Pairwise Conditional Gradients without Swap Steps and Sparser Kernel Herding

Kazuma Tsuji [1] **Ken'ichiro Tanaka** [2 3] **Sebastian Pokutta** [4]

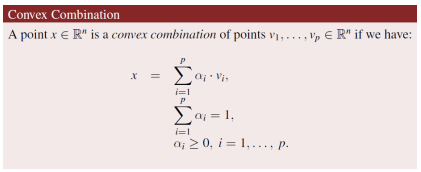
Abstract

The Pairwise Conditional Gradients (PCG) algorithm is a powerful extension of the Frank-Wolfe algorithm leading to particularly sparse solutions, which makes PCG very appealing for problems such as sparse signal recovery, sparse regression, and kernel herding. Unfortunately, PCG exhibits so-called swap steps that might not provide sufficient primal progress. The number of these bad steps is bounded by a function in the dimension and as such known guarantees do not generalize to the infinite-dimensional case, which would be needed for kernel herding. We propose a new variant of PCG, the so-called Blended Pairwise Conditional Gradients (BPCG). This new algorithm does not exhibit any swap steps, is very easy to implement, and does not require any internal gradient alignment procedures. The convergence rate of BPCG is basically that of PCG if no drop steps would occur and as such is no worse than PCG but improves and provides new rates in many cases. Moreover, we observe in the numerical experiments that BPCG's solutions are much sparser than those of PCG. We apply BPCG to the kernel herding setting, where we derive nice quadrature rules and provide numerical results demonstrating the performance of our method.

1. Introduction

Conditional Gradients (Levitin and Polyak, 1966) (also: Frank-Wolfe algorithms (Frank and Wolfe, 1956)) are an important class of first-order methods for constrained convex minimization, i.e., solving

$$\min_{x \in P} f(x),$$



where P is a compact convex feasible region. These methods usually form their iterates as convex combinations of feasible points and as such do not require (potentially expensive) projections onto the feasible region P. Moreover the access to the feasible region is solely realized by means of a so-called *Linear Minimization Oracle (LMO)*, which upon presentation with a linear function c returns $\arg\min_{x \in P} c^\intercal x$. Another significant advantage is that the iterates are typically formed as *sparse* convex combinations of extremal points of the feasible region (sometimes also called *atoms*) which makes this class of optimization algorithms particularly appealing for problems such as sparse signal recovery, structured regression, SVM training, and also kernel herding. Over the recent years there have been significant advances in Frank-Wolfe algorithms providing even faster convergence rates and higher sparsity (in terms of the number of atoms participating in the convex combination) and in particular the *Pairwise Conditional Gradients (PCG)* algorithm (Lacoste-Julien and Jaggi, 2015) provides a very high convergence speed (both theoretically and in computations) and sparsity. The PCG algorithm exhibits so-called *swap steps*, which are steps in which weight is shifting from one atom to another. These steps do not guarantee sufficient primal progress and hence usually convergence analyses bound their number in order to argue that there is a sufficient number of steps with good progress. Unfortunately, these bounds depend exponentially on the dimension of the feasible region (Lacoste-Julien and Jaggi, 2015) and require P to be a polytope for this bound to hold at all. This precludes application of PCG to the infinite-dimensional setting and even in the polyhedral setting the worst-case bound might be unappealing. Recently, several works (Rinaldi and Zeffiro, 2020; Combettes and Pokutta, 2020; Mortagy et al., 2020) suggested 'enhancement procedures' to potentially overcome swap steps by improving the descent directions, however at the cost of significantly more complex algorithms and (still often dimension-dependent) analysis. In contrast, we propose a much simpler modification of the PCG algorithm by combining it with the blending criterion from the Blended Conditional Gradients Algorithm

[1]MUFG Bank, Ltd., Tokyo, Japan [2]Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan [3]PRESTO Japan Science and Technological Agency (JST) Tokyo, Japan [4] AISST, Zuse Institute Berlin and Institute of Mathematics, Technische Universität Berlin, Berlin, Germany. Correspondence to: Kazuma Tsuji <kazumatsuji.research@gmail.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

(BCG) of Braun et al. (2019a). This modified PCG algorithm, which we refer to as *Blended Pairwise Conditional Gradients* does not exhibit swap steps anymore and the convergence rates that we obtain are that which the original PCG algorithm would achieve if swap steps would not occur. As such it improves the convergence rates of PCG and moreover, by eschewing swap steps, this modification provides natural convergence rates for the infinite-dimensional setting, which is important for our application to kernel herding (Welling, 2009; Chen et al., 2010; Bach et al., 2012). *Kernel herding* is a particular method for constructing a quadrature formula by running a Conditional Gradient algorithm on the convex subset of a Reproducing Kernel Hillbert Space (RKHS); the obtained solution and the associated convex combination correspond to the respective approximation. It can be considered within the framework of kernel quadrature methods that have been studied in a long line of works, such as e.g., (Huszár and Duvenaud, 2012; Oettershagen, 2017; Briol et al., 2019). Accurate quadrature rules with a *small* number of nodes are often desired, and as such the achievable sparsity of a given Conditional Gradient method when used for kernel herding is crucial.

Related Work

There is a broad literature on conditional gradient algorithms and recently this class of methods regained significant attention with many new results and algorithms for specific setups. Most closely related to our work however are the Pairwise Conditional Gradients algorithm introduced in (Lacoste-Julien and Jaggi, 2015) as well as the Away-step Frank-Wolfe algorithm (Wolfe, 1970; Guélat and Marcotte, 1986); the Pairwise Conditional Gradients algorithm arises as a natural generalization of the Away-step Frank-Wolfe algorithm. Moreover, we used the blending criterion from (Braun et al., 2019a) to efficiently blend together local PCG steps with global Frank-Wolfe steps.

On the other hand there is only a limited number of works attempting to use and extend Conditional Gradients to kernel herding. Lacoste-Julien et al. (2015) studied the practical performance of several variants of kernel herding that correspond to the variants of Conditional gradients, such as the Away-step Frank-Wolfe algorithm and Pairwise Conditional Gradients. More recently, Tsuji and Tanaka (2021) proposed a new variant of kernel herding with the explicit goal of obtaining sparse solutions.

Contribution

Our contribution can be roughly summarized as follows.

Pairwise Conditional Gradients without swap steps. We present a new Pairwise Conditional Gradients algorithm, the Blended Pairwise Conditional Gradients (BPCG) (Algorithm 1), that does not exhibit swap steps and provide convergence analyses of this method. In particular, the BPCG algorithm provides improved rates compared to PCG; the lack of swap steps makes the difference of the constant factor. Here we focus on the general smooth convex case and the strongly convex case over polytopes. The results for the general smooth convex case is also applicable to the infinite-dimensional general convex domains. We hasten to stress though that our convergence analyses can be immediately extended to sharp functions (a generalization of strongly convex functions) as well as uniformly convex feasible regions by combining (Kerdreux et al., 2019) and (Kerdreux et al., 2021), respectively, with our arguments here; due to space limitations this is beyond the scope of this paper. Additionally, numerical experiments suggest that the BPCG algorithm outputs fairly sparse solutions practically. BPCG offers superior sparsity of the iterates both due to the pairwise steps as well as the BCG criterion that favors local steps, which maintain the sparsity of the solution.

We also provide a lazified version (see (Braun et al., 2017; 2019b)) of BPCG (Algorithm 2) that significantly reduces the number of LMO calls by reusing previously visited atoms at the expense of a small constant factor loss in the convergence rates. The lazified variant is in particular useful when the LMO is expensive as the number of required calls to the LMO can be reduced dramatically in actual computations; see e.g., (Braun et al., 2017; 2019b;a) for the benefits of lazification.

Sparser Kernel Herding. We demonstrate the effectiveness of applying our BPCG algorithm to kernel herding. From a theoretical perspective, we can apply the convergence guarantees for the general smooth convex case mentioned before and obtain state-of-the-art guarantees. Moreover, in numerical experiments we demonstrate that the practical performance of BPCG for kernel herding much exceed the theoretical guarantees. BPCG and the lazified version achieve very fast convergence in the number of nodes which are competitive with optimal convergence rates. In addition, the lazification contributes to reducing computational cost because the LMO in kernel herding is computationally rather expensive.

Computational Results. We complement our theoretical analyses as well as the numerical experiments for kernel herding with general purpose computational experiments demonstrating the excellent performance of BPCG across several problems of interest.

2. Preliminaries

Let $f : \mathbf{R}^d \to \mathbf{R}$ be a differentiable, convex, and L-smooth function. Recall that f is *L-smooth* if

$$f(y) - f(x) - \langle \nabla f(x),\, y - x \rangle \leq \frac{L}{2} \|y - x\|^2$$

Convex Hull

The set of all convex combinations of points $v_1, \ldots, v_p \in \mathbb{R}^n$ is usually called *convex hull* of $v_1, \ldots, v_p$ and usually denoted by $conv(v_1, \ldots, v_p)$.

