Abstract. We treat uncertain linear programming problems by utilizing the notion of weighted analytic centers and notions from the area of multi-criteria decision making. After introducing our approach, we develop interactive cutting-plane algorithms for robust optimization, based on concave and quasi-concave utility functions. In addition to practical advantages, due to the flexibility of our approach, we are able to prove that under a theoretical framework due to Bertsimas and Sim [12], which establishes the existence of certain convex formulation of robust optimization problems, the robust optimal solutions generated by our algorithms are at least as desirable to the decision maker as any solution generated by many other robust optimization algorithms in the theoretical framework. We present some probabilistic bounds for feasibility of robust solutions and evaluate our approach by means of computational experiments.

1. Introduction

Optimization problems are widespread in real life decision making situations. However, data perturbations as well as uncertainty in at least part of the data are very difficult to avoid in practice. Therefore, in most cases we have to deal with the reality that some aspects of the data of the optimization problem are uncertain. This uncertainty is caused by many sources such as forecasting, or approximations in models, or data approximation, or noise in measurements. In order to handle optimization problems under uncertainty, several techniques have been proposed. The most common, widely-known approaches are

- **Sensitivity analysis**: typically, the influence of data uncertainty is initially ignored, and then the obtained solution is justified based on the data perturbations [14].
- **Chance constrained programming**: we use some stochastic models of uncertain data to replace the deterministic constraints by their probabilistic counterparts [35, 40, 19]. It is a natural way of converting the uncertain optimization problem into a deterministic one. However, most of the time the result is a computationally intractable problem [4].

\[\text{Date: October 6, 2014.}\]
Mehdi Karimi: (m7karimi@uwaterloo.ca) Department of Combinatorics and Optimization, Graduate Student, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Research of this author was supported in part by a Discovery Grant from NSERC and by ONR Research Grant N00014-12-10049.
Somayeh Moazeni: (somayeh@princeton.edu) Postdoctoral Research Associate, Department of Operations Research and Financial Engineering, Princeton University, Sherrerd Hall, Charlton Street, Princeton, New Jersey, 08544, U.S.A. Research of this author was supported in part by Discovery Grants from NSERC.
Levent Tunçel: (ltuncel@math.uwaterloo.ca) Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Research of this author was supported in part by Discovery Grants from NSERC and by ONR Research Grant N00014-12-10049.
• **Stochastic programming:** the goal is to find a solution that is feasible for all (or almost all) possible instances of the data and to optimize the expectation of some function of the decisions and the random variables [46].

• **Robust optimization:** robust optimization is the method that is most closely related to our approach. Generally speaking, robust optimization can be applied to any optimization problem where the uncertain data can be separated from the problem’s structure. This method is applicable to convex optimization problems including semidefinite programming [4]. Our focus in this paper is on uncertain linear programming problems. Uncertainty in the data means that the exact values of the data are not known, at the time when the solution has to be determined. In robust optimization framework, uncertainty in the data is described through *uncertainty sets*, which contain all (or most of) possible values that may be realized for the uncertain parameters. Generally speaking, the distinction between robust optimization and stochastic programming is that robust optimization does not require the specification of the exact distribution. Stochastic programming performs well when the distributions of the uncertainties are exactly known, and robust optimization is efficient when there is little information about those distributions. Recently, a new method has been proposed, called *distributionally robust* optimization, that tries to cover the gap between robust optimization and stochastic programming [24]. In this approach, one seeks a solution that is feasible for the worst-case probability distribution in a set of possible distributions.

Since the interest in robust formulations was revived in the 1990s, many researchers have introduced new formulations for robust optimization framework in linear programming and general convex programming [48, 7, 6, 8, 5, 12, 13, 11, 9, 39]. Ben-Tal and Nemirovski [7, 6] provided some of the first formulations for robust LP with detailed mathematical analysis. Bertsimas and Sim [12] proposed an approach that offers control on the degree of conservatism for every constraint as well as the objective function. Bertsimas et al. [9] characterize the robust counterpart of an LP problem with uncertainty set described by an arbitrary norm. By choosing appropriate norms, they recover the formulations proposed in the above papers [7, 6, 9].

The goal of classical robust optimization is to find a solution that is capable to cope best of all with *all* realizations of the data from a given (usually bounded) uncertainty set [4, 3]. By the classical definition of robustness [4, 8, 11, 21], a *robust optimal solution* is the solution of the following problem:

\[
\max \left\{ \inf_{\tilde{\mathbf{c}} \in \mathcal{C}} \langle \tilde{\mathbf{c}}, x \rangle : \tilde{\mathbf{A}}x \leq \tilde{\mathbf{b}}, \forall \tilde{\mathbf{b}} \in \mathcal{B}, \forall \tilde{\mathbf{A}} \in \mathcal{A} \right\},
\]

where \(\mathcal{C}, \mathcal{A},\) and \(\mathcal{B}\) are given uncertainty sets for \(\tilde{\mathbf{c}}, \tilde{\mathbf{A}},\) and \(\tilde{\mathbf{b}},\) respectively. Throughout this paper, we refer to the formulation of (1) as *classical robust formulation*.

1.1. **Some drawbacks of robust optimization.** Classical robust optimization is a powerful method to deal with optimization problems with uncertain data, however, we can raise some valid criticisms. One of the assumptions for robust optimization is that the uncertainty set must be precisely specified before solving the problem. Even if the uncertainty is only in the RHS, expecting the Decision Maker (DM) to construct accurately an ellipsoid or even a hypercube for uncertainty set may not always be reasonable. Another main criticism to classical robust optimization is that satisfying all of the constraints, if not make the problem infeasible, may lead to an objective value very far from the optimal value of the nominal problem. This problem is more critical for large deviations. As an example, [6, 36]
considered some of the problems in the NETLIB library (under reasonable assumptions on uncertainty of certain entries) and showed that classical robust counterparts of most of the problems in NETLIB become infeasible for a small perturbation. Moreover, in many other problems, objective value of the classical robust optimal solution is very low and may be unsatisfactory for the decision maker.

Several modifications of classical robust optimization have been introduced to deal with this issue. One, for example, is globalized robust counterparts introduced in Section 3 of [4]. The idea is to consider some constraints as “soft” whose violation can be tolerated to some degree. In this method, we take care of what happens when data leaves the nominal uncertainty set. In other words, we have “controlled deterioration” of the constraint. These modified approaches have more flexibility than the classical robust methodology, but we have the problem that the modified robust counterpart of uncertain problems may become computationally intractable. Although the modified robust optimization framework rectifies this drawback to some extent, it intensifies the first criticism by putting more pressure on the DM to specify deterministic uncertainty sets before solving the problem.

Another criticism of the classical robust optimization is that it gives the same “weight” to all the constraints. In practice, this is not the case as some constraints are more important for the DM. There are some options in classical robust optimization like changing the uncertainty set which again intensifies the first criticism. We will see that our approach can alleviate this difficulty.

1.2. Contributions and overview of this paper. We present a framework which allows a fine-tuning of the classical tradeoff between robustness and conservativeness and engages DM continuously and in a more effective way throughout the optimization process. Under a suitable theoretical modelling setup, we prove that the classical robust optimization approach is a special case of our framework. We demonstrate that it is possible to efficiently perform optimization under this framework and finally, we illustrate the use of our methods numerically.

One of the main contributions of this paper is the development of cutting-plane algorithms for robust optimization using the notion of weighted analytic centers in a small dimensional weight-space. We also design algorithms in the slack variable space as a theoretical stepping stone towards the more applicable weight-space cutting-plane algorithms. Ultimately, we are proposing that our approach be used in practice with a small number (say somewhere in the order of 1 to 20) of driving factors that really matter to the DM. These driving factors are independent of the number of variables and constraints, and determine the dimension of the weight space (for interaction with the DM). Working in a low dimensional weight-space not only simplifies the interaction for the DM, but also makes the cutting-plane algorithm more efficient.

The notion of moving across a weight space has been widely used in the area of multi-criteria decision making: when we have several competing objective values to optimize, a natural approach is to optimize a weighted sum of them [30], [27]. Authors in [30] presented an algorithm for evaluating and ranking items with multiple attributes. [30] is related to our work as the proposed algorithm is a cutting-plane one. However, our algorithm uses the concept of weighted analytic center which is completely different. Authors in [27] proposed a family of models (denoted my McRow) for multi-expert multi-criteria decision making. Their work is close to ours as they derived compact formulations of the McRow model by assuming some structure for the weight region, such as polyhedral or conic
descriptions. Our work has fundamental differences with [27]: cutting-plane algorithms in the weight-space find a weight vector \( w \) in a fixed weight region (the unit simplex) such that the weighted analytic center of \( w \), say \( x(w) \), is the desired solution for the DM. The algorithms we design in this paper make it possible to implement the ideas we mentioned above to help overcome some of the difficulties for robust optimization to reach a broader, practicing user base. For some further details and related discussion, also see Moazeni [36] and Karimi [31].

In Section 2, we explain our approach, and also introduce the notion of weighted analytic centers. In Section 3, we design the cutting-plane algorithms, and explain some practical uses of our approach. In Section 4, we prove that, under a theoretical framework due to Bertsimas and Sim [12], our approach is as least as strong as the classical robust optimization approach. Some preliminary computational results are presented in Section 5. In Section 6, we briefly talk about the extension of the approach to semidefinite programming and quasi-concave utility functions, and then conclude the paper.

1.3. Notations and assumptions. Before introducing our approach in the next section, let us first explain some of the assumptions and notations we are going to use. Much of the prior work on robust linear programming addresses the uncertainty through the coefficient matrix. Bertsimas and Sim [13] considered linear programming problems in which all data except the right-hand-side (RHS) vector is uncertain. In [8, 7, 11], it is assumed that the uncertainty affects the coefficient matrix and the RHS vector. Some papers deal with uncertainty only in the coefficient matrix [6, 12, 9]. Optimization problems in which all of the data in the objective function, RHS vector and the coefficient matrix are subject to uncertainty, have been considered in [5]. As we explain in Section 2, the nominal data and a rough approximation of the uncertainty set are enough for our approach. However, the structure of uncertainty region is useful for the probability analysis. In this paper, we deal with the general setup that any part of the data \((A,b,c)\) may be subject to uncertainty; however, we handle the uncertainty in \(A\) by pushing it onto uncertainty in \(b\). Moreover, in at least some applications, the amount of uncertainty in \(A\) is limited whereas the uncertainty in the RHS and the objective function vectors may be very significant. Some of the supporting arguments for this viewpoint are:

(1) Instead of specifying uncertainty for each local variable, we can handle the whole uncertainty with some global variables. These global variables can be, for example, the whole budget, human resources, availability of certain critical raw materials, government quotas, etc. It may be easier for the DM to specify the uncertainty set for these global variables. Then, we can approximate the uncertainty in the coefficient matrix with the uncertainty in the RHS and the objective function. In other words, we may fix the coefficient matrix on one of the samples from the uncertainty set and then handle the uncertainty by introducing uncertainty to the RHS vector as in [10].

(2) A certain coefficient matrix is typical for many real world problems. In many applications of planning and network design problems such as scheduling, manufacturing, electric utilities, telecommunications, inventory management and transportation, uncertainty might only affect costs (coefficients of the objective function) and demands (the RHS vector)[38, 44]. Transportation systems: in some problems, the nodes and the arcs are fixed. However, the cost associated to each arc is not known precisely. Traffic assignment problems: we can assume that the drivers have perfect information about the arcs and nodes. However, their route choice behavior makes the travelling time uncertain. Distribution systems: in some applications,
the locations of warehouses and their capacities (in inventory planning and distribution problems) are well-known and fixed for the DM. However, the size of orders and the demand rate of an item could translate to an uncertain RHS vector. Holding costs, set up costs and shortage costs, which affect the optimal inventory cost, are also typically uncertain. These affect at least the objective function. **Medical/health applications:** in these applications (see for instance, [18, 15, 47, 17]) the DM may be a group of people (including medical doctors and a patient who are more comfortable with a few, say 4-20, driving factors which may be more easily handled by the mathematical model, if these factors could be represented as uncertain RHS values.

In the aforementioned applications, well-understood existing resources, reliable structures (well-established street and road networks, warehouses, and machines which are not going to change), and logical components of the formulation are translated into a certain coefficient matrix. The data in the objective function and the RHS vector are usually estimated by statistical techniques by the DM, or affected by uncertain elements such as institutional, social, or economical market conditions. Therefore, determining these coefficients with precision is often difficult or practically impossible. Hence, considering uncertainty in the objective function and the RHS vector seems to be very applicable, and motivates us to consider such formulation in LP problems separately.

(3) In our approach, we need the uncertainty sets for probabilistic analysis. Uncertainty in the RHS and the objective value is easier to handle mathematically.

By the above explanation, for the rest of this paper, we assume that the uncertainty in $A$ has already been pushed into $b$; thus we may fix the coefficient matrix $A$. It is clear that changing each entry of $A$ could change the geometry of the feasible region. On the other hand, neither we nor the DM know how each coefficient may affect the optimal solution before starting to solve the problem. Therefore, to fix matrix $A$, we rely on the nominal values (expected values) of the coefficients estimated by a method agreed by the DM. An uncertain linear programming problem with deterministic coefficient matrix $A \in \mathbb{R}^{m \times n}$, is of the form:

\[
\begin{align*}
\text{max} & \quad \langle \tilde{c}, x \rangle \\
\text{s.t.} & \quad Ax \leq \tilde{b}, \\
& \quad x \in \mathbb{R}^n,
\end{align*}
\]

where $\tilde{c} \in \mathcal{C}$ and $\tilde{b} \in \mathcal{B}$ are an $n$-vector and an $m$-vector respectively, whose entries are subject to uncertainty. $\mathcal{C}$ and $\mathcal{B}$ are called uncertainty sets. In this paper, we deal with problem (2) and suppose that the data uncertainty affects only the elements of the vectors $\tilde{b}$ and $\tilde{c}$. In our probabilistic analysis, we assume entries of $\tilde{c}$ and $\tilde{b}$ are random variables with unknown distributions, as it is impractical to assume that the exact distribution is explicitly known. By classical view of robust optimization, classical robust counterpart of problem (2) is defined in (1) with a certain $A$. Feasible/Optimal solutions of problem (1) are called classical robust feasible/classical robust optimal solutions of problem (2) [4].
Let $c^{(0)}$ and $b^{(0)}$ be the expected value of $c$ and $b$, respectively. The following LP program is the framework of our algorithms:

$$\begin{align*}
\max & \quad \langle c^{(0)}, x \rangle \\
\text{s.t.} & \quad Ax \leq b^{(0)}, \\
& \quad x \in \mathbb{R}^n,
\end{align*}$$

Here, without loss of generality, we impose the following restrictions on the problem (2) (for details, see [36]): The matrix $A$ has full column rank, i.e., $\text{rank}(A) = n \leq m$. The set $\{x \in \mathbb{R}^n : Ax \leq b^{(0)}\}$ is bounded and has nonempty interior.

In this paper, vectors and matrices are denoted, respectively, by lower and uppercase letters. The matrices $Y$ and $S$ represent diagonal matrices, having the components of vectors $y$ and $s$ on their main diagonals, respectively. The letter $e$ and $e_i$ denote a vector of ones and a vector that is everywhere zero except at the $i$-th entry with the appropriate dimension, respectively. The rows of a matrix are shown by superscripts of the row, i.e., $a^{(i)}$ is the $i$-th row of the matrix $A$. The inner product of two vectors $a, b \in \mathbb{R}^n$ is shown both by $\langle a, b \rangle$ and $a^\top b$. For a matrix $A$, we show the range of $A$ with $\mathcal{R}(A)$ and the null space of $A$ with $\mathcal{N}(A)$.

2. A utility function approach and weighted analytic centers

2.1. A utility function approach. In Section 1, we introduced different methods for dealing with LP problems under uncertainty. For each method, we explained the drawbacks and practical difficulties. In this subsection, we introduce our new approach that helps us overcome some of these difficulties. Let us focus on the robust optimization method that from many points of view is the strongest among the methods we introduced in Section 1. One of the main problems with robust optimization is that the uncertainty region must be specified before solving the problem. As we explained, in practice, even if the uncertainty is only in the RHS, expecting the DM to construct accurately an ellipsoid or a hypercube for uncertainty set may not be reasonable. The proposed method removes DM’s anxiety about determining the uncertainty set precisely, and a nominal value of the data is enough.

