update step is similar to that for an unconstrained problem, as in §2. However, ignoring designs that violate constraints and lie just beyond the neighborhood might lead to a nonrobust design. This issue is taken into account when determining an update direction $\mathbf{d}^*_{\text{cost}}$, as illustrated in Figure 3(b). This update direction $\mathbf{d}^*_{\text{cost}}$ can be found by solving the SOCP:

$$\begin{aligned} \min_{\mathbf{d},\,\beta}\ & \beta \\ \text{s.t.}\ & \|\mathbf{d}\|_2 \le 1, \\ & \mathbf{d}'\left(\frac{\mathbf{x}_i - \hat{\mathbf{x}}}{\|\mathbf{x}_i - \hat{\mathbf{x}}\|_2}\right) \le \beta \quad \forall\, \mathbf{x}_i \in \mathcal{M}, \\ & \mathbf{d}'\left(\frac{\mathbf{y}_i - \hat{\mathbf{x}}}{\|\mathbf{y}_i - \hat{\mathbf{x}}\|_2}\right) \le \beta \quad \forall\, \mathbf{y}_i \in \mathcal{Y}_+, \\ & \beta \le -\epsilon, \end{aligned} \tag{15}$$

where $\mathcal{M}$ contains neighbors with the highest cost within the neighborhood, and $\mathcal{Y}_+$ is the set of known infeasible designs lying in the slightly enlarged neighborhood $\mathcal{N}_+$,

$$\mathcal{N}_+ := \{\mathbf{x} \mid \|\mathbf{x} - \hat{\mathbf{x}}\|_2 \le (1+\delta)\Gamma\}, \tag{16}$$

δ being a small positive scalar for designs that lie just beyond the neighborhood, as illustrated in Figure 3(b). Since $\hat{\mathbf{x}}$ is robust, there are no infeasible designs in the neighborhood $\mathcal{N}$. Therefore, all infeasible designs in $\mathcal{Y}_+$ lie at a distance between Γ and $(1+\delta)\Gamma$.

Termination Criteria. We shall first define the robust local minimum for a problem with constraints.

Definition 1. $\mathbf{x}^*$ is a robust local minimum for the problem with constraints if the following conditions apply:

(i) *Feasible under perturbations*: $\mathbf{x}^*$ remains feasible under perturbations,

$$h_j(\mathbf{x}^* + \Delta\mathbf{x}) \le 0 \quad \forall\, j,\ \forall\, \Delta\mathbf{x} \in \mathcal{U}, \quad \text{and} \tag{17}$$

(ii) *No descent direction*: There are no improving directions $\mathbf{d}^*_{\text{cost}}$ at $\mathbf{x}^*$.

Given the above definition, we can only terminate at Step (3b), where $\mathbf{x}^*$ is feasible under perturbations. Furthermore, for there to be no direction $\mathbf{d}^*_{\text{cost}}$ at $\mathbf{x}^*$, it must be surrounded by neighbors with high cost and infeasible designs in $\mathcal{N}_+$.

3.3. Enhancements When Constraints Are Convex

In this section, we review the case when $\mathbf{h}_i$ is explicitly given as a convex function. If problem (13) is convex, it can be solved with techniques that are more efficient than multiple gradient ascents. Table 1 summarizes the required procedures for solving problem (13).

Table 1 Algorithms to Solve Problem (13)

$\mathbf{h}_j(\mathbf{x})$	Problem (13)	Required computation
$\mathbf{a}'\mathbf{x}+b$	$\mathbf{a}'\hat{\mathbf{x}}+\Gamma\|\mathbf{a}\|_2+b\le 0$	Solve LP
$\mathbf{x}'\mathbf{Q}\mathbf{x}+2\mathbf{b}'\mathbf{x}+c$, $\mathbf{Q}$ symmetric	Single-trust region problem	One SDP in the worst case
$-\mathbf{h}_j$ is convex	Convex problem	One gradient ascent

For symmetric constraints, the resulting single-trust region problem can be expressed as $\max_{\Delta\mathbf{x}\in\mathcal{U}} \Delta\mathbf{x}'\mathbf{Q}\Delta\mathbf{x}+2(\mathbf{Q}\hat{\mathbf{x}}+\mathbf{b})'\Delta\mathbf{x}+\hat{\mathbf{x}}\mathbf{Q}'\hat{\mathbf{x}}+2\mathbf{b}'\hat{\mathbf{x}}+c$. The possible improvements to the robust local search are as follows:

(1) *Neighborhood search*. Solve problem (13) with the corresponding method of Table 1 instead of multiple gradient ascents to improve the computational efficiency.

(2) *Check feasibility under perturbations*. If $h_j^{\text{rob}}(\hat{\mathbf{x}}) \equiv \max_{\Delta\mathbf{x}\in\mathcal{U}} h_j(\hat{\mathbf{x}}+\Delta\mathbf{x}) > 0$, $\hat{\mathbf{x}}$ not feasible under perturbations.

(3) *Robust local move*. To warrant that all designs in the new neighborhood are feasible, the direction should be chosen such that it points away from the infeasible regions. The corresponding vectors describing the closest points in $h_j^{\text{rob}}(\hat{\mathbf{x}})$ are given by $\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}})$ as illustrated in Figure 4. Therefore, $\mathbf{d}$ has to satisfy

$$\mathbf{d}'_{\text{feas}}\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}}) < \beta\|\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}})\|_2$$

and

$$\mathbf{d}'_{\text{cost}}\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}}) < \beta\|\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}})\|_2$$

in SOCP (14) and SOCP (15), respectively. Note that $\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}}) = \nabla_{\mathbf{x}} h(\hat{\mathbf{x}}+\Delta\mathbf{x}_j^*)$, which can be evaluated easily.

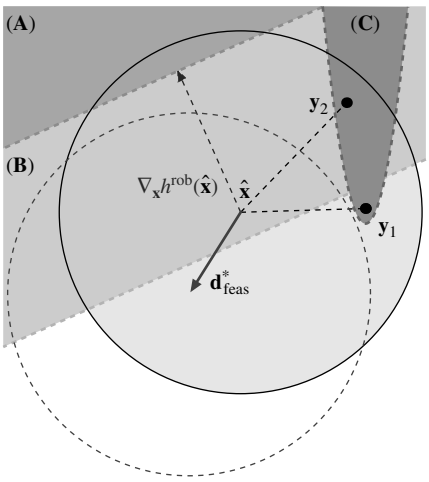

Figure 4 A Two-Dimensional Illustration of the Neighborhood When One of the Violated Constraints Is a Linear Function

Notes. (A) denotes the infeasible region. Because $\hat{\mathbf{x}}$ has neighbors in region (A), $\hat{\mathbf{x}}$ lies in the infeasible region of its robust counterpart (B). $\mathbf{y}_i$ denotes neighbors that violate a nonconvex constraint, shown in region (C). $\mathbf{d}^*_{\text{feas}}$ denotes a direction that would reduce the infeasible region within the neighborhood and points away from the gradient of the robust counterpart and all bad neighbors $\mathbf{y}_i$. The dashed circle represents the updated neighborhood.

In particular, if h_j is a linear constraint, then $h_j^{\text{rob}}(\mathbf{x}) = \mathbf{a}'\mathbf{x}+\Gamma\|\mathbf{a}\|_2+b \le 0$ is the same for all $\mathbf{x}$. Consequently, we can replace the constraint

$$\max_{\Delta\mathbf{x}\in\mathcal{U}} h_j(\mathbf{x}+\Delta\mathbf{x}) = \max_{\Delta\mathbf{x}\in\mathcal{U}} \mathbf{a}'(\mathbf{x}+\Delta\mathbf{x}) \le 0$$

with its robust counterpart $h_j^{\text{rob}}(\mathbf{x})$. Here, $h_j^{\text{rob}}(\mathbf{x})$ is a constraint on $\mathbf{x}$ without any uncertainties, as illustrated in Figure 4.

3.4. Constrained Robust Local Search Algorithm

In this section, we use the methods outlined in §§3.2 and 3.3 to formalize the overall algorithm:

Algorithm 2 [Constrained Robust Local Search].

Step 0. *Initialization*: Set $k := 1$. Let $\mathbf{x}^1$ be an arbitrary decision vector.

Step 1. *Neighborhood Search*:

(i) Find neighbors with high cost through $n+1$ gradient ascent sequences, where n is the dimension of $\mathbf{x}$. Record all evaluated neighbors and their costs in a history set $\mathcal{H}^k$, together with $\mathcal{H}^{k-1}$.

(ii) Let $\mathcal{J}$ be the set of constraints to the convex constraint maximization problem (13) that are convex. Find optimizer $\Delta\mathbf{x}_j^*$ and the highest constraint value $h_j^{\text{rob}}(\mathbf{x}^k)$ for all $j \in \mathcal{J}$, according to the methods listed in Table 1. Let $\bar{\mathcal{J}} \subseteq \mathcal{J}$ be the set of constraints that are violated under perturbations.