Extreme Point

Given a convex set $C \subseteq \mathbb{R}^n$ a point $\bar{x} \in C$ is an *extreme point* of C if $\bar{x}$ cannot be given as a proper convex combination of two points in C. That is, if we cannot find two points $y, z \in C$ such that $\bar{x} = \alpha y + (1-\alpha)z$, with $\alpha \in (0, 1)$.

for all $x, y \in \mathbf{R}^d$. In addition, f is *μ-strongly convex* if

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle \geq \frac{\mu}{2}\|y - x\|^2$$

for all $x, y \in \mathbf{R}^d$. Let $P \subset \mathbb{R}^d$ be a convex feasible region and the subset $V(P) \subset P$ satisfy $P = \text{conv}(V(P))$. The notation $\text{conv}(V(P))$ means the convex hull of $V(P)$. We assume that P is bounded and its diameter D is given by $D := \sup_{x,y \in P} \|x - y\|$. Furthermore, let $\delta_P > 0$ be the pyramidal width of P (Lacoste-Julien and Jaggi, 2015); we drop the index if the feasible region is clear from the context. For a convex combination $x = \sum_{i=1}^{n} c_i v_i$, as later maintained by the algorithm, let $c[x](v_i) := c_i$. In the following let $x^* \in P$ denote the (not necessarily unique) minimizer of f over P.

3. Blended Pairwise Conditional Gradients Algorithm

We will now present the algorithm and its convergence analysis.

3.1. Algorithm

We consider the Blended Pairwise Conditional Gradients algorithm (BPCG) shown in Algorithm 1. The BPCG algorithm is the same type of algorithm as the Blended Conditional Gradients algorithm in (Braun et al., 2019a). In Algorithm 1, if the local pairwise gap $\langle \nabla f(x_t), a_t - s_t \rangle$ is smaller than the Frank-Wolfe gap $\langle \nabla f(x_t), x_t - w_t \rangle$, a *FW step* (line 16-18) is taken. Otherwise, the weights of the active atoms in S_t are optimized by the Pairwise Conditional Gradients (PCG) *locally*. If the step size λ_t is larger than Λ_t^*, the away vertex is removed from the active set S_t and we call the step *drop step* (line 13). Otherwise we call the step *descent step* (line 11). Descent step and drop step are referred to as *pairwise step* all together.

By the structure of the BPCG algorithm, the sparsity of the solutions is expected since the new atoms are not added to S_t until the local pairwise gap decreases sufficiently. Moreover, since the PCG is implemented locally, BPCG does not exhibit *swap steps* in which weight is shifting from the away atom to the Frank-Wolfe atom. In the PCG, swap steps do not show much progress and the number of swap steps is bounded by the dimension-dependent constant. Therefore, the local implementation of the PCG is significant to extend the pairwise type algorithms to infinite-dimensional cases.

Note that both Algorithm 1 and Algorithm 2 which is introduced in subsection 3.3 use line search here to simplify the presentation. However both can be run alternatively with the short-step rule, which minimizes the quadratic inequality arising from smoothness (this is precisely the λ_t^* in Lemma 3.4) but requires knowledge of L or with the adaptive step-size strategy of (Pedregosa et al., 2020), which offers a performance similar to line search at the fraction of the cost; our analysis applies to these two step-size strategies as well.

Algorithm 1 Blended Pairwise Conditional Gradients (BPCG)

Require: convex smooth function f, start vertex $x_0 \in V(P)$.
Ensure: points $x_1, \ldots, x_T$ in P.

1: $S_0 \leftarrow \{x_0\}$
2: **for** $t = 0$ to $T - 1$ **do**
3: $\quad a_t \leftarrow \text{argmax}_{v \in S_t} \langle \nabla f(x_t), v \rangle$ {away vertex}
4: $\quad s_t \leftarrow \text{argmin}_{v \in S_t} \langle \nabla f(x_t), v \rangle$ {local FW}
5: $\quad w_t \leftarrow \text{argmin}_{v \in V(P)} \langle \nabla f(x_t), v \rangle$ {global FW}
6: $\quad$ **if** $\langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), x_t - w_t \rangle$ **then**
7: $\quad\quad d_t = a_t - s_t$
8: $\quad\quad \Lambda_t^* \leftarrow c[x_t](a_t)$
9: $\quad\quad \lambda_t \leftarrow \text{argmin}_{\lambda \in [0, \Lambda_t^*]} f(x_t - \lambda d_t)$
10: $\quad\quad$ **if** $\lambda_t < \Lambda_t^*$ **then**
11: $\quad\quad\quad S_{t+1} \leftarrow S_t$ {descent step}
12: $\quad\quad$ **else**
13: $\quad\quad\quad S_{t+1} \leftarrow S_t \setminus \{a_t\}$ {drop step}
14: $\quad\quad$ **end if**
15: $\quad$ **else**
16: $\quad\quad d_t = x_t - w_t$
17: $\quad\quad \lambda_t \leftarrow \text{argmin}_{\lambda \in [0, 1]} f(x_t - \lambda d_t)$
18: $\quad\quad S_{t+1} \leftarrow S_t \cup \{w_t\}$ (or $S_{t+1} \leftarrow \{w_t\}$ if $\lambda_t = 1$) {FW step}
19: $\quad$ **end if**
20: $\quad x_{t+1} \leftarrow x_t - \lambda_t d_t$
21: **end for**

Exact line search

α_k is the value obtained by exactly minimizing f along d_k, that is

$$\alpha_k = \arg\min_{\alpha} \phi(\alpha) = f(x_k + \alpha d_k) .$$

This is a viable option only when the function has some structure.

3.2. Convergence analysis

The following theorems provide convergence properties of the BPCG algorithm. We first state the general smooth case.

Theorem 3.1. Let P be a convex feasible domain of diameter D. Assume that f is convex and L-smooth. Let $\{x_i\}_{i=0}^T \subset P$ be the sequence given by the BPCG algorithm (Algorithm 1). Then, it holds that

$$f(x_T) - f(x^*) \leq \frac{4LD^2}{T}. \tag{3.1}$$

In the case of strongly convex functions f and polyhedral feasible regions P we obtain the following improved convergence rates. Note that in contrast to the analysis of the Pairwise Conditional Gradients algorithm in (Lacoste-Julien and Jaggi, 2015) we do not encounter swap steps.

Theorem 3.2. Let P be a polytope with pyramidal width δ and diameter D. Furthermore, let f be μ-strongly convex and L-smooth and consider the sequence $\{x_i\}_{i=0}^T \subset P$ obtained by the BPCG algorithm (Algorithm 1). Then, it

holds that

$$f(x_T) - f(x^*) \leq (f(x_0) - f(x^*)) \exp\left(-c_{f,P}\, T\right), \tag{3.2}$$

where $c_{f,P} := \frac{1}{2}\min\{\frac{1}{2}, \frac{\mu\delta^2}{4LD^2}\}$.

We prove these theorems by using the following lemmas.

Lemma 3.3 (Geometric Strong Convexity, ((Lacoste-Julien and Jaggi, 2015), Inequalities (23) and (28)))**.** Assume that f is μ-strongly convex and P is a polytope with pyramidal width δ. Then, with the notation of Algorithm 1 the following inequality holds:

$$f(x_t) - f(x^*) \leq \frac{\langle \nabla f(x_t), a_t - w_t\rangle^2}{2\mu\delta^2}. \tag{3.3}$$

Lemma 3.4. With the notation of Algorithm 1, suppose that step t is not a drop step (line 13). Let $\lambda_t^* = \frac{\langle \nabla f(x_t), d_t\rangle}{L\|d_t\|^2}$.

(a) If step t is either a FW step (line 16-18) with $\lambda_t^* < 1$ or a descent step (line 11), we have

$$f(x_t) - f(x_{t+1}) \geq \frac{\langle \nabla f(x_t),\, d_t\rangle^2}{2LD^2}. \tag{3.4}$$

(b) If step t is a FW step (line 16-18) with $\lambda_t^* \geq 1$, we have

$$f(x_t) - f(x_{t+1}) \geq \frac{1}{2}\langle \nabla f(x_t),\, d_t\rangle. \tag{3.5}$$

The proof of Lemma 3.4 is given in section B in the appendix.

Lemma 3.5. For each step t, an inequality $2\langle \nabla f(x_t),\, d_t\rangle \geq \langle \nabla f(x_t),\, a_t - w_t\rangle$ holds.

The proof of Lemma 3.5 is written in section B in the appendix.

We are now in a position to prove Theorems 3.2 and 3.1. Let $h_t := f(x_t) - f(x^*)$. Then, by the convexity of f and the definition of w_t, we have

$$h_t \leq \langle \nabla f(x_t),\, x_t - x^*\rangle \leq \langle \nabla f(x_t),\, x_t - w_t\rangle. \tag{3.6}$$

We start with the slightly more involved proof of Theorem 3.2.

Proof of Theorem 3.2. First, we focus on the case that step t is not a drop step, which is considered in Lemma 3.4. Suppose that step t is case (a) of Lemma 3.4. By combining inequality (3.4), Lemmas 3.5 and 3.3, we have

$$\begin{aligned} h_t - h_{t+1} &\geq \frac{\langle \nabla f(x_t),\, d_t\rangle^2}{2LD^2} \geq \frac{\langle \nabla f(x_t), a_t - w_t\rangle^2}{8LD^2} \\ &\geq \frac{\mu\delta^2}{4LD^2} h_t. \end{aligned} \tag{3.7}$$

Then, consider case (b) of Lemma 3.4, where $d_t = x_t - w_t$. By inequalities (3.5) and (3.6), we have

$$h_t - h_{t+1} \geq \frac{1}{2}\langle \nabla f(x_t),\, d_t\rangle \geq \frac{1}{2} h_t. \tag{3.8}$$

Therefore by letting $\hat{c}_{f,P} = \min\{\frac{1}{2}, \frac{\mu\delta^2}{4LD^2}\}$ we can deduce from (3.7) and (3.8) that

$$h_{t+1} \leq (1 - \hat{c}_{f,P}) h_t$$

if step t is not a drop step.