Consider problem (3); for any feasible point $x$, we define the slack vector $\bar{s} = b^{(0)} - Ax$. In this paper, we prefer to work only with the slack variables. Hence, we add a constraint that represents the objective function. This constraint is $\langle c^{(0)}, x \rangle \geq v$, where $v$ is a lower bound specified by the information from the DM. For example, if the DM decides that the objective value must not be below a certain value, we can put $v$ equal to that value. By the above definitions, LP problem (3) is equivalent to the following problem:

$$\begin{align*}
\max & \quad s_0 \\
\text{s.t.} & \quad Ax + \bar{s} = b^{(0)}, \\
& \quad \langle -c^{(0)}, x \rangle + s_0 = -v, \\
& \quad s := (\bar{s}, s_0)^\top \geq 0.
\end{align*}$$

Let us define $B_s \subseteq \mathbb{R}^{m+1}$ as the set of all feasible slack vectors. Then we can write (4) as

$$\begin{align*}
\max & \quad U(s) \\
\text{s.t.} & \quad s \in B_s,
\end{align*}$$
where \( U(s) := s_0 \). This \( U(s) \), which we denote as utility function, is the simplest one that takes into account only maximizing the objective function. Intuitively, we can cover a huge class of problems by using more complicated utility functions in (5). In this paper, we try to solve (5) for a general utility function \( U : \mathbb{R}^{m+1} \to \mathbb{R} \) that models all the preferences of the DM. We do not have access to this utility function, however assume that, for a slack vector \( s \), we can ask the DM questions to extract some information about the function. In many applications, robustness of a solution may be a monotone function of the slack variables (this typically corresponds to quasi-concave utility function in our theoretical development); however, this kind of property of the utility function is not as restrictive in our approach as it may seem since we can also have quasi-concave utility functions applied to any linear function of \( x \) or \( s \), using our approach (see subsection 3.4). We can also use modelling techniques from goal programming (see [29]). Assuming that \( U(s) \) is concave or quasi-concave, we retrieve the supergradient of \( U(s) \) at some points through a sequence of simple questions such as pairwise comparison questions (see for instance [33, 32, 34]).

In classical robust optimization approach, DM’s preferences, expertise etc. are formulated/captured by the uncertainty regions (this interaction takes place once). In our approach, we start with the axiom that the DM’s preferences, expertise etc. can be approximated by a multivariate utility function (however, we do not need to have this function, we do not need to reverse engineer it, we only assume its existence, which is quite typical in decision making situations and in interaction with the DM). Instead of requiring the DM to provide the regions of uncertainty at once, we interactively ask DM trade-off questions to extract enough information to construct approximate supergradients to the unknown utility function. This interactive approach has the additional benefit that in case the DM is inconsistent in his/her answers, since our approach is interactive and operates with very local information, we can provide the DM with a better chance of correcting mistakes as well as learning throughout the interactive process, what is possible within the given constraints and preferences.

We can make a connection between the feasible slack vectors of an LP and the notion of weighted-analytic-centers. There are strong justifications for using weight space (\( w \)-space) instead of \( s \)-space that we will see when we design the algorithms. Besides them, by using the notion of weighted center, we benefit from the differentiability. Weight-space and weighted-analytic-centers approach embeds a “highly differentiable” structure into the algorithms. Such tools are extremely useful in both the theory and applications of optimization. In contrast, classical robust optimization and other competing techniques usually end up delivering a final solution where differentiability cannot be expected; this happens because their potential solutions located on the boundary of some of the structures defining the problem.

2.2. Definition of weighted center. For every \( i \in \{1, 2, \ldots, m\} \), let \( F_i \) be a closed convex subset of \( \mathbb{R}^n \) such that \( \mathcal{F} := \bigcap_{i=1}^{m} F_i \) is bounded and has nonempty interior.

Let \( F_i : \text{int}(F_i) \to \mathbb{R} \) be a self-concordant barrier for \( F_i, i \in \{1, 2, \ldots, m\} \) (For a definition of self-concordant barrier functions see [42]). For every \( w \in \mathbb{R}^m_{++} \), we define the \( w \)-center of \( \mathcal{F} \) as

\[
\arg \min \left\{ \sum_{i=1}^{m} w_i F_i(x) : x \in \mathcal{F} \right\}.
\]

Consider the special case when each \( F_i \) is a closed half-space in \( \mathbb{R}^n \). Then the following result is well-known.
Theorem 2.1. Suppose for every \( i \in \{1, 2, \ldots, m\} \), \( a^{(i)} \in \mathbb{R}^n \setminus \{0\} \) and \( b_i \in \mathbb{R} \) are given such that:

\[ \mathcal{F} := \{ x \in \mathbb{R}^n : \langle a^{(i)}, x \rangle \leq b_i, \forall i \in \{1, 2, \ldots, m\} \}, \]

is bounded and \( \text{int}(\mathcal{F}) \) is nonempty. Also, for every \( i \in \{1, 2, \ldots, m\} \) define

\[ F_i(x) := -\ln(b_i - \langle a^{(i)}, x \rangle). \]

Then for every \( w \in \mathbb{R}^m_+ \), there exists a unique \( w \)-center in the interior of \( \mathcal{F} \), \( x(w) \). Conversely, for every \( x \in \text{int}(\mathcal{F}) \), there exists some weight vector \( w(x) \in \mathbb{R}^m_+ \) such that \( x \) is the unique \( w(x) \)-center of \( \mathcal{F} \).

Define the following convex optimization problems:

\[
\begin{align*}
\text{min} & \quad \langle c, x \rangle - \sum_{i=1}^{m} w_i \ln(s_i) \\
\text{s.t.} & \quad Ax + s = b, \\
& \quad s \in \mathbb{R}^m_+, x \in \mathbb{R}^n,
\end{align*}
\]

and

\[
\begin{align*}
\text{min} & \quad \langle b, y \rangle - \sum_{i=1}^{m} w_i \ln(y_i) \\
\text{s.t.} & \quad A^\top y = c, \\
& \quad y \in \mathbb{R}^m_+,
\end{align*}
\]

where \( A \in \mathbb{R}^{m \times n} \), \( b \in \mathbb{R}^m \), and \( c \in \mathbb{R}^n \). For every weight vector \( w > 0 \), the objective functions of the above problems are strictly convex on their domains. Moreover, the objective function values tend to \(+\infty\) along any sequence of their interior points (strictly feasible points), converging to a point on their respective boundary. So, the above problems have minimizers in the interior of their respective feasible regions. Since the objective functions are strictly convex, the minimizers are unique. Therefore, for every given \( w > 0 \), the above problems have unique solutions \((x(w), s(w))\) and \( y(w) \). These solutions can be used to define many primal-dual weighted-central-paths as the solution set \( \{(x(w), y(w), s(w)) : w > 0\} \) of the following system of equations and strict inequalities:

\[
\begin{align*}
Ax + s &= b, \quad s > 0, \\
A^\top y &= c, \\
Sy &= w,
\end{align*}
\]

where \( S := \text{Diag}(s) \). When we set \( w := te, t > 0 \), we obtain the usual primal-dual weighted-central-path. Figure 1 illustrates some weighted central paths.

As we mentioned above, we embed the objective function into matrix \( A \) to work only with the slack variables. In view of (4), let us define

\[ A := \begin{bmatrix} A \\ -\omega^{(0)} \end{bmatrix}, \quad b := \begin{bmatrix} b^{(0)} \\ -v \end{bmatrix}. \]

From now on, we may assume that \( A \in \mathbb{R}^{m \times n} \) also contains the last added constraint. As we embedded the objective function in \( A \), we can put \( c := 0 \) in (8), and the solutions of the following system form
Figure 1. Primal-dual central paths.

For every given weight vector $w$, $(x(w), y(w), s(w))$ is obtained uniquely and $x(w)$ is called the *weighted center* of $w$. For every given $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, $y > 0$, that satisfy the above system, $w$ and $s(w)$ are obtained uniquely. However, for a given $x \in \mathbb{R}^n$, there are many weight vectors $w$ that give $x$ as the $w$-center of the corresponding polytope. Without loss of generality, we restrict ourselves to the weights on the unit simplex, i.e., we consider weighted center $(x, y, s)$ corresponding to weight vectors $w$ such that $\sum_{i=1}^m w_i = 1$. A special case can be $w = \frac{1}{m} e$, where $e$ is the vector of all ones. We will show that this subset of weight vectors is enough to represent the feasible region. We call this simplex of weight vectors $W$:

$$W := \{w \in \mathbb{R}^m : w > 0, \ e^\top w = 1\}.$$  

We define the following notion for future reference:

**Definition 2.1.** A vector $s \in \mathbb{R}^m$ or $y \in \mathbb{R}^m$ is called centric if there exists $x$ such that $(x, y, s)$ satisfies (9) for a weight vector $w > 0$ where $e^\top w = 1$.

Let $s$ and $y$ be centric. First, we note that the simplex of the weight vectors can be divided into regions of constant $y$-vector ($W_y$) and constant $s$-vector ($W_s$). By using Lemma A.2, if $(\hat{x}, \hat{y}, \hat{s})$ is the solution of system (9) corresponding to the weight vector $\hat{w} \in W$, and $\bar{y} > 0$ is any centric $y$-vector, then $(\hat{x}, \bar{y}, \hat{s})$ is the solution of system (9) corresponding to the weight vector $Y(\bar{Y})^{-1}\hat{w}$. This means that for every centric vector $\hat{s}$ and any centric vector $y$, $\hat{S}y$ is a weight vector in the simplex.
For every pair of centric vectors $s$ and $y$, $W_s$ and $W_y$ are convex. To see this, let $(x, \bar{y}, s)$ and $(x, y, s)$ be the weighted centers of $\hat{w}$ and $w$. Then, it is easy to see that for every $\beta \in [0, 1]$, $(x, \beta \bar{y} + (1 - \beta)y, s)$ is the weighted center of $\beta \hat{w} + (1 - \beta)w$. Different properties of $W_s$ and $W_y$ are studied in Appendix A, but the following simple examples make the geometry of $W_s$ and $W_y$ clearer. We present two examples with $m = 3$, $n = 1$.

**Example 2.1.** For the first example, let $b := [1 \ 0 \ 0]^\top$ and $A := [1 \ -1 \ -1]^\top$. By using (9), the set of centric $s$-vectors is $B_s = \{(1 - x), x, x\}^\top : x \in (0, 1)\}$. The set of centric $y$-vectors is specified by solving $A^\top y = 0$ and $b^\top y = 1$, while $y > 0$. We can see that in this example, as shown in Figure 2, $W_s$s are parallel line segments while $W_y$s are line segments which all intersect at $[1 \ 0 \ 0]^\top$. For the second example, let $A := [1 \ -1 \ 0]^\top$ and $b := [1 \ 0 \ 1]^\top$. The set of $W_s$s and $W_y$s are shown in Figure 3 derived by solving (9). As can be seen, this time $W_y$s are parallel line segments and $W_s$s are line segments which intersect at the point $[0 \ 0 \ 1]^\top$.

These examples show that the affine hulls of $W_{y1}$ and $W_{y2}$ might not intersect for two centric $y$-vectors $y^1$ and $y^2$. This is also true for the affine hulls of $W_{s1}$ and $W_{s2}$ for two centric $s$-vectors $s^1$ and $s^2$.

**Example 2.2.** For the second example, let $A := [3 \ -3 \ -2]^\top$ and $b := [1 \ 1 \ 0]^\top$. The set of $W_s$s and $W_y$s are shown in Figure 4, derived by solving (9). In this example, none of $W_y$s, $W_s$s, or their affine hulls intersect in a single point.

### 3. Algorithms

In this section, we develop some cutting-plane algorithms which find an optimal solution for the DM, using the facts we established in the previous sections. As we mentioned in Section 2, we assume
that the DM’s preferences, knowledge, wisdom, expertise, etc. is modeled by a utility function (as a function of the slack variables $s$), i.e., $U(s)$, and our problem is to maximize this utility function over the set of centric (Definition 2.1) $s$-vectors $B_s$. (Of course, we do not assume to have access to this

Figure 3. $W_s$s and $W_y$s for the second example in Example 2.1.

Figure 4. $W_s$s and $W_y$s for Example 2.2.
function $U$, except through our limited interactions with the DM.) Therefore, our problem becomes

$$\max \quad U(s)$$
$$s.t. \quad s \in B_s.$$  

(10)

In the following, we denote an optimal solution of (10) with $s^{opt}$. In many applications, it is possible to capture choices with concave, quasi-concave, or nondecreasing utility functions. We are going to start with the assumption of concave $U(s)$. We see in Subsection 6.2 that the algorithm can easily be refined to be used for quasi-concave functions. Here, we use the concept of supergradient we introduce shortly. Supergradients (subgradients for convex functions) have been widely used before to design cutting-plane and ellipsoid algorithms. Our goal is to use the concept to design cutting-plane algorithms.

Suppose we start the algorithm from a point $w^0 \in \mathbb{R}^m$ with the corresponding $s$-vector $s^0 \in \mathbb{R}^m$. Using the idea of supergradient, we can introduce cuts in the $s$-space or $w$-space to shrink the set of $s$-vectors or $w$-vectors, such that the shrunken space contains an optimal point. In the following subsections, we discuss these algorithms in $s$-space and $w$-space. Our main algorithm is the one in the $w$-space, however, the $s$-space algorithm helps us understand the other better.

As mentioned above, our algorithms are based on the notion of supergradient of a concave function. Therefore, before stating the algorithms, we express a summary of the results we want to use. These properties are typically proven for convex functions in the literature [16, 45], however we can translate all of them to concave functions. It is a well-known fact that for a concave function $f : \mathbb{R}^n \to \mathbb{R}$, any local maximizer is also a global maximizer. If a strictly concave function attains its global maximizer, it is unique. The following theorem is fundamental for developing our cutting-plane algorithms.

**Theorem 3.1.** Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is a concave function and let $x^0 \in \text{relint}(\text{dom} f)$. Then there exists $g \in \mathbb{R}^n$ such that

$$f(x) \leq f(x^0) + g^\top (x - x^0), \quad \forall x \in \mathbb{R}^n.$$  

(11)

If $f$ is differentiable at $x^0$, then $g$ is unique, and $g = \nabla f(x^0)$.

The vector $g$ that satisfies (11) is called the supergradient of $f$ at $x^0$. The set of all supergradients of $f$ at $x^0$ is called the superdifferential of $f$ at $x^0$, and is denoted $\partial f(x^0)$. By Theorem 3.1, if $f$ is differentiable at $x^0$, then $\partial f(x^0) = \{\nabla f(x^0)\}$. The following lemma about supergradient, which is a simple application of the chain rule, is also useful.

**Lemma 3.1.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a concave function, and $D \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ be arbitrary matrices. Then, $g(x) := f(Dx + b)$ is a concave function and we have:

$$\partial g(x) = D^\top \partial f(Dx + b).$$

3.1. **Cutting-plane algorithm in the $s$-space.** Assume that we have a starting point $s^0$ and we can obtain a supergradient of $U$ at $s^0$ from the DM, e.g. $g^0$, $(g^0 = \nabla U(s^0)$ if $U$ is differentiable at $s^0$). By using (11), for every $s$,

$$U(s) - U(s^0) \geq 0 \Rightarrow (g^0)^\top (s - s^0) \geq 0.$$  

(12)

This means that all optimal points are in the half-space $(g^0)^\top (s - s^0) \geq 0$. So, by adding this cut, we can shrink the $s$-space and guarantee that there exists an optimal solution in the shrunken part. We
can translate this cut to a cut in the $x$-space by using (9):

$$(g^0)^\top(s - s^0) = (g^0)^\top(b - Ax - b + Ax^0) = (g^0)^\top A(x^0 - x).$$

Using this equation, we consider the cut as a new constraint of the original problem:

$$(g^0)^\top Ax \leq (g^0)^\top Ax^0.$$ Let us define $a^{m+1} = (g^0)^\top A$ and $b_{m+1} = (g^0)^\top Ax^0$. We redefine $F$ by adding this new constraint and find the weighted center for a chosen weight vector $w^i$. The step-by-step algorithm is as follows:

**$S$-space Algorithm:**

- **Step 1:** Set $w^0 = \frac{1}{m} e$ and find the $w^0$-centers $(x^0, y^0, s^0)$ with respect to $F$.
- **Step 2:** Set $k = 0$, $A_0 = A$, $b^0 = b$, and $F_0 = F$.
- **Step 3:** If $s^k$ satisfies the DM, return $(x^k, y^k, s^k)$ and stop.
- **Step 4:** Set $k = k + 1$. Find $g_{k-1}$, the supergradient of $U(s)$ at $s^{k-1}$. Set

$$A_k = \begin{bmatrix} A_{k-1} \\ g^\top_{k-1} A_{k-1} \end{bmatrix}, \quad b^k = \begin{bmatrix} b^{k-1} \\ g^\top_{k-1} A_{k-1} x^{k-1} \end{bmatrix},$$

$$F_k := \left\{ x \in \mathbb{R}^n : \langle a^{(i)}_k, x \rangle \leq b^k_i, \forall i \in \{1, 2, \ldots, m + k\} \right\}. \tag{13}$$

- **Step 5:** Set $w^k = \frac{1}{m} e$ for $i \in \{m + 1, \ldots, m + k\}$ and $w^k = \frac{1}{m} - \frac{k}{m^2}$ for $i \in \{1, \ldots, m\}$. Find the $w^k$-center $(x^k, y^k, s^k)$ with respect to $F_k$. Return to Step 3.