(iii) For every constraint $j \notin \mathcal{J}$, find infeasible neighbors by applying $n+1$ gradient ascent sequences on problem (13), with $\hat{\mathbf{x}} = \mathbf{x}^k$. Record all infeasible neighbors in a history set $\mathcal{Y}^k$, together with set $\mathcal{Y}^{k-1}$.

Step 2. *Check Feasibility Under Perturbations*: $\mathbf{x}^k$ is not feasible under perturbations if either $\mathcal{Y}^k$ or $\bar{\mathcal{J}}$ is not empty.

Step 3. *Robust Local Move*:

(i) If $\mathbf{x}^k$ is not feasible under perturbations, solve SOCP (14) with additional constraints $\mathbf{d}'_{\text{feas}}\nabla_{\mathbf{x}} h_j^{\text{rob}}(\mathbf{x}^k) < \beta\|\nabla_{\mathbf{x}} h_j^{\text{rob}}(\mathbf{x}^k)\|_2$ for all $j \in \bar{\mathcal{J}}$. Find direction d^*_{feas} and set $\mathbf{x}^{k+1} := \mathbf{x}^{k+1} + t^k d^*_{\text{feas}}$.

(ii) If $\mathbf{x}^k$ is feasible under perturbations, solve SOCP (15) to find a direction d^*_{cost}. Set $\mathbf{x}^{k+1} := \mathbf{x}^{k+1} + t^k d^*_{\text{feas}}$. If no direction d^*_{cost} exists, reduce the size of $\mathcal{M}$; if the size is below a threshold, terminate.

In Steps 3(i) and 3(ii), t^k is the minimum distance chosen such that the undesirable designs are excluded from the neighborhood of the new iterate $\mathbf{x}^{k+1}$. Finding t^k requires solving a simple geometric problem. For more details, refer to Bertsimas et al. (2009).

4. Generalization to Include Parameter Uncertainties

4.1. Problem Definition

Consider the nominal problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \bar{\mathbf{p}}) \\ \text{s.t.} \quad & h_j(\mathbf{x}, \bar{\mathbf{p}}) \le 0 \quad \forall j, \end{aligned} \tag{18}$$

where $\bar{\mathbf{p}} \in \mathbb{R}^m$ is a coefficient vector of the problem parameters. For our purpose, we can restrict $\bar{\mathbf{p}}$ to parameters with perturbations only. For example, if problem (18) is given by

$$\begin{aligned} \min_{\mathbf{x}} \quad & 4x_1^3 + x_2^2 + 2x_1^2 x_2 \\ \text{s.t.} \quad & 3x_1^2 + 5x_2^2 \le 20, \end{aligned}$$

then we can extract

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{and} \quad \bar{\mathbf{p}} = \begin{pmatrix} 4 \\ 1 \\ 2 \\ 3 \\ 5 \\ 20 \end{pmatrix}.$$

Note that uncertainties can even be present in the exponent, e.g., 3 in the monomial $4x_1^3$.

In addition to implementation errors, there can be perturbations $\Delta\mathbf{p}$ in parameters $\bar{\mathbf{p}}$ as well. The true, but unknown, parameter $\mathbf{p}$ can then be expressed as $\bar{\mathbf{p}} + \Delta\mathbf{p}$. To protect the design against both types of perturbations, we formulate the robust problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \max_{\Delta\mathbf{z} \in \mathcal{U}} f(\mathbf{x} + \Delta\mathbf{x}, \bar{\mathbf{p}} + \Delta\mathbf{p}) \\ \text{s.t.} \quad & \max_{\Delta\mathbf{z} \in \mathcal{U}} h_j(\mathbf{x} + \Delta\mathbf{x}, \bar{\mathbf{p}} + \Delta\mathbf{p}) \le 0 \quad \forall\, j, \end{aligned} \tag{19}$$

where $\Delta\mathbf{z} = \binom{\Delta\mathbf{x}}{\Delta\mathbf{p}}$. Here, $\Delta\mathbf{z}$ lies within the uncertainty set

$$\mathcal{U} = \left\{ \Delta\mathbf{z} \in \mathbb{R}^{n+m} \,\middle|\, \|\Delta\mathbf{z}\|_2 \le \Gamma \right\}, \tag{20}$$

where $\Gamma > 0$ is a scalar describing the size of perturbations we want to protect the design against. Similar to problem (10), a design is robust only if no constraints are violated under the perturbations. Among these robust designs, we seek to minimize the worst-case cost

$$g(\mathbf{x}) := \max_{\Delta\mathbf{z} \in \mathcal{U}} f(\mathbf{x} + \Delta\mathbf{x}, \bar{\mathbf{p}} + \Delta\mathbf{p}). \tag{21}$$

4.2. Generalized Constrained Robust Local Search Algorithm

Problem (19) is equivalent to the following problem with implementation errors only:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \max_{\Delta\mathbf{z} \in \mathcal{U}} f(\mathbf{z} + \Delta\mathbf{z}) \\ \text{s.t.} \quad & \max_{\Delta\mathbf{z} \in \mathcal{U}} h_j(\mathbf{z} + \Delta\mathbf{z}) \le 0 \quad \forall\, j, \\ & \mathbf{p} = \bar{\mathbf{p}}, \end{aligned} \tag{22}$$

where $\mathbf{z} = \binom{\mathbf{x}}{\mathbf{p}}$. The idea behind generalizing the constrained robust local search algorithm is analogous to the approach described in §2 for the unconstrained problem, discussed in Bertsimas et al. (2009). Consequently, the necessary modifications to Algorithm 2 are as follows:

(1) *Neighborhood search*. Given $\hat{\mathbf{x}}$, $\hat{\mathbf{z}} = \binom{\hat{\mathbf{x}}}{\bar{\mathbf{p}}}$ is the decision vector. Therefore, the neighborhood can be described as

$$\mathcal{N} := \left\{ \mathbf{z} \,\middle|\, \|\mathbf{z} - \hat{\mathbf{z}}\|_2 \le \Gamma \right\} = \left\{ \begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} \,\middle|\, \left\| \begin{pmatrix} \mathbf{x} - \hat{\mathbf{x}} \\ \mathbf{p} - \bar{\mathbf{p}} \end{pmatrix} \right\|_2 \le \Gamma \right\}. \tag{23}$$

(2) *Robust local move*. Let $\mathbf{d}^* = \binom{\mathbf{d}_x^*}{\mathbf{d}_p^*}$ be an update direction in the $\mathbf{z}$ space. Because $\mathbf{p}$ is not a decision vector but a given system parameter, the algorithm has to ensure that $\mathbf{p} = \bar{\mathbf{p}}$ is satisfied at every iterate. Thus, $\mathbf{d}_p^* = \mathbf{0}$.

When finding the update direction, the condition $\mathbf{d}_p = \mathbf{0}$ must be included in either SOCP (14) or SOCP (15) along with the feasibility constraints $\mathbf{d}' \nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}}) < \beta \|\nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{x}})\|_2$. As discussed earlier, we seek a direction $\mathbf{d}$ that points away from the worst-case and infeasible neighbors. We achieve this objective by maximizing the angle between $\mathbf{d}$ and all worst-case neighbors as well as the angle between $\mathbf{d}$ and the gradient of all constraints. For example, if a design $\mathbf{z}$ is not feasible under perturbations, the SOCP is given by

$$\begin{aligned} \min_{\mathbf{d}=(\mathbf{d}_x, \mathbf{d}_p),\, \beta} \quad & \beta \\ \text{s.t.} \quad & \|\mathbf{d}\|_2 \le 1, \\ & \mathbf{d}'(\mathbf{z}_i - \hat{\mathbf{z}}) \le \beta \|\mathbf{z}_i - \hat{\mathbf{z}}\|_2 \quad \forall\, \mathbf{y}_i \in \mathcal{Y}, \\ & \mathbf{d}' \nabla_{\mathbf{z}} h_j^{\text{rob}}(\hat{\mathbf{z}}) < \beta \|\nabla_{\mathbf{z}} h_j^{\text{rob}}(\hat{\mathbf{z}})\|_2 \quad \forall\, j \in \bar{\mathcal{F}}, \\ & \mathbf{d}_p = \mathbf{0}, \\ & \beta \le -\epsilon. \end{aligned}$$

Here, $\mathcal{Y}^k$ is the set of infeasible designs in the neighborhood. Since the p-component of $\mathbf{d}$ is zero, this problem reduces to

$$\begin{aligned} \min_{\mathbf{d}_x,\, \beta} \quad & \beta \\ \text{s.t.} \quad & \|\mathbf{d}_x\|_2 \le 1, \\ & \mathbf{d}_x'(\mathbf{x}_i - \hat{\mathbf{x}}) \le \beta \|\mathbf{z}_i - \hat{\mathbf{z}}\|_2 \quad \forall\, \mathbf{y}_i \in \mathcal{Y}, \\ & \mathbf{d}_x' \nabla_{\mathbf{x}} h_j^{\text{rob}}(\hat{\mathbf{z}}) < \beta \|\nabla_{\mathbf{z}} h_j^{\text{rob}}(\hat{\mathbf{z}})\|_2 \quad \forall\, j \in \bar{\mathcal{F}}, \\ & \beta \le -\epsilon. \end{aligned} \tag{24}$$

A similar approach is carried out for the case where $\mathbf{z}$ is robust. Consequently, both $\mathbf{d}_{\text{feas}}^*$ and $\mathbf{d}_{\text{cost}}^*$ satisfy $\mathbf{p} = \bar{\mathbf{p}}$ at every iteration. This is illustrated in Figure 5.