Next, we take the drop steps into account. If step t is a drop step, it is clear that $h_{t+1} \leq h_t$. In addition, we bound the number of the drop steps in the algorithm. Let T_{FW}, T_{desc}, and T_{drop} be the numbers of the FW steps, descent steps, and drop steps, respectively. Note that $T = T_{\mathrm{FW}} + T_{\mathrm{desc}} + T_{\mathrm{drop}}$. Since $\#S_T \geq 1$, an inequality $T_{\mathrm{drop}} \leq T_{\mathrm{FW}}$ holds. Therefore we have

$$\begin{aligned} T &= T_{\mathrm{FW}} + T_{\mathrm{desc}} + T_{\mathrm{drop}} \leq 2T_{\mathrm{FW}} + T_{\mathrm{desc}} \\ &\leq 2(T_{\mathrm{FW}} + T_{\mathrm{desc}}). \end{aligned}$$

Finally, by combining the above arguments, we have

$$\begin{aligned} h_T &\leq (1 - \hat{c}_{f,P})^{T_{\mathrm{FW}} + T_{\mathrm{desc}}} h_0 \leq (1 - \hat{c}_{f,P})^{T/2} h_0 \\ &\leq \exp\left(-\frac{\hat{c}_{f,P}}{2} T\right) h_0. \end{aligned}$$

□

The proof of Theorem 3.1 can be done in the similar way and the proof is written in section B in the appendix. Although we showed Theorem 3.1 in the Euclidean case for the simplicity of the argument, it is easy to see from the proof of Theorem 3.1 that the convergence guarantee is also applicable to the infinite-dimensional general convex feasible regions.

As shown in Theorem 3.1 and Theorem 3.2, the BPCG algorithm achieves a convergence rate which is no worse than that of the PCG. In addition, since BPCG does not exhibit swap steps, the convergence rate of BPCG including (possibly dimension-dependent) constant factors is considered to be better than that of the PCG especially when the dimension of the feasible region is high: PCG's convergence rate (see (Lacoste-Julien and Jaggi, 2015)) is of the form $h_t \leq h_0 \exp(-\rho k(t))$, where $\rho > 0$ and $k(t) \geq t/(3|\mathcal{A}|! + 1)$, where $\mathcal{A}$ is the set of vertices generating the polytope. Even for $0/1$-polytopes in n-dimensional space, this can be as bad as 2^n and the factor $(2^n)!$ is even worse. In fact, this dimension-dependence in the rate is also the reason the convergence proof of PCG does not generalize to infinite-dimensional cases. As such BPCG's

convergence rate is much more in line with that of the Away-Step Frank-Wolfe algorithm (see (Lacoste-Julien and Jaggi, 2015)). Moreover, since the iterations of BPCG include many local updates in which no new atoms are added, it is expected that the BPCG algorithm outputs sparser solutions than the PCG algorithm in terms of the support size of the supporting convex combination.

Remark 3.1. We clarify the advantages of the BPCG algorithm in comparison to other CG variants.

Compared to the vanilla CG: The BPCG algorithm ensures linear convergence for the strongly convex polyhedral case, whereas the vanilla CG (Frank and Wolfe, 1956; Levitin and Polyak, 1966) does not. Although both algorithms guarantee a $O(1/k)$ convergence for infinite-dimensional cases, we expect the BPCG algorithm to output much sparser solutions due to the aggressive dropping of unnecessary vertices from the convex combination and adding new vertices only when they are needed for primal progress.

Compared to the Away CG algorithm: BPCG and Away CG (Wolfe, 1970; Guélat and Marcotte, 1986) have the same convergence guarantee for the general smooth convex case and strongly convex polyhedral case however BPCG empirically produces significantly sparser solutions. There are also many more CG variants but typically they are not general purpose and apply special cases only.

Compared to the BCG: The constant factor $\frac{1}{2LD^2}$ of the bound (3.4) in Lemma 3.4 does not depend on the dimension of feasible domains, which is important to guarantee $O(\frac{1}{T})$ convergence in infinite-dimensional cases. The similar BCG algorithm in (Braun et al., 2019a) employs Simplex Gradient Descent (SiGD) instead of local PCG steps. However, the lower bound for the progress of SiGD includes a dimension-dependent term and we cannot guarantee $O(\frac{1}{T})$ convergence in infinite-dimensional cases in general for BCG.

In short, BPCG fixes the swap step issue of PCG, making pairwise steps the de-facto choice for CG algorithms with best-known convergence rates across the board save for the Fully-Corrective Frank-Wolfe algorithm (FCFW) (e.g., (Holloway, 1974; Jaggi, 2013)) that usually has excessive cost per iteration and is therefore rarely implemented. In fact BPCG (as well as BCG) are in some sense inexact versions of the FCFW. BPCG also applies to the infinite-dimensional setting and its (empirical) sparsity is very high making it a prime candidate for problems such as sparse signal recovery, matrix completion, and numerical integration. Moreover, BPCG inherits the right rates for sharp functions over uniformly convex sets.

3.3. Lazified version of the BPCG

We can consider the lazified version (see (Braun et al., 2017; 2019b)) of the BPCG as shown by Algorithm 2 in the appendix. It employs the estimated Frank-Wolfe gap Φ_t instead of computing the Frank-Wolfe gap in each iteration. The lazification technique helps reduce the number of access to LMOs, which improves the computational efficiency of Algorithm 2 since we only need to access the active set S_t when the pairwise gap $\langle \nabla f(x_t), a_t - s_t \rangle$ is larger than Φ_t.

The theoretical analysis of the lazified BPCG can be done in the almost same way as the proof of Theorem 3.1 in (Braun et al., 2019a) for the strongly convex case; the general smooth case follows similarly. For the descent step, the analysis differs, but we can use Lemma 3.4. For the only L-smooth case, we can show in a similar as the proof of the strongly convex case. As a result, we can show the following theorem. The proof is provided in section B in the appendix.

Theorem 3.6. Let P be a convex feasible domain with diameter D. Furthermore, let f be a L-smooth convex function and consider the sequence $\{x_i\}_{i=0}^T \subset P$ obtained from the lazified BPCG algorithm (Algorithm 2).

Case (A) If f is μ-strongly convex and P is a polytope with pyramidal width $\delta > 0$, we have

$$f(x_T) - f(x^*) = O\left(\exp\left(-cT\right)\right) \quad (T \to \infty)$$

for a constant $c > 0$ independent of T.

Case (B) If f is only convex and L-smooth, we have

$$f(x_T) - f(x^*) = O\left(\frac{1}{T}\right) \quad (T \to \infty).$$

As well as Theorem 3.1, it is easy to see from the proof in section B that the convergence guarantee of Case (B) in Theorem 3.6 can be also applicable to infinite-dimensional general convex feasible domains.

4. Application to kernel quadrature

Kernel herding is a well-known method for kernel quadrature and can be regarded as a Frank-Wolfe method on a reproducing kernel Hilbert space (RKHS). In the following we will use the BPCG algorithm for kernel herding with aim to exploit its sparsity. To describe this application, we consider the following notation for kernel quadrature on an RHKS. Let $\Omega \subset \mathbf{R}^d$ be a compact region and let a continuous function $K : \Omega \times \Omega \to \mathbf{R}$ be a symmetric positive-definite kernel. Then, the RKHS $\mathcal{H}_K(\Omega)$ for K on Ω is uniquely determined. We denote the inner product in $\mathcal{H}_K(\Omega)$ by $\langle \cdot, \cdot \rangle_K$. We consider integration of a function $f \in \mathcal{H}_K(\Omega)$ with respect to a probability measure on Ω. By letting $\mathscr{M}^+(1)$ be a set of

probability measures on Ω, we consider $I = \int_\Omega f(x) d\mu(x)$ for $\mu \in \mathscr{M}^+(1)$. To approximate I, we consider a quadrature formula $I_{X,W} = \sum_{x\in X} w_x f(x)$ by taking finite sets $X \subset \Omega$ and $W = \{w_x\}_{x\in X} \subset [0,1]$ with $\sum_{x\in X} w_x = 1$. The formula I_X can be regarded as an integral of f by the discrete probability measure $\xi = \sum_{x\in X} w_x \delta_x \in \mathscr{M}^+(1)$, where δ_x is the Dirac measure for $x \in X$. According to the reproducing property of the RKHS, the Dirac measure δ_x corresponds to the function $K(x,\cdot) \in \mathcal{H}_K(\Omega)$. The accuracy of the formula $I_{X,W}$ is estimated by the maximum mean discrepancy (MMD) $\gamma_K(\xi,\mu) := \mathscr{E}_K(\xi-\mu)^{1/2}$, where

$$\mathscr{E}_K(\eta) := \int_\Omega \int_\Omega K(x,y)\,\mathrm{d}\eta(x)\,\mathrm{d}\eta(y)$$

is the K-energy of a signed measure η.

