



# A simple method for convex optimization in the oracle model

Daniel Dadush<sup>1</sup> · Christopher Hojny<sup>2</sup> · Sophie Huiberts<sup>3</sup> · Stefan Weltge<sup>4</sup>

Received: 27 July 2022 / Accepted: 11 July 2023  
© The Author(s) 2023

## 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  $O\left(\frac{(RL)^2}{\varepsilon^2} \cdot \frac{R^2}{r^2}\right)$  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

**Mathematics Subject Classification** 90C05 · 90C25

---

A preliminary version of this article has appeared in the conference proceedings of IPCO 2022.

---

✉ Christopher Hojny  
c.hojny@tue.nl

Daniel Dadush  
dadush@cwi.nl

Sophie Huiberts  
sophie@huiberts.me

Stefan Weltge  
weltge@tum.de

- <sup>1</sup> Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
- <sup>2</sup> Eindhoven University of Technology, Eindhoven, The Netherlands
- <sup>3</sup> Columbia University, New York, USA
- <sup>4</sup> Technical University of Munich, Munich, Germany

## 1 Introduction

We consider the problem of minimizing a convex function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  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  $K$  is either too large to store or not fully known. The SO model was first introduced in [29] where it was shown that an additive  $\varepsilon$ -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 [27] 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 et al. [23], where many applications to combinatorial optimization were provided. Further progress on the SO model was given by Vaidya [36], 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 [25, 28].

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 [26], Goldstein-Cheney [9] as well as Gomory [22] (in the integer programming context), and the analytic center cutting plane method [34] (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 with  $O(n \log(RL/(r\varepsilon))^2)$  convergence exist, achieved by judiciously dropping constraints [1], the more practical variants have worse guarantees. For instance, if  $K$  is the ball of radius  $R$ , the standard variant of ACCPM is only shown to achieve  $O(n(RL/\varepsilon)^2 \log(RL/\varepsilon))$  convergence [30].

In this paper, we describe a new method for convex optimization in the SO model that computes an additive  $\varepsilon$ -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 [3, 5], 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,  $\ell_2$  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 [19] (see [24] 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 [20, 31, 33]. 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 multi-commodity flow [20, 33], 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  $\Sigma$  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 [15]), and a variant of this method, the perceptron algorithm [32] 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 the 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  $\rho$  is the so-called width of  $\Sigma$  (the radius of the largest ball contained in  $\Sigma$  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, 10, 17], which iteratively “rescale” the norm to speed up the convergence. These rescaled variants can also be applied in the oracle setting [4, 11, 14] with appropriate adaptations. The main shortcoming of existing conic approaches is that they are currently not well-adapted for solving optimization problems rather than 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  $(c, \text{UB})$  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  $(c, \text{UB})$  appears with a nonzero coefficient and hence  $(-c, -\text{UB})$  is a nonnegative combination of the points  $(a_i, b_i)$ , which in turn shows that  $\langle -c, x \rangle \leq -\text{UB}$  is implied by the linear inequalities  $\langle a_i, x \rangle \leq b_i$ , i.e., UB is equal to the minimum of  $f$  over  $K$ . In view of this, we will consider a potential  $\Phi: \mathbb{R}^{n+1} \rightarrow \mathbb{R}_+$  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  $\Phi$  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  $\Phi$  and scalings of inequalities. We will also see that general convex functions can be directly handled in the same manner by simply replacing  $(c, \text{UB})$  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 distinction 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`<sup>1</sup> 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.

A preliminary version of this article has appeared in the conference proceedings of IPCO 2022 [13]. The current article extends on this version by providing missing

<sup>1</sup> [https://github.com/christopherhojny/supplement\\_simple-iterative-methods-linopt-convex-sets](https://github.com/christopherhojny/supplement_simple-iterative-methods-linopt-convex-sets).

proofs and presenting more detailed numerical experiments. Moreover, we generalize Algorithm 1 by no longer requiring explicit knowledge on the Lipschitz constant  $L$ .

