A Simple Method for Convex Optimization in the Oracle Model

Daniel Dadush[1⋆], Christopher Hojny[2], Sophie Huiberts[1], and Stefan Weltge[3]

[1] Centrum Wiskunde & Informatica, Netherlands
{dadush,s.huiberts}@cwi.nl
[2] Eindhoven University of Technology, Netherlands
c.hojny@tue.nl
[3] Technical University of Munich, Germany
weltge@tum.de

Abstract. We give a simple and natural method for computing approximately optimal solutions for minimizing a convex function f over a convex set K given by a separation oracle. Our method utilizes the Frank–Wolfe algorithm over the cone of valid inequalities of K and subgradients of f. Under the assumption that f is L-Lipschitz and that K contains a ball of radius r and is contained inside the origin centered ball of radius R, using $O(\frac{(RL)^2}{\varepsilon^2} \cdot \frac{R^2}{r^2})$ iterations and calls to the oracle, our main method outputs a point $x \in K$ satisfying $f(x) \leq \varepsilon + \min_{z \in K} f(z)$. Our algorithm is easy to implement, and we believe it can serve as a useful alternative to existing cutting plane methods. As evidence towards this, we show that it compares favorably in terms of iteration counts to the standard LP based cutting plane method and the analytic center cutting plane method, on a testbed of combinatorial, semidefinite and machine learning instances.

Keywords: convex optimization · separation oracle · cutting plane method

1 Introduction

We consider the problem of minimizing a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ over a compact convex set $K \subseteq \mathbb{R}^n$. We assume that K contains an (unknown) Euclidean ball of radius $r > 0$ and is contained inside the origin centered ball of radius $R > 0$, and that f is L-Lipschitz. We have first-order access to f that yields $f(x)$ and a subgradient of f at x for any given x. Moreover, we only have access to K through a separation oracle (SO), which, given a point $x \in \mathbb{R}^n$, either asserts that $x \in K$ or returns a linear constraint valid for K but violated by x.

Convex optimization in the SO model is one of the fundamental settings in optimization. The model is relevant for a wide variety of implicit optimization problems, where an explicit description of the defining inequalities for K

⋆ This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement QIP–805241)

is either too large to store or not fully known. The SO model was first introduced in [28] where it was shown that an additive ε-approximate solution can be obtained using $O(n \log(LR/(\varepsilon r)))$ queries via the center of gravity method and $O(n^2 \log(LR/(\varepsilon r)))$ queries via the ellipsoid method. This latter result was used by Khachiyan [26] to give the first polynomial time method for linear programming. The study of oracle-type models was greatly extended in the classic book of Grötschel, Lovász, and Schrijver [22], where many applications to combinatorial optimization were provided. Further progress on the SO model was given by Vaidya [35], who showed that the $O(n \log(LR/(\varepsilon r)))$ oracle complexity can be efficiently achieved using the so-called volumetric barrier as a potential function, where the best current running time for such methods was given very recently [27,24].

From the practical perspective, two of the most popular methods in the SO model are the standard linear programming (LP) based cutting plane method, independently discovered by Kelley [25], Goldstein-Cheney [9] as well as Gomory [21] (in the integer programming context), and the analytic center cutting plane method [33] (ACCPM).

The LP based cutting plane method, which we henceforth dub the standard cut loop, proceeds as follows: starting with finitely many linear underestimators of f and linear constraints valid for K, in each iteration it solves a linear program that minimizes the lower envelope of f subject to the current linear relaxation of K. The resulting point x is then used to query f and the SO to obtain a new underestimator for f and a new constraint valid for K. Note that if f is a linear function, it repeatedly minimizes f over linear relaxations of K. While it is typically fast in practice, it can be unstable, and no general quantitative convergence guarantees are known for the standard cut loop.

To link to integer programming, in that context K is the convex hull of integer points of some polytope P and the objective is often linear, and the method is initialized with a linear description of P. A crucial difference there is that the separator SO is generally only efficient when queried at vertices of the current relaxation.

ACCPM is a barrier based method, in which the next query point is the minimizer of the barrier for the current inequalities in the system. ACCPM is in general a more stable method with provable complexity guarantees. Interestingly, while variants of ACCPM achieving $O(n^2 \log(1/\varepsilon))$ exist, achieved by judiciously dropping constraints [1], the more practical variants achieve only $O(n/\varepsilon^2)$ convergence [29].

In this paper, we describe a new method for convex optimization in the SO model that computes an additive ε-approximate solution within $O(R^4 L^2/r^2\varepsilon^2)$ iterations. Our algorithm is easy to implement, and we believe it can serve as a useful alternative to existing methods. In our experimental results, we show that it compares favorably in terms of iteration counts to the standard cut loop and the analytic center cutting plane method, on a testbed of combinatorial, semidefinite and machine learning instances.

Before explaining our approach, we review the relevant work in related models. To begin, there has been a tremendous amount of work in the context of first-order methods [5,3], where the goal is to minimize a possibly complicated function, given by a gradient oracle, over a *simple domain* K (e.g., the simplex, cube, ℓ_2 ball). These methods tend to have cheap iterations and to achieve $\text{poly}(1/\varepsilon)$ convergence rates. They are often superior in practice when the requisite accuracy is low or moderate, e.g., within 1% of optimal. For these methods, often variants of (sub-)gradient descent, it is generally assumed that computing (Euclidean) projections onto K as well as linear optimization over K are easy. If one only assumes access to a linear optimization (LO) oracle on K, K can become more interesting (e.g., the shortest-path or spanning-tree polytope). In this context, one of the most popular methods is the so-called *Frank–Wolfe* algorithm [18] (see [23] for a modern treatment), which iteratively computes a convex combination of vertices of K to obtain an approximate minimizer of a smooth convex function.