To construct a good discrete measure ξ, we consider minimization of the squared MMD $F_{K,\mu}(\nu) := \gamma_K(\nu,\mu)^2$ as a function of $\nu \in \mathscr{M}^+(1)$. Now, kernel herding is essentially running a Frank-Wolfe method to minimize $F_{K,\mu}$ over $\mathscr{M}^+(1)$, in which we start with a Dirac measure $\xi = \delta_{x_0}$ and sequentially add new Dirac measures $\delta_{x_1}, \delta_{x_2}, \dots$ and form iterates as convex combinations. As such, we can naturally apply BPCG to the kernel herding. Here we note that the function $F_{K,\mu}$ is a 2-smooth and convex function on $\mathscr{M}^+(1)$. In particular, we have

$$\begin{aligned} &F_{K,\mu}(\theta + \alpha(\eta-\zeta)) \\ &= F_{K,\mu}(\theta) + \alpha \langle \nabla F_{K,\mu}(\theta),\, \eta-\zeta\rangle_K + \alpha^2\, \mathscr{E}_K(\eta-\zeta) \end{aligned} \tag{4.1}$$

for any $\alpha \in \mathbf{R}$ and $\theta, \eta, \zeta \in \mathscr{M}^+(1)$ such that $\theta + \alpha(\eta-\zeta) \in \mathscr{M}^+(1)$. In addition, if $K(x,x) = 1$ and $K(x,y) \geq 0$ for any $x, y \in \Omega$, we have $\mathscr{E}_K(\eta-\zeta) \leq 2$ for any $\eta, \zeta \in \mathscr{M}^+(1)$. Therefore the term $\mathscr{E}_K(\eta-\zeta)$ is bounded in (4.1). This fact means that the diameter of $\mathscr{M}^+(1)$ with respect to $\mathscr{E}_K$ is bounded.

We note that in this problem setting, the function $\int_\Omega K(x,\cdot)\,\mathrm{d}\mu(x)$ which is the embedding of μ to $\mathcal{H}_K$ satisfies $\int_\Omega K(x,\cdot)\,\mathrm{d}\mu(x) \in \overline{\mathrm{conv}(\{K(x,\cdot) \mid x \in \Omega\})}$, where the closure is taken with respect to the norm of $\mathcal{H}_K$ (see (Tsuji and Tanaka, 2021)).

In Algorithm 3 we describe the BPCG algorithm applied to kernel herding. In addition, we present its lazified version Algorithm 4 in the appendix. In these algorithms, we can regard the Dirac measures δ_x (i.e., $K(x,\cdot)$) as vertices or atoms in the RKHS $\mathcal{H}_K(\Omega)$. We assume the existence of the solution in line 5 in Algorithm 3.

Recall that the function $F_{K,\mu}$ is 2-smooth and convex as indicated in (4.1). Then, under the assumption that $K(x,x) = 1$ and $K(x,y) \geq 0$ for any $x, y \in \Omega$, we obtain the following theorem in a similar manner to Theorems 3.1 and 3.6.

Algorithm 3 BPCG algorithm for kernel herding

Require: the function $F_{K,\mu}$, start measure $\xi_0 \in \mathscr{M}^+(1)$ with $\operatorname{supp}\xi_0 = \{x_0\}$
Ensure: discrete measures $\xi_1, \dots, \xi_T \in \mathscr{M}^+(1)$

1: $X_0 \leftarrow \operatorname{supp}\xi_0$
2: **for** $t = 0$ to $T-1$ **do**
3: $\quad x_t^{\mathrm{A}} \leftarrow \operatorname{argmax}_{x\in X_t} \langle \nabla F_{K,\mu}(\xi_t),\, \delta_x\rangle_K$
4: $\quad x_t^{\mathrm{S}} \leftarrow \operatorname{argmin}_{x\in X_t} \langle \nabla F_{K,\mu}(\xi_t),\, \delta_x\rangle_K$
5: $\quad x_t^{\mathrm{W}} \leftarrow \operatorname{argmin}_{x\in \Omega} \langle \nabla F_{K,\mu}(\xi_t),\, \delta_x\rangle_K$
6: $\quad$ **if** $\left\langle \nabla F_{K,\mu}(\xi_t),\, \delta_{x_t^{\mathrm{A}}} - \delta_{x_t^{\mathrm{S}}} \right\rangle_K \geq \left\langle \nabla F_{K,\mu}(\xi_t),\, \xi_t - \delta_{x_t^{\mathrm{W}}} \right\rangle_K$ **then**
7: $\quad\quad \eta_t \leftarrow \delta_{x_t^{\mathrm{A}}} - \delta_{x_t^{\mathrm{S}}}$
8: $\quad\quad \alpha_t \leftarrow \operatorname{argmin}_{\alpha\in[0,\, \xi_t(\{x_t^{\mathrm{A}}\})]} F_{K,\mu}(\xi_t - \alpha\,\eta_t)$
9: $\quad\quad$ **if** $\alpha_t < \xi_t(\{x_t^{\mathrm{A}}\})$ **then**
10: $\quad\quad\quad X_{t+1} \leftarrow X_t$
11: $\quad\quad$ **else**
12: $\quad\quad\quad X_{t+1} \leftarrow X_t \setminus \{x_t^{\mathrm{A}}\}$
13: $\quad\quad$ **end if**
14: $\quad$ **else**
15: $\quad\quad \eta_t \leftarrow \xi_t - \delta_{x_t^{\mathrm{W}}}$
16: $\quad\quad \alpha_t \leftarrow \operatorname{argmin}_{\alpha\in[0,1]} F_{K,\mu}(\xi_t - \alpha\eta_t)$
17: $\quad\quad X_{t+1} \leftarrow X_t \cup \{x_t^{\mathrm{W}}\}$
18: $\quad$ **end if**
19: $\quad \xi_{t+1} \leftarrow \xi_t - \alpha_t\eta_t$
20: **end for**

Theorem 4.1. Suppose that $K(x,x) = 1$ and $K(x,y) \geq 0$ for any $x, y \in \Omega$. Let $\{\xi_t\}_{t=0}^T \in \mathscr{M}^+(1)$ be the discrete probability measure on Ω given by Algorithms 3 or Algorithm 4. Then, we have

$$\gamma_K(\xi_T,\mu)^2 = O\left(\frac{1}{T}\right) \quad (T \to \infty).$$

5. Numerical experiments

To demonstrate the practical effectiveness of the BPCG algorithm, we compare the performance of BPCG to other state-of-the-art algorithms. To this end we present first numerical experiments in finite-dimensional spaces and compare the performance to (vanilla) Conditional Gradients, Away-Step Conditional Gradients, and Pairwise Conditional Gradients. We then consider the kernel herding setting and compare BPCG to other kernel quadrature methods. In Appendix C we present a minor modification of BPCG's step selection criterion that further improves sparsity, while changing the convergence rate only by a constant factor. The BPCG algorithm has been also integrated into the FrankWolfe.jl Julia package (Besançon et al., 2022) and is now the *de facto* default choice for active set based FW variants.

5.1. Finite dimensional optimization problems

We report both general primal-dual convergence behavior in iterations and time as well as consider the sparsity of the iterates. In the all experiments, the same initial point x_0 is used for the five methods in each experiment. For the lazified BPCG, we use the accuracy parameter $J = 2$.

PROBAILITY SIMPLEX

Let $\Delta(n)$ be a probability simplex, i.e.,

$$\Delta(n) := \left\{ x \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} x_i = 1, x_i \geq 0 \; (i = 1, \ldots, n) \right\}.$$

We consider the following optimization problem:

$$\min_{x \in \mathbb{R}^n} \|x - x_0\|_2^2$$
$$\text{s.t. } x \in \Delta(n),$$

where $x_0 \in \Delta(n)$. Figure 1 shows the result for the case $n = 200$. The x-axes in Figure 1 exhibit the number of iterations and computational time and the y-axes exhibit the primal and dual gaps. In terms of convergence in iterations, BPCG has almost the same performance as the Pairwise variant, however in terms of computational time BPCG has the best performance. In this case the LMO is so cheap that using the lazified BPCG algorithm is not advantageous. In

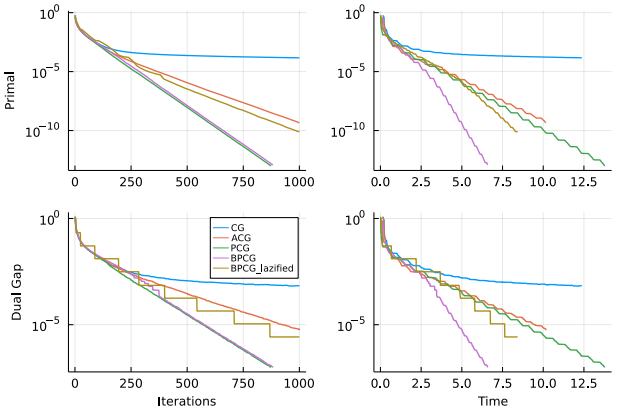


Figure 1. Probability simplex $n = 200$

addition, to assess the sparsity of generated solution of each algorithm, we also considered larger setups with $n = 500$. The result are shown in Figure 2. The x-axes exhibit the number of atoms that form x_t and the y-axes exhibit the primal and dual gaps. It can be observed that both BPCG methods perform slightly (but not much) better than the others with respect to the sparsity, which is expected in this case due to known lower bounds for this type of instance (see (Jaggi, 2013)).