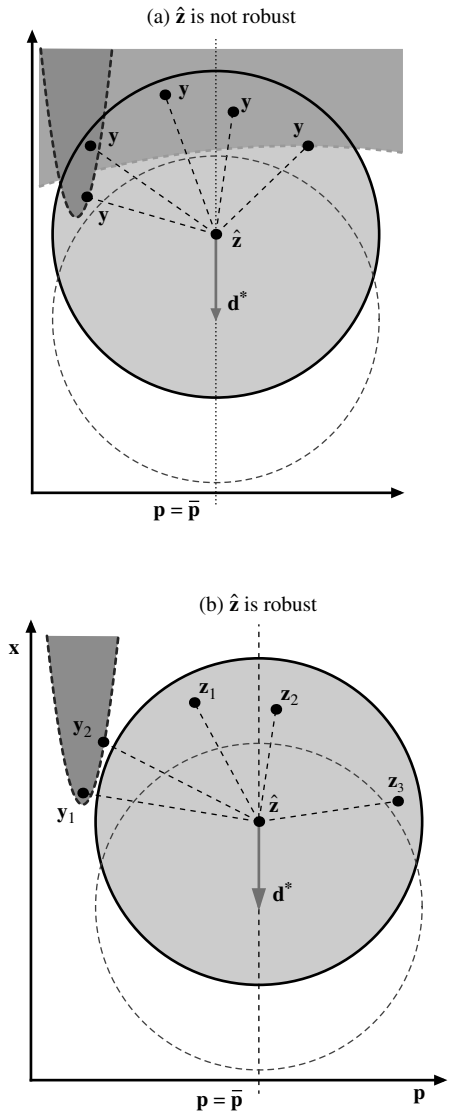

Figure 5 A Two-Dimensional Illustration of the Robust Local Move for Problems with Both Implementation Errors and Parameter Uncertainties

Notes. The neighborhood spans the $\mathbf{z} = (\mathbf{x}, \mathbf{p})$ space: (a) the constrained counterpart of Figure 3(a), and (b) the constrained counterpart of Figure 3(b). Note that the direction found must lie within the hyperplanes $\mathbf{p} = \bar{\mathbf{p}}$.

Now we have arrived at the constrained robust local search algorithm for problem (19) with both implementation errors and parameter uncertainties:

Algorithm 3 [Generalized Constrained Robust Local Search].

Step 0. *Initialization*: Set $k := 1$. Let $\mathbf{x}^1$ be an arbitrary initial decision vector.

Step 1. *Neighborhood Search*: Same as Step 1 in Algorithm 2, but over the neighborhood (23).

Step 2. *Check Feasibility Under Perturbations*: $\mathbf{z}^k$, and equivalently $\mathbf{x}^k$, is feasible under perturbations, if $\mathcal{Y}^k$ and $\bar{\mathcal{F}}^k$ are empty.

Step 3. *Robust Local Move*:

(i) If $\mathbf{z}^k$ is not feasible under perturbations, find a direction d^*_{feas} by solving SOCP (24) with $\hat{\mathbf{z}} = \mathbf{z}^k$. Set $\mathbf{z}^{k+1} := \mathbf{z}^{k+1} + t^k d^*_{\text{feas}}$.

(ii) If $\mathbf{x}$ is feasible under perturbations, solve the SOCP:

$$
\begin{aligned}
\min_{\mathbf{d}_x, \beta} \quad & \beta \\
\text{s.t.} \quad & \|\mathbf{d}_x\|_2 \le 1, \\
& \mathbf{d}_x'(\mathbf{x}_i - \mathbf{x}^k) \le \beta \left\| \begin{pmatrix} \mathbf{x}_i - \mathbf{x}^k \\ \mathbf{p}_i - \bar{\mathbf{p}} \end{pmatrix} \right\|_2 \\
& \qquad\qquad \forall\, \mathbf{z}_i \in \mathcal{M}^k,\ \mathbf{z}_i = \begin{pmatrix} \mathbf{x}_i \\ \mathbf{p}_i \end{pmatrix}, \\
& \mathbf{d}_x' \left(\mathbf{x}_i - \mathbf{x}^k\right) \le \beta \left\| \begin{pmatrix} \mathbf{x}_i - \mathbf{x}^k \\ \mathbf{p}_i - \bar{\mathbf{p}} \end{pmatrix} \right\|_2 \\
& \qquad\qquad \forall\, \mathbf{y}_i \in \mathcal{Y}^k_+,\ \mathbf{y}_i = \begin{pmatrix} \mathbf{x}_i \\ \mathbf{p}_i \end{pmatrix}, \\
& \mathbf{d}_x' \nabla_{\mathbf{x}} h_j^{\text{rob}}(\mathbf{z}^k) < \beta \|\nabla_{\mathbf{z}} h_j^{\text{rob}}(\mathbf{z}^k)\|_2 \quad \forall\, j \in \bar{\mathcal{F}}_+, \\
& \beta \le -\epsilon,
\end{aligned}
\tag{25}
$$

to find a direction d^*_{cost}. $\mathcal{Y}^k_+$ is the set of infeasible designs in the enlarged neighborhood $\mathcal{N}^k_+$ as in Equation (16). $\bar{\mathcal{F}}_+$ is the set of constraints that are not violated in the neighborhood of $\hat{\mathbf{x}}$ but are violated in the slightly enlarged neighborhood $\mathcal{N}_+$. Set $\mathbf{z}^{k+1} := \mathbf{z}^{k+1} + t^k d^*_{\text{feas}}$. If no direction d^*_{cost} exists, reduce the size of $\mathcal{M}$; if the size is below a threshold, terminate.

We have finished introducing the robust local search method with constraints. In the following sections, we will present two applications that showcase the performance of this method.

5. Application I: Problem with Polynomial Cost Function and Constraints

5.1. Problem Description

The first problem is sufficiently simple so as to develop intuition into the algorithm. Consider the nominal problem

$$
\begin{aligned}
\min_{x,\,y} \quad & f_{\text{poly}}(x, y) \\
\text{s.t.} \quad & h_1(x, y) \le 0, \\
& h_2(x, y) \le 0,
\end{aligned}
\tag{26}
$$

where

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

$$h_1(x, y) = (x - 1.5)^4 + (y - 1.5)^4 - 10.125,$$
$$h_2(x, y) = -(2.5 - x)^3 - (y + 1.5)^3 + 15.75.$$

Given implementation errors $\Delta = \binom{\Delta x}{\Delta y}$ such that $\|\Delta\|_2 \le 0.5$, the robust problem is

$$\begin{aligned} \min_{x,y} \quad & \max_{\|\Delta\|_2 \le 0.5} f_{\text{poly}}(x + \Delta x, y + \Delta y) \\ \text{s.t.} \quad & \max_{\|\Delta\|_2 \le 0.5} h_1(x + \Delta x, y + \Delta y) \le 0, \\ & \max_{\|\Delta\|_2 \le 0.5} h_2(x + \Delta x, y + \Delta y) \le 0. \end{aligned} \tag{27}$$

To the best of our knowledge, there are no practical ways to solve such a robust problem, given today's technology (see Lasserre 2006). If the relaxation method for polynomial optimization problems is used, as in Henrion and Lasserre (2003), problem (27) leads to a large polynomial semidefinite program (SDP) problem, which cannot yet be solved in practice (see Kojima 2003, Lasserre 2006). In Figure 6, contour plots of the nominal and the estimated worst-case cost surface along with their local and global extrema are shown to generate intuition for the performance of the robust optimization method. The computation takes less than 10 minutes on an Intel Xeon 3.4 GHz to terminate, thus it is fast enough for a prototype problem. Three different initial designs with their respective neighborhoods are sketched as well.