In the context of combinatorial optimization, there has been a considerable line of work on solving (implicit) packing and covering problems using the so-called multiplicative weights update (MWU) framework [32,30,19]. In this framework, one must be able to implement an MWU oracle, which in essence computes optimal solutions for the target problem after the "difficult" constraints have been aggregated according to the current weights. This framework has been applied for getting fast $(1 \pm \varepsilon)$-approximate solutions to multicommodity flow [32,19], packing spanning trees [8], the Held–Karp approximation for TSP [7], and more, where the MWU oracle computes shortest paths, minimum cost spanning trees, minimum cuts respectively in a sequence of weighted graphs. The MWU oracle is in general just a special type of LO oracle, which can often be interpreted as a SO that returns a maximally violated constraint. While certainly related to the SO model, it is not entirely clear how to adapt MWU to work with a general SO, in particular in settings unrelated to packing and covering.

A final line of work, which directly inspires our work, has examined simple iterative methods for computing a point in the interior of a cone $\mathcal{I}$ that directly apply in the SO model. The application of simple iterative methods for solving conic feasibility problems can be traced to Von Neumann in 1948 (see [14]), and a variant of this method, the perceptron algorithm [31] is still very popular today. Von Neumann's algorithm computes a convex combination of the defining inequalities of the cone, scaled to be of unit length, of nearly minimal Euclidean norm. The separation oracle is called to find an inequality violated by the current convex combination, and this inequality is then used to make current convex combination shorter, in an analogous way to Frank–Wolfe. This method is guaranteed to find a point in the cone in $O(1/\rho^2)$ iterations, where ρ is the so-called width of $\mathcal{I}$ (the radius of the largest ball contained in $\mathcal{I}$ centered at a point of norm 1). Starting in 2004, polynomial time variants of this and related methods (i.e., achieving $\log 1/\rho$ dependence) have been found [6,16,10], which iteratively "rescale" the norm to speed up the convergence. These rescaled variants can

also be applied in the oracle setting [4,11,13] with appropriate adaptations. The main shortcoming of existing conic approaches is that they are currently not well-adapted for solving optimization problems rather then feasibility problems.

Our approach. In this work, we build upon von Neumann's approach and utilize the Frank–Wolfe algorithm over the cone of valid inequalities of K as well as the subgradients of f in a way that yields a clean, simple, and flexible framework for solving general convex optimization problems in the SO model. For simpler explanation, let us assume that $f(x) = \langle c, x\rangle$ is a linear function and that we know an upper bound UB on the minimum of f over K. Given some linear inequalities $\langle a_i, x\rangle \leq b_i$ that are valid for all $x \in K$, our goal is to find convex combinations p of the *homogenized* points (c, UB) and (a_i, b_i) that are "close" to the origin. Note that if $p = \mathbf{0}$, the fact that K is full-dimensional implies that (c, UB) appears with a nonzero coefficient and hence $(-c, -\mathrm{UB})$ is a nonnegative combination of the points (a_i, b_i), which in turn shows that UB is equal to the minimum of f over K. In view of this, we will consider a potential $\Phi\colon \mathbb{R}^{n+1} \to \mathbb{R}_+$ with the property that if $\Phi(p)$ is sufficiently small, then the convex combination will yield an explicit certificate that UB is close to the minimum of f over K.

Given a certain convex combination p, note that the gradient of Φ at p provides information about whether moving towards one of the known points will (significantly) decrease $\Phi(p)$. However, if no such known point exists, it turns out that the "dehomogenization" of the gradient (a scaling of its projection onto the first n coordinates) is a natural point $x \in \mathbb{R}^n$ to query the SO with. In fact, if $x \in K$, it will have improved objective value with respect to f. Otherwise, the SO will provide a linear inequality such that moving towards its homogenization decreases $\Phi(p)$.

In this work, we will show that the above paradigm immediately yields a rigorous algorithm for various natural choices of Φ and scalings of inequalities. We will also see that general convex functions can be directly handled in the same manner by simply replacing (c, UB) with all subgradient cuts of f learned throughout the iterations. The same applies to pure feasibility problems for which we set $f = \mathbf{0}$. The convergence analysis of our algorithm is simple and based on standard estimates for the Frank–Wolfe algorithm.

Besides its conceptual simplicity and distiction to existing methods for convex optimization in the SO model, we also regard it as a practical alternative. In fact, in terms of iterations, our vanilla implementation in `Julia`[4] performs similarly and often even better than the standard cut loop and the analytic center cutting plane method evaluated on a testbed of oracle-based linear optimization problems for matching problems, semidefinite relaxations of the maximum cut problem, and LPBoost. Moreover, the flexibility of our framework leaves several degrees of freedom to obtain optimized implementations that outperform our naive implementation.

Acknowledgments We would like to thank Robert Luce and Sebastian Pokutta for their very valuable feedback on our work.