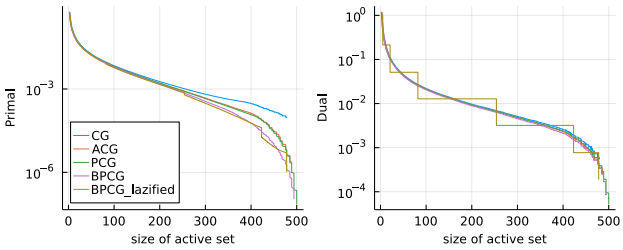


Figure 2. Sparsity: Probability simplex $n = 500$

BIRKHOFF POLYTOPE

Let $B(n)$ be the Birkhoff polytope in $\mathbb{R}^{n \times n}$, which is the set of $n \times n$ real values matrices whose entries are all non-negative and whose rows and columns each add up to 1. We consider the following optimization problem

$$\min_{X \in \mathbb{R}^{n \times n}} \|X - X_0\|_2^2$$
$$\text{s.t. } X \in B(n),$$

where $X_0 \in \mathbb{R}^{n \times n}$ and $\|\cdot\|_2$ is the Frobenius norm. The result is Figure 3: For the primal value, the five methods

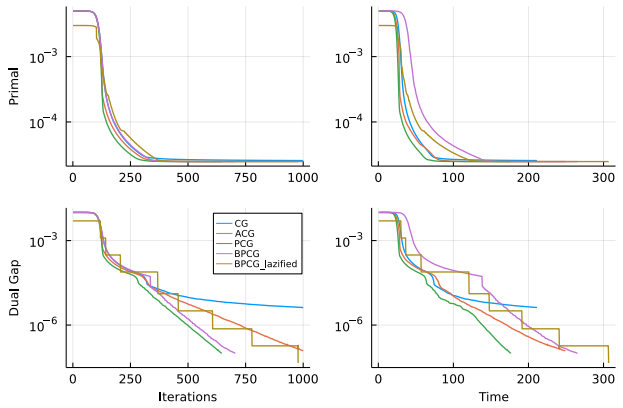


Figure 3. Birkhoff polytope $n = 200$

have almost same performance. For the dual gap, we see that the Pairwise variant and BPCG are superior to other methods for the number of iterations and for computational time. Here the pairwise variant performed best.

In addition, we compare solution sparsity in Figure 4. The two BPCG algorithms perform much better than other methods in terms of the sparsity.

MATRIX COMPLETION

We also considered matrix completion instances over the spectrahedron $S = \{X \succeq 0 \mid \mathrm{Tr}(X) = 1\} \subset \mathbb{R}^{n \times n}$. The

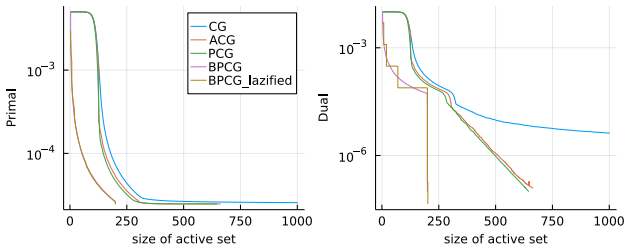


Figure 4. Sparsity: Birkhoff polytope $n = 200$

problem is written as

$$\min_{X \in S} \sum_{(i,j) \in L} (X_{i,j} - T_{i,j})^2,$$

where $D = \{T_{i,j} \mid (i,j) \in L\}$ is an observed data set. We used the data set MovieLens Latest Datasets `http://files.grouplens.org/datasets/movielens/ml-latest-small.zip`. The result is given in Figure 5. We can observe the effectiveness of

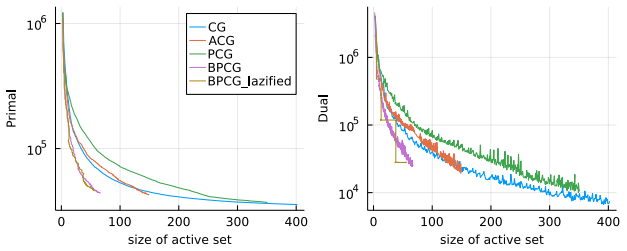


Figure 5. Sparsity: Matrix completion

BPCG methods in obtaining very sparse solutions.

5.2. Kernel herding

Next, we show the results of the application of BPCG to the kernel herding. We compare the BPCG algorithm and the lazified version with the ordinary kernel herding methods "linesearch" and "equal-weight" which correspond to the vanilla Frank Wolfe algorithm whose step size α_t is defined by line search and $\alpha_t = \frac{1}{t+1}$. In addition, we compare BPCG with the Away and Pairwise variants of "linesearch"; recall that for the latter a theoretical convergence in the infinite-dimensional case is not known. To evaluate the application of BPCG to kernel herding as a quadrature method, we compare it with the popular kernel quadrature method *Sequential Bayesian Quadrature (SBQ)* (Huszár and Duvenaud, 2012) as well as the Monte Carlo method. In the kernel herding experiments, we use the accuracy parameter $J = 1$ for the lazified BPCG.

5.2.1. Matérn kernel case

We consider the case that the kernel is the Matérn kernel, which has the form

$$K(x,y) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|x-y\|_2}{\rho} \right)^{\nu} B_\nu \left(\sqrt{2\nu} \frac{\|x-y\|_2}{\rho} \right),$$

where B_ν is the modified Bessel function of the second kind and ρ and ν are positive parameters. The Matérn kernel is closely related to Sobolev space and the RKHS $\mathcal{H}_K$ generated by the kernel with smoothness ν is norm equivalent to the Sobolev space with smoothness $s = \nu + \frac{d}{2}$ (see e.g., (Kanagawa et al., 2018; Wendland, 2004)). In addition, the optimal convergence rate of the MMD in Sobolev space is known as $n^{-\frac{s}{d}}$ (Novak, 2006). In this section, we use the parameter $(\rho, \nu) = (\sqrt{3}, \frac{3}{2}), (\sqrt{5}, \frac{5}{2})$ since the kernel has explicit forms with these parameters.

The domain Ω is $[-1,1]^d$ and the probability distribution is a uniform distribution. First, we see the case $\nu = \frac{3}{2}$ and $d = 2$. In this case, the optimal rate is $n^{-\frac{5}{4}}$. The result is Figure 6. Figure 6 shows the convergence of MMD between apprroximate measures and the target measure for the number of iterations, computational time and the number of nodes that form the approximate measure ξ_t. For sparsity, BPCG methods have significant performance and they achieve the convergence speed comparable to the optimal rate. For computational time, the lazified BPCG have good performance. This is because in the lazified algorithm, we only access the active set and this reduces the computational cost considerably. Although, BPCG methods are not superior to other methods for the number of the iterations, we can see the effectiveness of BPCG methods.

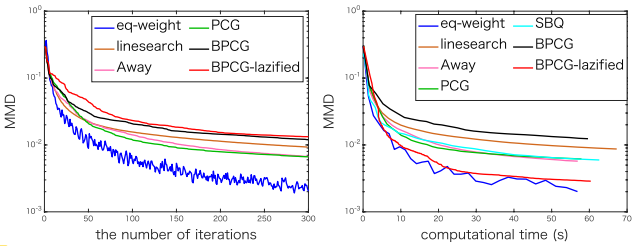
Moreover, the same good performance for sparsity can be seen in the case $\nu = \frac{5}{2}$ and $d = 2$, which is shown in Figure 7. BPCG algorithms also achieves the convergence rate which is competitive with the optimal rate $n^{-\frac{7}{4}}$.

5.2.2. Gaussian kernel

Next, we treat the gaussian kernel $K(x,y) = \exp(-\|x - y\|_2^2)$. The domain is $\Omega = [-1,1]^d$ and the density function of the distribution on Ω is $\frac{1}{C}\exp\left(-\|x\|^2\right)$, where $C = \int_\Omega \exp\left(-\|x\|^2\right) \mathrm{d}x$. In this case, we see the case $d = 2$ and the result is Figure 7. We note that $\exp(-n^{\frac{1}{2}})$ is the exponential factor of the upper bound of convergence rate which appears in several works on gaussian kernel, for example, (Wendland, 2004; Karvonen et al., 2021). We see that BPCG methods are competitive with SBQ and the $\exp(-n^{\frac{1}{2}})$ rate and outperform significantly other methods.