5.2. Computation Results

For the constrained problem (26), the nonconvex cost surface and the feasible region are shown in Figure 6(a). Note that the feasible region is not convex because h_2 is not a convex constraint. Let $g_{\text{poly}}(x, y)$ be the worst-case cost function given as

$$g_{\text{poly}}(x, y) := \max_{\|\Delta\|_2 \le 0.5} f_{\text{poly}}(x + \Delta x, y + \Delta y).$$

Figure 6(b) shows the worst-case cost estimated by using sampling on the cost surface f_{poly}. In the robust problem (27), we seek to find a design that minimizes $g_{\text{poly}}(x, y)$ such that its neighborhood lies within the unshaded region. An example of such a design is the point C in Figure 6(b).

Two separate robust local searches were carried out from initial designs A and B. The initial design A exemplifies initial configurations whose neighborhood contains infeasible designs and is close to a local minimum. The design B represents only configurations whose neighborhood contains infeasible designs. Figure 7 shows that in both instances, the algorithm terminated at designs that are feasible under perturbations and have significantly lower worst-case costs. However, it converged to different robust local minima in the two instances, as shown in Figure 7(c). The presence of multiple robust local minima is not surprising because $g_{\text{poly}}(x, y)$ is nonconvex. Figure 7(c) also shows that both robust local minima I and II satisfy the terminating conditions as stated in §3.2:

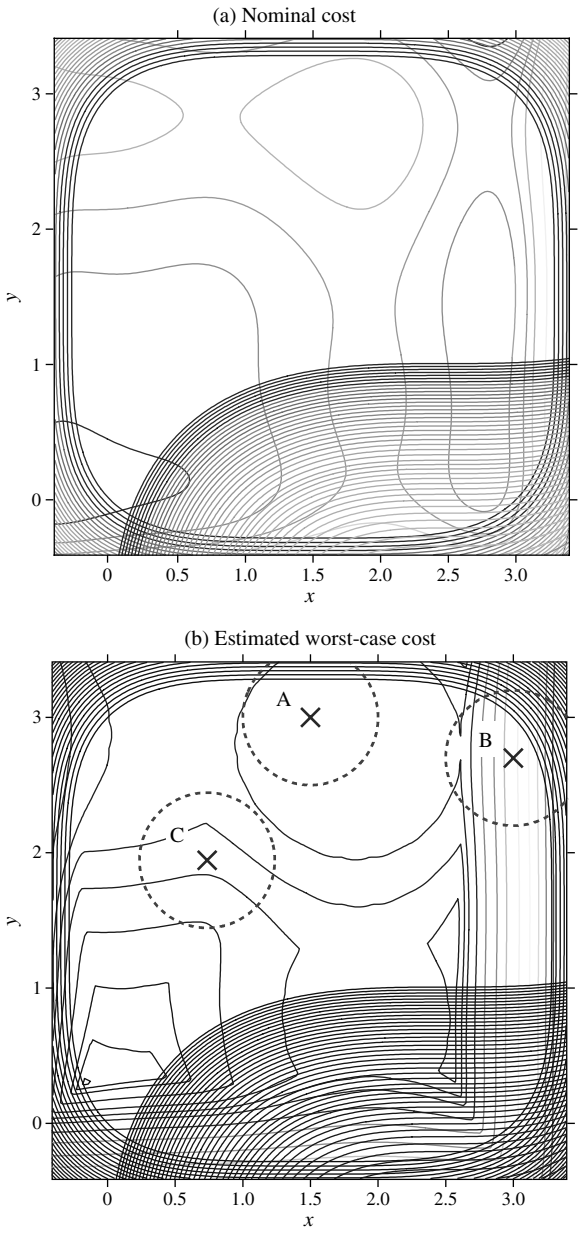

Figure 6 Contour Plot of (a) The Nominal Cost Function and (b) The Estimated Worst-Case Cost Function in Application I

Notes. The shaded regions denote designs that violate at least one of the two constraints, h_1 and h_2. Although points A and B are feasible, they are not feasible under perturbations because of their infeasible neighbors. Point C, on the other hand, remains feasible under perturbations.

(1) *Feasible under perturbations*. Both their neighborhoods do not overlap with the shaded regions.

(2) *No direction* $\mathbf{d}^*_{\text{cost}}$ *found*. Both designs are surrounded by bad neighbors and infeasible designs lying just outside their respective neighborhoods. Note that for robust local minimum II, the bad neighbors lie on the same contour line even though they

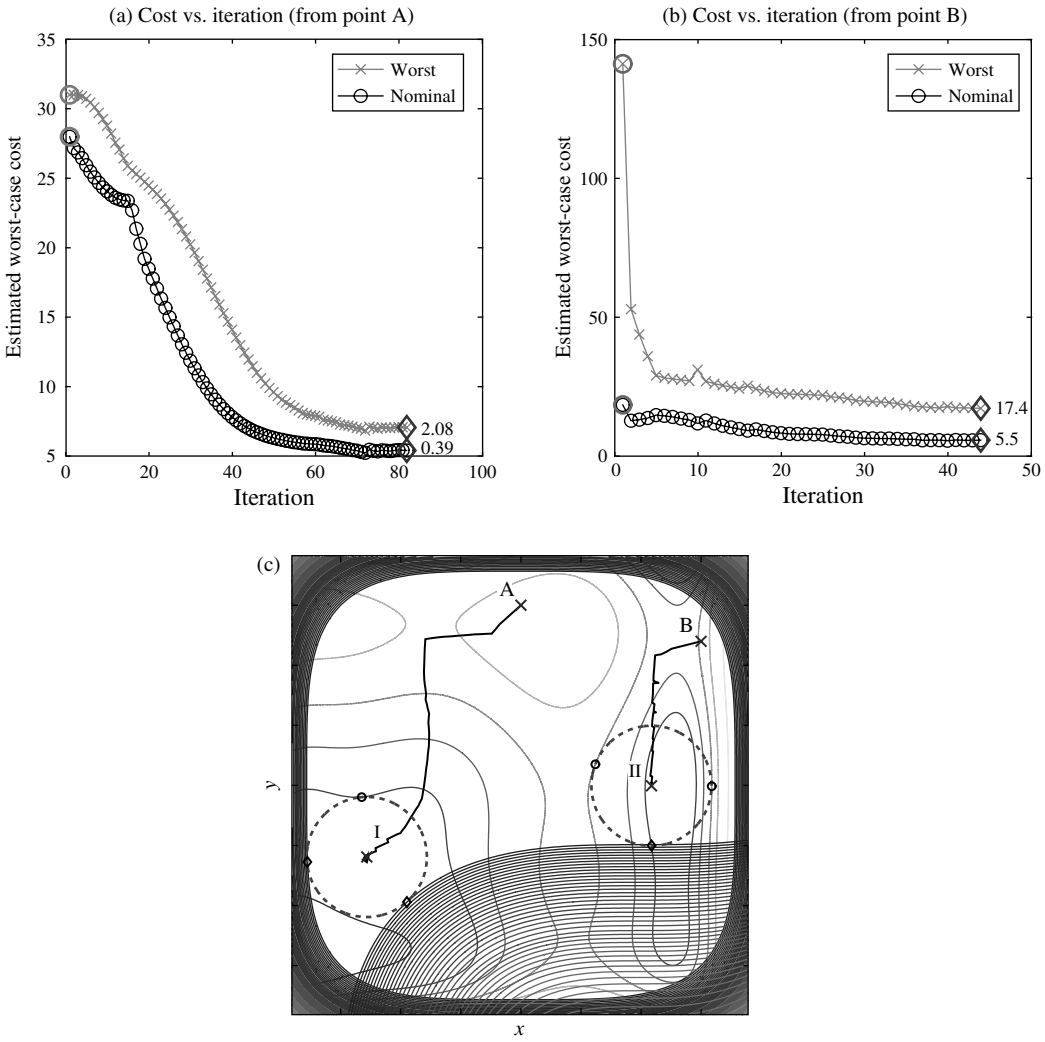

Figure 7 Performance of the Robust Local Search Algorithm in Application I from Two Different Starting Points, A and B

Notes. The circle marker indicates the starting and the diamond marker the final design. (a) Starting from point A, the algorithm reduces the worst-case cost and the nominal cost. (b) Starting from point B, the algorithm converges to a different robust solution, which has a significantly larger worst-case cost and nominal cost. (c) The broken circles sketch the neighborhood of minima. For each minimum, (i) there is no overlap between its neighborhood and the shaded infeasible regions, and (ii) there is no improving direction because it is surrounded by neighbors of high cost (bold circle) and infeasible designs (bold diamond) residing just beyond the neighborhood. Two bad neighbors of minimum II (starting from B) share the same cost, since they lie on the same contour line.

are apart, indicating that any further improvement is restricted by the infeasible neighboring designs.