[4] https://github.com/christopherhojny/supplement_simple-iterative-methods-linopt-convex-sets

2 Algorithm

Recall that we are given first-order access to a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ that we want to minimize over a convex body $K \subseteq \mathbb{R}^n$. In the case where f is not differentiable, with a slight abuse of notation we interpret $\nabla f(x)$ to be *any* subgradient of f at x. We can access K by a separation oracle that, given a point $x \in \mathbb{R}^n$, either asserts that $x \in K$ or returns a point $(a, b) \in \mathcal{A} \subseteq \mathbb{R}^{n+1}$ with $\langle a, x\rangle > b$ such that $\langle a, y\rangle \leq b$ holds for all $y \in K$. Here, $\langle \cdot, \cdot \rangle$ denotes the standard scalar product and we assume that all points in $\mathcal{A}$ correspond to linear constraints valid for K. To state our algorithm, let $\|\cdot\|$ denote any norm on $\mathbb{R}^{n+1}$ and $\|\cdot\|_*$ its dual norm. Moreover, let $\Phi\colon \mathbb{R}^{n+1} \to \mathbb{R}_+$ be any strictly convex and differentiable function with $\min_{x\in\mathbb{R}^{n+1}} \Phi(x) = \Phi(0) = 0$. Our method is given in Algorithm 1, in which we denote the number of iterations by T for later reference. However, T does not need to be specified in advance, and the algorithm may be stopped at any time, e.g., when a solution or bound of desired accuracy has been found.

Algorithm 1

1: UB $\leftarrow \infty$, $A_1 \leftarrow \{(\mathbf{0}, 1)/\|(\mathbf{0}, 1)\|_*\}$, $G_1 \leftarrow \emptyset$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad p_t \leftarrow \arg\min\{\Phi(p) : p \in \operatorname{conv}(A_t \cup G_t)\}$
4: $\quad$ **if** $p_t = \mathbf{0}$ **then return** UB.
5: $\quad x_t \leftarrow -\nabla\Phi(p_t)[1:n]/\nabla\Phi(p_t)[n+1]$
6: $\quad$ **if** $x_t \in K$ **then**
7: $\quad\quad$ UB $\leftarrow \min\{\text{UB}, f(x_t)\}$
8: $\quad\quad A_{t+1} \leftarrow A_t$.
9: $\quad\quad G_{t+1} \leftarrow G_t \cup \{(\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle)\}$
10: $\quad$ **else**
11: $\quad\quad$ get $(a, b) \in \mathcal{A}$, with $\langle a, x_t\rangle > b$ and $\|(a, b)\|_* = 1$
12: $\quad\quad A_{t+1} \leftarrow A_t \cup \{(a, b)\}$.
13: $\quad\quad G_{t+1} \leftarrow G_t$.
14: **return** UB.

In Line 5, $\nabla\Phi(p_t)[1:n]$ denotes the first n components of $\nabla\Phi(p_t)$, and $\nabla\Phi(p_t)[n+1]$ denotes the last component of $\nabla\Phi(p_t)$. The sets A_t and G_t denote the already known/separated inequalities and objective gradients during iteration t.

Lemma 1. *When $x_t \in \mathbb{R}^n$ is computed in iteration t of Algorithm 1, it is well-defined and we have $\langle c, x_t\rangle \leq d$ for every $(c, d) \in A_t \cup G_t$.*

Proof. Since p_t minimizes Φ over $\operatorname{conv}(A_t \cup G_t)$, for every $q \in \operatorname{conv}(A_t \cup G_t)$ we have $\langle \nabla\Phi(p_t), q - p_t\rangle \geq 0$. If $p_t \neq \mathbf{0}$ then from strict convexity of Φ and

$\min_{x\in\mathbb{R}^{n+1}} \Phi(x) = \Phi(\mathbf{0}) = 0$ we get

$$\langle \nabla\Phi(p_t), q\rangle \geq \langle \nabla\Phi(p_t), p_t\rangle > 0. \tag{1}$$

First, apply this inequality to $q = (\mathbf{0},1)/\|(\mathbf{0},1)\|_* \in A_t$ and conclude $\nabla\Phi(p_t)[n+1] > 0$. This makes sure that x_t can be computed. Second, we apply (1) to $q = (c,d) \in A_t \cup G_t$ and find that $d - \langle c, x_t\rangle = \frac{1}{\nabla\Phi(p_t)[n+1]}\langle \nabla\Phi(p_t), (c,d)\rangle > 0$, thus x_t satisfies $\langle c, x_t\rangle \leq d$ for all $(c,d) \in A_t \cup G_t$. □

Note that, for the sake of presentation, in Line 3 we require p_t to be the convex combination of minimum Φ-value. However, it is usually not necessary to compute such a minimum. The same convergence rates can be obtained if, in every iteration, p_t is a suitable convex combination of p_{t-1} and some $(c,d) \in A_t \cup G_t$ with $\langle \nabla\Phi(p_{t-1}), (c,d)\rangle < 0$. If the last coordinate of p_{t-1}, as discussed in the above proof, is not positive, then such an update can be made towards $(\mathbf{0},1)/\|(\mathbf{0},1)\|_* \in A_t$. Any such update will significantly decrease $\Phi(p_t)$, and the computation in Line 3 is guaranteed to make at least that much progress. This shows that simple updates of p_t, which may be more preferable in practice, still suffice to achieve the claimed convergence rates.

Lemma 2. *Suppose that Φ is 1-smooth with respect to $\|\cdot\|_*$ and that*

$$\|(\nabla f(x), \langle \nabla f(x), x\rangle)\|_* \leq 1$$

for every $x \in K$. Then for every $t = 1,\dots,T$, Algorithm 1 satisfies $\Phi(p_t) \leq \frac{8}{t+2}$.

Recall that Φ is 1-smooth with respect to $\|\cdot\|_*$ if $\|\nabla\Phi(x) - \nabla\Phi(y)\| \leq \|x-y\|_*$ holds for all $x, y \in \mathbb{R}^{n+1}$. The proof of the above lemma is in line with standard proofs for the analysis of Frank–Wolfe algorithms, see, e.g., Theorem 1 in [23]. For this reason, the proof is deferred to a later version.