## 2 Algorithm

Recall that we are given first-order access to a convex function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  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  $\Phi: \mathbb{R}^{n+1} \rightarrow \mathbb{R}_+$  be any strictly convex and differentiable function with  $\min_{x \in \mathbb{R}^{n+1}} \Phi(x) = \Phi(\mathbf{0}) = 0$ . 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  $t = 1, 2, \dots, T$  do
3:    $p_t \leftarrow \arg \min\{\Phi(p) : p \in \text{conv}(A_t \cup G_t)\}$ 
4:   if  $p_t = \mathbf{0}$  then return  $UB$ .
5:    $x_t \leftarrow -\nabla\Phi(p_t)[1 : n]/\nabla\Phi(p_t)[n + 1]$ 
6:   if  $x_t \in K$  then
7:      $UB \leftarrow \min\{UB, f(x_t)\}$ 
8:      $g_t \leftarrow (\nabla f(x_t), \langle \nabla f(x_t), x_t \rangle)$ 
9:     if  $g_t = \mathbf{0}$  then return  $UB$ .
10:     $A_{t+1} \leftarrow A_t$ .
11:     $G_{t+1} \leftarrow G_t \cup \{g_t/\|g_t\|_*\}$ 
12:   else
13:     get  $(a, b) \in \mathcal{A}$ , with  $\langle a, x_t \rangle > b$  and  $\|(a, b)\|_* = 1$ 
14:      $A_{t+1} \leftarrow A_t \cup \{(a, b)\}$ .
15:      $G_{t+1} \leftarrow G_t$ .
16: 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  $\Phi$  over  $\text{conv}(A_t \cup G_t)$ , for every  $q \in \text{conv}(A_t \cup G_t)$  we have  $\langle \nabla\Phi(p_t), q - p_t \rangle \geq 0$ . In particular, if  $p_t \neq \mathbf{0}$ , then we have

$$\langle \nabla\Phi(p_t), q \rangle \geq \langle \nabla\Phi(p_t), p_t \rangle \geq -\Phi(p_t) + \langle \nabla\Phi(p_t), p_t \rangle$$

$$= -(\Phi(p_t) + \langle \nabla \Phi(p_t), \mathbf{0} - p_t \rangle) > -\Phi(\mathbf{0}) = 0, \tag{1}$$

First, apply (1) 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 Inequality (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  $(c, d) \in A_t \cup G_t$ .  $\square$

Notice that throughout our algorithm UB is always an upper bound on  $\text{OPT} = \min_{x \in K} f(x)$ . We remark that when UB is returned in Line 9, then  $\nabla f(x_t) = \mathbf{0}$  and hence  $f(x_t) = \text{UB} = \min_{x \in \mathbb{R}^n} f(x) = \text{OPT}$ . In what follows, we see that UB is generally getting closer to OPT as the number of iterations increases. To this end, we will first show that  $\Phi(p_t)$  decreases at a specific rate.

Note that, for the sake of presentation, in Line 3 we require  $p_t$  to be the convex combination of minimum  $\Phi$ -value. As is implicit in the proof of Lemma 2, it is not necessary to compute such a minimum to ensure the desired convergence rate. More precisely, it suffices to let  $p_t$  be 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  $\Phi$  is 1-smooth with respect to  $\|\cdot\|_*$ . Then for every  $t = 1, \dots, T$ , Algorithm 1 satisfies  $\Phi(p_t) \leq \frac{8}{t+2}$ .*

**Proof** Suppose that  $p_t \neq \mathbf{0}$ . Then, by Line 3 in Algorithm 1,  $\|p_t\|_* \leq 1$  since  $p_t$  is a convex combination of points with norm 1. We claim that in every iteration with  $p_t \neq \mathbf{0}$  we add a point  $q_t \in A_{t+1} \cup G_{t+1}$  with  $\|q_t\|_* = 1$  such that

$$\langle \nabla \Phi(p_t), q_t \rangle \leq 0 \tag{2}$$

holds. To see this, recall from the proof of the previous lemma that  $\nabla \Phi(p_t)[n + 1] > 0$  holds. If  $x_t \in K$ , then  $q_t = g_t / \|g_t\|_*$  is added to  $G_{t+1}$ , which satisfies

$$\begin{aligned} \|g_t\|_* \langle \nabla \Phi(p_t), q_t \rangle &= \langle \nabla \Phi(p_t), g_t \rangle \\ &= \langle \nabla \Phi(p_t), (\nabla f(x_t), \langle \nabla f(x_t), x_t \rangle) \rangle \\ &= \langle \nabla \Phi(p_t)[1 : n], \nabla f(x_t) \rangle + \nabla \Phi(p_t)[n + 1] \langle \nabla f(x_t), x_t \rangle \\ &= 0, \end{aligned}$$

where the last equality is due to the definition of  $x_t$ . Otherwise, our algorithm receives some  $(a, b) \in \mathcal{A}$  with  $\langle a, x_t \rangle > b$  and  $\|(a, b)\|_* = 1$  and hence setting  $q_t = (a, b) \in A_{t+1}$  we see that

$$\langle \nabla \Phi(p_t), q_t \rangle = \langle \nabla \Phi(p_t)[1 : n], a \rangle + \nabla \Phi(p_t)[n + 1] \cdot b$$

$$\begin{aligned}
 &= \nabla\Phi(p_t)[n + 1] \cdot (-\langle x_t, a \rangle + b) \\
 &< 0
 \end{aligned}$$

holds. This establishes the claim.