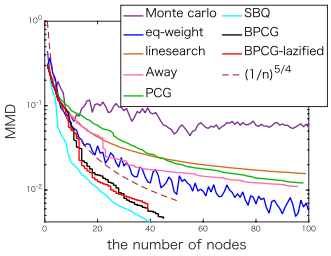
6. Conclusion

The proposed Blended Pairwise Conditional Gradient (BPCG) algorithms yield very sparse solutions fast with very



(a) MMD for the number of iterations

(b) MMD for computational time



(c) MMD for the number of nodes

Figure 6. Matérn kernel ($\nu = 3/2$)

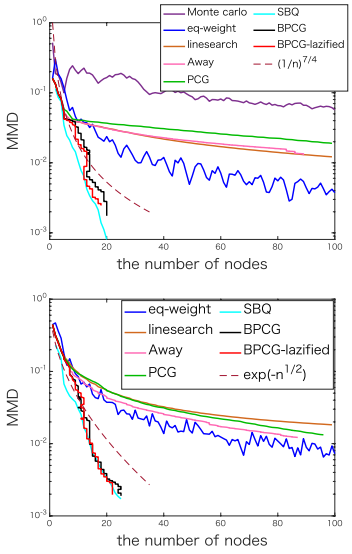


Figure 7. Matérn kernel ($\nu = 5/2$) (top) and Gaussian kernel (bottom)

high speed both in convergence in iterations as well as time. It does not exhibit any swap steps, which provides state-of-the-art convergence guarantees for the strongly convex case as well as applies application to the infinite-dimensional case. We have analyzed its convergence property and exemplified its real performance via the numerical experiments. The BPCG works well for application to kernel herding, the infinite-dimensional case, in that it provides small MMD with a small number of nodes. A main avenue for future work will be tighter estimates for the convergence rate of the BPCG in various cases, in particular with regards to sparsity.

References

Bach, F., Lacoste-Julien, S., and Obozinski, G. (2012). On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1355–1362, Madison, WI, USA. Omnipress.

Besançon, M., Carderera, A., and Pokutta, S. (2022). FrankWolfe.jl: a high-performance and flexible toolbox for Frank-Wolfe algorithms and Conditional Gradients. *INFORMS Journal on Computing*.

Braun, G., Pokutta, S., Tu, D., and Wright, S. (2019a). Blended conditional gradients: the unconditioning of conditional gradients. In *Proceedings of the 36th International Conference on Machine Learning (PMLR)*, volume 97, pages 735–743.

Braun, G., Pokutta, S., and Zink, D. (2017). Lazifying conditional gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, pages 566–575.

Braun, G., Pokutta, S., and Zink, D. (2019b). Lazifying conditional gradient algorithms. *Journal of Machine Learning Research (JMLR)*, 20(71):1–42.

Briol, F.-X., Oates, C. J., Girolami, M., Osborne, M. A., and Sejdinovic, D. (2019). Rejoinder: Probabilistic Integration: A Role in Statistical Computation? *Statistical Science*, 34(1):38 – 42.

Chen, Y., Welling, M., and Smola, A. (2010). Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 109–116, Arlington, Virginia, USA. AUAI Press.

Combettes, C. W. and Pokutta, S. (2020). Boosting Frank-Wolfe by chasing gradients. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2111–2121.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110.

Guélat, J. and Marcotte, P. (1986). Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119.

Holloway, C. A. (1974). An extension of the frank and wolfe method of feasible directions. *Mathematical Programming*, 6(1):14–27.

Huszár, F. and Duvenaud, D. (2012). Optimally-weighted herding is bayesian quadrature. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pages 377–386, Arlington, Virginia, USA. AUAI Press.

Jaggi, M. (2013). Revisiting Frank-Wolfe: projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 427–435.

Kanagawa, M., Hennig, P., Sejdinovic, D., and Sriperumbudur, B. K. (2018). Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*.

Karvonen, T., Oates, C., and Girolami, M. (2021). Integration in reproducing kernel hilbert spaces of gaussian kernels. *Mathematics of Computation*.

Kerdreux, T., d'Aspremont, A., and Pokutta, S. (2019). Restarting Frank-Wolfe. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 1275–1283.

Kerdreux, T., d'Aspremont, A., and Pokutta, S. (2021). Projection-free optimization on uniformly convex sets. In *Proc. Artificial Intelligence and Statistics (AISTATS)*.

Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of Frank-Wolfe optimization variants. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 496–504. Curran Associates, Inc.

Lacoste-Julien, S., Lindsten, F., and Bach, F. (2015). Sequential kernel herding: Frank-wolfe optimization for particle filtering. In *Artificial Intelligence and Statistics*, pages 544–552. PMLR.

Levitin, E. S. and Polyak, B. T. (1966). Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50.

Mortagy, H., Gupta, S., and Pokutta, S. (2020). Walking in the Shadow: A New Perspective on Descent Directions for Constrained Minimization. *to appear in Proceedings of NeurIPS*.

Novak, E. (2006). Deterministic and stochastic error bounds in numerical analysis.

Oettershagen, J. (2017). *Construction of optimal cubature algorithms with applications to econometrics and uncertainty quantification*.

Pedregosa, F., Negiar, G., Askari, A., and Jaggi, M. (2020). Linearly convergent Frank–Wolfe with backtracking linesearch. In *Proc. Artificial Intelligence and Statistics (AISTATS)*.

Rinaldi, F. and Zeffiro, D. (2020). A unifying framework for the analysis of projection-free first-order methods under a sufficient slope condition. *arXiv preprint arXiv:2008.09781*.

Tsuji, K. and Tanaka, K. (2021). Acceleration of the kernel herding algorithm by improved gradient approximation. *arXiv preprint arXiv:2105.07900*.

Welling, M. (2009). Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128.

Wendland, H. (2004). *Scattered data approximation*, volume 17. Cambridge university press.

Wolfe, P. (1970). Convergence theory in nonlinear programming. In *Integer and Nonlinear Programming*, pages 1–36. North-Holland, Amsterdam.

A. Algorithms

Algorithm 2 shown below is a lazified version of the BPCG algorithm.

Algorithm 2 Lazified BPCG

Require: convex smooth function f, start vertex $x_0 \in V(P)$, accuracy $J \geq 1$.
Ensure: points $x_1, \ldots, x_T$ in P

$\Phi_0 \leftarrow \max_{v \in P} \langle \nabla f(x_0),\, x_0 - v \rangle / 2$
$S_0 \leftarrow \{x_0\}$
for $t = 0$ to $T - 1$ **do**
  $a_t \leftarrow \operatorname{argmax}_{v \in S_t} \langle \nabla f(x_t), v \rangle$ {away vertex}
  $s_t \leftarrow \operatorname{argmin}_{v \in S_t} \langle \nabla f(x_t), v \rangle$ {local FW}
  if $\langle \nabla f(x_t), a_t - s_t \rangle \geq \Phi_t$ **then**
    $d_t = a_t - s_t$
    $\Lambda_t^* \leftarrow c[x_t](a_t)$
    $\lambda_t \leftarrow \operatorname{argmin}_{\lambda \in [0,\, \Lambda_t^*]} f(x_t - \lambda d_t)$
    $x_{t+1} \leftarrow x_t - \lambda_t d_t$
    $\Phi_{t+1} \leftarrow \Phi_t$
    if $\lambda_t < \Lambda_t^*$ **then**
      $S_{t+1} \leftarrow S_t$ {descent step}
    else
      $S_{t+1} \leftarrow S_t \setminus \{a_t\}$ {drop step}
    end if
  else
    $w_t \leftarrow \operatorname{argmin}_{v \in V(P)} \langle \nabla f(x_t), v \rangle$ {global FW}
    if $\langle \nabla f(x_t), x_t - w_t \rangle \geq \Phi_t / J$ **then**
      $d_t = x_t - w_t$
      $\lambda_t \leftarrow \operatorname{argmin}_{\lambda \in [0,1]} f(x_t - \lambda d_t)$
      $x_{t+1} \leftarrow x_t - \lambda_t d_t$
      $\Phi_{t+1} \leftarrow \Phi_t$
      $S_{t+1} \leftarrow S_t \cup \{w_t\}$ {FW step}
    else
      $x_{t+1} \leftarrow x_t$
      $\Phi_{t+1} \leftarrow \Phi_t / 2$
      $S_{t+1} \leftarrow S_t$ {gap step}
    end if
  end if
end for

The fundamental idea of lazification is reducing the number of accesses to the LMO (the oracle providing the global FW vertex in Line 18). By introducing Φ_t in place of the FW gap $\langle \nabla f(x_t), x_t - w_t \rangle$, we only need to access the active set as long as the inequality in Line 6 is satisfied. In this case, we do not need to access the LMO. If the inequality is not satisfied and the FW gap is larger than Φ_t/J where J is the quantity of accuracy, we take an FW step. Otherwise, Φ_t is updated. The lazification is effective especially in the case that the LMO is heavy.

Algorithm 4 shown below is a lazified BPCG algorithm applied to kernel herding.