The following lemma yields conditions under which a small value of $\Phi(p_t)$ implies that UB is close to the minimum of f over K. Note in particular that it proves that if $\|p_t\| = 0$ then UB = OPT.

Lemma 3. *Assume that $\|(x,-1)\| \leq 2$ holds for every $x \in K$, and there exist $z \in K$ and $\alpha \in (0,1]$ such that $\langle (a,b), (-z,1)\rangle \geq \alpha\|(-z,1)\|\|(a,b)\|_*$ holds for every $(a,b) \in \mathcal{A} \cup \{(\mathbf{0},1)\}$. Moreover, assume that $\|(\nabla f(x), \langle \nabla f(x), x\rangle)\|_* \leq 1$ holds for every $x \in K$. If $\|p_T\| \leq \alpha/2$ in Algorithm 1, then the returned value satisfies* $\mathrm{UB} \geq \mathrm{OPT} \geq \mathrm{UB} - \frac{4\|p_T\|_*(1+\alpha)}{\alpha}$.

Proof. Let $x^* \in K$ minimize $f(x)$ over $x \in K$ and let $F \subset [T-1]$ be the set of iterations (except the last one) in which $x_t \in K$. Now write the point p_T as a convex combination

$$p_T = \sum_{(a,b)\in A_T} \lambda_{(a,b)}(a,b) + \sum_{t\in F} \gamma_t(\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle)$$

where $\lambda \geq 0, \gamma \geq 0$ and $\|(\lambda, \gamma)\|_1 = 1$. Then we have

$$\begin{aligned}
\sum_{t\in F} \gamma_t(f(x_t) - f(x^*)) &\leq \sum_{t\in F} \gamma_t \langle \nabla f(x_t), x_t - x^* \rangle \\
&= \Big\langle \sum_{t\in F} \gamma_t(\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle),\ (-x^*, 1) \Big\rangle \\
&\leq \Big\langle \sum_{t\in F} \gamma_t(\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle) + \sum_{(a,b)\in A_T} \lambda_{(a,b)}(a,b),\ (-x^*, 1) \Big\rangle \\
&= \langle p_T, (-x^*, 1)\rangle \\
&\leq \|p_T\|_* \cdot \|(-x^*, 1)\| \leq 2\|p_T\|_*.
\end{aligned}$$

Here, the inequalities respectively arise from convexity of f, that $x^* \in K$ satisfies $\langle (a,b), (-x^*, 1)\rangle \geq 0$ for every $(a,b) \in A_T$, and the Cauchy–Schwarz inequality. In particular, we find that $\min_{t\in F} f(x_t) - f(x^*) \leq \frac{2\|p_T\|_*}{\sum_{t\in F}\gamma_t}$ whenever $\sum_{t\in F}\gamma_t > 0$. To lower bound this latter quantity, we use the assumptions on z to derive the inequalities

$$\begin{aligned}
\alpha\left(1 - \sum_{t\in F}\gamma_t\right)\|(-z,1)\| &= \alpha\|(-z,1)\| \sum_{(a,b)\in A_T} \lambda_{(a,b)} \\
&\leq \langle \sum_{(a,b)\in A_T} \lambda_{(a,b)}(a,b), (-z,1)\rangle \quad (\text{ since } \|(a,b)\|_* = 1\ ) \\
&= \langle p_T, (-z,1)\rangle - \sum_{t\in F} \gamma_t \langle (\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle),\ (-z,1)\rangle \\
&\leq \|p_T\|_* \cdot \|(-z,1)\| + \sum_{t\in F} \gamma_t \|(\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle)\|_* \cdot \|(-z,1)\|.
\end{aligned}$$

Now observe that $\|(\nabla f(x_t), \langle \nabla f(x_t), x_t\rangle)\|_* \leq 1$ for every $t \in F$ and divide through by $\|(-z,1)\|$ to find $\alpha(1 - \sum_{t\in F}\gamma_t) \leq \|p_T\|_* + \sum_{t\in F}\gamma_t$. Hence, if $\|p_T\|_* \leq \frac{\alpha}{2}$ then $\alpha/2 \leq (\alpha+1)\sum_{t\in F}\gamma_t$. This lower bound on $\sum_{t\in F}\gamma_t$ suffices to prove the lemma. □

Combining the previous two lemmas, we obtain the following convergence rate of our algorithm:

Theorem 1. *Assume that $\beta > 0$ is such that $\Phi(x) \geq \beta\|x\|_*^2$ for all $x \in \mathbb{R}^{n+1}$. Under the assumptions of Lemmas 2 and 3, Algorithm 1 computes, for every $T \geq \frac{32}{\beta\alpha^2}$, a value $\mathrm{UB} < \infty$ satisfying $\mathrm{UB} \geq \min_{x\in K} f(x) \geq \mathrm{UB} - \frac{16}{\sqrt{\beta(T+2)}} \cdot \frac{1+\alpha}{\alpha}$.*

Proof. After T iterations, we have $\beta\|p_T\|_*^2 \leq \Phi(p_T) \leq \frac{8}{T+2} \leq \beta\alpha^2/4$ per Lemma 2. Since then $\|p_T\|_* \leq \frac{\sqrt{8}}{\sqrt{\beta(T+2)}} \leq \alpha/2$, Lemma 3 tells us that $\mathrm{OPT} \geq \mathrm{UB} - \frac{16(1+\alpha)}{\sqrt{\beta(T+2)}\alpha}$. □

Let us now apply the previous findings to a concrete setting, in which we assume that the objective function f is L-Lipschitz, i.e., $|f(x)-f(y)| \le L\|x-y\|_2$ for all $x, y \in \mathbb{R}^n$.