The logic behind Step 5 is that we want to give smaller weights to the new constraints than the original ones (however, our choices above are just examples; implementers should make suitable, practical choices that are tailored to their specific application). A main problem with the algorithm is that the dimension of the weight-space is increased by one every time we add a constraint. We show that this problem is solved by our $w$-space algorithm in the following subsection.

### 3.2. Cutting-plane algorithm in the $w$-space.

In this subsection, we consider the cuts in the $w$-space. To do that, we first try a natural way of extending the algorithm in the $s$-space to the one in the $w$-space. We show that this extension only works for a limited subset of utility functions. Then, we develop an algorithm applicable to all concave utility functions.

Like the $s$-space, we try to use the supergradients of $U(s)$. Let $U_w$ denote the utility function as a function of $w$. From (9) we have $Ys = w$; so, $U_w(w) = U(s) = U(Y^{-1}w)$. If $Y$ were constant for all weight vectors, $U_w(w)$ would be a concave function, and we could use Lemma 3.1 to find the supergradient at each point. The problem here is that $Y$ is not necessarily the same for different weight vectors. Assume that we confine ourselves to weight vectors in the simplex $W$ with the same $y$-vector ($W_y$). $U_w(w)$ is a concave function on $W_y$, so, we can define its supergradient. By Lemma 3.1, we conclude that $\partial U_w(w) = Y^{-1}\partial U(s)$ for all $w \in W_y$.

Suppose we start at $w^0$ with the weighted center $(x^0, y^0, s^0)$. Let us define $g^{0w} := (Y^0)^{-1}g^0$, where $g_0$ is a supergradient of $U(s)$ at $s^0$. Then from (11) we have,

$$U_w(w) \leq U_w(w^0) + (g^{0w})^\top (w - w^0), \forall w \in W_{y_0}. \tag{14}$$
If we confine the weight-space to $W_y$, by the same procedure used for $s$-space, we can introduce cuts in the $w$-space using (14). The problem is that we do not have a proper characterization of $W_y$. On the other hand, $U_w$ may not be a concave function on the whole simplex. Assume that $s^{opt}$ is an optimal solution of (10), and $W_{s^{opt}}$ is the set of weight vectors in the simplex with $s$-vector $s^{opt}$. It is easy to see that $W_{s^{opt}}$ is convex. We also have the following lemma:

**Lemma 3.2.** Let $(x', y', s')$ be the weighted center corresponding to $w'$, $s^{opt}$ be an optimal solution of (10), and $g'$ be the supergradient of $U(s)$ at $s'$. Then, $S^{opt}y'$ is in the half-space $g'_w(w - w') \geq 0$, where $g'_w = Y^{t-1}g'$.

Proof. We have $g'_w(S^{opt}y' - w') = g'_w Y^{t-1}(S^{opt}y' - S'y') = g'_w(s^{opt} - s') \geq 0$. The last inequality follows from the fact that $s^{opt}$ is a maximizer and $g'$ is a supergradient of $U(s)$ at $s'$.

The above lemma shows that using hyperplanes of the form $g'_w Y^{t-1}(w - w')$, we can always keep a point from $W_{s^{opt}}$. Now, using the fact that $W_{s^{opt}}$ is convex and the above lemma, the question is: if we use a sequence of these hyperplanes, can we always keep a point from $W_{s^{opt}}$? A simpler question is: We start with $w^0$ and shrink the simplex $W$ into the intersection of the half-space $(g^{bw})^\top(w - w^0) \geq 0$ and the simplex, say $W_0$. Then we choose an arbitrary weight vector $w^1$ with weighted center $(x^1, y^1, s^1)$ from the shrunken space $W_0$. If $g^1$ is a supergradient of $U(s)$ at $s^1$, then we shrink $W_0$ into the intersection of $W_0$ and the half-space $(g^{bw})^\top(w - w^1) \geq 0$, where $g^{bw} = (Y^{t-1})^{-1}g^1$, and call the last shrunken space $W_1$. Is it always true that a weight vector with $s$-vector $s^{opt}$ exists in $W_1$? In the following, we show that this is true for some utility functions, but not true in general. We define a special set of functions that have good properties for cuts in the $w$-space, and the above algorithm works for them.

**Definition 3.1.** A function $f : \mathbb{R}^{m+}_{++} \to \mathbb{R}$ is called Non-Decreasing under Affine Scaling (NDAS) if for every $d \in \mathbb{R}^{m+}_+$ we have:

1. $f(s) \leq \max\{f(Ds), f(D^{-1}s)\}, \quad \forall s \in \mathbb{R}^{m+}_+.$
2. If for a single $s^0 \in \mathbb{R}^{m+}_+$ we have $f(s^0) \leq f(Ds^0)$, then $f(s) \leq f(Ds)$ for all $s \in \mathbb{R}^{m+}_+$.

For every $t \in \mathbb{R}^m$ the function $f_1(s) := \sum_{i=1}^m t_i \log s_i$ is NDAS. Indeed, for every $s, d \in \mathbb{R}^{m+}_+$ we have:

$$f_1(s) - f_1(Ds) = \sum_{i=1}^m t_i \log d_i,$$

$$f_1(s) - f_1(D^{-1}s) = -\sum_{i=1}^m t_i \log \frac{1}{d_i} = \sum_{i=1}^m t_i \log d_i,$$

and so we have $2f_1(s) = f_1(Ds) + f_1(D^{-1}s)$. The second property is also easy to verify and the function is NDAS. $f_1(s)$ is also important due to its relation to a family of classical utility functions in mathematical economics: Cobb-Douglas production function which is defined as $U_{cd}(s) = \prod_{i=1}^m s_i^{t_i}$, where $t \in \mathbb{R}^{m+}_+$. Usage of this function to simulate problems in economics goes back to 1920’s. Maximization of $U_{cd}(s)$ is equivalent to the maximization of its logarithm which is equal to $f_1(s) = \ln(U_{cd}(s)) = \sum_{i=1}^m t_i \log s_i$. Authors in [30] considered the Cobb-Douglas utility function to present an algorithm for evaluating and ranking items with multiple attributes. [30] is related to our
work as the proposed algorithm is a cutting-plane one. [30] also used the idea of weight-space as the utility function is the weighted sum of the attributes. However, our algorithm uses the concept of weighted analytic center which is different. Now, we have the following proposition.

**Proposition 3.1.** Assume that \( U(s) \) is a NDAS concave function. Let \( (x^0, y^0, s^0) \) and \( (x^1, y^1, s^1) \) be the weighted centers of \( w^0 \) and \( w^1 \), and \( g^0 \) and \( g^1 \) be the supergradients of \( U(s) \) at \( s^0 \) and \( s^1 \), respectively. Then we have

\[
\left\{ w : (g^0 w)^\top (w - w^0) \geq 0, \ (g^1 w)^\top (w - w^1) \geq 0 \right\} \cap W_{s^{\text{opt}}} \neq \emptyset,
\]

where \( g^0 w = (Y^0)^{-1} g^0 \) and \( g^1 w = (Y^1)^{-1} g^1 \).

**Proof.** See Appendix D. \qed

By Proposition 3.1, using the first two hyperplanes, the intersection of the shrunken space and \( W_{s^{\text{opt}}} \) is not empty. Now, we want to show that we can continue shrinking the space and have nonempty intersection with \( W_{s^{\text{opt}}} \).

**Proposition 3.2.** Assume that \( U(s) \) is a NDAS concave function. Let \( (x^i, y^i, s^i) \) be the weighted centers of \( w^i \), \( i \in \{0, \ldots, k\} \), and \( g^i \) be the supergradients of \( U(s) \) at \( s^i \). Let us define

\[
W^i := \left\{ w : (g^i w)^\top (w - w^i) \geq 0 \right\} \cap W,
\]

where \( g^i w = (Y^i)^{-1} g^i \). Assume we picked the points such that

\[
w^i \in \text{relint} \left( \bigcap_{j=0}^{i-1} W^j \right), \quad i \in \{1, \ldots, k\}.
\]

Then we have

\[
\left( \bigcap_{j=0}^{k} W^j \right) \cap W_{s^{\text{opt}}} \neq \emptyset,
\]

where \( s^{\text{opt}} \) is an optimal solution of \((10)\).

**Proof.** See Appendix D. \qed

Proposition 3.2 shows that the above-mentioned cutting-plane algorithm works for the NDAS functions. It would be very helpful in designing a cutting-plane algorithm in the \( w \)-space if this Proposition were true in general. However, this is not true for a general concave function. For a counter example, see Example D.1 in Appendix D. To be able to perform a cutting-plane algorithm in the \( w \)-space, we have to modify the definition of cutting hyperplanes. In the next two propositions, we introduce a new set of cutting-planes.

**Proposition 3.3.** For every point \( Y^0 s^0 \in W \), there exists a hyperplane \( P \) passing through it such that:

1- \( P \) contains all the points in \( W_{s^0} \), and
2- \( P \) cuts \( W_{s^0} \) the same way as \( (g^0)^\top (Y^0)^{-1} (w - Y^0 s^0) = 0 \) cuts it; the intersections of \( P \) and
\[(g^0)^\top(Y^0)^{-1}(w - Y^0 s^0) = 0\] with \(W_y^0\) is the same, and the projections of their normals onto \(W_y^0\) have the same direction.

**Proof.** See Appendix D. \qed

**Proposition 3.4.** Assume that we choose the points \(Y^0 s^0, Y^1 s^1 \in W\). The hyperplane \(P\) passing through \(Y^1 s^1\), with the normal vector \(u^1 := (S^1)^{-1}Ah^1\), \(h^1 = (A^\top Y^0(S^1)^{-1}A)^{-1}A^\top g^1\) satisfies the following properties:

1. \(P\) contains all the points in \(W_s^1\), and
2. \((u^1)^\top(Y^0 s_{\text{opt}} - Y^1 s^1) \geq 0\) for every feasible maximizer of \(U(s)\).

**Proof.** See Appendix D. \qed

By Proposition 3.4, we can create a sequence of points and hyperplanes such that the corresponding half-spaces contain \(Y^0 s_{\text{opt}}\). The algorithm is as follows:

**W-space Algorithm:**

- Step 1: Set \(w^0 = \frac{1}{m}e\) and find the \(w^0\)-centers \((x^0, y^0, s^0)\) with respect to \(\mathcal{F}\).
- Step 2: Set \(k = 0\), and \(W_0 = W\).
- Step 3: If \(s^k\) satisfies the optimality condition, return \((x^k, y^k, s^k)\) and stop.
- Step 4: Find \(g^k\), the supergradient of \(U(s)\) at \(s^k\). Find \(h^k\) by solving the following equation
  \[
  A^\top Y^0(S^k)^{-1}Ah^k = A^\top g^k.
  \]
- Step 5: Set \(u^k = (S^k)^{-1}Ah^k\) and \(W_{k+1} = W_k \cap \{w : (u^k)^\top(w - w^k) \geq 0\}\). Pick an arbitrary point \(w^{k+1}\) from \(W_{k+1}\) and find the \(w^{k+1}\)-center \((x^{k+1}, y^{k+1}, s^{k+1})\) with respect to \(\mathcal{F}\). Set \(k = k + 1\) and return to Step 3.

A clear advantage of this algorithm over the one in the \(s\)-space is that we do not have to increase the dimension of the \(w\)-space at each pass and subsequently we do not have to assign weights to the new added constraints. So, the above algorithm is straightforward to implement. We provided a discussion how to choose the next weight vector in the shrunken set in Appendix B. We can also use the properties of the weighted center we derived in Appendix A to improve the performance of the algorithm. We introduce two modified algorithms in Appendix B.

### 3.3. Convergence of the algorithm

Introduction of cutting-plane algorithms goes back at least to 1960’s and one of the first appealing ones is the center of gravity version [43]. The center of gravity algorithm has not been used in practice because computing the center of gravity, in general, is difficult. However, it is noteworthy due to its theoretical properties. For example, Grünbaum [25] proved that by using any cutting-plane through the center, more than 0.37 of the feasible set is cut out which guarantees a geometric convergence rate with a sizeable constant. Many different types of centers have been proposed in the literature. A group of algorithms use the center of a specific localization set, which is updated at each step. One of them is the ellipsoid method [51] where the localization set is represented by an ellipsoid containing an optimal solution. Ellipsoid method can be related to our algorithm as we can use it to find the new weight vectors at each iteration. The cutting-plane
method which is most relevant to our algorithm is the analytic center one, see [23] for a survey. In this method, the new point at each iteration is an approximate analytic center of the remaining polytope. The complexity of such algorithms has been widely studied in the literature. Nesterov [41] proved the $\epsilon$-accuracy bound of $O(L^2R^2/\epsilon^2)$ when the objective function is Lipschitz continuous with constant $L$, and the optimal set lies in a ball of diameter $R$. Goffin, Luo, and Ye [22] considered the feasibility version of the problem and derived an upper bound of $O(n^2\epsilon^2)$ calls to the cutting-plane oracle. Another family of cutting-plane algorithms are based on volumetric barriers or volumetric centers [49, 50, 1]. Vaidya used the volumetric center to design a new algorithm for minimizing a convex function over a convex set [49]. More sophisticated algorithms have been developed based on Vaidya’s volumetric cutting plane method [50, 1].

3.4. Some implementation ideas. In the previous subsections, we introduced an algorithm that is highly cooperative with the DM and proved many interesting features about it. In this subsection, we set forth some implementation ideas.

3.4.1. Driving factors. As we mentioned, one of our main criticisms of classical robust optimization is that it is not practical to ask the DM to specify an $m$-dimensional ellipsoid for the uncertainty set. Our approach improves this situation by asking easier questions. The idea is similar to those used in the area of multi-criteria optimization. Consider the system of inequalities $Ax \leq b$ and the corresponding slack vector $s = b - Ax$ representing the problem. The DM might prefer to directly consider only a few factors that really matter, we call them Driving Factors. For example, the driving factors for a DM might be budget amount, profit, allocated human resources, etc. We can represent $k$ driving factors by $(c^i)^\top x$, $i \in \{1, \ldots, k\}$, and the problem for the DM is to maximize the utility function $U((c^1)^\top x, \ldots, (c^k)^\top x)$. Similar to the way we added the objective of the linear program to the constraints, we can add $k$ constraints to problem and write (10) as:

$$
\begin{align*}
\max & \quad U(\xi_1, \ldots, \xi_k) \\
\text{s.t.} & \quad \xi_i = b_i - (c^i)^\top x \geq 0, \quad i \in \{1, \ldots, k\} \\
& \quad s = b - Ax, \quad s \geq 0.
\end{align*}
$$

(18)

As can be seen, the supergradient vector has only $k$ nonzero elements which makes it much easier for the DM to specify it for $k \ll m$. $k$ is usually small and we can figure out approximate gradients by asking pair-wise comparison questions among the driving factors. However, it still may have the problem that the cutting plane algorithm is in a high-dimensional space and it might be slow. We can take one step further to resolve this difficulty.

Assume that the slack vector for our driving factors $\xi$ is a linear function of the slack vector of the original problem as $\xi = Cs$ for a matrix $C$. Our goal is to solve problem

$$
\begin{align*}
\max & \quad U(\xi_1, \ldots, \xi_k) \\
\text{s.t.} & \quad \xi \in B_\xi.
\end{align*}
$$

(19)

instead of (18), where $B_\xi$ is the polytope of slack variable for the driving factors. Without loss of generality, we can assume that $U(\xi_1, \ldots, \xi_k)$ is a monotone non-decreasing function of $\xi_1, \ldots, \xi_k$ (it can be done by adjusting matrix $C$). (19) is a problem in 1–20 dimension and can be solve efficiently with cutting-plane algorithms. The DM deals only with problem (19), however, an optimizer/expert
needs to translate the cuts in $w_\xi$-space into cuts in $w$-space and/or coordinate the search between the $w_\xi$-space and $w$-space.

\[ u_i := U(s + \epsilon_i e_i) \approx U(s) + \frac{\partial U(s)}{\partial s_i} \epsilon_i \]

\[ \Rightarrow \frac{\partial U(s)}{\partial s_i} \approx \frac{u_i - u_0}{\epsilon_i}, \quad u_0 := U(s). \]

Assume that we have $m + 1$ points $s$ and $s + \epsilon_i e_i$, $i \in \{1, \ldots, m\}$. By the above equations, if we have the value of $U(s)$ at these points, we can find the approximate gradient. But in the absence of true utility function, we have to find these values through proper questions from the DM. Here, we assume that we can ask the DM about the relative preference for the value of the function at these $m + 1$ points. For example, DM can use a method called Analytic Hierarchy Process (AHP) to assess relative preference. We use these relative preferences to find the approximate gradient.