Next, recall that  $\Phi$  is 1-smooth with respect to  $\|\cdot\|_*$  if

$$\langle \nabla\Phi(x), x - y \rangle \leq \Phi(x) - \Phi(y) + \frac{1}{2}\|x - y\|_*^2 \tag{3}$$

holds for all  $x, y \in \mathbb{R}^{n+1}$ . In particular, for  $x = \mathbf{0}$  and  $y = p_1$  this yields

$$\Phi(p_t) \leq \Phi(p_1) \leq \Phi(\mathbf{0}) + \langle \nabla\Phi(\mathbf{0}), p_1 \rangle + \frac{1}{2}\|p_1\|_*^2 = \frac{1}{2}\|p_1\|_*^2 = \frac{1}{2},$$

where the first inequality follows from Line 3 in Algorithm 1 as well as  $A_1 \subseteq A_t$  and  $G_1 \subseteq G_t$ . Moreover, setting  $\lambda := \frac{1}{4}\Phi(p_t)$  we obtain  $\lambda \leq 1$  and

$$\begin{aligned}
 \Phi(p_{t+1}) &\leq \Phi(p_t + \lambda(q_t - p_t)) \leq \Phi(p_t) - \lambda\langle \nabla\Phi(p_t), p_t - q_t \rangle + \frac{1}{2}\lambda^2\|p_t - q_t\|_*^2 \\
 &\leq \Phi(p_t) - \lambda\langle \nabla\Phi(p_t), p_t \rangle + \frac{1}{2}\lambda^2\|p_t - q_t\|_*^2 \\
 &\leq \Phi(p_t) - \lambda\Phi(p_t) + \frac{1}{2}\lambda^2\|p_t - q_t\|_*^2 \\
 &\leq \Phi(p_t) - \lambda\Phi(p_t) + 2\lambda^2 \\
 &= \Phi(p_t) - \frac{1}{8}\Phi(p_t)^2,
 \end{aligned}$$

where the second inequality follows from (3), the third inequality follows from (2), the fourth inequality follows from convexity and  $\Phi(\mathbf{0}) = 0$ , and the last inequality holds since  $\|p_t\|_* \leq 1$  and  $\|q_t\|_* \leq 1$ . From this we derive

$$\frac{1}{\Phi(p_{t+1})} \geq \frac{1}{\Phi(p_t)} \cdot \frac{1}{1 - \frac{1}{8}\Phi(p_t)} \geq \frac{1}{\Phi(p_t)} \left( 1 + \frac{1}{8}\Phi(p_t) \right) = \frac{1}{\Phi(p_t)} + \frac{1}{8}.$$

It follows that  $\frac{1}{\Phi(p_t)} \geq \frac{1}{\Phi(p_1)} + \frac{1}{8}(t - 1)$  for all  $t$ , which yields the claim since  $\Phi(p_1) \leq \frac{1}{2}$ . □

The following lemma yields conditions under which a small value of  $\Phi(p_t)$  implies that UB is close to OPT. Note in particular that it proves that if  $\|p_t\| = 0$  then UB = OPT. Recall that a function  $f$  is  $L$ -Lipschitz on a set  $K \subset \mathbb{R}^n$  when  $|f(x) - f(y)| \leq L\|x - y\|_2$  for all  $x, y \in K$ .

**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  $(a, b) \in \mathcal{A} \cup \{(\mathbf{0}, 1)/\|(\mathbf{0}, 1)\|_*\}$ . If  $\|p_T\|_* \leq \alpha/2$  in Algorithm 1, then the returned value satisfies  $\text{UB} \geq \text{OPT} \geq \text{UB} - \frac{8U\|p_T\|_*}{\alpha}$ , where  $U := \max_{x \in K} \|(\nabla f(x), \langle \nabla f(x), x \rangle)\|_*$ .*

**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

$$p_T = \sum_{(a,b) \in A_T} \lambda_{(a,b)}(a, b) + \sum_{t \in F} \gamma_t g_t / \|g_t\|_*,$$

where  $\lambda \geq 0, \gamma \geq 0$  and  $\|(\lambda, \gamma)\|_1 = 1$ , and where  $g_t = (\nabla f(x_t), \langle \nabla f(x_t), x_t \rangle)$  as defined in Algorithm 1. Then we have

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

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

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

Now 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  $\alpha/2 \leq (\alpha + 1) \sum_{t \in F} \gamma_t \leq 2 \sum_{t \in F} \gamma_t$ . This lower bound on  $\sum_{t \in F} \gamma_t$  suffices to prove the lemma.  $\square$

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  $UB < \infty$  satisfying  $UB \geq \min_{x \in K} f(x) \geq UB - \frac{32U}{\alpha\sqrt{\beta(T+2)}}$ .