Theorem 2. *Let $K \subset \mathbb{R}^n$ be a convex body satisfying $z + r\mathbb{B}_2^n \subset K \subset R\mathbb{B}_2^n$, given by a separation oracle $\mathcal{A}$, and let $f\colon \mathbb{R}^n \to \mathbb{R}$ be an L-Lipschitz convex function given by a subgradient oracle.*

Apply Algorithm 1 to the function $\frac{1}{RL}f$ using norm $\|(x,y)\| \coloneqq \sqrt{2}\|(x/R, y)\|_2$ and potential $\Phi(a,b) \coloneqq \frac{1}{4}\|(Ra,b)\|_2^2$. Then, for every $\varepsilon > 0$, after

$$T = O\left(\frac{R^2}{r^2} \cdot \frac{R^2L^2}{\varepsilon^2}\right)$$

iterations we have $\mathrm{UB} \ge \min_{x\in K} f(x) \ge \mathrm{UB} - \varepsilon$.

Proof. By replacing $f(x)$ by $f(Rx)/(RL)$, K by K/R, ε by $\varepsilon/(RL)$, r by r/R, we may assume that $R = L = 1$, that $r \in (0,1]$. After this rescaling, note $\|(x,y)\| \coloneqq \sqrt{2}\|(x,y)\|_2$ and $\Phi(a,b) \coloneqq \frac{1}{4}\|(a,b)\|_2^2 = \frac{1}{2}\|(a,b)\|_*^2$. Crucially, note that Algorithm 1 is invariant under the above replacement.

We now claim that our choice of input satisfies the conditions of Theorem 1 with $\beta = 1/2$ and $\alpha = r/4$. Given the claim, Theorem 1 directly proves the result. To prove the claim, apart from verifying that the bounds on β and α hold, we must verify smoothness of Φ with respect to the dual norm, a bound of 2 on the norm of $(-x, 1)$ for $x \in K$, as well as a dual norm bound of 1 on $(\nabla f(x), \langle \nabla f(x), x\rangle)$ for $x \in K$.

The setting $\beta = 1/2$ is direct by definition of Φ. Since $\|\cdot\|_*$ is a Euclidean norm, it is immediate that Φ is 1-smooth with respect to $\|\cdot\|_*$. For each $x \in K$, using that $R = L = 1$, we may also verify that

$$\|(x,1)\| = \sqrt{2}\|(x,1)\|_2 = \sqrt{2}\sqrt{\|x\|_2^2 + 1} \le \sqrt{2}\sqrt{R^2+1} = 2,$$

and

$$\begin{aligned}\|(\nabla f(x), \langle \nabla f(x), x\rangle)\|_* &= \frac{1}{\sqrt{2}}\|(\nabla f(x), \langle \nabla f(x), x\rangle)\|_2 \\ &\le \frac{1}{\sqrt{2}}\sqrt{\|\nabla f(x)\|_2^2 + \|\nabla f(x)\|^2\|x\|^2} \\ &\le \frac{1}{\sqrt{2}}\sqrt{L^2 + L^2R^2} = 1.\end{aligned}$$

We now show the lower bound $\alpha \ge r/4$. Firstly, since $\|(-z,1)\|\|(\mathbf{0},1)\|_* = \|(-z,1)\|_2\|(\mathbf{0},1)\|_2 \le \sqrt{2}$, we see that $\langle(-z,1),(\mathbf{0},1)\rangle = 1 \ge \frac{1}{2}\|(-z,1)\|\|(\mathbf{0},1)\|_*$. Next, any (a,b) returned by the oracle is normalized so that $\|(a,b)\|_* = 1 \Leftrightarrow \|(a,b)\|_2 = \sqrt{2}$. Note then that $\|(-z,1)\|\|(a,b)\|_* \le 2$. From here, we observe that

$$\langle(a,b),(-z,1)\rangle = b - \langle a, z\rangle = b - \langle a, z + ra/\|a\|_2\rangle + r\|a\|_2 \ge r\|a\|_2,$$

since $z+ra/\|a\|_2 \in K$ by assumption. Furthermore, $b-\langle a,z\rangle \geq b-\|a\|_2\|z\|_2 \geq b-\|a\|_2$ and $0 \leq b-\langle a,z\rangle \leq b+\|a\|_2$. Thus, $b-\langle a,z\rangle \geq \max\{r\|a\|_2, b-\|a\|_2\}$. We now examine two cases. If $\|a\|_2 \geq 1/2$, then $b-\langle a,z\rangle \geq r/2 \geq r/4 \cdot \|(-z,1)\|\|(a,b)\|_*$. If $\|a\|_2 \leq 1/2$, then $|b| \geq 1$ since $\|(a,b)\|_2^2 = 2$. Since $b+\|a\|_2 \geq 0 \Rightarrow b \geq 1$. This gives $b-\langle a,z\rangle \geq b-\|a\|_2 \geq 1/2 \geq r/2$. Thus, $\alpha \geq r/4$, as needed. □

3 Computational experiments

In this section, we provide a computational comparison of our method with the standard cut loop, the ellipsoid method, and the analytic center cutting plane method on a testbed of linear optimization instances. For comparison purposes, all four methods are embedded into a common cutting plane framework such that the same termination criteria apply.