Assume that the DM provides us with the priority vector $p$, then we have the following relationship between $p$ and $u_i$’s

\[ \frac{u_i}{u_j} = \frac{p_i}{p_j}, \quad i, j \in \{0, \ldots, m\}, \]

\[ \Rightarrow \frac{u_i - u_0}{u_0} = \frac{p_i - p_0}{p_0}, \]

\[ \Rightarrow u_i - u_0 = \beta(p_i - p_0), \quad \beta := \frac{u_0}{p_0}. \]
Now, we can substitute the values of \( u_i - u_0 \) from (21) into (20) and we have

\[
\nabla U(s) = \beta_0 \left[ \frac{p_1 - p_0}{\epsilon_1} \ldots \frac{p_m - p_0}{\epsilon_m} \right]^T.
\]

The problem here is that we do not have the parameter \( \beta_0 \). However, this parameter is not important in our algorithm because we are looking for normals to our proper hyperplanes and, as it can be seen in Propositions 3.3 and 3.4, a scaled gradient vector can also be used to calculate \( h^0 \) and \( h^1 \). Therefore, we can simply ignore \( \beta_0 \) in our algorithm.

### 4. Improvised Robust Optimization via Utility Functions

In previous sections, we introduced our new methodology to deal with LP problems with uncertainty. We explained in Section 2 that our approach has many good features in terms of interaction with the decision maker and usability, and its practical advantages over the classical robust optimization approach are clear. In this section, we prove, utilizing the approach by Bertsimas and Sim \( [12] \), that the robust optimal solutions generated by our algorithms are at least as desirable to the decision maker as any solution generated by many other robust optimization algorithms. We show that by choosing a suitable utility function \( U(s) \) we can model many of the classical robust formulations. In other words, we can find a solution of a classical robust optimization problem by solving

\[
\begin{align*}
\text{max} & \quad g(x) := U(b - Ax) \\
\text{s.t.} & \quad a_i^T x \leq b_i, \ i \in \{1, \ldots, m\}.
\end{align*}
\]

Many classical robust optimization models and their approximations can be written as follows

\[
\begin{align*}
\text{max} & \quad c^T x \\
\text{s.t.} & \quad a_i^T x + f_i(x) \leq b_i, \ i \in \{1, \ldots, m\},
\end{align*}
\]

where \( f_i(x), i \in \{1, \ldots, m\} \), is a convex function such that \( f_i(x) \geq 0 \) for all feasible \( x \). If the convex uncertainty set \( U_i \) is known for each \( i \in \{1, \ldots, m\} \) and \( a_i \in U_i \), then we have \( f_i(x) := \sup_{\bar{a} \in U_i} \bar{a}^T x - a_i^T x \). By changing \( f_i(x) \), different formulations can be derived. In the following we bring some examples.

Assume that for each entry \( A_{ij} \) of matrix \( A \) we have \( A_{ij} \in [a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}] \). It can easily be seen \( [12] \) that the classical robust optimization problem is equivalent to (24) for \( f_i(x) = \hat{a}_{ij}^T |x| \). For the second example, assume that \( A \in \{ A : \|M(\text{vec}(A) - \text{vec}(\hat{A}))\| \leq \Delta \} \) for a given \( \hat{A} \) where \( \| \cdot \| \) is a general norm and \( M \) is an invertible matrix. \( \text{vec}(A) \) is a vector in \( \mathbb{R}^{mn \times 1} \) created by stacking the columns of \( A \) on top of one another. It is proved in \( [9] \) that many approximate robust optimization models can be formulated by changing the norm. It is also proved in \( [9] \) that this robust optimization model can be formulated as (24) by \( f_i(x) = \Delta \|M^{-T} x_i\|_s \), where \( \| \cdot \|_s \) is the dual norm and \( x_i \in \mathbb{R}^{mn \times 1} \) is a vector that contains \( x \) in entries \((i-1)n + 1\) through \( in \), and 0 everywhere else.

Now, utilizing Karush-Kuhn-Tucker (KKT) theorem, we prove that for every robust optimization problem that can be put into form (24), there exists a concave utility function \( U \) for which (23) has the same optimal solution as (24).

**Theorem 4.1.** Assume that (24) has Slater points. Then, there exists a concave function \( g(x) \) (or equivalently \( U(s) \)) such that optimization problems (23) and (24) have the same optimal solutions.
Proof. For the optimality condition of (24) we have: There exists $\lambda \in \mathbb{R}_+^m$ such that
\[ c - \sum_{i=1}^m \lambda_i (a_i + \nabla f_i(x)) = 0 \]  
(25)  
\[ \lambda_i (a_i^T x + f_i(x) - b_i) = 0, \quad i \in \{1, \ldots, m\}. \]
Since the Slater condition holds for (24), optimality conditions (25) are necessary and sufficient. Let $x^*$ be an optimal solution of (24), and let $J \subseteq \{1, \ldots, m\}$ denote the set of indices for which $\lambda_i \neq 0, i \in J$. Let us define $g(x)$ as follows:
\[ g(x) := c^T x + \sum_{i \in J} \mu_i \ln(b_i + t_i - a_i^T x - f_i(x)), \]  
(26)  
where $t_i > 0, i \in J$, are arbitrary numbers. We claim that $g(x)$ is concave. $b_i + t_i - a_i^T x - f_i(x)$ is a concave function and $\ln(x)$ is increasing concave, hence $\ln(b_i + t_i - a_i^T x - f_i(x))$ is a concave function for $i \in \{1, \ldots, m\}$. $g(x)$ is the summation of an affine function and some concave functions and so is concave. The gradient of $g(x)$ is
\[ \nabla g(x) = c - \sum_{i \in J} \frac{\mu_i}{b_i + t_i - a_i^T x - f_i(x)} (a_i + \nabla f_i(x)). \]  
(27)  
Now define $\mu_i, i \in J$, as
\[ \mu_i := \lambda_i \left[ b_i + t_i - a_i^T x^* - f_i(x^*) \right]. \]  
(28)  
Using (28) and comparing (27) and (25), we conclude that $x^*$ is a solution of (23), as we wanted. The other direction can be proved similarly. \[ \square \]

The above argument proves the existence of a suitable utility function. A remaining question is that can we construct such a utility function without having a solution of (25)? In the following, we construct a function with objective value arbitrarily close to the objective value of (24). Assume that strong duality holds for (24). Let us define $g(x) := c^T x + \mu \sum_{i=1}^m \ln(b_i - a_i^T x - f_i(x))$ and assume that $\hat{x}$ is the maximizer of $g(x)$. We have
\[ \nabla g(\hat{x}) = c - \sum_{i=1}^m \frac{\mu}{b_i - a_i^T \hat{x} - f_i(\hat{x})} (a_i + \nabla f_i(\hat{x})) = 0. \]  
(29)  
This means that $\hat{x}$ is the maximizer of the Lagrangian of the problem in (24), $L(\lambda, x)$, for $\hat{\lambda}_i := \mu/(b_i - a_i^T \hat{x} - f_i(\hat{x})), i \in \{1, \ldots, m\}$. So by strong duality, we have
\[ c^T x^* \leq L(\hat{\lambda}, \hat{x}) = c^T \hat{x} + \sum_{i=1}^m \frac{\mu}{b_i - a_i^T \hat{x} - f_i(\hat{x})} (b_i - a_i^T \hat{x} - f_i(\hat{x})) \]  
(30)  
(30) shows that by choosing $\mu$ small enough, we can construct $g(x)$ such that the optimal objective value of (23) is arbitrarily close to the optimal objective value of (24).

Note that many other approaches to robust optimization and decision making under uncertainty (including the generalized robust counterpart introduced by Ben-Tal and Nemirovski [4], and the approach of Iancu and Trichakis [28] using the notion of pareto robust optimization) can be included.
as a special case of our framework. A good starting point to prove the existence of a utility function is to start with indicator functions of sets encoding feasibility conditions. This approach first leads to utility functions that are not continuous; however, as we did above, these functions can be smoothed by use of barriers which then lead to differentiable utility functions with desired properties.

5. Illustrative Preliminary Computational Experiments

In this section, we present some numerical results to show the performance of the algorithms in the \( w \)-space designed in Section 3. LP problems we use are chosen from the NETLIB library of LPs. Most of these LP problems are not in the format we have used throughout the paper which is the standard inequality form. Hence, we convert each problem to the standard equality form and then use the dual problem. In this section, the problem \( \max \{ (c(0)^\top x : Ax \leq b(0)) \} \) is the converted one.

Example 1: In this example, we consider a simple problem of maximizing a quadratic function. Consider the ADLITTLE problem (in the converted form) with 139 constraints and 56 variables. We apply the algorithm to function \( U_{ij}(s) = -(s_i - s_j)^2 \) which makes two slack variables as close as possible. This function may not have any practical application, however, shows a simple example difficult to solve by classical robust optimization.

The stopping criteria is \( \| g \| \leq 10^{-6} \). For \( U_{23} \) the algorithm takes 36 iterations and returns \( U_{23} = -5 \times 10^{-11} \). For \( U_{34} \) the algorithm takes 35 iterations and returns \( U_{34} = -2.4 \times 10^{-12} \).

Example 2: Consider the ADLITTLE problem and assume that three constraints \{68, 71, 74\} are important for the DM. Assume that the DM estimates that there is 20 percent uncertainty in the RHS of these inequalities. We have \((b_{68}, b_{71}, b_{74}) = (500, 493, 506)\) and so the desired slack variables are around \((s_{68}, s_{71}, s_{74}) = (100, 98, 101)\). By using the classical robust optimization method that satisfies the worst case scenario, the optimal objective value is \( obj_c = 1.6894 \times 10^5 \).

Now assume that the following utility function represents DM’s preferences:

\[
U_1(s) = t_{68} \ln(s_{68}) + t_{71} \ln(s_{71}) + t_{74} \ln(s_{74}) + t_m \ln(s_m).
\]

This function is a NDAS function that we defined in Definition 3.1. Assume that the DM set \( t_m = 10 \) and \( t_{68} = t_{71} = t_{74} = 1 \). By using our algorithm, we get the objective value of \( obj_1 = 1.7137 \times 10^5 \) with the slack variables \((s_{68}, s_{71}, s_{74}) = (82, 83, 132)\). As we observe, the objective value is higher than the classical robust optimization method while two of the slack conditions are not satisfied. However, the slack variables are close to the desired ones. If the DM sets \( t_m = 20 \), we get the objective value of \( obj_2 = 1.9694 \times 10^5 \) with the slack variables \((s_{68}, s_{71}, s_{74}) = (40, 41, 79)\). However, all the iterates might be interesting for the DM. The following results are also returned by the algorithm before the optimal one:

\[
obj_3 = 1.8847 \times 10^5, \quad (s_{68}, s_{71}, s_{74}) = (56, 58, 83),
\]

\[
obj_4 = 1.7 \times 10^5, \quad (s_{68}, s_{71}, s_{74}) = (82, 84, 125).
\]
Now assume that the DM wants to put more weight on constraints 68 and 71 and so set $t_{68} = t_{71} = 2$, $t_{74} = 1$ and $t_m = 20$. In this case, the algorithm returns $obj_5 = 1.8026 \times 10^5$ with the slack variables $(s_{68}, s_{71}, s_{74}) = (82, 84, 64)$.

**Example 3:** In this example, we consider the DEGEN2 problem (in the converted form) with 757 constraints and 442 variables. The optimal solution of this LP is $obj_1 = -1.4352 \times 10^3$. Assume that constraints 245, 246, and 247 are important for the DM who wants them as large as possible, however, at the optimal solution we have $s(245) = s(246) = s(247) = 0$. The DM also wants the optimal objective value to be at least $-1.5 \times 10^3$. As we stated before, we add the objective function as a constraint to the system. To have the objective value at least $-1.5 \times 10^3$, we can add this constraint as $c^\top x = -1500 + s_{m+1}$. For the utility function, the DM can use the NDAS function

$$U(s) = \ln(s_{245}) + \ln(s_{246}) + \ln(s_{247}).$$

By running the algorithm for the above utility function, we get $(s_{245}, s_{246}, s_{247}) = (7.75, 17.31, 17.8)$ with objective value $obj_2 \approx -1500$ after 50 iterations and $(s_{245}, s_{246}, s_{247}) = (15.6, 27.58, 27.58)$ with $obj_3 \approx -1500$ after 100 iterations.

**Example 4:** We include a stopping criterion in the algorithm based on the norm of the super-gradient. The DM should also have some control over the stopping criteria (perhaps because of being satisfied or getting tired of the process). In this example, we consider the SCORPION problem (in the converted form) with 466 constraints and 358 variables. The optimal objective value of this LP is $obj_1 = 1.8781 \times 10^3$. We consider the following two NDAS utility functions:

$$U_1(s) = \ln(s_{m+1}) + \sum_{i=265}^{274} \ln(s_i),$$

$$U_2(s) = 5 \ln(s_{m+1}) + \sum_{i=265}^{274} \ln(s_i).$$

In this example, we apply the original and the 2nd modified algorithms to both $U_1(s)$ and $U_2(s)$. The improvement in the utility function value after 200 iterations is shown in Figure 6. As can be seen, the rate of increase in the utility function is decreasing after each iteration. For this problem, the algorithm does not stop by itself and continues until the satisfaction of the DM. The DM can stop the algorithm, for example, when the rate of increase is less than a specified threshold. For this example, the rate of improvement for the 2nd modified algorithm is almost as good as the original one. However, in the modified algorithm, the weighted center is computed around 40 times during the 200 iterations which is much less computational work.

**Example 5:** In this example, we consider utility functions introduced at the end of Section C. Consider problem SCORPION with optimal objective value of $obj_1 = 1.8781 \times 10^3$. Assume that the uncertainty in constraints 211 to 215 are important for the DM and we have $\|\Delta b_i\|_1 = 0.7b_i(0)$, $i \in \{211, \ldots, 215\}$, where $\Delta b_i$ was defined in (43). Let $\hat{x}$ be the solution of MATLAB’s LP solver, then we have $s_{211} = \cdots = s_{215} = 0$ which is not satisfactory for the DM. Besides, assume that the DM wants the objective value to be at least 1800. To satisfy that, we add the $(m + 1)$th constraint as $s_{m+1} = -1800 + (c(0))^\top x$ which guarantees $(c(0))^\top x \geq 1800$. For the utility function, first we define $u_i(s_i), i \in \{211, \ldots, 215\}$ similar to Figure 8 with $\epsilon_i^1 = \|\Delta b_i\|_1 = 0.7b_i(0)$ and $\epsilon_i^2 = \infty$. So we have for
AN IMPROVISED APPROACH TO ROBUSTNESS IN LINEAR OPTIMIZATION

$\mathbf{u}(s_i) = \begin{cases} s_i & s_i < \|\Delta b_i\|_1 \\ \|\Delta b_i\|_1 & s_i \geq \|\Delta b_i\|_1 \end{cases}$

(32)

Now, we can define $U(s) := \sum_{i=211}^{215} \ln u_i(s_i)$. By running the algorithm, the supergradient goes to zero after 65 iterations and the algorithm stops. Denote the solution by $x^*$, then the results are as follows:

$(c^{(0)})^\top x^* = 1800.3,$

$b^{(0)}_{211} = 3.86, \quad b^{(0)}_{212} = 48.26, \quad b^{(0)}_{211} = 21.81, \quad b^{(0)}_{211} = 48.26, \quad b^{(0)}_{211} = 3.86,$

$s^*_{211} = 3.29, \quad s^*_{212} = 19.47, \quad s^*_{211} = 7.39, \quad s^*_{211} = 16.97, \quad s^*_{211} = 3.24.$

(33)

Now, assume that the DM wants the objective value to be at least 1850 and the $(m+1)$th constraint becomes $s_{m+1} = -1850 + (c^{(0)})^\top x$. In this case, the norm of the supergradient reaches zero, after 104 iterations. The norm of supergradients versus the number of iterations are shown in Figure 7 for these two cases. Denote the solution after 100 iterations by $\bar{x}$, then we have:

$(c^{(0)})^\top \bar{x}^* = 1850,$

$s^*_{211} = 1.22, \quad s^*_{212} = 16.74, \quad s^*_{211} = 6.80, \quad s^*_{211} = 14.54, \quad s^*_{211} = 1.25.$

(34)

Let $\bar{x}$ be the returned value in the second case after 65 iterations. It is clearly not robust feasible; however, we can use bound (47) to find an upper bound on the probability of infeasibility. Assume that $N = 10$ and all the entries of $\Delta b_i$ are equal. Then, bound (47) reduces to $B(N, \delta_i N)$, where $\delta_i = \frac{s_i}{\|\Delta b_i\|_1}$. The probabilities of infeasibility of $\bar{x}$ for constraints 211 to 215 are given in Table 5 (using bound (47)).
Figure 7. Norm of the supergradient versus the number of iterations for Example 5.