5.3. When Constraints Are Linear

In §3.3, we argued that the robust local search can be more efficient if the constraints are explicitly given as convex functions. To illustrate this, suppose that the constraints in problem (26) are linear and given by

$$\begin{aligned} h_1(x, y) &= 0.6x - y + 0.17, \\ h_2(x, y) &= -16x - y - 3.15. \end{aligned} \tag{28}$$

As shown in Table 1, the robust counterparts of the constraints in Equation (28) are

$$\begin{aligned} h_1^{\text{rob}}(x, y) &= 0.6x - y + 0.17 + 0.5831 \le 0, \\ h_2^{\text{rob}}(x, y) &= -16x - y - 3.15 + 8.0156 \le 0. \end{aligned} \tag{29}$$

The benefit of using the explicit counterparts in Equation (29) is that the algorithm terminates in only 96 seconds as opposed to 3,600 seconds when using the initial linear constraints in Equation (28).

6. Application II: A Problem in Intensity-Modulated Radiation Therapy for Cancer Treatment

Radiation therapy is a key component in cancer treatment today. In this form of treatment, ionizing radiation is directed onto cancer cells with the objective of destroying their DNA and consequently causing cell death. Unfortunately, healthy and noncancerous cells are exposed to the destructive radiation as well, since cancerous tumors are embedded within the patient's

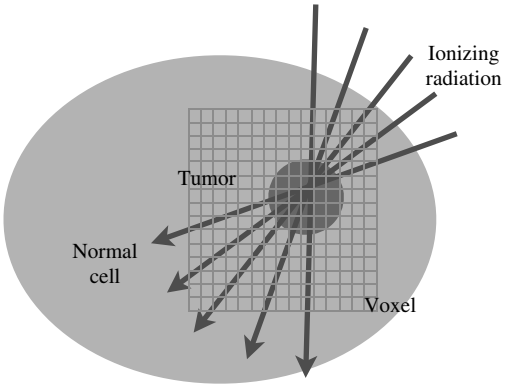

Figure 8 **Multiple Ionizing Radiation Beams Are Directed at Cancerous Cells**

body. Even though healthy cells can repair themselves, an important objective behind the planning process is to minimize the total radiation received by the patient ("objective") while ensuring that the tumor is subjected to a sufficient level of radiation ("constraints").

Most radiation oncologists adopt the technique of intensity-modulated radiation therapy (IMRT; see Bortfeld 2006). In IMRT, the dose distribution is controlled by two sets of decisions. First, instead of a single beam, multiple beams of radiation from different angles are directed onto the tumor. This is illustrated in Figure 8. In the actual treatment, this is accomplished using a rotatable oncology system, which can be varied in angle on a plane perpendicular to the patient's length-axis. Furthermore, the beam can be regarded as assembled by a number of beamlets. By choosing the beam angles and the beamlet intensities ("decision variables"), it is desirable to make the treated volume conform as closely as possible to the target volume, thereby minimizing radiation dosage to possible organ-at-risk (OAR) and normal tissues. For a detailed introduction to IMRT and various related techniques, see Bortfeld (2006) and the references therein.

The area of simultaneous optimization of beam intensity and beam angle in IMRT has been studied in the recent past, mainly by successively selecting a set of angles from a set of predefined directions and optimizing the respective beam intensities (Djajaputra et al. 2003). So far, however, the issue of robustness has only been addressed for a fixed set of beam angles, e.g., in Chan et al. (2006). In this work, we address the issue of robustly optimizing both the beam angles and the beam intensities—to the best of our knowledge, for the first time. We apply the presented robust optimization method to a clinically relevant case that has been downsized due to numerical restrictions.

Optimization Problem. We obtained our optimization model through a joint research project with the Radiation Oncology group of the Massachusetts General Hospital at the Harvard Medical School. In our model, all affected body tissues are divided into volume elements called voxels v (see Deasy et al. 2003). The voxels belong to three sets:

- $\mathcal{T}$: set of tumor voxels, with $|\mathcal{T}| = 145$;
- $\mathcal{O}$: set of organ-at-risk voxels, with $|\mathcal{O}| = 42$;
- $\mathcal{N}$: set of normal tissue voxels, with $|\mathcal{N}| = 1{,}005$.

Let the set of all voxels be $\mathcal{V}$. Therefore, $\mathcal{V} = \mathcal{T} \cup \mathcal{O} \cup \mathcal{N}$ and $|\mathcal{V}| = 1{,}192$; there are a total of 1,192 voxels, determined by the actual sample case we have used for this study. Moreover, there are five beams from five different angles. Each beam is divided into 16 beamlets. Let $\theta \in \mathbb{R}^5$ denote the vector of beam angles and let $\mathcal{I}$ be the set of beams. In addition, let $\mathcal{B}_i$ be the set of beamlets b corresponding to beam i, $i \in \mathcal{I}$. Furthermore, let x_i^b be the intensity of beamlet b, with $b \in \mathcal{B}_i$, and let $\mathbf{x} \in \mathbb{R}^{16\times 5}$ be the vector of beamlet intensities,

$$\mathbf{x} = \left(\mathbf{x}_1^1, \ldots, \mathbf{x}_1^{16}, \mathbf{x}_2^1, \ldots, \mathbf{x}_5^{16}\right)'. \tag{30}$$

Finally, let $D_v^b(\theta_i)$ be the absorbed dosage per unit radiation in voxel v introduced by beamlet b from beam i. Thus, $\sum_i \sum_b D_v^b(\theta_i) x_i^b$ denotes the total dosage deposited in voxel v under a treatment plan $(\theta, \mathbf{x})$. The objective is to minimize a weighted sum of the radiation dosage in all voxels while ensuring that (1) a minimum dosage l_v is delivered to each tumor voxel $v \in \mathcal{T}$, and (2) the dosage in each voxel v does not exceed an upper limit u_v. Consequently, the nominal optimization problem is

$$\begin{aligned}
\min_{\mathbf{x},\,\theta} \quad & \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} c_v D_v^b(\theta_i) x_i^b \\
\text{s.t.} \quad & \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} D_v^b(\theta_i) x_i^b \ge l_v \quad \forall\, v \in \mathcal{T}, \\
& \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} D_v^b(\theta_i) x_i^b \le u_v \quad \forall\, v \in \mathcal{V}, \\
& x_i^b \ge 0 \quad \forall\, b \in \mathcal{B}_i,\ \forall\, i \in \mathcal{I},
\end{aligned} \tag{31}$$

where term c_v is the penalty of a unit dose in voxel v. The penalty for a voxel in OAR is set much higher than the penalty for a voxel in the normal tissue. Note that if θ is given, problem (31) reduces to a linear program (LP), and the optimal intensities $\mathbf{x}^*(\theta)$ can be found more efficiently. However, the problem is nonconvex in θ because varying a single θ_i changes $D_v^b(\theta_i)$ for all voxel v and for all $b \in \mathcal{B}_i$.

Let θ' represent the available discretized beam angles. To get $D_v^b(\theta')$, the values at $\theta' = 0°, 2°, \ldots, 358°$ were derived using CERR, a numerical solver for radiotherapy research introduced by Deasy et al. (2003). Subsequently, for a given $\hat{\theta}$, $D_v^b(\hat{\theta})$ is obtained by the linear interpolation

$$D_v^b(\hat{\theta}) = \frac{\theta' - \hat{\theta} + 2°}{2°} \cdot D_v^b(\theta') + \frac{\hat{\theta} - \theta'}{2°} \cdot D_v^b(\theta' + 2°), \tag{32}$$

where $\theta' = 2\lfloor \hat{\theta}/2 \rfloor$. It is not practical to use the numerical solver to evaluate $D_v^b(\hat{\theta})$ directly during optimization because the computation takes too much time.

Model of Uncertainty. When a design $(\theta, \mathbf{x})$ is implemented, the realized design can be erroneous and take the form $(\theta + \Delta\theta, \mathbf{x} \otimes (\mathbf{1} \pm \delta))$, where $\otimes$ refers to an element-wise multiplication. The sources of errors include equipment limitations, differences in patient's posture when measuring and irradiating, and minute body movements. These perturbations are estimated to be normally and independently distributed:

$$\begin{aligned} \delta_i^b &\sim \mathcal{N}(0, 0.01), \\ \Delta\theta_i &\sim \mathcal{N}\left(0, \tfrac{1}{3}^\circ\right). \end{aligned} \tag{33}$$

Note that by scaling δ_i^b by 0.03, we obtain $\delta_i^b \sim \mathcal{N}(0, \frac{1}{3})$, and hence all components of the vector $\binom{\delta/0.03}{\Delta\theta}$ obey an $\mathcal{N}(0, \frac{1}{3})$ distribution. Under the robust optimization approach, we define the uncertainty set:

$$\mathcal{U} = \left\{ \begin{pmatrix} \delta \\ 0.03 \\ \Delta\theta \end{pmatrix} \middle| \left\| \begin{matrix} \delta \\ 0.03 \\ \Delta\theta \end{matrix} \right\|_2 \le \Gamma \right\}. \tag{34}$$