Algorithm 4 Lazified BPCG algorithm for kernel herding

Require: the function $F_{K,\mu}$, start measure $\xi_0 \in \mathscr{M}^+(1)$ with $\operatorname{supp} \xi_0 = \{x_0\}$, accuracy $J \geq 1$
Ensure: discrete measures $\xi_1, \ldots, \xi_T \in \mathscr{M}^+(1)$
1: $\Phi_0 \leftarrow \max_{x \in \Omega} \langle \nabla F_{K,\mu}(\xi_0), \xi_0 - \delta_x \rangle_K / 2$
2: $X_0 \leftarrow \operatorname{supp} \xi_0$
3: **for** $t = 0$ to $T - 1$ **do**
4: $\quad x_t^{\mathrm{A}} \leftarrow \operatorname{argmax}_{x \in X_t} \langle \nabla F_{K,\mu}(\xi_t), \delta_x \rangle_K$
5: $\quad x_t^{\mathrm{S}} \leftarrow \operatorname{argmin}_{x \in X_t} \langle \nabla F_{K,\mu}(\xi_t), \delta_x \rangle_K$
6: $\quad$ **if** $\left\langle \nabla F_{K,\mu}(\xi_t), \delta_{x_t^{\mathrm{A}}} - \delta_{x_t^{\mathrm{S}}} \right\rangle_K \geq \Phi_t$ **then**
7: $\quad\quad \eta_t \leftarrow \delta_{x_t^{\mathrm{A}}} - \delta_{x_t^{\mathrm{S}}}$
8: $\quad\quad \alpha_t \leftarrow \operatorname{argmin}_{\alpha \in [0,\, \xi_t(\{x_t^{\mathrm{A}}\})]} F_{K,\mu}(\xi_t - \alpha\, \eta_t)$
9: $\quad\quad \xi_{t+1} \leftarrow \xi_t - \alpha_t\, \eta_t$
10: $\quad\quad \Phi_{t+1} \leftarrow \Phi_t$
11: $\quad\quad$ **if** $\alpha_t < \xi_t(\{x_t^{\mathrm{A}}\})$ **then**
12: $\quad\quad\quad X_{t+1} \leftarrow X_t$
13: $\quad\quad$ **else**
14: $\quad\quad\quad X_{t+1} \leftarrow X_t \setminus \{x_t^{\mathrm{A}}\}$
15: $\quad\quad$ **end if**
16: $\quad$ **else**
17: $\quad\quad x_t^{\mathrm{W}} \leftarrow \operatorname{argmin}_{x \in \Omega} \langle \nabla F_{K,\mu}(\xi_t), \delta_x \rangle_K$
18: $\quad\quad$ **if** $\left\langle \nabla F_{K,\mu}(\xi_t), \xi_t - \delta_{x_t^{\mathrm{W}}} \right\rangle_K \geq \Phi_t / J$ **then**
19: $\quad\quad\quad \eta_t \leftarrow \xi_t - \delta_{x_t^{\mathrm{W}}}$
20: $\quad\quad\quad \alpha_t \leftarrow \operatorname{argmin}_{\alpha \in [0,1]} F_{K,\mu}(\xi_t - \alpha \eta_t)$
21: $\quad\quad\quad \xi_{t+1} \leftarrow \xi_t - \alpha_t \eta_t$
22: $\quad\quad\quad \Phi_{t+1} \leftarrow \Phi_t$
23: $\quad\quad\quad X_{t+1} \leftarrow X_t \cup \{x_t^{\mathrm{W}}\}$
24: $\quad\quad$ **else**
25: $\quad\quad\quad \xi_{t+1} \leftarrow \xi_t$
26: $\quad\quad\quad \Phi_{t+1} \leftarrow \Phi_t / 2$
27: $\quad\quad\quad X_{t+1} \leftarrow X_t$
28: $\quad\quad$ **end if**
29: $\quad$ **end if**
30: **end for**

B. Proofs

Proof of Lemma 3.4

Proof. For $\lambda \geq 0$, it follows from the L-smoothness that

$$f(x_t - \lambda d_t) \leq f(x_t) - \lambda \langle \nabla f(x_t),\, d_t \rangle + \frac{\lambda^2}{2} L \|d_t\|^2. \tag{B.1}$$

(a) We begin with a FW step with $\lambda_t^* < 1$. By letting $\lambda = \lambda_t^*$ in the RHS of (B.1), we have inequality (3.4) as follows.

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) - \frac{\langle \nabla f(x_t),\, d_t \rangle^2}{2L\|d_t\|^2} \\ &\leq f(x_t) - \frac{\langle \nabla f(x_t),\, d_t \rangle^2}{2LD^2}. \end{aligned} \tag{B.2}$$

Next, consider a descent step with $\lambda_t^* < \Lambda_t^*$. By $\lambda_t = \operatorname{argmin}_{\lambda \in [0,\, \Lambda_t^*]} f(x_t - \lambda d_t)$ and $\lambda_t^* < \Lambda_t^*$, we have $f(x_{t+1}) \leq f(x_t - \lambda_t^* d_t)$. By letting $\lambda = \lambda_t^*$ in the RHS of (B.1), we derive the desired inequality.

Finally, consider a descent step with $\lambda_t^* \geq \Lambda_t^*$. Since step t is not a drop step, $\Lambda_t^* > \lambda_t$ holds. Here λ_t is a global minimizer of the convex function $f(x_t - \lambda d_t)$. Therefore we have $f(x_{t+1}) \leq f(x_t - \lambda_t^* d_t)$ and this RHS is bounded by that of (B.2) owing to (B.1).

(b) The condition $\lambda_t^* = \frac{\langle \nabla f(x_t),\, d_t \rangle}{L\|d_t\|^2} \geq 1$ implies $\langle \nabla f(x_t),\, d_t \rangle \geq L\|d_t\|^2$. By letting $\lambda = 1$ in the RHS of (B.1), we have inequality (3.5) as follows.

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) - \langle \nabla f(x_t),\, d_t \rangle + \frac{1}{2} L \|d_t\|^2 \\ &\leq f(x_t) - \frac{1}{2} \langle \nabla f(x_t),\, d_t \rangle. \end{aligned}$$

□

Proof of Lemma 3.5

Proof. First, suppose that step t is the pairwise step. By the definition of the algorithm, we have $\langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), x_t - w_t \rangle$. In addition, by the definitions of x_t and s_t, we have

$$\langle \nabla f(x_t), x_t \rangle \geq \langle \nabla f(x_t), s_t \rangle . \tag{B.3}$$

Hence we have $\langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), s_t - w_t \rangle$ and by adding the LHS of this inequality to both sides we have

$$2 \langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), a_t - w_t \rangle .$$

Next, suppose that step t is the Frank-Wolfe step. By the definition of the algorithm, we have $\langle \nabla f(x_t), x_t - w_t \rangle > \langle \nabla f(x_t), a_t - s_t \rangle$. It follows from this inequality and (B.3) that $\langle \nabla f(x_t), x_t - w_t \rangle > \langle \nabla f(x_t), a_t - x_t \rangle$. By adding the LHS of this inequality to both sides we have

$$2 \langle \nabla f(x_t), x_t - w_t \rangle > \langle \nabla f(x_t), a_t - w_t \rangle .$$

□

Proof of Theorem 3.1

Proof. First, suppose that step t is a descent step. By the algorithm of BPCG, it holds

$$\langle \nabla f(x_t), d_t \rangle \geq \langle \nabla f(x_t), x_t - w_t \rangle .$$

Combining this with Lemma 3.4 (a) and $0 \leq h_t \leq \langle \nabla f(x_t), x_t - w_t \rangle$ ((3.6)), we have

$$\begin{aligned} h_t - h_{t+1} &\geq \frac{1}{2LD^2} \langle \nabla f(x_t), d_t \rangle^2 \\ &\geq \frac{1}{2LD^2} \langle \nabla f(x_t), x_t - w_t \rangle^2 \\ &\geq \frac{1}{2LD^2} h_t^2. \end{aligned} \tag{B.4}$$

Next consider the case where step t is a FW step. First, by the L-smoothness of f and (3.6), we have

$$\begin{aligned} f(x_t - \lambda(x_t - w_t)) &\leq f(x_t) - \lambda \langle \nabla f(x_t), x_t - w_t \rangle + \frac{L}{2} \lambda^2 \|x_t - w_t\|^2 \\ &\leq f(x_t) - \lambda \langle \nabla f(x_t), x_t - w_t \rangle + \frac{\lambda^2}{2} LD^2 \\ &\leq f(x_t) - \lambda h_t + \frac{\lambda^2}{2} LD^2 \end{aligned}$$

for $\lambda \geq 0$. Subtracting $f(x^*)$ from both sides, we get

$$f(x_t - \lambda(x_t - w_t)) - f(x^*) \leq h_t - \lambda h_t + \frac{\lambda^2}{2} LD^2. \tag{B.5}$$

Consider the following two cases:

(i) $h_t \leq LD^2$

By the definition of λ_t,

$$h_{t+1} = f(x_t - \lambda_t(x_t - w_t)) - f(x^*) \leq f(x_t - \lambda(x_t - w_t)) - f(x^*) \tag{B.6}$$

for $\lambda \in [0, 1]$. Using (B.5) and (B.6) for $\lambda = \frac{h_t}{LD^2} \leq 1$, we have

$$h_{t+1} \leq f\left(x_t - \frac{h_t}{LD^2}(x_t - w_t)\right) - f(x^*) \leq h_t - \frac{h_t^2}{2LD^2}. \tag{B.7}$$

(ii) $h_t \geq LD^2$

Combining (B.5) with (B.6) for $\lambda = 1$, we have

$$h_{t+1} \leq f\left(x_t - 1 \cdot (x_t - w_t)\right) - f(x^*) \leq \frac{LD^2}{2}. \tag{B.8}$$

By (B.7) and (B.8), we have

$$h_{t+1} \leq \begin{cases} h_t - \frac{h_t^2}{2LD^2} & (h_t \leq LD^2), \\ \frac{LD^2}{2} \leq \frac{h_t}{2} & (\text{otherwise}). \end{cases} \tag{B.9}$$

For a iteration T, we define $T_{\mathrm{FW}}, T_{\mathrm{desc}}$ and T_{drop} in the same way as the proof of Theorem 3.2. Using (B.4) and (B.9), we can show

$$h_T \leq \frac{2LD^2}{T_{\mathrm{desc}} + T_{\mathrm{FW}}}. \tag{B.10}$$

This can be shown just in the same way as the proof of Corollary 4.2 in (Braun et al., 2019a); the value $4L_f$ in the proof is replaced by $2LD^2$ in this case.