Table 1. The probability of infeasibility of $\bar{x}$ for constraints 211 to 215.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Pr(\langle a_j, \bar{x} \rangle &gt; b_j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>0</td>
</tr>
<tr>
<td>212</td>
<td>0.0827</td>
</tr>
<tr>
<td>213</td>
<td>0.0018</td>
</tr>
<tr>
<td>214</td>
<td>0.0866</td>
</tr>
<tr>
<td>215</td>
<td>0</td>
</tr>
</tbody>
</table>

6. Extensions and conclusion

6.1. Extension to Semidefinite Optimization (SDP). Semidefinite Programming is a special case of Conic Programming where the cone is a direct product of semidefinite cones. Many convex optimization problems can be modeled by SDP. Since our method is based on a barrier function for a polytope in $\mathbb{R}^n$, it can be generalized and used as an approximation method for robust semidefinite programming that is NP-hard for ellipsoidal uncertainty sets. An SDP problem can be formulated as follows

$$\sup \langle \tilde{c}, x \rangle,$$

s.t. $\sum_{j=1}^{l_i} A_i^{(j)} x_j + S_i = \tilde{B}_i, \quad \forall i \in \{1, 2, ..., m\},$

$S_i \succeq 0, \quad \forall i \in \{1, 2, \ldots, m\},$
where $A_i^{(j)}$ and $\tilde{B}_i$ are symmetric matrices of appropriate size, and $\succeq$ is the Löwner order; for two square, symmetric matrices $C_1$ and $C_2$ with the same size, we have $C_1 \succeq C_2$ iff $C_1 - C_2$ is a semidefinite matrix. For every $i \in \{1, \ldots, m\}$, define

$$F_i := \{x \in \mathbb{R}^n : \sum_{j=1}^{t_i} A_i^{(j)} x_j \preceq \tilde{B}_i\}.$$  

Assume that $\text{int}(F_i) \neq \emptyset$ and let $F_i : \text{int}(F_i) \to \mathcal{R}$ be a self-concordant barrier for $F_i$. The typical self-concordant barrier for SDP is $F_i(x) = -\ln \left( \det \left( \tilde{B}_i - \sum_{j=1}^{t_i} A_i^{(j)} x_j \right) \right)$. Assume

$$F := \bigcap_{i=1}^{m} F_i$$

is bounded and its interior is nonempty. Now, as in the definition of the weighted center for LP, we can define a weighted center for SDP. For every $w \in \mathbb{R}_+^m$, we can define the weighted center as follows:

$$(35) \arg \min \left\{ \sum_{i=1}^{m} w_i F_i(x) : x \in F \right\}$$

The problem with this definition is that we do not have many of the interesting properties we proved for LP. The main one is that the weighted centers do not cover the relative interior of the whole feasible region and we cannot sweep the whole feasible region by moving in the $w$-space. There are other notions of weighted centers that address this problem; however, they are more difficult to work with algorithmically. Extending the results we derived for LP to SDP can be a good future research direction to follow.

6.2. Quasi-concave utility functions. The definition of the quasi-concave function is as follows:

**Definition 6.1.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is quasi-concave if its domain is convex, and for every $\alpha \in \mathbb{R}$, the set

$$\{x \in \text{dom} f : f(x) \geq \alpha\}$$

is also convex.

All concave functions are quasi-concave, however, the converse is not true. Quasi-concave functions are important in many fields such as game theory and economics. In microeconomics, many utility functions are modeled as quasi-concave functions. For differentiable functions, we have the following useful proposition:

**Proposition 6.1.** A differentiable function $f$ is quasi-concave if and only if the domain of $f$ is convex and for every $x$ and $y$ in $\text{dom} f$ we have:

$$(36) \quad f(y) \geq f(x) \Rightarrow (\nabla f(x))^\top (y - x) \geq 0$$

(36) is similar to (12), which is the property of the supergradient we used to design our algorithms. The whole point is that for a differentiable quasi-concave function $U(s)$ and any arbitrary point $s^0$, the maximizers of $U(s)$ are in the half-space

$$(\nabla U(s^0))^\top (s - s^0) \geq 0.$$  

This means that we can extend our algorithms to differentiable quasi-concave utility functions simply by replacing supergradient with gradient, and all the results for $s$-space and $w$-space stay valid.
6.3. Conclusion. In this paper, we presented new algorithms in a framework for robust optimization designed to mitigate some of the major drawbacks of robust optimization in practice. Our algorithms have the potential of increasing the applicability of robust optimization. Some of the advantages of our new algorithms are:

(1) Instead of a single, isolated, and very demanding interaction with the DM, our algorithms interact continuously with the DM throughout the optimization process with more reasonable demands from the DM in each iteration. One of the benefits of our approach is that the DM “learns” what is feasible to achieve throughout the process. Another benefit is that the DM is more likely to be satisfied (or at least be content) with the final solution. Moreover, being personally involved in the production of the final solution, the DM bears some responsibility for it and is more likely to adapt it in practice.

(2) Our algorithms operate in the weight-space using only driving factors with the DM. This helps reduce the dimension of the problem, simplify the demands on the DM while computing the most important aspect of the problem at hand.

(3) Weight-space and weighted-analytic-centers approach embeds a “highly differentiable” structure into the algorithms. Such tools are extremely useful in both the theory and applications of optimization. In contrast, classical robust optimization and other competing techniques usually end up delivering a final solution where differentiability cannot be expected.

Note that many elements of our approach can be partly utilized in other approaches to robust optimization and decision making situations under uncertainty. Moreover, our work creates natural connections between robust optimization and multi-attribute utility theory, elicitation methods used in multi-criteria decision making problems and goal programming theory (see [33, 37, 29]).

Developing similar algorithms for semidefinite programming is left as a future research topic. As we explained in Subsection 6.1, we can define a similar notion of weighted center for SDP. However, these weighted centers do not have some of the properties we used for LP, and we may have to switch to other notions of weighted centers that are more difficult to work with algorithmically, and have fewer desired properties compared to the LP setting.

References


Appendix A. Properties of \( w \)-space

In this appendix, we study the properties of weight space as well as \( W_s \) and \( W_y \) regions. Let us start from the following well-known lemma:

**Lemma A.1.** Let \((x, y, s)\) and \((\hat{x}, \hat{y}, \hat{s})\) be the solutions of system (9) corresponding to the weight vectors \( w, \hat{w} \in \mathbb{R}^m_{++} \), respectively. For every \( \bar{y} \) in the null space of \( A^\top \) we have:

\[
\langle \hat{s}, \bar{y} \rangle = \langle s, \bar{y} \rangle.
\]

**Proof.** From (9), we have \( s = b - Ax \) and \( \hat{s} = b - A\hat{x} \), which results in \( s - \hat{s} = A(x - \hat{x}) \). Hence we have \( s - \hat{s} \in \mathcal{R}(A) \). As the null space of \( A^\top \) and the range of \( A \) are orthogonal, for every \( \bar{y} \in \mathcal{N}(A^\top) \) we can write:

\[
\langle s - \hat{s}, \bar{y} \rangle = 0 \Rightarrow \langle \hat{s}, \bar{y} \rangle = \langle s, \bar{y} \rangle.
\]

Let \((\hat{x}, \hat{y}, \hat{s})\) be the solution of system (9) corresponding to the weight vector \( \hat{w} \). Moreover, assume that \( \hat{y} > 0 \) is such that \( A^\top \hat{y} = 0 \). Then, by using Lemma A.1, we can show that \((\hat{x}, \hat{y}, \hat{s})\) is the solution of system (9) corresponding to the weight vector \( \hat{Y} (\hat{Y})^{-1} \hat{w} \). Hence, there may be many weight vectors that give the same \( w \)-center. A stronger result is the following lemma which shows that in some cases, we can find the weighted center for a combination of weight vectors by using the combination of their weighted centers.

**Lemma A.2.** Let \((x^{(i)}, y^{(i)}, s^{(i)}), i \in \{1, \ldots, \ell\} \) be solutions of system (9), corresponding to the weights \( w^{(i)} \). Then, for every set of \( \beta_i \in [0, 1], i \in \{1, \ldots, \ell\} \), such that \( \sum_{i=1}^\ell \beta_i = 1 \), and for every
\[ j \in \{1, \ldots, \ell\}, \text{ we have } (\sum_{i=1}^{\ell} \beta_i x^{(i)}, y^{(j)}, \sum_{i=1}^{\ell} \beta_i s^{(i)}) \text{ is the } w\text{-center of } \mathcal{F}, \text{ where} \]
\[
w := \sum_{i=1}^{\ell} \beta_i Y^{(j)} (Y^{(i)})^{-1} w^{(i)}.\]

Moreover,
\[
\sum_{i=1}^{m} w_i = \sum_{i=1}^{m} w^{(j)}.
\]

**Proof.** According to the assumptions, for every \( i \in \{1, \ldots, \ell\}, \) we have
\[
A x^{(i)} + s^{(i)} = b^{(0)}, \quad s > 0,
\]
\[
A^T y^{(i)} = 0,
\]
\[
S^{(i)} y^{(i)} = w^{(i)}.
\]

Now, it can be seen that \( (\sum_{i=1}^{\ell} \beta_i x^{(i)}, y^{(j)}, \sum_{i=1}^{\ell} \beta_i s^{(i)}) \) satisfies the system:
\[
A(\sum_{i=1}^{\ell} \beta_i x^{(i)}) + (\sum_{i=1}^{\ell} \beta_i s^{(i)}) = b^{(0)}, \quad (\sum_{i=1}^{\ell} \beta_i s^{(i)}) > 0,
\]
\[
A^T y^{(j)} = 0,
\]
\[
(37) \quad (\sum_{i=1}^{\ell} \beta_i S^{(i)}) y^{(j)} = \sum_{i=1}^{\ell} \beta_i Y^{(j)} (Y^{(i)})^{-1} w^{(i)}.
\]

Since the \( w\)-center of \( \mathcal{F} \) is unique, the proof for the first part is done.

For the second part, from (37) we can write
\[
\sum_{i=1}^{m} w_i = \sum_{i=1}^{\ell} \sum_{p=1}^{\ell} \beta_p s_i^{(p)} y_i^{(j)} = \sum_{i=1}^{\ell} \beta_p \sum_{i=1}^{m} s_i^{(p)} y_i^{(j)} = \sum_{p=1}^{\ell} \beta_p \langle s^{(p)}, y^{(j)} \rangle.
\]

By Lemma A.1, we have \( \langle s^{(p)}, y^{(j)} \rangle = \langle s^{(i)}, y^{(j)} \rangle \). Therefore, we can continue the above series of equations as follows:
\[
\sum_{i=1}^{m} w_i = \sum_{p=1}^{\ell} \beta_p \langle s^{(j)}, y^{(j)} \rangle = \sum_{i=1}^{\ell} \beta_p \sum_{i=1}^{m} s_i^{(j)} y_i^{(j)} = \sum_{i=1}^{m} w_i^{(j)} \sum_{p=1}^{\ell} \beta_p = \sum_{i=1}^{m} w_i^{(j)}.
\]

\[\square\]

**A.1. Properties of \( w\)-space.** In this subsection, we study the structure of the \( w\)-space, which is important for the design of the algorithms in Section 3. Let \( s \) and \( y \) be centric. First, we note that the simplex of the weight vectors can be divided into regions of constant \( y\)-vector \( (W_y) \) and constant \( s\)-vector \( (W_s) \). By using Lemma A.2, if \( (\tilde{x}, \tilde{y}, \tilde{s}) \) is the solution of system (9) corresponding to the weight vector \( \tilde{w} \in W \), and \( \tilde{y} > 0 \) is any centric \( y\)-vector, then \( (\hat{x}, \hat{y}, \hat{s}) \) is the solution of system (9) corresponding to the weight vector \( \hat{Y}(\hat{Y})^{-1} \hat{w} \). This means that for every centric vector \( \hat{s} \) and any centric vector \( y \), \( \hat{S} y \) is a weight vector in the simplex.
For every pair of centric vectors $s$ and $y$, $W_s$ and $W_y$ are convex. To see this, let $(x, y, s)$ and $(x, y, s)$ be the weighted centers of $\hat{w}$ and $w$. Then, it is easy to see that for every $\beta \in [0, 1]$, $(x, \beta y + (1 - \beta) s)$ is the weighted center of $\beta \hat{w} + (1 - \beta) w$. With a similar reasoning, $W_y$ is convex for every centric $y$.

Using (9), we can express $W_s$ and $W_y$ as follows:

$$W_y = Y[(\mathcal{R}(A) + b) \cap \mathbb{R}^m_{++}] \cap B_1(0, 1),$$

(38)

$$W_s = S[N(A^\top) \cap \mathbb{R}^m_{++}] \cap B_1(0, 1),$$

(39)

where $B_1(0, 1)$ is the unit ball in 1-norm centered at zero vector. Here, we want to find another formulation for $W_y$ that might work better in some cases. We use the following lemma.

**Lemma A.3.** Assume that the rows of $B_y \in \mathbb{R}^{(m-n) \times m}$ make a basis for the null space of $A^\top Y$. Then there exists $x \in \mathbb{R}^n$ such that $YAx + w = Yb$ if and only if $B_yw = B_yYb$. I.e., $(Yb - w) \in \mathcal{R}(YA)$ iff $(Yb - w) \in N(B_y)$.

**Proof.** Assume that there exists $x$ such that $YAx + w = Yb$. By multiplying both sides with $B_y$ from the left and using the fact that $B_yYA = 0$ we have the result. For the other direction, assume that $B_yw = B_yYb$. Then $B_y(w - Yb) = 0$ which means $w - Yb$ is in the null space of $B_y$. Then, using the orthogonal decomposition theorem, we have $N(B_y) = \mathcal{R}(B_y^\top) = N(A^\top Y)^\perp = \mathcal{R}(YA)$. Thus, there exists $x$ such that $YAx + w = Yb$. \hfill \Box

Assume that $B \in \mathbb{R}^{(m-n) \times m}$ is such that its rows make a basis for the null space of $A^\top$. For every vector $y$, we have $A^\top y = A^\top Y(Y^{-1}y)$, so if $y$ is in the null space of $A^\top$, $Y^{-1}y$ is in the null space of $A^\top Y$. Hence, if the rows of $B$ make a basis for the null space of $A^\top$, the rows of $BY^{-1}$ make a basis for the null space of $A^\top Y$ and we can write $B_y = BY^{-1}$. Using Lemma A.3, there exists $x$ such that $YAx + w = Yb$ if and only if $BY^{-1}w = BY^{-1}Yb = Bb$, and we can write (38) as:

$$W_y = \left\{ w > 0 : BY^{-1}w = Bb, \; e^\top w = 1 \right\}.$$  

(40)

Let us denote the affine hull with $\text{aff}(.)$. We can prove the following lemma about $W_s$ and $W_y$.

**Lemma A.4.** Assume that $s$ and $y$ are centric, we have

$$W_s = \text{aff}(W_s) \cap W \quad \text{and} \quad W_y = \text{aff}(W_y) \cap W.$$

**Proof.** We prove the first one and our proof for the second one is the same. Clearly we have $W_s \subseteq \text{aff}(W_s) \cap W$. To prove the other side, assume by contradiction that there exist $w \in \text{aff}(W_s) \cap W$ such that $w \notin W_s$. Pick an arbitrary $\hat{w} \in \text{relint}(W_s)$ and consider all the points $w(\beta) = \beta w + (1 - \beta) \hat{w}$ for $\beta \in [0, 1]$. Both $w$ and $\hat{w}$ are in $\text{aff}(W_s)$, so all the points $w(\beta)$ are also in $\text{aff}(W_s)$. $w(0) \in W_s$ and $w(1) \notin W_s$, so let $\beta$ be $\text{sup} \{ \beta : w(\beta) \in W_s \}$.