Given this uncertainty set, the corresponding robust problem can be expressed as

$$\begin{aligned} \min_{\mathbf{x},\,\theta} \quad & \max_{(\delta,\,\Delta\theta)\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} c_v D_v^b(\theta_i + \Delta\theta_i) x_i^b (1+\delta_i^b) \\ \text{s.t.} \quad & \min_{(\delta,\,\Delta\theta)\in\mathcal{U}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} D_v^b(\theta_i + \Delta\theta_i) x_i^b (1+\delta_i^b) \ge l_v \\ & \qquad\qquad \forall\, v \in \mathcal{T}, \\ & \max_{(\delta,\,\Delta\theta)\in\mathcal{U}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} D_v^b(\theta_i + \Delta\theta_i) x_i^b (1+\delta_i^b) \le u_v \\ & \qquad\qquad \forall\, v \in \mathcal{V}, \\ & x_i^b \ge 0 \quad \forall\, b \in \mathcal{B}_i,\ \forall\, i \in \mathcal{I}. \end{aligned} \tag{35}$$

Approximating the Robust Problem. Since the clinical data in $D_v^b(\theta)$ are already available for angles $\theta' = 0^\circ, 2^\circ, 4^\circ, \ldots, 358^\circ$ with a resolution of $\Delta\theta = \pm 2^\circ$, it is not practical to apply the robust local search on problem (35) directly. Instead, we approximate the problem with a formulation that can be evaluated more efficiently. Because $D_v^b(\theta_i)$ is obtained through a linear interpolation, it can be rewritten as

$$D_v^b(\theta_i \pm \Delta\theta) \approx D_v^b(\theta_i) \pm \frac{D_v^b(\phi + 2^\circ) - D_v^b(\phi)}{2^\circ} \Delta\theta, \tag{36}$$

where $\phi = 2\lfloor \theta_i/2 \rfloor$. $D_v^b(\phi)$, $D_v^b(\phi + 2^\circ)$ are values obtained from the numerical solver. Note that since $\theta_i \pm \Delta\theta \in [\phi, \phi + 2^\circ]$ for all $\Delta\theta$, Equation (36) is exact.

Let $(\partial/\partial\theta) D_v^b(\theta_i) = (D_v^b(\phi + 2^\circ) - D_v^b(\phi))/2^\circ$. Then,

$$\begin{aligned} & D_v^b(\theta_i + \Delta\theta_i) \cdot x_i^b \cdot (1 + \delta_i^b) \\ & \quad \approx \left(D_v^b(\theta_i) + \frac{\partial}{\partial\theta} D_v^b(\theta_i) \cdot \Delta\theta_i \right) \cdot x_i^b \cdot (1 + \delta_i^b) \\ & \quad = D_v^b(\theta_i) \cdot x_i^b + D_v^b(\theta_i) \cdot x_i^b \cdot \delta_i^b + \frac{\partial}{\partial\theta} D_v^b(\theta_i) \cdot x_i^b \cdot \Delta\theta_i \\ & \qquad + \frac{\partial}{\partial\theta} D_v^b(\theta_i) \cdot x_i^b \cdot \Delta\theta_i \cdot \delta_i^b \\ & \quad \approx D_v^b(\theta_i) \cdot x_i^b + D_v^b(\theta_i) \cdot x_i^b \cdot \delta_i^b + \frac{\partial}{\partial\theta} D_v^b(\theta_i) \cdot x_i^b \cdot \Delta\theta_i. \end{aligned} \tag{37}$$

In the final approximation step, the second-order terms are dropped. Using Equation (37) repeatedly leads to

$$\begin{aligned} & \max_{(\delta,\,\Delta\theta)\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} c_v \cdot D_v^b(\theta_i + \Delta\theta_i) \cdot x_i^b \cdot (1 + \delta_i^b) \\ & \approx \max_{(\delta,\,\Delta\theta)\in\mathcal{U}} \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} \Big(c_v \cdot D_v^b(\theta_i) \cdot x_i^b + c_v \cdot D_v^b(\theta_i) \cdot x_i^b \cdot \delta_i^b \\ & \qquad\qquad + c_v \cdot \frac{\partial}{\partial\theta} D_v^b(\theta_i) \cdot x_i^b \cdot \Delta\theta_i \Big) \\ & = \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \\ & \quad + \max_{(\delta,\,\Delta\theta)\in\mathcal{U}} \left\{ \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} \left(\sum_{v\in\mathcal{V}} c_v D_v^b(\theta_i) \right) x_i^b \delta_i^b \right. \\ & \qquad \left. + \sum_{i\in\mathcal{I}} \left(\sum_{b\in\mathcal{B}_i} \left(\sum_{v\in\mathcal{V}} c_v \frac{\partial}{\partial\theta} D_v^b(\theta_i) \right) x_i^b \right) \Delta\theta_i \right\} \\ & = \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{I}} \sum_{b\in\mathcal{B}_i} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \\ & \quad + \Gamma \left\| \begin{matrix} 0.03 \cdot \sum_{v\in\mathcal{V}} c_v \cdot D_v^1(\theta_1) \cdot x_1^1 \\ \vdots \\ 0.03 \cdot \sum_{v\in\mathcal{V}} c_v \cdot D_v^{16}(\theta_1) \cdot x_1^{16} \\ 0.03 \cdot \sum_{v\in\mathcal{V}} c_v \cdot D_v^1(\theta_2) \cdot x_2^1 \\ \vdots \\ 0.03 \cdot \sum_{v\in\mathcal{V}} c_v \cdot D_v^{16}(\theta_5) \cdot x_5^{16} \\ \sum_{b\in\mathcal{B}_1} \sum_{v\in\mathcal{V}} c_v \cdot \frac{\partial}{\partial\theta} D_v^b(\theta_1) \cdot x_1^b \\ \vdots \\ \sum_{b\in\mathcal{B}_5} \sum_{v\in\mathcal{V}} c_v \cdot \frac{\partial}{\partial\theta} D_v^b(\theta_5) \cdot x_5^b \end{matrix} \right\|_2. \end{aligned} \tag{38}$$

Note that the maximum in the second term is determined via the boundaries of the uncertainty set in Equation (34). For better reading, the beamlet and angle components are explicitly written out. Because all the terms in the constraints are similar to those in

the objective function, the constraints can be approximated using the same approach. To simplify the notation, the 2-norm term in Equation (38) will be represented by

$$\left\| \begin{array}{l} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_1} \sum_{v \in \mathcal{V}} c_v \cdot \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{array} \right\|_2 .$$

Using this procedure, we obtain the nonconvex robust problem

$$\begin{aligned} \min_{\mathbf{x},\theta} \quad & \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \\ & + \Gamma \left\| \begin{array}{l} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_i} \sum_{v \in \mathcal{V}} c_v \cdot \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{array} \right\|_2 \\ \text{s.t.} \quad & \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} D_v^b(\theta_i) \cdot x_i^b \\ & - \Gamma \left\| \begin{array}{l} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_i} \sum_{v \in \mathcal{V}} \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{array} \right\|_2 \geq l_v \qquad (39) \\ & \qquad \forall v \in \mathcal{T}, \\ & \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} D_v^b(\theta_i) \cdot x_i^b \\ & + \Gamma \left\| \begin{array}{l} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_i} \sum_{v \in \mathcal{V}} \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{array} \right\|_2 \leq u_v \\ & \qquad \forall v \in \mathcal{V}, \\ & x_i^b \geq 0 \quad \forall b \in \mathcal{B}_i, \forall i \in \mathcal{I}, \end{aligned}$$

which closely approximates the original robust problem (35). Note that when θ and $\mathbf{x}$ are known, the objective cost and all the constraint values can be computed efficiently.

6.1. Computation Results

We used the following algorithm to find a large number of robust designs $(\theta^k, \mathbf{x}^k)$, for $k = 1, 2, \ldots$.

Algorithm 4 [Algorithm Applied to the IMRT Problem].

Step 0. *Initialization*: Let $(\theta^0, \mathbf{x}^0)$ be the initial design and let Γ^0 be the initial value. Set $k := 1$.

Step 1. Set $\Gamma^k := \Gamma^{k-1} + \Delta\Gamma$, where $\Delta\Gamma$ is a small scalar and can be negative.

Step 2. Find a robust local minimum by applying Algorithm 2 with

(i) initial design $(\theta^{k-1}, \mathbf{x}^{k-1})$, and

(ii) uncertainty set (34) with $\Gamma = \Gamma^k$.

Step 3. $(\theta^k, \mathbf{x}^k)$ is the robust local minimum.

Step 4. Set $k := k + 1$. Go to Step 1; if $k > k_{\max}$, terminate.