**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  $OPT \geq UB - \frac{32U}{\sqrt{\beta(T+2)}\alpha}$ .  $\square$

**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  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  be an  $L$ -Lipschitz convex function given by a subgradient oracle.

Apply Algorithm 1 to the function  $f$  using norm  $\|(x, y)\| := \sqrt{2}\|(x/R, y)\|_2$  and potential  $\Phi(a, b) := \frac{1}{4}\|(Ra, b)\|_2^2$ . 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  $UB \geq \min_{x \in K} f(x) \geq UB - \varepsilon$ .

**Proof** Note that our choice of norm implies that  $\|(a, b)\|_* = \frac{1}{\sqrt{2}}\|(Ra, b)\|_2$ . We claim that our choice of input satisfies the conditions of Theorem 1 with  $\beta = 1/2$  and  $\alpha = r/4R$ . Given the claim, Theorem 1 directly proves the result. To prove the claim, apart from verifying that the bounds on  $\beta$  and  $\alpha$  hold, we must verify smoothness of  $\Phi$  with respect to the dual norm, a bound of 2 on the norm of  $(-x, 1)$  for  $x \in K$ , as well as  $U \leq LR$ .

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

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

and

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

which implies that  $U \leq LR$ . We now show the lower bound  $\alpha \geq r/4R$ . Firstly, we see that

$$\left\langle (-z, 1), \frac{(\mathbf{0}, 1)}{\|(\mathbf{0}, 1)\|_*} \right\rangle = 1 \geq \frac{1}{2}\|(-z, 1)\| \frac{\|(\mathbf{0}, 1)\|}{\|(\mathbf{0}, 1)\|_*} \|_*.$$

Next, any  $(a, b)$  returned by the oracle is normalized so that  $\|(a, b)\|_* = 1 \Leftrightarrow \|(Ra, b)\|_2 = \sqrt{2}$ . Note then that  $\|(-z, 1)\| \|(a, b)\|_* \leq 2$ . From here, we observe that

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

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

### 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, see Algorithm 2 which we describe in the next paragraph in more detail.

#### 3.1 Framework

Our generic cutting plane framework gets the objective  $\langle c, x \rangle$  which is to be minimized, the radius  $R$  of an outer ball containing the feasible region  $K$ , and two oracles as input. The separation oracle SO is equipped with a set of initial linear inequalities valid for  $K$  (such as bounds on variables). These inequalities define an initial polyhedral relaxation  $P$  of  $K$ , which will be refined whenever new inequalities are separated. The point oracle PO implements the points that are to be separated by the different methods, e.g., the oracle for the standard cut loop will return a point in  $\operatorname{argmin}\{\langle c, x \rangle : x \in P\}$ . Moreover, for each instance, we will be given a finite upper bound UB and incorporate the linear inequality  $\langle c, x \rangle \leq \text{UB}$  in a similar way. This upper bound gets updated

---

#### Algorithm 2 Generic cutting plane method.

---

- 1: **Input:** point oracle (PO), separation oracle (SO), objective  $\langle c, x \rangle$ , outer radius  $R > 0$
  - 2:  $\text{UB} \leftarrow R\|c\|$
  - 3:  $P \leftarrow$  polyhedron defined by inequalities  $A_1$  queried from SO as well as  $\langle c, x \rangle \leq \text{UB}$
  - 4: **while**  $\text{UB} - \text{LB} > 10^{-3}$  **do**
  - 5:    $\text{LB} \leftarrow \min\{\langle c, x \rangle : x \in P\}$
  - 6:    $x' \leftarrow$  ask PO for next point to be separated
  - 7:    $\langle a, x \rangle \leq b \leftarrow$  ask SO for inequality that is maximally violated by  $x'$
  - 8:   **if**  $\langle a, x' \rangle > b$  **then**
  - 9:      $P \leftarrow P \cap \{x \in \mathbb{R}^n : \langle a, x \rangle \leq b\}$
  - 10:   **else**
  - 11:      $\text{UB} = \min\{\text{UB}, \langle c, x' \rangle\}$
  - return**  $\text{UB}, \text{LB}$
-

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). Such an oracle is often automatically available and can have a positive impact on the performance of the considered algorithms. For the problems we consider, the actual implementation of a smart oracle will be specified below.