Note that all the points in $W_s$ has the same $s$-vector, so we have $w(\beta) = Sy(\beta)$ for $\beta \in [0, \beta)$. By using (9) we must also have $w(\beta) \in W_s$. We want to prove that $\beta = 1$. Assume that $\beta < 1$. All the points on the line segment between $w(0)$ and $w(\beta)$ have the same $s$-vector and we can write them as $S(\gamma y(0) + (1 - \gamma)y(\beta))$ for $\gamma \in [0, 1]$. But note that $y(\beta) > 0$, so there is a small enough $\epsilon > 0$ such that $y_k = (-\epsilon y(0) + (1 + \epsilon)y(\beta)) > 0$ and hence $Sy_k$ is a weight vector in $W_s$. However, it is also a
vector on the line segment between \( w(\hat{\beta}) \) and \( w \) which is a contradiction to \( \hat{\beta} = \sup \{ \beta : w(\beta) \in W_s \} \). So \( \hat{\beta} = 1 \) and \( w = w(1) \in W_s \) which is a contradiction. Hence \( W_s \supseteq \text{aff}(W_s) \cap W \) and we are done. \( \square \)

We conclude that \( W \) is sliced in two ways by \( W_y \)'s and \( W_s \)'s for centric \( s \) and \( y \) vectors. For each centric \( s \) and each centric \( y \), \( W_y \) and \( W_s \) intersect at a single point \( S(y) \) on the simplex. We want to prove that the smallest affine subspace containing \( W_s \) and \( W_y \) is \( \text{aff}(W) = \{ w : e^\top w = 1 \} \). To that end, we prove some results on the intersection of affine subspaces. We start with the following definition:

**Definition A.1.** The recession cone of a convex set \( C \in \mathbb{R}^n \) is denoted by \( \text{rec}(C) \) and defined as:

\[
\text{rec}(C) := \{ y \in \mathbb{R}^n : (x + y) \in C, \ \forall x \in C \}.
\]

The lineality space of a convex set \( C \) is denoted by \( \text{lin}(C) \) and defined as:

\[
\text{lin}(C) := \{ (\text{rec}(C)) \cap (-\text{rec}(C)) \}.
\]

Let \( U \) be an affine subspace of \( \mathbb{R}^m \). If \( y \in \text{rec}(U) \), then \( -y \in \text{rec}(U) \), which means \( (\text{rec}(U)) = (-\text{rec}(U)) \). Therefore, by Definition A.1, we have \( \text{lin}(U) = \text{rec}(U) \). Then, by using the definition of the affine space we have:

\[
\text{lin}(U) := \{ u_1 - u_2 : \forall u_1, u_2 \in U \}.
\]

In other words, \( \text{lin}(U) \) is a linear subspace such that \( U = u + \text{lin}(U) \) for all \( u \in U \) where \( '+' \) is the Minkowski sum. The following two lemmas are standard, see, for instance, [20].

**Lemma A.5.** Given a pair of nonempty affine subspaces \( U \) and \( V \) in \( \mathbb{R}^n \), the following facts hold:

1. \( U \cap V \neq \emptyset \) if and only if for every \( u \in U \) and \( v \in V \), we have \( (v - u) \in \text{lin}(U) + \text{lin}(V) \).
2. \( U \cap V \) consists of a single point if and only if for every \( u \in U \) and \( v \in V \), we have \( (v - u) \in \text{lin}(U) + \text{lin}(V) \) and \( \text{lin}(U) \cap \text{lin}(V) = \{ 0 \} \).
3. For every \( u \in U \) and \( v \in V \), we have

\[
\text{lin}(\text{aff}(U \cup V)) = \text{lin}(U) + \text{lin}(V) + \{ \alpha(v - u) : \alpha \in \mathbb{R} \}.
\]

**Lemma A.6.** Let \( U \) and \( V \) be nonempty affine subspaces in \( \mathbb{R}^n \). Then we have the following properties:

1. If \( U \cap V = \emptyset \), then

\[
\dim(\text{aff}(U \cup V)) = \dim(U) + \dim(V) + 1 - \dim(\text{lin}(U) \cap \text{lin}(V)),
\]

2. If \( U \cap V \neq \emptyset \), then

\[
\dim(\text{aff}(U \cup V)) = \dim(U) + \dim(V) - \dim(U \cap V).
\]

Using the above lemmas, we deduce the following proposition.

**Proposition A.1.** Assume that \( s \) and \( y \) are centric \( s \)-vector and \( y \)-vector, respectively. Then the smallest affine subspace containing \( W_s \) and \( W_y \) is \( \text{aff}(W) = \{ w : e^\top w = 1 \} \).
Proof. We assumed that $A \in \mathbb{R}^{m \times n}$ has full column rank, i.e., rank($A$) = $n \leq m$ and the interior of \{x : Ax \leq b\} is not empty. Let $B_s$ denote the set of all centric s-vectors, i.e., the set of s-vectors for which there exist $(x, y, s)$ satisfies all the equations in (9). We claim that $B_s = \{s > 0 : s = b - Ax\}$.

For every $s \in \{s > 0 : s = b - Ax\}$, pick an arbitrary $y > 0$ such that $A^\top y = 0$. For every scalar $\alpha$ we have $A^\top(\alpha y) = 0$, so we can choose $\alpha$ such that $\alpha y^\top s = 1$. Hence $(x, \alpha y, s)$ satisfies (9) and we conclude that $B_s = \{s > 0 : s = b - Ax\}$. The range of $A$ has dimension $n$ and since $B_s$ is not empty; it is easy to see that the dimension of $B_s$ is also $n$. Moreover, we have $W_y = YB_s$ and since $Y$ is non-singular, we have dim$(W_y) = n$.

Now denote by $B_y$ the set of centric y-vectors. By (9), we have $A^\top y = 0$. The dimension of the null space of $A^\top$ is $(n - m)$. In addition, we have to consider the restriction $e^\top w = 1$; we have
\[ 1 = e^\top w = e^\top (Ys) = s^\top y = (b - Ax)^\top y = b^\top y - x^\top A^\top y = b^\top y. \]

So, we have $b^\top y = 1$ for centric y-vectors which reduces the dimension by one (since $b \notin \mathbb{R}(A)$), and dim$(B_y) = m - n - 1$. We have $W_s = SB_y$ and so by the same explanation dim$(W_s) = m - n - 1$.

We proved that $W_s$ and $W_y$ intersect at only a single point $w = Sy$, so dim$(W_s \cap W_y) = 0$. By using Lemma A.6-(2) the dimension of the smallest affine subspace containing $W_s$ and $W_y$ is
\[ \text{dim}(W_s) + \text{dim}(W_y) - \text{dim}(W_s \cap W_y) = n + m - n - 1 = m - 1. \]

The dimension of aff$(W)$ is also $m - 1$, so by Lemma A.4 aff$(W)$ is the least affine subspace containing $W_s$ and $W_y$. 

\[ \square \]

Appendix B. Some technical results and implementation details for $w$-space algorithm

In this appendix, we first have a discussion on choosing the next weight vector in implementing the weight space algorithm. After that we introduce two modified algorithms using the properties of the weight space we proved in Appendix A.

B.1. Choosing the next weight vector. In the above algorithm, we did not explain about choosing the next weight vector in the shrunked space. In the case of little information about the function, different centers can be chosen to achieve better convergence, as we explain in Subsection 3.3. If we have enough information about the utility function, we might be able to choose a more appropriate weight vector. Assume that $U(s) := \sum_{i=1}^{m} t_i \log s_i$ where $t \in \mathbb{R}_{+}^{m}$. By comparison of (10) with the optimization problem for the weighted analytic center, we see that our problem is actually finding the weighted analytic center for the weight vector $t$. Hence, if we had $t$, our problem would be finding the weighted center of $t$. However, $t$ can be computed by using the gradient of the function.

Assume that we start with $w^0$ with the weighted center $(x^0, s^0, y^0)$. Defining $g^0 := \nabla U(s^0)$, it is easy to see that $t = s^0 g^0$. Now we can choose $w^1 = \beta s^0 g^0$ where $\beta$ is the scaling factor such that $\beta e^\top s^0 g^0 = 1$, and the s-vector of $w^1$ is the solution of the problem. The same idea can be used if we know that the utility function is close to the sum of the logarithms.
Assume that $U(s)$ is non-decreasing on each entry, i.e., $\nabla U(s) \geq 0$ for all $s \in R^n_+$. Consider $W$-space algorithm introduced above and the point $w = S^k y^k$ from the simplex. The corresponding half-space is $(u^k)^T (w - w^k) \geq 0$ where $w^k = (S^k)^{-1} A h^k$, and $h^k$ is the solution of $A^T Y^0 (S^k)^{-1} A h^k = A^T g^k$. It is easy to show that $S^k y^k$ lies in that half space. We have:

$$
(u^k)^T (S^k y^k - w^k) = (u^k)^T (S^k y^k) = (h^k)^T A^T (S^k)^{-1} (S^k y^k) \\
= (g^k)^T A (A^T Y^0 (S^k)^{-1} A)^{-1} A^T g^k \geq 0,
$$

(42)

where the last inequality is from the fact that $A (A^T Y^0 (S^k)^{-1} A)^{-1} A^T$ is positive semidefinite. The problem here is that $\beta^k S^k y^k$ may not be in the shrunken space, where $\beta^k$ is again the scaling factor. So, we can perform a line search to find a point on the line segment $[S^k y^k, \beta S^k y^k]$ in the interior of the shrunken space.

### B.2. Modified algorithm in the $w$-space.

We designed a cutting-plane algorithm in the $w$-space for maximizing the utility function. In this subsection, we are going to use the properties of the weighted center we derived in Section 2 to improve the performance of the algorithm. We introduce two modified versions of the $w$-space algorithms in this subsection.

#### B.2.1. First modified algorithm.

As we proved in Section 2, for every centric $y$-vector $\hat{y}$ and any centric $s$-vector $\hat{s}$, $\bar{w} = \bar{Y} \hat{s}$ is a weight vector in the simplex $W$. As we are maximizing $U(s)$ over $s$, roughly speaking, only the $s$-vector of the weighted center is important for us for each $w \in W$. This is somehow explicit in our algorithm as, for example, the normal to the cutting-plane at each step, given in Proposition 3.4, depends on $s$ and $y^0$ which is the $y$-vector of the starting point $w^0$. The algorithm also guarantees to keep $Y^0 s^{\text{opt}}$ in the shrunken region at each step. Hence, we lose nothing if we try to work with weight vectors with $y = y^0$.

Consider Lemma A.2 which is about the convex combination of weight vectors. Assume that we have weight vectors $w^i$, $i \in \{1, \ldots, l\}$, with weighted centers $(x^i, y^0, s^i)$, which means they have the same $y$-vector. By Lemma A.2, for every set of $\beta_i \in [0, 1]$, $i \in \{1, \ldots, l\}$, such that $\sum_{i=1}^l \beta_i = 1$, we have $(\sum_{i=1}^l \beta_i x^i, y^0, \sum_{i=1}^l \beta_i s^i)$ is the $w$-center where $w := \sum_{i=1}^l \beta_i w^i$. In other words, when the $y$-vectors are the same, $s$-vector (equivalently $x$-vector) of the convex combination of $w^i$ is equal to the convex combination of $s^i$, $i \in \{1, \ldots, l\}$. This is interesting because if we can update the weight vectors by using the convex combination, we do not need to compute the weighted center. We are going to use this to modify our algorithm.

Assume that the starting point is $w^0$ with weighted center $(x^0, y^0, s^0)$. The modified algorithm is similar to the algorithm in Subsection 3.2 and the normal to the cutting-plane is derived by using (61). However, in the modified one, all $w^i$ have $y$-vector equal to $y^0$. The modified algorithm has two modules:

**Module 1:** Assume that at Step $i$, we have $w^i = Y^0 s^i$ and $w^{i-1} = Y^0 s^{i-1}$ with the corresponding normals of the cutting-planes $u^i$ and $u^{i-1}$. By the choice of $w^i$, we must have $(w^{i-1})^T (w^i - w^{i-1}) \geq 0$. In the modified algorithm, if we have $(w^i)^T (w^{i-1} - w^i) \geq 0$, then we put $w^{i+1} = (w^i + w^{i-1})/2$ (it is easy to see this weight vector is in the required cut simplex). In this case, we have $y^{i+1} = y^0$ and $s^{i+1} = (s^i + s^{i-1})/2$. 

If we have $(w^i)^\top(w^{i-1} - w^i) \geq 0$, then the line segment $[w^{i-1}, w^i]$ is no longer in the required cut simplex. However, there exists $t > 0$ such that $\hat{w} := w^i + t(w^i - w^{i-1})$ is in the required cut simplex. We can do a line search to find $t$ and then we set $w^{i+1} = \hat{w}$. In this case, we have $y^{i+1} = y^0$ and $s^{i+1} = s^i + t(s^i - s^{i-1})$.

**Module 2:** In Module 1, the algorithm always moves along a single line. When the weight vectors in Module 1 get close to each other, we perform Module 2 to get out of that line. To do that, we choose a constant $\epsilon > 0$ and whenever in Module 1 we have $\|w^i - w^{i-1}\|_2 \leq \epsilon$, we perform Module 2. In Module 2, like the algorithm in Subsection 3.2, we pick an arbitrary weight vector $\hat{w}$ in the remaining cut simplex and compute the weighted center $(\hat{x}, \hat{y}, \hat{s})$. The problem now is that $y$-vector is not necessarily equal to $y^0$. However, we said that $\hat{s}$ is important for our algorithm; hence, we consider the weight vector $Y^0\hat{s}$. This new weight vector is not necessarily in the required cut simplex. To solve this problem, we use the same technique as in Module 1. We consider the line containing the line segment $[\hat{w}, Y^0\hat{s}]$ and do a line search to find an appropriate weight vector on this line. To simplify the line search, we consider $(w^i)^\top(Y^0\hat{s} - w^i) \geq 0$ and $(w^i)^\top(Y^0\hat{s} - w^i) < 0$ separately.

At the end of Module 2, we again come back to Module 1 to continue the algorithm. As can be seen, we only have to find a weighted center in Module 1 which makes the modified algorithm computationally more efficient than the original algorithm, in practice.

B.2.2. Second modified algorithm. Consider the main algorithm and the proof of Proposition 3.4. We constructed normal vectors that satisfy (60). By using the supergradient inequality, $(g^1)^\top(\hat{s} - s^1) \leq 0$ results in $U(\hat{s}) \leq U(s^1)$. Assume that a sequence of $s$-vectors $\{s^0, s^1, \ldots, s^j\}$ has been created by the algorithm up to iteration $j$. We may not have access to the value of $U(s_i)$, $i \in \{1, \ldots, j\}$, however, we know that there exists $p \in \{1, \ldots, j\}$ such that $U(s^p) \geq U(s^i)$ for all $i \in \{1, \ldots, j\}$. By the supergradient inequality we must have $(g^j)^\top(s^p - s^i) \geq 0$ for all $i \in \{1, \ldots, j\}$ and from (60)

$$(w^j)^\top(Y^0s^p - Y^i s^i) = (g^j)^\top(s^p - s^i) \geq 0, \quad \forall i \in \{1, \ldots, j\}.$$ 

This means that $Y^0s^p$ is a weight vector in the desired cut simplex. Let $\{p_1, \ldots, p_k\}$ be the indices that $(g^j)^\top(s^{p_l} - s^i) \geq 0$ for all $i \in \{1, \ldots, j\}, l \in \{1, \ldots, k\}$. By the above explanation, we know that $k \geq 1$ (the $s$-vector with the largest value so far is in this set.). The idea of the modified algorithm is that when $k > 1$, we put a convex combination of these $s$-vectors as the new $s$-vector. We can divide the new algorithm into three modules.

**Module 1:** $k > 1$: Define $s^{j+1} := \frac{1}{k}(s^{p_1} + \cdots + s^{p_k})$ and $w^{j+1} := Y^0s^{j+1}$.

**Module 2:** $k = 1$. We only have one point $s^p$ that $(g^j)^\top(s^p - s^i) \geq 0$ for all $i \in \{1, \ldots, j\}$ and by the above explanation we have $U(s^p) \geq U(s^i)$ for all $i \in \{1, \ldots, j\}$. Hence, $s^p$ is our best point so far and we use it to find the next one. To do that, we choose a direction $ds$ such that $s^{j+1} = s^p + \alpha ds$. $s^{j+1} - s^p = \alpha ds$ must be in $\mathcal{R}(A)$ and therefore a good choice is the projection of $g^p$ on $\mathcal{R}(A)$. Let us define $P_A$ as the projection matrix to $\mathcal{R}(A)$, then we define $ds = P_A g^p$ and do a line search to find the appropriate $\alpha$ such that $s^{j+1} := s^p + \alpha ds$ is in the desired cut simplex. We also have $w^{j+1} := Y^0s^{j+1}$.