Framework. Each method has access to a separation oracle that is equipped with a set of initial linear inequalities valid for K (such as bounds on variables), which are incorporated within each method in a straightforward way. For instance, we initialize our algorithm by adding these constraints to the set A_1. Moreover, for each instance, we will be given a finite upper bound UB and incorporate the linear inequality $f(x) \leq \text{UB}$ in a similar way. This upper bound gets updated whenever a feasible solution of better objective value was found. Our framework collects all inequalities queried by the current method and computes the resulting lower bound on the optimum value in every iteration. Each method is stopped whenever the difference of upper and lower bound is below 10^{-3}.

We will also inspect the possibility of a *smart* oracle that, regardless of whether a given point x is feasible, may still provide a valid inequality as well as a feasible solution (for instance, by modifying x in a simple way so that it becomes feasible). For some problems we consider, such an oracle is available and will be specified below.

Implementation. The framework has been implemented in `julia 1.6.2` using `JuMP` and `Gurobi 9.1.1`. To guarantee a fair comparison, all four methods have been implemented in a straightforward fashion. We use the textbook implementation of the ellipsoid method, and Badenbroek's [2]implementation of the analytic center cutting plane method. Our method is implemented [5] in the spirit of Theorem 2, where p_t is computed using `Gurobi`.

Test sets. We use three problem classes in our experiments: linear programming formulations of the maximum-cardinality matching problem, semidefinite relaxations of the maximum cut problem, and LPBoost instances for classification problems.

[5] https://github.com/christopherhojny/supplement_simple-iterative-methods-linopt-convex-sets

For the maximum-cardinality matching problem, we consider the linear program

$$\max \Big\{ \sum\nolimits_{e\in E} x_e : x \in [0,1]^E,\ \sum\nolimits_{e\in\delta(v)} x_e \le 1 \text{ for all } v \in V,\ \sum\nolimits_{e\in E[U]} x_e \le \tfrac{|U|-1}{2} \text{ for all } U \subseteq V \text{ with } |U| \text{ odd} \Big\},$$

due to Edmonds [17], where $G = (V, E)$ is a given undirected graph, $\delta(v)$ is the set of all edges incident to v, and $E[U]$ is the set of all edges with both endpoints in U. The latter constraints are handled within an oracle that computes an inequality minimizing $(|U| - 1)/2 - \sum_{e\in E[U]} x_e$, whereas the other inequalities are provided as initial constraints. For the above problem, the smart version of the oracle does not provide a feasible point since there is no obvious way of transforming a given point into a feasible one. However, the smart version always provides the minimizing inequality.

We consider 16 random instances with 500 nodes, generated as follows. For each $r \in \{30, 33, \dots, 75\}$ we build an instance by sampling r triples of nodes $\{u, v, w\}$ and adding the edges of the induced triangles to the graph. We believe that these instances are interesting because the r triangles give rise to many constraints to be added by the oracle. Moreover, we selected all 13 instances from the Color02 symposium [12] with less than 300 edges.

Our second set of instances is based on the semidefinite relaxation of Goemans and Williamson [20] for the maximum cut problem

$$\max \Big\{ \sum\nolimits_{\{v,w\}\in E} c(v,w)(1 - X_{v,w})/2 : X_{v,w} = X_{w,v} \text{ for all } v, w \in V,\ X_{v,v} = 1 \text{ for all } v \in V,\ X \text{ is positive semidefinite} \Big\},$$

where c are edge weights on the edges of (V, E). We add the constraints $X \in [-1, 1]^{V\times V}$ to the initial constraints and handle the semidefiniteness constraint by a separation oracle that, given X, computes an eigenvector h of X of minimum eigenvalue and returns the inequality $\langle hh^\intercal, X\rangle \ge 0$.

Within the smart version of the oracle, this constraint is returned regardless of the feasibility of X. If X is not feasible, the semidefinite matrix $\frac{1}{\lambda-1}X - \frac{\lambda}{\lambda-1}I$ is returned, where λ denotes the minimum eigenvalue and I the identity matrix. We generated 10 complete graphs with edge weights chosen uniformly at random in $[0, 1]$.

Our third set of instances arises from LPBoost [15], a classifier algorithm based on column generation. To solve the pricing problem in column generation, the following linear program is solved:

$$\max \Big\{ \gamma : (\gamma, \lambda) \in [-1, 1] \times [0, D]^n, \langle \mathbf{1}, \lambda\rangle = 1, \sum_{i=1}^{m} y_i h(x^i, \omega)\lambda_i \le -\gamma \text{ for } \omega \in \Omega \Big\},$$

where Ω is a set of parameters, for $i \in [m]$, x^i is a data point labeled as $y_i = \pm 1$, $h(\cdot, \omega)$ is a classifier parameterized by $\omega \in \Omega$ that predicts the label of x^i

Table 1: Comparison of iterations and dual/primal integral without smart oracles.

	#iterations				dual integral				primal integral		
instance	LP	ellipsoid	analytic	our	LP	ellipsoid	analytic	our	ellipsoid	analytic	our
matching	175.44	500.00	500.00	99.81	48.34	473.02	22.13	21.10	52.12	9.29	4.40
matching02	283.77	460.77	491.69	47.15	257.76	339.67	194.26	21.64	23.41	5.91	2.13
maxcut	265.30	500.00	500.00	193.30	7.72	44.32	3.48	6.14	21.15	9.04	6.32
LPboost	91.94	489.06	479.12	278.06	3.15	13.62	20.65	53.15	459.97	100.71	64.08

as $h(x^i, \omega) \in \{-1, +1\}$, and $D > 0$ is a parameter. In our experiments, we restrict $h(\cdot, \omega)$ to be a decision tree of height 1, so-called tree stumps, and choose $D = \frac{5}{n}$. To separate a point (γ', λ'), we use `julia`'s `DecisionTree` module to compute a decision stump with score function λ' that weights the data points, whose corresponding inequality classifies (γ', λ') as feasible or not. A smart oracle always returns the computed inequality and decreases γ' until (γ', λ') becomes feasible according to the found decision stump.