### 3.2 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 implementation of the analytic center cutting plane method [2]. Our method is implemented<sup>2</sup> in the spirit of Theorem 2, where  $p_t$  is computed using `Gurobi`. Note that the number reported in Table 1 deviate slightly from those reported in [13], because we have rerun the experiments in a different computational environment and Algorithm 1 does not longer require to normalize objective gradients.

### 3.3 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.

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

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e \\ & \sum_{e \in \delta(v)} x_e \leq 1, & \text{for all } v \in V, \\ & \sum_{e \in E[U]} x_e \leq \frac{|U| - 1}{2}, & \text{for all } U \subseteq V \text{ with } |U| \text{ odd} \\ & x \in [0, 1]^E, \end{aligned}$$

due to Edmonds [18], 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

<sup>2</sup> [https://github.com/christopherhojny/supplement\\_simple-iterative-methods-linopt-convex-sets](https://github.com/christopherhojny/supplement_simple-iterative-methods-linopt-convex-sets).

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, forming the test set `matching`. 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, yielding the test set `matching02`.

Our second set of instances is based on the semidefinite relaxation of Goemans and Williamson [21] for the maximum cut problem

$$\max \left\{ \sum_{\{v,w\} \in E} c(v,w)(1 - X_{v,w})/2 : X_{v,w} = X_{w,v} \text{ for all } v, w \in V, \right. \\ \left. X_{v,v} = 1 \text{ for all } v \in V, X \succeq 0 \right\}$$

where  $c$  are edge weights on the edges of  $(V, E)$ . We add the box 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  $\langle hh^T, X \rangle \geq 0$ .

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  $\lambda$  denotes the minimum eigenvalue of  $X$  and  $I$  the identity matrix. We generated 10 complete graphs on 10 nodes with edge weights chosen uniformly at random in  $[0, 1]$ .

Our third set of instances arises from LPBoost [16], a classifier algorithm based on column generation. To solve the pricing problem in column generation, the following linear program is solved:

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

where  $\Omega$  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  $x^i$  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  $(\gamma', \lambda')$ , we use `julia's DecisionTree` module to compute a decision stump with score function  $\lambda'$  that weights the data points, whose corresponding inequality classifies  $(\gamma', \lambda')$  as feasible or not. A smart oracle always returns the computed inequality and decreases  $\gamma'$  until  $(\gamma', \lambda')$  becomes feasible according to the found decision stump.

We extracted all data sets from the UC Irvine Machine Learning Repository [35] 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