For comparison, two initial designs $(\theta^0, \mathbf{x}^0)$ were used:

(i) "nominal best," which is a local minimum of the nominal problem; and

(ii) "strict interior," which is a design lying in the strict interior of the feasible set of the nominal problem. It is determined by a local minimum to the following problem:

$$\begin{aligned} \min_{\mathbf{x},\theta} \quad & \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} c_v D_v^b(\theta_i) x_i^b \\ \text{s.t.} \quad & \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} D_v^b(\theta_i) x_i^b \geq l_v + \text{buffer} \quad \forall v \in \mathcal{T}, \\ & \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} D_v^b(\theta_i) x_i^b \leq u_v - \text{buffer} \quad \forall v \in \mathcal{V}, \\ & x_i^b \geq 0 \quad \forall b \in \mathcal{B}_i, \forall i \in \mathcal{I}. \end{aligned}$$

From the nominal best, Algorithm 4 is applied with an increasing Γ^k. We choose $\Gamma^0 = 0$ and $\Delta\Gamma = 0.001$ for all k. $k_{\max}$ was set to be 250. It is estimated that beyond this value a further increase of Γ would simply increase the cost without reducing the probability any further. Because the nominal best is an optimal solution to the LP, it lies on the extreme point of the feasible set. Consequently, even small perturbations can violate the constraints. In every iteration of Algorithm 4, Γ^k is increased slowly. With each new iteration, the terminating design will remain feasible under a larger perturbation.

The strict interior design, on the other hand, will not violate the constraints under larger perturbations because of the *buffer* introduced. However, this increased robustness comes with a higher nominal cost. By evaluating problem (39), the strict interior was found to satisfy the constraints for $\Gamma \leq 0.05$. Thus, we apply Algorithm 4 using this initial design twice:

(i) $\Gamma^0 = 0.05$ and $\Delta\Gamma = 0.001$, for all k. $k_{\max}$ was set to 150, and

(ii) $\Gamma^0 = 0.051$ and $\Delta\Gamma = -0.001$, for all k. $k_{\max}$ was set to 50.

All designs $(\theta^k, \mathbf{x}^k)$ were assessed for their performance under implementation errors, using 10,000 normally distributed random scenarios, as in Equation (33).

Pareto Frontiers. In general, an increase in robustness of a design often results in higher cost.

For randomly perturbed cases, the average performance of a plan is the clinically relevant measure. Therefore, when comparing robust designs, we look at the mean cost and the probability of violation. The results for the mean cost are similar to those of the worst simulated cost. Therefore, we only report on mean costs. Furthermore, based on empirical

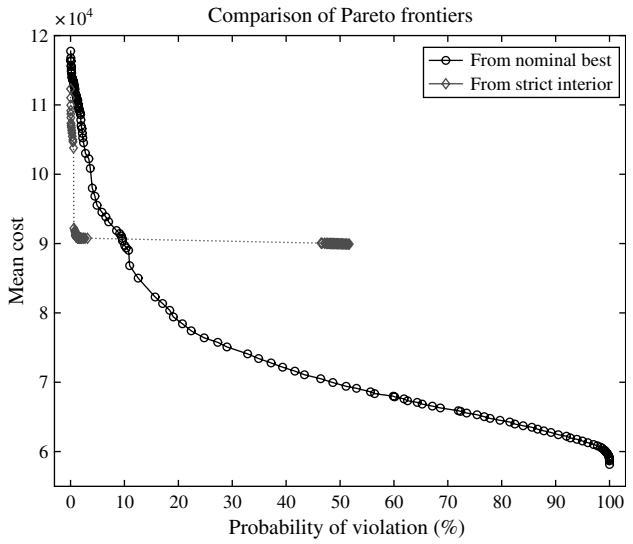

Figure 9 Pareto Frontiers Attained by Algorithm 4 for Different Initial Designs: Nominal Best and Strict Interior

Notes. Starting from nominal best, the designs have lower costs when the required probability of violation is high. When the required probability is low, however, the designs found from strict interior perform better.

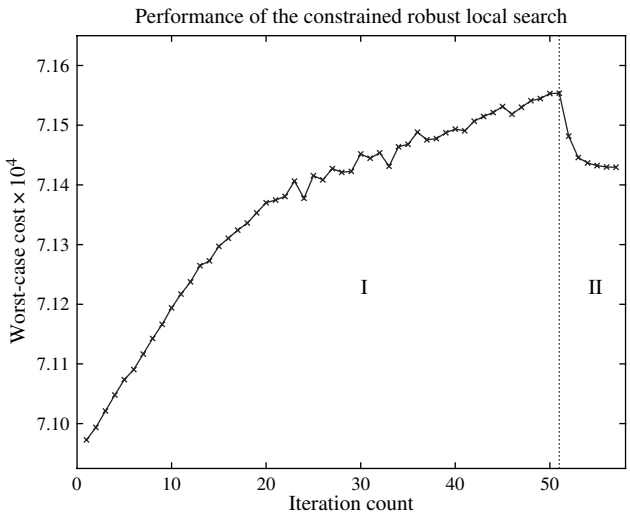

Figure 10 A Typical Robust Local Search Carried Out in Step 2 of Algorithm 4

Notes. In phase I, the algorithm searches for a robust design without considering the worst-case cost. Because of the trade-off between cost and feasibility, the worst-case cost increases during this phase. At the end of phase I, a robust design is found. In phase II, the algorithm improves the worst-case cost.

evidence, random sampling is not a good gauge for the worst-case cost. To get an improved worst-case cost estimate, multiple gradient ascents are necessary. However, this is not practical in this context due to the large number of designs involved.

Because multiple performance measures are considered, the best designs lie on a Pareto frontier that reflects the trade-off between these objectives. Figure 9 shows two Pareto frontiers, nominal best and strict interior, as initial designs. When the probability of violation is high, the designs found starting from the nominal best have lower costs. However, if the constraints have to be satisfied with a higher probability, designs found from the strict interior perform better. Furthermore, the strategy of increasing Γ slowly in Algorithm 4 provides trade-offs between robustness and cost, thus enabling the algorithm to map out the Pareto frontier in a single sweep, as indicated in Figure 9.

This Pareto frontier allows clinicians to choose the best robustly optimized plan based on a desired probability of violation; e.g., if the organs at risk are not very critical, this probability might be relaxed to attain a plan with a lower mean cost, thus delivering less mean dose to all organs.

Different Modes in a Robust Local Search. The robust local search has two distinct phases. When iterates are not robust, the search first seeks a robust design with no consideration of worst-case costs. (See Step 3(i) in Algorithm 3.) After a robust design has been found, the algorithm then improves the worst-case cost until a robust local minimum has been found. (See Step 3(ii) in Algorithm 3.) These two phases are illustrated in Figure 10 for a typical robust local search carried out in Application II. Note that the algorithm takes around 20 hours on an Intel Xeon 3.4 GHz to terminate.

6.2. Comparison with Convex Robust Optimization Techniques

When θ is fixed, the resulting subproblem is convex. Therefore, convex robust optimization techniques can be used even though the robust problem (39) is not convex. Moreover, the resulting subproblem becomes a SOCP problem when θ is fixed in problem (39). Therefore, we are able to find a robustly optimized intensity $\mathbf{x}^*(\theta)$. Since all the constraints are addressed, $(\theta, \mathbf{x}^*(\theta))$ is a robust design. Thus, the problem reduces to finding a local minimum θ. We use a steepest descent algorithm with a finite-difference estimate of the gradients. $J_{\text{rob}}(\theta^k)$ shall denote the cost of problem (39) for $\theta := \theta^k$. The algorithm can be summarized as follows:

Algorithm 5 [Algorithm Using Convex Techniques in the IMRT Problem].

Step 0. *Initialization*: Set $k := 1$. θ^1 denotes the initial design.

Step 1. Obtain $\mathbf{x}^k(\theta^k)$ by

(a) Solving problem (39) with $\theta := \theta^k$.

(b) Setting $\mathbf{x}^k$ to the optimal solution of the subproblem.

Step 2. Estimate gradient $(\partial/\partial\theta)J_{\text{rob}}(\theta^k)$ using finite differences:

(a) Solve problem (39) with $\theta = \theta^k \pm \epsilon \cdot \mathbf{e}_i$, for all $i \in \mathcal{I}$, where ϵ is a small positive scalar and $\mathbf{e}_i$ is a unit vector in the ith coordinate.

(b) For all $i \in \mathcal{I}$,

$$\frac{\partial J_{\text{rob}}}{\partial \theta_i} = \frac{J(\theta^k + \epsilon \mathbf{e}_i) - J(\theta^k - \epsilon \mathbf{e}_i)}{2 \cdot \epsilon}.$$

Step 3. Check terminating condition:

(a) If $\|\partial J_{\text{rob}}/\partial \theta\|_2$ is sufficiently small, terminate. Else, take the steepest descent step

$$\theta^{k+1} := \theta^k - t^k \frac{\partial J_{\text{rob}}}{\partial \theta},$$

where t^k is a small and diminishing step size.

(b) Set $k := k + 1$, go to Step 1.