We extracted all data sets from the UC Irvine Machine Learning Repository [34] that are labeled as multivariate, classification, ten-to-hundred attributes, hundred-to-thousand instances. Data sets with alpha-numeric values or too many missing values have been discarded.

Results. In what follows, we report on the number of iterations, i.e., oracle calls, each method needs to obtain a gap (upper bound minus lower bound) below 10^{-3}. We impose a limit of 500 iterations per instance. Since we are testing naive implementations of each method, we do not report on running time.

To get more insights on the primal and dual performance of the tested methods, we also report on their *primal and dual integrals*. Note that we are solving maximization problems in this section, as opposed to minimization problems in Section 2. That is, primal (dual) solutions provide lower (upper) bounds on OPT. If ℓ_i is the lower bound on the optimal objective value OPT in iteration i, the *primal integral* is $\sum_{i=1}^{500} \frac{\text{OPT}-\ell_i}{\text{OPT}-\ell_1}$. The *dual integral* is computed analogously. If an integral is small, this indicates quick progress in finding the correct value of the corresponding bound.

Table 1 summarizes our results without smart oracles, where all numbers are average values. Here, "matching" refers to the random instances and "matching02" to the instances from the Color02 symposium. The standard cut loop is referred to as "LP", the ellipsoid method as "ellipsoid", the analytic center method as "analytic", and Algorithm 1 as "our". Note that Table 1 does not report on the primal integral of "LP" since the standard cut loop is a dual method.

We see that the ellipsoid and analytic center methods are struggling with solving any instance within 500 iterations independent from the problem class. Our algorithm solves the instances of the matching and max-cut problem much faster than the standard cut loop. Only for LPBoost, the standard cut loop

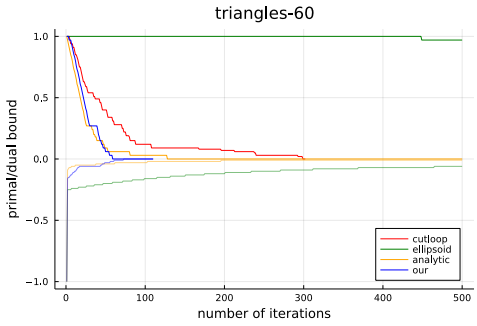

Fig. 1: Typical primal/dual bounds for a random matching instance.

clearly dominates our algorithm. To better understand this behavior, the integrals reveal that our algorithm is better in improving the primal bound than the dual bound, with the only exception being LPBoost. The analytic center method, however, performs significantly worse than our algorithm in improving the primal bound. Regarding the dual bound, it performs better than our algorithm (with the exception of matching02). The ellipsoid method is much worse in improving the primal bound in comparison with the analytic center method and our algorithm. Regarding the dual bound, a similar trend can be observed with LPBoost being an exception.

In summary, the analytic center cutting plane method improves the dual bound more quickly than our algorithm. It can find a good primal solution early as the primal integral is small, however it fails to close the remaining gap within the iteration limit. Our algorithm is able to close the primal gap faster, with the trade-off of a slightly slower dual convergence. A typical plot of the of the relative primal and dual gaps is given in Figure 1.

In a second experiment, we investigate the effect of smart oracles. As Table 2 shows, there is no impact of smart oracles on the matching instances. For maxcut, our algorithm gets slightly slower and the other methods do not seem to be affected by smartness. For LPBoost, all methods benefit from a smart oracle with the biggest effect for analytic center and our algorithm. The reason for the positive effect for LPBoost might be in the particular structure of these instances: the objective just consists of γ and every truncated convex combination λ is feasible.

Table 2: Comparison of iterations and dual/primal integral with smart oracles.

	#iterations				dual integral				primal integral		
instance	LP	ellipsoid	analytic	our	LP	ellipsoid	analytic	our	ellipsoid	analytic	our
matching	175.44	500.00	500.00	99.81	48.34	473.02	22.13	21.10	52.12	9.29	4.40
matching02	283.77	460.77	491.69	47.15	257.76	339.67	194.26	21.64	23.41	5.91	2.13
maxcut	265.30	500.00	500.00	231.00	7.72	42.90	3.48	6.15	20.42	8.91	5.59
LPboost	86.94	346.38	88.00	127.00	3.04	13.50	5.54	5.46	25.41	6.83	6.95