**Module 3:** In the first two modules, we do not have to calculate the weighted center. In this module, like the first modified algorithm, when $\|w^{j+1} - w^j\|$ in Module 1 or 2 is smaller than a specified value, we perform an iteration like the original algorithm; pick an arbitrary point inside the cut simplex and compute the weighted center for that.
AN IMPROVISED APPROACH TO ROBUSTNESS IN LINEAR OPTIMIZATION

APPENDIX C. Probabilistic Analysis

Probabilistic analysis is tied to robust optimization. One of the recent trends in robust optimization research is the attempt to try reducing conservatism to get better results, and at the same time keeping a good level of robustness. In other words, we have to show that our proposed answer has a low probability of infeasibility. In this section, we derive some probability bounds for our algorithms based on weight and slack vectors. These bounds can be given to the DM with each answer and the DM can use them to improve the next feedback.

C.1. Representing the robust feasible region with weight vectors. Before starting the probabilistic analysis, want to relate the notion of weights to the parameters of the uncertainty set. As we explained in Subsection 1.3, we consider our uncertainty sets as follows:

\[ B_i := \left\{ \hat{b}_i : \exists \tilde{z} = (\tilde{z}_i^1, \ldots, \tilde{z}_i^{N_i}) \in [-1, 1]^{N_i} \text{ s.t. } \hat{b}_i = b_i^{(0)} + \sum_{l=1}^{N_i} \Delta b_i^l \tilde{z}_i^l \right\}, \]

where \( \{\tilde{z}_i\}_{i=1}^{N_i} \), \( i \in \{1, \ldots, m\} \) are independent random variables, and \( \Delta b_i^l \) is the scaling factor of \( \tilde{z}_i^l \).

We assume that the support of \( \tilde{z}_i^l \) contains \( \tilde{z}_i^l = -1 \), i.e., \( Pr\{\tilde{z}_i^l = -1\} \neq 0 \). Let us define another set which is related to the weight vectors:

\[ W := \left\{ (w_1, \ldots, w_m) : w_i \in [y_i(w)\|\Delta b_i\|_1, 1], \sum_{i=1}^{m} w_i = 1 \right\}, \]

where \( y(w) \) is the \( y \)-vector of \( w \). Our goal is to explicitly specify a set of weights whose corresponding \( w \)-center makes the feasible solution of the robust counterpart.

**Proposition C.1.** Let \( x \) satisfy \( Ax \leq \hat{b} \) for every \( \hat{b} \in B_1 \times B_2 \times \cdots \times B_m \). Then there exists some \( w \in W \), so that \( x \) is the weighted analytic center with respect to the weight vector \( w \), i.e., \( x = x(w) \).

In other words,

\[ \left\{ x : Ax \leq \hat{b}, \forall \hat{b} \in B_1 \times B_2 \times \cdots \times B_m \right\} \subseteq \left\{ x(w) : w \in W \right\}. \]

**Proof.** Let \( \hat{w} > 0 \) be an arbitrary vector such that \( \sum_{i=1}^{m} \hat{w}_i = 1 \), and let \( (\hat{x}, \hat{y}, \hat{s}) \) be the weighted center corresponding to it. Assume that \( x \) is in the robust feasible region; we must have \( \langle a_i, x \rangle \leq b_i^{(0)} + \langle \Delta b_i, \tilde{z}_i \rangle \) for every \( \tilde{z}_i \) with nonzero probability, particularly for \( \tilde{z}_i = -e \) where \( e \) is all ones vector. So

\[ \langle a_i, x \rangle - b_i^{(0)} \leq \langle \Delta b_i, \tilde{z}_i \rangle = \langle \Delta b_i, -e \rangle = -\|\Delta b_i\|. \]

Define \( s_i := b_i^{(0)} - \langle a_i, x \rangle \). Thus, from the above equation, for every \( i \in \{1, \ldots, m\} \) we have

\[ 0 < \|\Delta b_i\|_1 \leq s_i, \]

and consequently \( \hat{y}_i\|\Delta b_i\|_1 \leq \hat{y}_i s_i \) using the fact that \( \hat{y}_i > 0 \). For every \( i \in \{1, \ldots, m\} \), we set

\[ w_i := \hat{y}_i s_i. \]

Since \( (x, \hat{y}, s) \) satisfies the optimality conditions, we have \( x = x(w) \). It remains to show that \( w \in W \). First note that:

\[ \sum_{i=1}^{m} w_i = \sum_{i=1}^{m} s_i \hat{y}_i = \sum_{i=1}^{m} \hat{s}_i \hat{y}_i = \sum_{i=1}^{m} \hat{w}_i = 1, \]
where for the second equality we used Lemma A.1. Now, using the fact that \( w_i \geq 0 \) for every \( i \in \{1, \ldots, m\} \), we have \( w_i < \sum_{j=1}^{m} w_j = 1 \). We already proved that \( \hat{y}_i \| \Delta b_i \|_1 \leq \hat{y}_i s_i = w_i \). These two inequalities prove that \( w_i \in [\hat{y}_i \| \Delta b_i \|_1, 1] \).

The above proposition shows that when the robust counterpart problem with respect to the uncertainty set \( B_1 \times B_2 \times \cdots \times B_m \) is feasible, the set \( \mathcal{W} \) is nonempty. In the next proposition we prove that the equality holds in the above inclusion.

**Proposition C.2.** (a) We have

\[
\{ x : Ax \leq \tilde{b}, \forall \tilde{b} \in B_1 \times B_2 \times \cdots \times B_m \} = \{ x(w) : w \in \mathcal{W} \}.
\]

(b) Assume that \( w > 0 \) satisfies \( \sum_{i=1}^{m} w_i = 1 \), and \( y \) is its corresponding \( y \)-vector. For every \( i \in \{1, \ldots, m\} \), we have

\[
w_i \geq y_i \| \Delta b_i \|_1 \Rightarrow \langle a_i, x(w) \rangle \leq \tilde{b}_i, \quad \forall \tilde{b}_i \in B_i.
\]

**Proof.** (a) \( \subset \) part was proved in Proposition C.1. For \( \supset \), let \( w \in \mathcal{W} \) and \( (x, y, s) \) be its corresponding weighted center. By \( w \in \mathcal{W} \) we have

\[
y_i \| \Delta b_i \|_1 \leq w_i = s_i y_i = (b_i^{(0)} - \langle a_i, x \rangle) y_i = \| \Delta b_i \|_1 \leq (b_i^{(0)} - \langle a_i, x \rangle).
\]

Therefore, for all \( \tilde{z}_i \in \times_{i=1}^{m} [-1, 1] \),

\[
\langle a_i, x \rangle \leq b_i^{(0)} - \| \Delta b_i \|_1 \leq b_i^{(0)} - \sum_{i=1}^{N_i} \Delta b_i^{l} \tilde{z}_i^{l} = b_i^{(0)} + \langle \tilde{z}_i, \Delta b_i \rangle,
\]

which proves \( x \) is a robust feasible solution with respect to the uncertainty set \( B_1 \times B_2 \times \cdots \times B_m \).

(b) Assume that \( w > 0 \) satisfies \( \sum_{i=1}^{m} w_i = 1 \), \( y \) is its corresponding \( y \)-vector, and there exists \( i \in \{1, \ldots, m\} \) such that \( w_i \geq y_i \| \Delta b_i \|_1 \). If there exists \( \tilde{b}_i \in B_i \) such that \( \langle a_i, x(w) \rangle > \tilde{b}_i \) where \( \tilde{b}_i = b_i^{(0)} + \sum_{l=1}^{N_i} \Delta b_i^{l} \tilde{z}_i^{l} \), by using \( \tilde{z}_i^{l} \geq -1 \) we have

\[
\langle a_i, x(w) \rangle > \tilde{b}_i \Rightarrow \langle a_i, x(w) \rangle > b_i^{(0)} + \sum_{l=1}^{N_i} \Delta b_i^{l} \tilde{z}_i^{l} \geq b_i^{(0)} - \sum_{l=1}^{N_i} \Delta b_i^{l}
\]

\[
\Rightarrow \sum_{l=1}^{N_i} \Delta b_i^{l} > b_i^{(0)} - \langle a_i, x(w) \rangle = s_i(w)
\]

\[
\Rightarrow y_i \sum_{l=1}^{N_i} \Delta b_i^{l} > y_is_i(w) = w_i \geq y_i \sum_{l=1}^{N_i} \Delta b_i^{l}
\]

\[
\Rightarrow \sum_{l=1}^{N_i} \Delta b_i^{l} > \sum_{l=1}^{N_i} \Delta b_i^{l},
\]

which is a contradiction. We conclude that \( \langle a_i, x(w) \rangle \leq \tilde{b}_i \) for all \( \tilde{b}_i \in B_i \). \( \square \)
C.2. Probability bounds. Without loss of generality, we make the following assumptions on \( \tilde{b} \) and \( \tilde{c} \):

- For every \( i \in \{1, 2, \ldots, m\} \), \( \tilde{b}_i \) can be written as \( \tilde{b}_i = b_i^{(0)} + \sum_{l=1}^{N_i} \Delta b_i^l \tilde{z}_i^l \) where \( \{\tilde{z}_i^l\}_{l=1}^{N_i} \) are independent random variables for every \( i \in \{1, \ldots, m\} \).
- For each \( \tilde{c}_i, i \in \{1, \ldots, n\} \), we have \( \tilde{c}_i = c_i^{(0)} + \sum_{l=1}^{N_{ic}} \Delta c_i^l \tilde{z}_i^l \) where \( \{\tilde{z}_i^l\}_{l=1}^{N_{ic}} \) are independent random variables.

As can be seen above, each variable \( \tilde{b}_i \) is the summation of a nominal value \( b_i^{(0)} \) with scaled random variables \( \{\tilde{z}_i^l\}_{l=1}^{N_i} \). In practice, the number of these random variables \( N_i \) is small compared to the dimension of \( A \) as we explained above: each random variable \( \tilde{z}_i^l \) represents a major source of uncertainty in the system.

Suppose we wish to find a robust feasible solution with respect to the uncertainty set \( B_1 \times B_2 \times \cdots \times B_m \), where \( B_i \) was defined in (43). By Proposition C.2, it is equivalent to finding the weighted center for a \( w \in \mathcal{W} \), where \( \mathcal{W} \) is defined in (44). However, finding such a weight vector is not straightforward as we do not have an explicit formula for \( \mathcal{W} \). Assume that we pick an arbitrary weight vector \( w > 0 \) such that \( \sum_{i=1}^m w_i = 1 \), with the weighted center \( (x, y, s) \). Let us define the vector \( \delta \) for \( w \) as

\[
\delta_i = \frac{w_i}{y_i \|\Delta b_i\|_1}, \quad i \in \{1, 2, \ldots, m\},
\]

where \( \Delta b_i \) was defined in (43). For each \( i \in \{1, \ldots, m\} \), if \( 1 \leq \delta_i \), by Proposition C.2-(b) we have \( \left< a_i, x(w) \right> \leq \tilde{b}_i \) for all \( \tilde{b}_i \in B_i \). So, the problem is with the constraints that \( 1 > \delta_i \). For every such constraint, we can find a bound on the probability that \( \left< a_j, x(w) \right> > \tilde{b}_j \). As in the proof of Proposition C.2-(b), in general we can write:

\[
\Pr\{\left< a_j, x \right> > \tilde{b}_j\} = \Pr\left\{-y_i \sum_{l=1}^{N_i} \Delta b_i^l \tilde{z}_i^l + w_i \delta_i \|\Delta b_i\|_1 \right\}
\]

\[
= \Pr\left\{-\sum_{l=1}^{N_i} \Delta b_i^l \tilde{z}_i^l > \delta_i \|\Delta b_i\|_1 \right\} \leq \exp\left(-\frac{\delta_i^2 (\|\Delta b_i\|_1)^2}{2 \sum_{l=1}^{N_i} (\Delta b_i^l)^2}\right),
\]

where the last inequality is derived by using Hoeffding’s inequality:

**Lemma C.1. (Hoeffding’s inequality[26])** Let \( v_1, v_2, \ldots, v_n \) be independent random variables with finite first and second moments, and for every \( i \in \{1, 2, \ldots, n\} \), \( \tau_i \leq v_i \leq \rho_i \). Then for every \( \varphi > 0 \)

\[
\Pr\left\{\sum_{i=1}^n v_i - E\left(\sum_{i=1}^n v_i\right) \geq n\varphi\right\} \leq \exp\left[-\frac{-2n^2 \varphi^2}{\sum_{i=1}^n (\rho_i - \tau_i)^2}\right].
\]

Bertsimas and Sim [12] derived the best possible bound, i.e., a bound that is achievable. The corresponding lemma proved in [12] is as follows:

**Lemma C.2.** (a) If \( \tilde{z}_i^l, l \in \{1, \ldots, N_i\} \), are independent and symmetrically distributed random variables in \([-1, 1]\), \( p \) is a positive constant, and \( \gamma_i \leq 1, l \in \{1, \ldots, N_i\} \), then

\[
\Pr\left\{\sum_{l=1}^{N_i} \gamma_{il} \tilde{z}_i^l \geq p\right\} \leq B(N_i, p),
\]

where \( B(N, p) \) is a binomial distribution.
where

\[ B(N_i, p) = \frac{1}{2^{N_i}} \left[ (1 - \mu) \left( \frac{N_i}{\nu}\right) + \sum_{i=\lceil \nu \rceil+1}^{N_i} \frac{(N_i}{i} \right), \]

where \( \nu := (N_i + p)/2 \), and \( \mu := \nu - \lceil \nu \rceil \).

(b) The bound in (46) is tight for \( z_i^l \) having a discrete probability distribution:

\[ \Pr\{z_i^l = 1\} = \Pr\{z_i^l = -1\} = 1/2, \quad \gamma_{il} = 1, \quad l \in \{1, \ldots, N_i\}, \]

an integral value of \( p \geq 1 \), and \( p + N_i \) being even.

We can use the bound for our relation (45) as follows. Assume that \( z_i^l, \quad l \in \{1, \ldots, N_i\}, \) are independent and symmetrically distributed random variables in \([-1, 1]\). Also denote by \( \max(\Delta b_i) \), the maximum entry of \( \Delta b_i \). Using (45), We can write

\[
\Pr\{\langle a_j, x \rangle > \tilde{b}_j\} = \Pr \left\{ \sum_{l=1}^{N_i} \Delta b_i^l z_i^l > \delta_i \|\Delta b_i\|_1 \right\} \\
\leq \Pr \left\{ \sum_{l=1}^{N_i} \frac{\Delta b_i^l}{\max(\Delta b_i)} z_i^l \geq \delta_i \|\Delta b_i\|_1 \right\} \\
\leq B \left( N_i, \delta_i \|\Delta b_i\|_1 \frac{\max(\Delta b_i)}{\max(\Delta b_i)} \right).
\]  

To compare these two bounds, assume that all the entries of \( \Delta b_i \) are equal. Bound (45) reduces to \( \exp(-\delta_i^2 N_i/2) \), and bound (47) reduces to \( B(N_i, \delta_i N_i) \). We can prove that bound (47) dominates bound (45). Moreover, bound (47) is somehow the best possible bound as it can be achieved by a special probability distribution as in Lemma C.2. The above probability bounds do not take part in our algorithm explicitly. However, for each solution, we can present these bounds to the DM and s/he can use them to improve the feedback to the algorithm. As an example of how these bounds may be used for the DM, we show how to construct a concave utility function \( U(s) \) based on these probability bounds. Bounds (45) and (47) are functions of \( \delta_i = \frac{w_i}{\|\Delta b_i\|_1} = \frac{s_i}{\|\Delta b_i\|_1} \) and as a result, functions of \( s \).

Now, assume that based on the probability bounds, the DM defines a function \( u_i(s_i) \) for each slack variable \( s_i \) as shown in Figure 8. \( u_i(s_i) \) increases as \( s_i \) increases, and then at the point \( c_i^l \) becomes flat. At \( s_i = c_i^l \) it starts to decrease to reach zero. Parameters \( c_i^l \) and \( c_i^r \) are specified by the DM’s desired bounds. Now, we can define the utility function as \( U(s) := \prod_{j=1}^{m} u_i(s_i) \). This function is not concave, but maximization of it is equivalent to the maximization of \( \ln(U(s)) \) which is concave.