Unfortunately, Algorithm 5 cannot be implemented because the subproblem cannot be computed efficiently, even though it is convex. With 1,192 SOCP constraints, it takes more than a few hours for CPLEX 9.1 to solve the problem. Given that 11 subproblems are solved in every iteration and more than a hundred iterations are carried in each run of Algorithm 5, we need an alternative subproblem.

Therefore, we refine the definition of the uncertainty set. Instead of an ellipsoidal uncertainty set (34), which describes the independently distributed perturbations, we use the polyhedral uncertainty set

$$\mathcal{U} = \left\{ \begin{pmatrix} \delta \\ 0.03 \\ \Delta\theta \end{pmatrix} \middle| \left\| \begin{matrix} \delta \\ 0.03 \\ \Delta\theta \end{matrix} \right\|_p \le \Gamma \right\}, \tag{40}$$

with norm $p = 1$ or $p = \infty$. The resulting subproblem becomes

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \\
& + \Gamma \left\| \begin{matrix} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} c_v \cdot D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_1} \sum_{v \in \mathcal{V}} c_v \cdot \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{matrix} \right\|_q \\
\text{s.t.} \quad & \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} D_v^b(\theta_i) \cdot x_i^b \\
& - \Gamma \left\| \begin{matrix} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_1} \sum_{v \in \mathcal{V}} \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{matrix} \right\|_q \ge l_v \qquad \forall v \in \mathcal{T}, \\
& \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{B}_i} D_v^b(\theta_i) \cdot x_i^b \\
& + \Gamma \left\| \begin{matrix} \left\{ 0.03 \cdot \sum_{v \in \mathcal{V}} D_v^b(\theta_i) \cdot x_i^b \right\}_{b,i} \\ \left\{ \sum_{b \in \mathcal{B}_1} \sum_{v \in \mathcal{V}} \frac{\partial}{\partial \theta} D_v^b(\theta_i) \cdot x_i^b \right\}_i \end{matrix} \right\|_q \le u_v \qquad \forall v \in \mathcal{V}, \\
& x_i^b \ge 0 \quad \forall b \in \mathcal{B}_i, \forall i \in \mathcal{I},
\end{aligned} \tag{41}$$

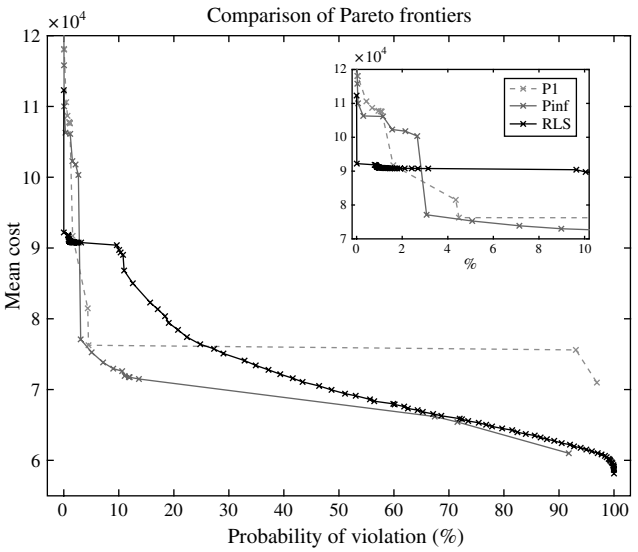

Figure 11 **Pareto Frontiers for the Trade-Off Between Mean Cost and Probability of Constraint Violation Using the Robust Local Search (RLS)**

Notes. For comparison to convex robust optimization techniques, the norm in Equation (40) is set to a 1-norm (P1) or an ∞-norm (Pinf). In the small figure, which is a magnification of the larger figure for a probability of violation less than or equal to 10%, we observe that for probability of violation less than 2%, RLS leads to lower mean costs and lower probability of violation, whereas for probability of violation above 2%, Pinf is the best solution, as shown in the larger figure.

where $1/p + 1/q = 1$. For $p = 1$ and $p = \infty$, problem (41) is an LP, which takes less than five seconds to solve. Note that θ is a constant in this formulation. Now, by replacing the subproblem (39) with problem (41) every time, Algorithm 5 can be applied to find a robust local minimum. For a given Γ, Algorithm 5 takes around one to two hours on an Intel Xeon 3.4 GHz to terminate.

Computation Results. We found a large number of robust designs using Algorithm 5 with different Γ and starting from nominal best and strict interior. Figure 11 shows the results for both $p = 1$ (P1) and $p = \infty$ (Pinf). It also illustrates the Pareto frontiers of all designs that were found under the robust local search (RLS), P1, and Pinf. When the required probability of violation is high, the convex techniques, in particular P1, find better robust designs. For lower probability, however, the designs found by the robust local search are better.

Compared to the robust local search, the convex approaches have inherent advantages in optimal robust designs $\mathbf{x}^*(\theta)$ for every θ. This explains why the convex approaches find better designs for a larger probability of violation. However, the robust local search is suited for far more general problems because it does not rely on convexities in subproblems. Nevertheless, its performance is comparable to the convex

approaches, especially when the required probability of violation is low.

7. Conclusions

We have generalized the robust local search technique to handle problems with constraints. The method consists of a neighborhood search and a robust local move in every iteration. If a new constraint is added to the problem with n-dimensional uncertainties, $n+1$ additional gradient ascents are required in the neighborhood search step; i.e., the basic structure of the algorithm does not change.

The robust local move is also modified to avoid infeasible neighbors. We apply the algorithm to an example with a nonconvex objective and nonconvex constraints. The method finds two robust local minima from different starting points. In both instances, the worst-case cost is reduced by more than 70%.

When a constraint results in a convex constraint maximization problem, we show that the gradient ascents can be replaced with more efficient procedures. This gain in efficiency is demonstrated on a problem with linear constraints. In this example, the standard robust local search takes 3,600 seconds to converge at the robust local minimum. The same minimum, however, was obtained in 96 seconds when the gradient ascents were replaced by the function evaluation.

The constrained version of the robust local search requires only a subroutine that provides the constraint value as well as the gradient. Because of this generic assumption, the technique is applicable to many real-world applications, including nonconvex and simulation-based problems. The generality of the technique is demonstrated on an actual healthcare problem in intensity-modulated radiation therapy for cancer treatment. This application has 85 decision variables and more than a thousand constraints. The original treatment plan, found using optimization without consideration for uncertainties, proves to always violate the constraints when uncertainties are introduced. Such constraint violations correspond to either an insufficient radiation in the cancer cells or an unacceptably high radiation dosage in the normal cells. Using the robust local search, we find a large number of robust designs using uncertainty sets of different sizes. By considering the Pareto frontier of these designs, a treatment planner can find the ideal trade-off between the amount of radiation introduced and the probability of violating the dosage requirements.

Acknowledgments

The authors thank T. Bortfeld, V. Cacchiani, and D. Craft for fruitful discussions related to the IMRT application in §6. This work is supported by DARPA-N666001-05-1-6030.

References

Ben-Tal, A., A. Nemirovski. 1998. Robust convex optimization. *Math. Oper. Res.* **23**(4) 769–805.

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

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

Bertsimas, D., M. Sim. 2006. Tractable approximations to robust conic optimization problems. *Math. Programming* **107**(1–2) 5–36.

Bertsimas, D., O. Nohadani, K. M. Teo. 2007. Robust optimization in electromagnetic scattering problems. *J. Appl. Phys.* **101**(7) 074507.

Bertsimas, D., O. Nohadani, K. M. Teo. 2009. Robust optimization for unconstrained simulation-based problems. *Oper. Res.* Forthcoming. http://www.optimization-online.org/DB_HTML/2007/08/1756.html.

Bortfeld, T. 2006. Review IMRT: A review and preview. *Phys. Medicine Biol.* **51**(13) R363–R379.

Chan, T. C. Y., T. Bortfeld, J. N. Tsitsikilis. 2006. A robust approach to IMRT. *Phys. Medicine Biol.* **51**(10) 2567–2583.

Deasy, J. O., A. I. Blanco, V. H. Clark. 2003. CERR: A computational environment for radiotherapy research. *Medical Phys.* **30**(5) 979–985.

Djajaputra, D., Q. Wu, Y. Wu, R. Mohan. 2003. Algorithm and performance of a clinical IMRT beam-angle optimization system. *Phys. Medicine Biol.* **48**(19) 3191–3212.

Henrion, D., J. B. Lasserre. 2003. Gloptipoly: Global optimization over polynomials with MATLAB and SeDuMi. *ACM Trans. Math. Software* **29** 165–194.

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

Lasserre, J. B. 2006. Robust global optimization with polynomials. *Math. Programming* **107**(1–2) 275–293.

Stinstra, E., D. den Hertog. 2007. Robust optimization using computer experiments. *Eur. J. Oper. Res.* **191**(3) 816–837.