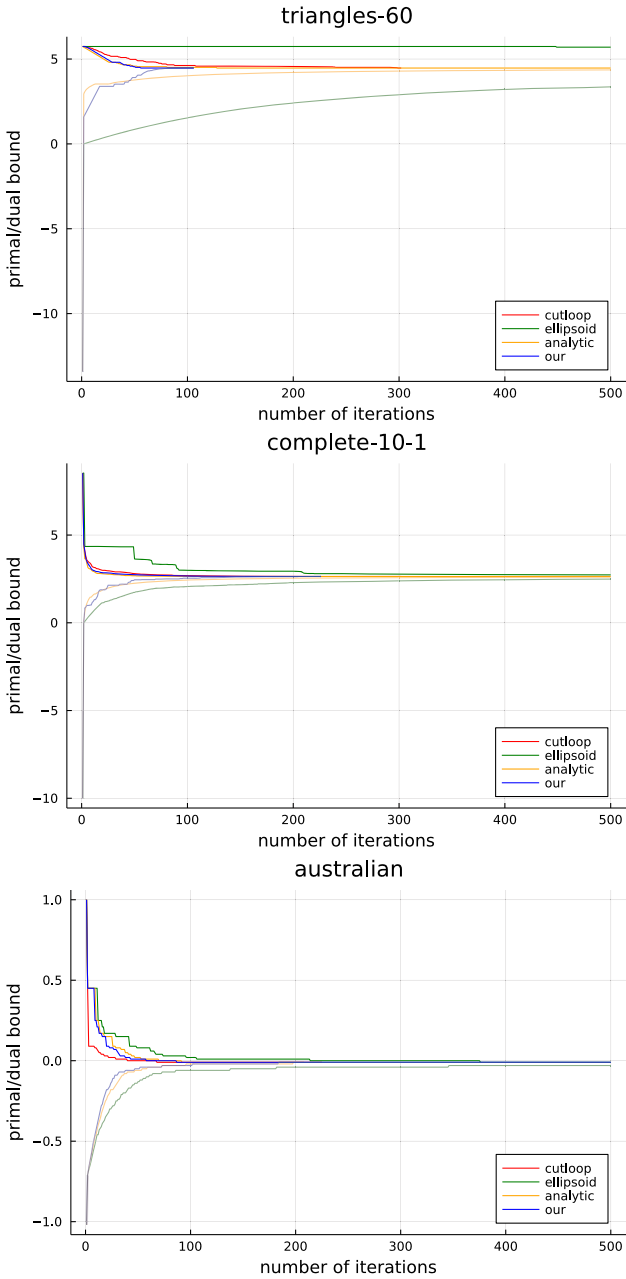


Fig. 1 Typical primal/dual bounds for a random matching, random max-cut, and LPBoost instance

have been discarded. If a selected data set contains data points with missing values, these are removed from the data set. Table 5 in Appendix 1 reports on more details of this test set.

### 3.4 Results

In the remainder of this section, we evaluate the performance of Algorithm 1 in comparison to the standard cut loop, the ellipsoid method, and the analytic center method. To this end, we have conducted three different experiments, which we will detail below. 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 Sect. 2. That is, primal (dual) solutions provide lower (upper) bounds on OPT. If  $\ell_i$  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.

Our first experiment compares the different algorithms using non-smart oracles and using  $R\|c\|$  as upper bound on the optimal objective value. Table 1 summarizes our results, 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 2 does not report on the primal integral of “LP” since the standard cut loop is a dual method.

For all tested instances, we observe that the ellipsoid and analytic center methods are struggling with solving any instance within 500 iterations. The standard cut loop is able to solve all problems more efficiently with, on average, between 90–284 iterations per instance. Our algorithm outperforms the standard cut loop on all test sets except for LPboost. For matching problems, our algorithm requires 44% (matching) and 84% less iterations than the standard cut loop, for maxcut problems it is 27% faster, and only for LPboost instances it requires three times as many iterations as the standard cut loop on average.

An explanation for this behavior is explained by the integral values, which are provided by Table 1. Among all tested algorithms, our method performs best in finding an optimal solution as the primal integrals are comparably very small. On the dual side, the picture is less clear, but in particular for matching instances our algorithm is able to quickly find the true dual value. For LPboost, however, our method converges much slower on the dual side than the other methods. We conjecture that this behavior is due to the special structure of the objective for LPboost instances. While the other problem classes have a completely dense objective, only one variable contributes to the objective for LPboost.

**Table 1** Comparison of iterations and dual/primal integral

Instance	#iterations			Dual integral			Primal integral				
	LP	Ellipsoid	Analytic	Algorithm 1	LP	Ellipsoid	Analytic	Algorithm 1	Ellipsoid	Analytic	Algorithm 1
	Matching	175.44	500.00	500.00	97.44	56.28	547.32	25.34	24.46	892.03	159.55
Matching02	283.77	460.77	491.69	46.46	8.46	7.64	4.05	0.56	359.95	90.00	30.49
Maxcut	265.30	500.00	500.00	194.30	39.48	225.43	17.78	27.39	274.48	117.13	81.34
LPboost	91.38	489.06	479.12	269.00	3.63	14.91	21.92	51.94	339.65	54.20	52.83

**Table 2** Comparison of iterations and dual/primal integral using smart oracles

Instance	#Iterations			Dual integral			Primal integral				
	LP	Ellipsoid	analytic	Algorithm 1	LP	ellipsoid	analytic	Algorithm 1	ellipsoid	analytic	Algorithm 1
Matching	175.44	500.00	500.00	97.44	56.28	547.32	25.34	24.46	892.03	159.55	74.15
Matching02	283.77	460.77	491.69	46.46	8.46	7.64	4.05	0.56	359.95	90.00	30.49
Maxcut	265.30	500.00	500.00	194.00	39.48	218.39	17.82	27.33	264.97	115.39	71.04
LPboost	89.56	346.38	88.00	126.75	3.77	14.75	6.14	6.12	23.25	6.00	6.26



**Table 3** Comparison of iterations and dual/primal integral using smart oracles and an optimal initialization of the upper bound value