References

1. Atkinson, D.S., Vaidya, P.M.: A cutting plane algorithm for convex programming that uses analytic centers. Mathematical Programming **69**(1), 1–43 (1995)
2. Badenbroek, R., de Klerk, E.: An analytic center cutting plane method to determine complete positivity of a matrix (2020)
3. Beck, A.: First-Order Methods in Optimization. Society for Industrial and Applied Mathematics (Oct 2017). https://doi.org/10.1137/1.9781611974997, https://doi.org/10.1137/1.9781611974997
4. Belloni, A., Freund, R.M., Vempala, S.: An efficient rescaled perceptron algorithm for conic systems. Mathematics of Operations Research **34**(3), 621–641 (2009)
5. Ben-Tal, A., Nemirovski, A.: Lectures on Modern Convex Optimization. Society for Industrial and Applied Mathematics (Jan 2001). https://doi.org/10.1137/1.9780898718829, https://doi.org/10.1137/1.9780898718829
6. Betke, U.: Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem. Discrete & Computational Geometry **32**(3) (May 2004). https://doi.org/10.1007/s00454-004-2878-4, https://doi.org/10.1007/s00454-004-2878-4
7. Chekuri, C., Quanrud, K.: Approximating the held-karp bound for metric TSP in nearly-linear time. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). IEEE (Oct 2017). https://doi.org/10.1109/focs.2017.78, https://doi.org/10.1109/focs.2017.78
8. Chekuri, C., Quanrud, K.: Near-linear time approximation schemes for some implicit fractional packing problems. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics (Jan 2017). https://doi.org/10.1137/1.9781611974782.51, https://doi.org/10.1137/1.9781611974782.51
9. Cheney, E.W., Goldstein, A.A.: Newton's method for convex programming and tchebycheff approximation. Numerische Mathematik **1**(1), 253–268 (1959)
10. Chubanov, S.: A strongly polynomial algorithm for linear systems having a binary solution. Mathematical Programming **134**(2), 533–570 (Feb 2011). https://doi.org/10.1007/s10107-011-0445-3, https://doi.org/10.1007/s10107-011-0445-3
11. Chubanov, S.: A polynomial algorithm for linear feasibility problems given by separation oracles. Optimization Online, Jan (2017)
12. Color02 - computational symposium: Graph coloring and its generalizations. available at (2002), http://mat.gsia.cmu.edu/COLOR02
13. Dadush, D., Végh, L.A., Zambelli, G.: Rescaling algorithms for linear conic feasibility. Mathematics of Operations Research **45**(2), 732–754 (May 2020). https://doi.org/10.1287/moor.2019.1011, https://doi.org/10.1287/moor.2019.1011
14. Dantzig, G.B.: Converting a converging algorithm into a polynomially bounded algorithm. Tech. rep., Technical report, Stanford University, 1992. 5.6, 6.1, 6.5 (1991)
15. Demiriz, A., Bennett, K.P., Shawe-Taylor, J.: Linear programming boosting via column generation. Machine Learning **46**(1), 225–254 (2002)
16. Dunagan, J., Vempala, S.: A simple polynomial-time rescaling algorithm for solving linear programs. Mathematical Programming **114**(1), 101–114 (Feb 2007). https://doi.org/10.1007/s10107-007-0095-7, https://doi.org/10.1007/s10107-007-0095-7

17. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. Journal of Research of the National Bureau of Standards **69B**(1–2), 125–130 (1964)
18. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval research logistics quarterly **3**(1-2), 95–110 (1956)
19. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM Journal on Computing **37**(2), 630–652 (Jan 2007). https://doi.org/10.1137/s0097539704446232, https://doi.org/10.1137/s0097539704446232
20. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115—-1145 (1995). https://doi.org/10.1145/227683.227684, https://doi.org/10.1145/227683.227684
21. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Amer. Math. Soc. **64**, 275–278 (1958)
22. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization, vol. 2. Springer-Verlag (1988). https://doi.org/10.1007/978-3-642-78240-4
23. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. Proceedings of Machine Learning Research, vol. 28, pp. 427–435. PMLR, Atlanta, Georgia, USA (17–19 Jun 2013), http://proceedings.mlr.press/v28/jaggi13.html
24. Jiang, H., Lee, Y.T., Song, Z., Wong, S.C.w.: An improved cutting plane method for convex optimization, convex-concave games, and its applications. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing. p. 944–953. STOC 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3357713.3384284, https://doi.org/10.1145/3357713.3384284
25. Kelley, Jr, J.E.: The cutting-plane method for solving convex programs. Journal of the society for Industrial and Applied Mathematics **8**(4), 703–712 (1960)
26. Khachiyan, L.G.: A polynomial algorithm in linear programming (in russian). Doklady Akademiia Nauk SSSR 224 **224**, 1093–1096 (1979), english Translation: Soviet Mathematics Doklady 20, 191-194.
27. Lee, Y.T., Sidford, A., Wong, S.C.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science. pp. 1049–1065 (2015). https://doi.org/10.1109/FOCS.2015.68
28. Nemirovsky, A., Yudin, D.: Informational complexity and efficient methods for solution of convex extremal problems. Ékonomika i Mathematicheskie Metody **12** (1983)
29. Nesterov, Y.: Cutting plane algorithms from analytic centers: efficiency estimates. Mathematical Programming **69**(1), 149–176 (1995)
30. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research **20**(2), 257–301 (May 1995). https://doi.org/10.1287/moor.20.2.257, https://doi.org/10.1287/moor.20.2.257
31. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65**(6), 386–408 (1958). https://doi.org/10.1037/h0042519, https://doi.org/10.1037/h0042519
32. Shahrokhi, F., Matula, D.W.: The maximum concurrent flow problem. J. ACM **37**(2), 318–334 (Apr 1990). https://doi.org/10.1145/77600.77620, http://doi.acm.org/10.1145/77600.77620

33. Sonnevend, G.: New algorithms in convex programming based on a notion of "centre" (for systems of analytic inequalities) and on rational extrapolation. In: Hoffmann, K.H., Zowe, J., Hiriart-Urruty, J.B., Lemarechal, C. (eds.) Trends in Mathematical Optimization: 4th French-German Conference on Optimization. pp. 311–326. Birkhäuser Basel, Basel (1988)
34. UC Irvine Machine Learning Repository. https://archive-beta.ics.uci.edu/ml/datasets, accessed September 3, 2021
35. Vaidya, P.M.: A new algorithm for minimizing convex functions over convex sets. Mathematical Programming **73**(3), 291–341 (Jun 1996). https://doi.org/10.1007/bf02592216, https://doi.org/10.1007/bf02592216