**APPENDIX D. PROOFS OF PROPOSITIONS 3.1, 3.2, 3.3, AND 3.4**

**Proof of Proposition 3.1**

**Proof.** Consider the weight vectors \( Y^0 s^{opt} \) and \( Y^1 s^{opt} \). Our two hyperplanes are

\[
P_0 := \{ w : (g^0)^\top (Y^0)^{-1} (w - Y^0 s^0) = 0 \}, \\
P_1 := \{ w : (g^1)^\top (Y^1)^{-1} (w - Y^1 s^1) = 0 \}.
\]
Figure 8. The function $u_i(s_i)$ defined for the slack variable $s_i$

By Lemma 3.2, $Y^0 s^{opt}$ is in the half-space $(g^0)\top (Y^0)^{-1}(w - Y^0 s^0) \geq 0$ and $Y^1 s^{opt}$ is in the half-space $(g^1)\top (Y^1)^{-1}(w - Y^1 s_1) \geq 0$. If one of these two points is also in the other half-space, then we are done. So, assume that

$$(g^0)\top (Y^0)^{-1}(Y^1 s^{opt} - Y^0 s^0) < 0 \quad \text{and} \quad (g^1)\top (Y^1)^{-1}(Y^0 s^{opt} - Y^1 s^1) < 0$$

(we are seeking contradiction), which is equivalent to

$$(g^0)\top ((Y^0)^{-1}Y^1 s^{opt} - s^0) < 0 \quad \text{and} \quad (g^1)\top ((Y^1)^{-1}Y^0 s^{opt} - s^1) < 0.$$  

Using (14) and (48) we conclude that

$$U((Y^0)^{-1}Y^1 s^{opt}) < U(s^0) \leq U(s^{opt}) \quad \text{and} \quad U((Y^1)^{-1}Y^0 s^{opt}) < U(s^1) \leq U(s^{opt}).$$

However, note that $(Y^0)^{-1}Y^1 = ((Y^1)^{-1}Y^0)^{-1}$ and this is a contradiction to Definition 3.1. So (48) is not true and at least one of $Y^0 s^{opt}$ and $Y^1 s^{opt}$ is in

$$\{ w : (g^{0w})\top (w - w^0) \geq 0, (g^{1w})\top (w - w^1) \geq 0 \}.$$  

□

Proof of Proposition 3.2

Proof. Among the three representations of $W_s$ were given in (39), we use the second one in the following. If (16) is not true, then the following system is infeasible:

$$A\top (s^{opt})^{-1}w = 0, \quad e\top w = 1, \quad w \geq 0, \quad (g^{iw})\top (w - w^i) \geq 0, \quad i \in \{0, \ldots, k\}.$$  

(49)
By Farkas’ Lemma, there exist $v \in \mathbb{R}^n$, $p \in \mathbb{R}$, and $q \in \mathbb{R}_+^k$ such that:

$$(S^{opt})^{-1}Av + pe - \sum_{i=0}^{k} q_ig_i^w \geq 0 \iff Av + ps^{opt} - \sum_{i=0}^{k} q_is^{opt}(Y^i)^{-1}g_i \geq 0,$$

(50)

$$p - \sum_{i=0}^{k} q_i(g_i^w)^\top w^i < 0 \iff p - \sum_{i=0}^{k} q_i(g_i)^\top s^i < 0.$$  

Now for each $j \in \{0, \ldots, k\}$, we multiply both sides of the first inequality in (50) with $e^\top Y^j$, then we have:

$$p - \sum_{i=0}^{k} q_i(s^{opt})^\top Y^j(Y^i)^{-1}g_i \geq 0, \quad \forall j \in \{0, \ldots, k\},$$

(51)

$$p - \sum_{i=0}^{k} q_i(g_i)^\top s^i < 0,$$

where we used the facts that $e^\top Y^j Av = (A^\top y)^\top v = 0$ and $e^\top Y^j s^{opt} = 1$. If we multiply the first set of inequalities in (51) with $-1$ and add it to the second one we have

$$q_j(g_j)^\top (s^{opt} - s^j) + \sum_{i \neq j} q_i(g_i)^\top (Y^j(Y^i)^{-1}s^{opt} - s^i) < 0,$$

(52)

for all $j \in \{0, \ldots, k\}$, $q \in \mathbb{R}_+^k$ and $(g_j)^\top (s^{opt} - s^j) \geq 0$ by supergradient inequality. Hence, from (52), for each $j \in \{0, \ldots, k\}$, there exists $\phi_j \in \{0, \ldots, k\} \setminus \{j\}$ such that $(g_{\phi_j})^\top (Y^j(Y^\phi_j)^{-1}s^{opt} - s^j) < 0$ which, using (12), means $U(Y^{j}(Y^{\phi_j})^{-1}s^{opt}) < U(s^{\phi_j}) \leq U(s^{opt})$. Therefore, by the first property of NDAS functions, we must have

$$U(Y^{\phi_j}(Y^j)^{-1}s^{opt}) \geq U(s^{opt}).$$

(53)

Now, it is easy to see that there exists a sequence $j_1, \ldots, j_t \in \{0, \ldots, k\}$ such that $\phi_{j_t} = j_{t+1}$ and $\phi_{j_1} = j_1$. By using (53) and the second property of NDAS functions $t - 1$ times we can write:

$$U(s^{opt}) \leq U(Y^{j_t}(Y^{j_{t-1}})^{-1}s^{opt}) \leq U(Y^{j_{t-1}}(Y^{j_{t-2}})^{-1}s^{opt}) \leq \cdots \leq U(Y^{j_2}(Y^{j_1})^{-1}s^{opt}) = U(Y^{j_t}(Y^{j_{t-1}})^{-1}s^{opt}).$$

(54)

However, we had $U(Y^{j_1}(Y^{j_1})^{-1}s^{opt}) = U(Y^{j_2}(Y^{j_2})^{-1}s^{opt}) < U(s^{opt})$ which is a contradiction to (54). This means the system (49) is feasible and we are done. \hfill \Box

**Example D.1.** The statement of Proposition 3.1 is not true for a general concave function.

**Proof.** Consider the first example of Example 2.1. We have $m = 3$, $n = 1$,

$A = [1, -1, -1]^\top$, and $b = [1, 0, 0]^\top$. Using (9), the set of centric $s$-vectors is

$$B_s = \{[1-z, \ z, \ z]^\top : z \in (0,1)\}.$$  

The set of centric $y$-vectors, $B_y$, is specified by solving $A^\top y = 0$ and $y^\top b = 1$ while $y > 0$ and we can see that $B_y = \{[1, \ z, \ 1-z]^\top : z \in (0,1)\}$. As shown in Figure 2, $W_s$s are parallel line segments while $W_y$s are line segments that all intersect at $[1, 0, 0]^\top$.  


Now, assume that the function $U(s)$ is as follows (does not depend on $s_3$)

$$U(s) = \begin{cases} 
3s_1 - s_2, & \text{if } s_1 \leq s_2; \\
-s_1 + 3s_2, & \text{if } s_1 > s_2.
\end{cases}$$  \hspace{1cm} (55)

This function is piecewise linear and it is easy to see that it is concave. $U(s)$ is also differentiable at all the points except the points $s_1 = s_2$. At any point that the function is differentiable, the supergradient is equal to the gradient of the function at that point. Hence, we have $\partial U(s) = \{3, -1, 0\}^\top$ for $s_1 < s_2$ and $\partial U(s) = \{-1, 3, 0\}^\top$ for $s_1 > s_2$.

If we consider $U(s)$ on $B_s$, we can see that the maximum of the function is attained at the point that $s_1 = s_2$, so $s_{\text{opt}} = [1/2, 1/2, 1/2]^\top$. Now assume that we start at $w^0 = S^0y^0 = [0.4, 0.1, 0.5]^\top$. Because we have $y_1 = 1$ for all centric $y$-vectors, $w^0_1 = s^0_1$, and we can easily find $s^0$ and $y^0$ as $s^0 = [0.4, 0.6, 0.6]^\top$ and $y^0 = [1, 1/6, 5/6]^\top$. The hyperplane passing through $w^0$ is $(g^0)^\top(Y^0)^{-1}(w - w^0) = 0$ and since $s^0 < s^0_2$ we have

$$\begin{align*}
(g^0)^\top(Y^0)^{-1} &= [3, -1, 0](Y^0)^{-1} = [3, -6, 0],
\end{align*}$$

and we can write the hyperplane as $3(w_1 - 0.4) - 6(w_2 - 0.1) = 0$. In the next step, we have to choose a point $w^1$ such that $(g^0)^\top(Y^0)^{-1}(w^1 - w^0) \geq 0$. Let us pick $w^1 = [0.6, 0.19, 0.21]^\top$ for which we can easily find $s^1 = [0.6, 0.4, 0.4]^\top$ and $y^1 = [1, 0.475, 0.525]^\top$. For this point we have $s^1_1 > s^1_2$, so $(g^1)^\top(Y^1)^{-1} = [-1, 6.32, 0]^\top$ and the hyperplane passing through $w^1$ is $- (w_1 - 0.6) + 6.32(w_2 - 0.19) = 0$. The intersection of two hyperplanes on the simplex can be found by solving the following system of equations:

$$\begin{align*}
\begin{cases} 
3w_1 - 6w_2 = 0.6 \\
-w_1 - 6w_2 = 0.6 \\
w_1 + w_2 + w_3 = 1
\end{cases} \Rightarrow w^* = \begin{bmatrix} 0.57 \\ 0.185 \\ 0.245 \end{bmatrix}.
\end{align*}$$  \hspace{1cm} (57)

The intersection of simplex and the hyperplanes $(g^0)^\top(Y^0)^{-1}(w - w^0) = 0$ and $(g^1)^\top(Y^1)^{-1}(w - w^1) = 0$ are shown in Figure 9. The intersection of simplex with 

$$\{ w : (g^0)^\top(Y^0)^{-1}(w - w^0) \geq 0, \ (g^1)^\top(Y^1)^{-1}(w - w^1) \geq 0 \}$$

is shown by hatching lines. As can be seen, we have:

$$\{ w : (g^0)^\top(w - w^0) \geq 0, \ (g^1)^\top(w - w^1) \geq 0 \} \cap W_{s_{\text{opt}}} = \emptyset.$$

\hfill \Box

**Proof of Proposition 3.3**

**Proof.** Assume that $w^0 = Y^0s^0$ is the point that is chosen and let $u^0$ be the normal vector to the desired hyperplane $P$. First, we want the hyperplane to contain $W_{s_{\text{opt}}}$. This means that for all centric $\hat{y}$, the vector $S^0y^0 - S^0\hat{y}$ is on $P$, i.e., we have $(u^0)^\top S^0(y^0 - \hat{y}) = 0$. Since $A^\top(y^0 - \hat{y}) = 0$, we can put $u^0 = (S^0)^{-1}Ah^0$ with an arbitrary $h^0$ and we have:

$$(u^0)^\top S^0(y^0 - \hat{y}) = (h^0)^\top A^\top(S^0)^{-1}S^0(y^0 - \hat{y}) = 0.$$  

Now, we want to find $h^0$ such that $(u^0)^\top(w - Y^0s^0)$ cuts $W_{s_{\text{opt}}}$ the same way as $(g^0)^\top(Y^0)^{-1}(w - Y^0s^0)$ cuts it. We actually want to find $h^0$ which satisfies the stronger property that
where \( \Pi \) is the orthogonal projection onto the range of \( (Y^0)^\top (Y^0)^{-1}(w - w^0) = 0 \) and \( (g^1)^\top (Y^1)^{-1}(w - w^1) = 0 \) in Example D.1.

\[
(u^0)^\top (w - Y^0 s^0) = (g^0)^\top (Y_0)^{-1}(w - Y^0 s^0) \text{ for all } w \in W_{y^0}. \text{ All the points in } W_{y^0} \text{ are of the form } Y^0 \hat{s}, \text{ so we must have } (u^0)^\top Y^0(\hat{s} - s^0) = (g^0)^\top (\hat{s} - s^0). \text{ Since } (\hat{s} - s^0) \text{ is in the range of } A, \text{ this equation is true if and only if:}
\]
\[
(u^0)^\top Y^0 Ax = (g^0)^\top Ax \Rightarrow ((u^0)^\top Y^0 - (g^0)^\top)Ax = 0, \forall x \in \mathbb{R}^n.
\]

This means that \( Y^0 u^0 - g^0 \) must be in the \( \mathcal{R}(A)^\perp = \mathcal{N}(A^\top) \), which means \( A^\top (Y^0 u^0 - g^0) = 0. \) However, we had from above that \( u^0 = (S^0)^{-1}Ah^0 \) and hence:

\[
A^\top Y^0 u^0 = A^\top g^0 \Rightarrow A^\top Y^0 (S^0)^{-1}Ah^0 = A^\top g^0 \Rightarrow h^0 = (A^\top Y^0 (S^0)^{-1}A)^{-1}A^\top g^0.
\]

So, the hyperplane with normal vector \( u^0 = (S^0)^{-1}Ah^0 \), where \( h^0 = (A^\top Y^0 (S^0)^{-1}A)^{-1}A^\top g^0 \) has the required properties. Since this hyperplane cuts \( W_{y^0} \) the same way as \( (g^0)^\top (Y^0)^{-1}(w - Y^0 s^0) \) does, we conclude that \( (u^0)^\top (Y^0 s^{opt} - Y^0 s^0) \geq 0. \) Therefore, \( Y^0 s^{opt} \) is in the half-space \( (u^0)^\top (w - Y^0 s^0) \geq 0. \)

The normal of the hyperplane derived in Proposition 3.3 has a nice interpretation with respect to orthogonal projection and the primal-dual scaling \( Y^{-1}S \). We have:

\[
u^0 = (S^0)^{-1/2}A(Y^0)^{-1}(S^0)^{-1/2}A^\top g^0
\]

\[
= (Y^0)^{-1/2}(S^0)^{-1/2}A^\top g^0
\]

\[
= (Y^0)^{-1/2}(S^0)^{-1/2}A^\top Y^0 (S^0)^{-1}A^\top (Y^0)^{-1/2}(S^0)^{-1/2}g_0
\]

\[
(59) \quad = (Y^0)^{-1/2}(S^0)^{-1/2}Y^{-1/2}(S^0)^{1/2}g_0,
\]

where \( \Pi \) is the orthogonal projection onto the range of \( (Y^0)^{1/2}(S^0)^{-1/2}A \). Note that a main benefit of the hyperplane in Proposition 3.3 is that when we choose a point, we can cut away all the points with the same \( s \)-vector. Now, we prove the following proposition which shows we can cut the simplex
with a sequence of hyperplanes such that the intersection of their corresponding half-spaces contain a point from \( W_{s^{opt}} \).

**Proof of Proposition 3.4**

Proof. As in the proof of Proposition 3.3, if we set \( u^1 = (S^1)^{-1}Ah^1 \), then the hyperplane contains all the points in \( W_{s^1} \). To satisfy the second property, we want to find \( h^1 \) with the stronger property that

\[
(u^1)^\top (Y^0\hat{s} - Y^1s^1) = (g^1)^\top (\hat{s} - s^1),
\]

for all the centric \( \hat{s} \). The reason is that we already have \((g^1)^\top (s_{opt} - s^1) \geq 0\). By the choice of \( u^1 = (S^1)^{-1}Ah^1 \), for every centric \( y \) we have

\[
(u^1)^\top Y^1s^1 = (u^1)^\top Y^0s^1 = 0
\]

and we can continue the above equation as follows:

\[
(g^1)^\top (\hat{s} - s^1) = (u^1)^\top (Y^0\hat{s} - Y^1s^1) = (u^1)^\top (Y^0\hat{s})
\]

\[
= (u^1)^\top (Y^0\hat{s} - Y^0s^1)
\]

\[
= (u^1)^\top Y^0(\hat{s} - s^1).
\]

Now we can continue in a similar way as in the proof of Proposition 3.3. Since \((\hat{s} - s^0)\) is in the range of \( A \), we must have:

\[
((u^1)^\top Y^0 - (g^1)^\top)Ax = 0, \quad \forall x \in \mathbb{R}^n.
\]

By the same reasoning, we have:

\[
A^\top Y^0u^1 = A^\top g^1 \Rightarrow A^\top Y^0(S^1)^{-1}Ah^1 = A^\top g^1 \Rightarrow h^1 = (A^\top Y^0(S^1)^{-1}A)^{-1}A^\top g^1.
\]

So, the hyperplane with normal vector \( u^1 = (S^1)^{-1}Ah^1 \), where \( h^1 = (A^\top Y^0(S^1)^{-1}A)^{-1}A^\top g^1 \) has the required properties. \( \square \)