Instance	#iterations				Dual integral				Primal integral			
	LP	Ellipsoid	Analytic	Algorithm 1	LP	Ellipsoid	Analytic	Algorithm 1	Ellipsoid	Analytic	Algorithm 1	Algorithm 1
Matching	166.19	500.00	30.81	29.81	56.28	578.48	17.51	17.46	0.00	0.00	0.00	0.00
Matching02	197.23	187.62	1.46	1.46	8.46	8.46	0.03	0.03	0.00	0.00	0.00	0.00
Maxcut	225.90	500.00	92.70	186.50	39.47	620.00	15.55	30.00	0.00	0.00	0.00	0.00
LPboost	56.75	303.06	45.56	101.56	3.61	14.75	3.79	3.91	0.00	-0.00	0.00	0.00

**Table 4** Comparison of iterations and dual/primal integral using smart oracles and an almost optimal initialization of the upper bound value

Instance	#iterations			Dual integral			Primal integral				
	LP	Ellipsoid	Analytic	Algorithm 1	LP	Ellipsoid	Analytic	Algorithm 1	Ellipsoid	Analytic	Algorithm 1
Matching	175.44	500.00	500.00	97.44	56.28	547.32	25.34	24.46	892.03	159.55	74.15
Matching02	283.77	460.77	491.69	46.46	8.46	7.64	4.05	0.56	359.95	90.00	30.49
Maxcut	265.30	500.00	500.00	194.00	39.48	218.39	17.82	27.33	264.97	115.39	71.04
LPboost	90.19	346.38	87.81	126.75	3.79	14.75	6.14	6.12	23.25	6.00	6.25

For this reason, our second experiment tested the impact of smart oracles on the performance of the four different methods. In particular for LPboost this might make a difference, because it is rather easy to turn an infeasible point into a feasible solution by setting the objective value  $\gamma$  on the corresponding value. Indeed, as Table 2 shows, there is almost no effect of smart oracles on the tested maxcut problems. Moreover, there is no effect on matching problems since there we did not implement smart oracles as discussed above. For LPboost, however, the ellipsoid and analytic center method as well as our algorithm benefit greatly from smart oracles. While the analytic center method and Algorithm 1 achieve the lowest primal integral values of approximately 6, the relative improvement is most pronounced for the ellipsoid method whose primal integral drops by 93%.

Finally, we investigated the impact of knowing a good feasible solution on the performance of the four different algorithms. To this end, we initialized the upper bound on the optimal objective value in our cutting plane framework either with the optimal value or the a value being 1% larger than the optimum value. In the first case, all algorithms only need to find a matching dual value, whereas in the second case also an (almost) optimal primal solution needs to be found. Tables 3 and 4 report on our results, respectively, using smart oracles.

We can see that initializing the upper bound with the optimal value greatly improves the performance of the analytic center method and our algorithm, and also the ellipsoid method performs faster for matching02 instances. The performance of the standard cut loop, however, does not show significant differences in comparison with the standard initialization. But if an almost optimal initialization of the upper bound is used, we cannot observe large differences in the average iteration count in comparison to the standard initialization. This shows that all tested methods perform rather well on the dual side, but are struggling in finding an almost optimal primal solution. Based on the primal integral values, however, we can see that our method performs on average better than the ellipsoid and analytic center method. The above findings are also supported by the plots in Fig. 1 that show the typical development of the relative primal and dual bounds for the different problem types.

**Acknowledgements** We would like to thank Robert Luce and Sebastian Pokutta for their very valuable feedback on our work. The first author has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement QIP-805241)

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

**Table 5** Overview of LPboost instances

Name	#Data points	#Features
00451	116	9
audit_risk	772	27
australian	690	14
colposkopy-green	98	68
echocardiogram	61	11
german	1000	24
heart	270	13
house-votes-84	232	16
ionsphere	351	34
mesothelioma	324	34
parkinsons	195	22
pop_failures-2	540	20
sonar	208	60
spect	187	22
tic-tac-toe	958	9
wdbc-1	194	33

## A Details about the LPboost test set

Table 5 reports about the characteristics of the LPboost test set. The names of the different data sets match those from the UC Irvine Machine Learning Repository [35] and columns “#data points” and “#features” report about the number of data points and the number of different features of each data point.

## References

1. Atkinson, D.S., Vaidya, P.M.: A cutting plane algorithm for convex programming that uses analytic centers. *Math. Program.* **69**(1), 1–43 (1995)
2. Badenbroek, R., de Klerk, E.: An analytic center cutting plane method to determine complete positivity of a matrix. *INFORMS J. Comput.* **34**, 1115–125 2022 . <https://doi.org/10.1287/ijoc.2021.1108>
3. Beck, A.: *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics (2017). <https://doi.org/10.1137/1.9781611974997>
4. Belloni, A., Freund, R.M., Vempala, S.: An efficient rescaled perceptron algorithm for conic systems. *Math. Oper. Res.* **34**(3), 621–641 (2009)
5. Ben-Tal, A., Nemirovski, A.: *Lectures on Modern Convex Optimization*. Society for Industrial and Applied Mathematics, London (2001). <https://doi.org/10.1137/1.9780898718829>
6. Betke, U.: Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem. *Discret. Comput. Geom.* (2004). <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 (2017). <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 (2017). <https://doi.org/10.1137/1.9781611974782.51>

9. Cheney, E.W., Goldstein, A.A.: Newton's method for convex programming and Tchebycheff approximation. *Numer. Math.* **1**(1), 253–268 (1959)
10. Chubanov, S.: A strongly polynomial algorithm for linear systems having a binary solution. *Math. Program.* **134**(2), 533–570 (2011). <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* (2017)
12. Color02—Computational Symposium: Graph Coloring and its Generalizations. Available at <http://mat.gsia.cmu.edu/COLOR02> (2002)
13. Dadush, D., Hojny, C., Huiberts, S., Weltge, S.: A simple method for convex optimization in the oracle model. In: Aardal, K., Sanità, L. (eds.) *Integer Programming and Combinatorial Optimization*, 23rd International Conference, IPCO 2022 (2022)
14. Dadush, D., Végh, L.A., Zambelli, G.: Rescaling algorithms for linear conic feasibility. *Math. Oper. Res.* **45**(2), 732–754 (2020). <https://doi.org/10.1287/moor.2019.1011>
15. Dantzig, G.B.: Converting a converging algorithm into a polynomially bounded algorithm. Technical report, Stanford University, 1992. 5.6, 6.1, 6.5 (1991)
16. Demiriz, A., Bennett, K.P., Shawe-Taylor, J.: Linear programming boosting via column generation. *Mach. Learn.* **46**(1), 225–254 (2002)
17. Dunagan, J., Vempala, S.: A simple polynomial-time rescaling algorithm for solving linear programs. *Math. Program.* **114**(1), 101–114 (2007). <https://doi.org/10.1007/s10107-007-0095-7>
18. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Natl. Bureau Stand.* **69B**(1–2), 125–130 (1964)
19. Frank, M., Wolfe, P.: An algorithm for quadratic programming. *Naval Res. Logist. Q.* **3**(1–2), 95–110 (1956)
20. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.* **37**(2), 630–652 (2007). <https://doi.org/10.1137/s0097539704446232>
21. 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>
22. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. *Bull. Am. Math. Soc.* **64**, 275–278 (1958)
23. Grötschel, M., Lovász, L., Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization*, vol. 2. Springer (1988). <https://doi.org/10.1007/978-3-642-78240-4>
24. Jaggi, M.: Revisiting Frank–Wolfe: projection-free sparse convex optimization. In: *Proceedings of Machine Learning Research*, vol. 28, pp. 427–435. PMLR, Atlanta, Georgia, USA (17–19, 2013). <http://proceedings.mlr.press/v28/jaggi13.html>
25. 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*, pp. 944–953. STOC 2020, Association for Computing Machinery, New York, NY, USA (2020). <https://doi.org/10.1145/3357713.3384284>
26. Kelley, J.E., Jr.: The cutting-plane method for solving convex programs. *J. Soc. Ind. Appl. Math.* **8**(4), 703–712 (1960)
27. Khachiyan, L.G.: A polynomial algorithm in linear programming (in Russian). *Doklady Akademii Nauk SSSR* **224**, 1093–1096 (1979), English Translation: *Soviet Mathematics Doklady* **20**, 191–194
28. 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>
29. Nesterov, Y.: Cutting plane algorithms from analytic centers: efficiency estimates. *Math. Program.* **69**(1), 149–176 (1995)
30. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.* **20**(2), 257–301 (1995). <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. *Psychol. Rev.* **65**(6), 386–408 (1958). <https://doi.org/10.1037/h0042519>
32. Shahrokhi, F., Matula, D.W.: The maximum concurrent flow problem. *J. ACM* **37**(2), 318–334 (1990). <https://doi.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 3 Sep 2021
  35. Vaidya, P.M.: A new algorithm for minimizing convex functions over convex sets. *Math. Program.* **73**(3), 291–341 (1996). <https://doi.org/10.1007/bf02592216>
  36. Yudin, D., Nemirovsky, A.: Informational complexity and efficient methods for solution of convex extremal problems. *Econ. Math. Methods* **12**, 357–369 (1976)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.