Finally, as shown in the proof of Theorem 3.2, $T \leq 2(T_{\mathrm{desc}} + T_{\mathrm{FW}})$ holds. By substituting this to (B.10), we have

$$h_T \leq \frac{4LD^2}{T}.$$

□

Proof of Theorem 3.6

Proof. The proof tracks that of (Braun et al., 2019a), especially for Case(A).

We first consider Case (B). In the same way as (Braun et al., 2019a), we divide the iteration into sequences of epochs that are demarcated by the *gap steps* in which the value Φ_t is halved. We bound the number of iterations in each epoch.

By the convexity of f and the definition of w_t, we have

$$f(x_t) - f(x^*) \leq \langle \nabla f(x_t), x_t - x^* \rangle \leq \langle \nabla f(x_t), x_t - w_t \rangle .$$

If iteration $t-1$ is a gap step, it holds

$$f(x_t) - f(x^*) \leq \langle \nabla f(x_t), x_t - w_t \rangle \leq \frac{2\Phi_t}{J} \leq 2\Phi_t. \tag{B.11}$$

We note that (B.11) also holds at $t = 0$ by the definition of Φ_0. By (B.11), if $2\Phi_t \leq \epsilon$ holds for $\epsilon > 0$, the primal gap $f(x_t) - f(x^*)$ is upper bounded by ϵ. Therefore, the total number of epochs N_Φ to achieve $f(x_t) - f(x^*) \leq \epsilon$ is bounded in the following way:

$$N_\Phi \leq \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil . \tag{B.12}$$

Next, we consider the epoch that starts from iteration t and use the notation u to index the iterations within the epoch. We note that $\Phi_t = \Phi_u$ within the epoch.

We divide each iteration into three cases according to types of steps. First, consider the case u iteration is a FW step that means $d_u = x_u - w_u$. Using Lemma 3.4 and the condition $\langle \nabla f(x_u), x_u - w_u \rangle \geq \Phi_u / J = \Phi_t / J$, we have

$$\begin{aligned} f(x_u) - f(x_{u+1}) &\geq \min \left\{ \frac{\langle \nabla f(x_u), x_u - w_u \rangle^2}{2LD^2}, \frac{1}{2} \langle \nabla f(x_u), x_u - w_u \rangle \right\} \\ &\geq \frac{\Phi_t}{2J} \min \left\{ 1, \frac{\Phi_t}{LD^2 J}. \right\}. \end{aligned} \tag{B.13}$$

Next, consider the case u iteration is a descent step that means $d_u = a_u - s_u$ and $\lambda_u < \Lambda_u^*$. Using Lemma 3.4 (a) and the inequality $\langle \nabla f(x_u), a_u - s_u \rangle \geq \Phi_u = \Phi_t$, we have

$$\begin{aligned} f(x_u) - f(x_{u+1}) &\geq \frac{\langle \nabla f(x_u), d_u \rangle^2}{2LD^2} \\ &\geq \frac{\Phi_t^2}{2LD^2}. \end{aligned} \tag{B.14}$$

Finally, consider the case u iteration is a drop step that means $d_u = a_u - s_u$ and $\lambda_u = \Lambda_u^*$. In this case, it holds

$$f(x_u) - f(x_{u+1}) \geq 0. \tag{B.15}$$

We bound the total number of iterations T to achieve $f(x_T) - f(x^*) \leq \epsilon$. Let $N_{\mathrm{FW}}, N_{\mathrm{desc}}, N_{\mathrm{drop}}, N_{\mathrm{gap}}$ be the number of FW steps, descent steps, drop steps and gap steps respectively. In addition, we denote the number of FW steps, descent steps in epoch t by $N_{\mathrm{FW}}^t, N_{\mathrm{desc}}^t$ respectively. Using $N_{\mathrm{drop}} \leq N_{\mathrm{FW}}$ and (B.12), we have

$$T \leq N_{\mathrm{FW}} + N_{\mathrm{desc}} + N_{\mathrm{drop}} + N_{\mathrm{gap}} \leq \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil + 2N_{\mathrm{FW}} + N_{\mathrm{desc}} \leq \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil + \sum_{t:\mathrm{epoch}} (2N_{\mathrm{FW}}^t + N_{\mathrm{desc}}^t). \tag{B.16}$$

Here, we bound $2N_{\mathrm{FW}}^t + N_{\mathrm{desc}}^t$ in epoch t.

Let t' be the index of iteration where epoch t starts. We consider the following two cases:

(I) $\Phi_t \geq LD^2J$

By (B.11),(B.13), (B.14) and the condition $\Phi_t \geq LD^2J$, we have

$$\begin{aligned}
2\Phi_t &\geq f(x_{t'}) - f(x^*) \\
&\geq N_{\mathrm{FW}}^t \cdot \frac{\Phi_t}{2J} + N_{\mathrm{desc}}^t \cdot \frac{\Phi_t^2}{2LD^2} \\
&\geq 2N_{\mathrm{FW}}^t \cdot \frac{\Phi_t}{4J} + N_{\mathrm{desc}}^t \cdot \frac{\Phi_t}{2LD^2} \cdot LD^2J \\
&= \Phi_t \left(2N_{\mathrm{FW}}^t \cdot \frac{1}{4J} + N_{\mathrm{desc}}^t \cdot \frac{J}{2} \right).
\end{aligned}$$

Thus,

$$2N_{\mathrm{FW}}^t + N_{\mathrm{desc}}^t \leq \max\{8J, \frac{4}{J}\}. \tag{B.17}$$

(II) $\Phi_t < LD^2J$

Using (B.11),(B.13) and (B.14), we have

$$2\Phi_t \geq f(x_{t'}) - f(x^*) \geq N_{\mathrm{FW}}^t \cdot \frac{\Phi_t^2}{2LD^2J^2} + N_{\mathrm{desc}}^t \cdot \frac{\Phi_t^2}{2LD^2} = \Phi_t^2 \left(2N_{\mathrm{FW}}^t \cdot \frac{1}{4LD^2J^2} + N_{\mathrm{desc}}^t \cdot \frac{1}{2LD^2} \right).$$

Thus, we have

$$2N_{\mathrm{FW}}^t + N_{\mathrm{desc}}^t \leq \frac{1}{\Phi_t} \max\{8LD^2J^2, 4LD^2\}. \tag{B.18}$$

Since Φ_t dose not increase and (B.17) holds, the number of iterations in which $\Phi_t \geq LD^2J$ holds is bounded. We define t_1 as the first epoch where $\Phi_t < LD^2J$ is satisfied and T_1 as the total number of the iterations by epoch t_1. Then, by (B.16) and (B.18), we have

$$T \leq T_1 + \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil + \sum_{t \geq t_1} (2N_{\mathrm{FW}}^t + N_{\mathrm{desc}}^t) \leq T_1 + \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil + \sum_{t:\mathrm{epoch}} \frac{C_1}{\Phi_t},$$

where $C_1 = \max\{8LD^2J^2, 4LD^2\}$. In addition, it holds that

$$\sum_{t:\mathrm{epoch}} \frac{C_1}{\Phi_t} \leq \frac{C_1}{\Phi_0} \sum_{n=0}^{N_\Phi} 2^n \leq \frac{C_1}{\Phi_0}(2^{N_\Phi+1} - 1) \leq \frac{C_1}{\Phi_0} 2^{\log \frac{2\Phi_0}{\epsilon}+2} = \frac{8C_1}{\epsilon}.$$

Therefore , we have

$$T \leq T_1 + \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil + \frac{8C_1}{\epsilon}.$$

Therefore we drive $\epsilon = O(\frac{1}{T})$.

Next, consider Case (A). The argument by (B.16) can be done exactly the same as Case (B). In the same way as Case (B), we consider the two cases (I) $\Phi_t \geq LD^2J$ and (II) $\Phi_t < LD^2J$. For the case (I), we can derive just the same result (B.17). For the case (II), we analyze in a different way from Case (B) in the following way:

(II-A) $\Phi_t < LD^2J$

If iteration $t-1$ is a gap step, using the same argument as (Braun et al., 2019a), we have

$$f(x_t) - f(x^*) \leq \frac{8\Phi_t^2}{\mu}. \tag{B.19}$$

Using (B.19), we drive the following bound for Case (A);

$$\begin{aligned}\frac{8\Phi_t^2}{\mu} &\geq f(x_{t'}) - f(x^*) \\ &\geq N_{\mathrm{FW}}^t \cdot \frac{\Phi_t^2}{2LD^2J^2} + N_{\mathrm{desc}}^t \cdot \frac{\Phi_t^2}{2LD^2} \\ &\geq \Phi_t^2 \left(2N_{\mathrm{FW}}^t \cdot \frac{1}{4LD^2J^2} + N_{\mathrm{desc}}^t \cdot \frac{1}{2LD^2} \right).\end{aligned}$$

Therefore, we have

$$2N_{\mathrm{FW}}^t + N_{\mathrm{desc}}^t \leq \frac{8}{\mu} \max\{4LD^2J^2, 2LD^2\}. \tag{B.20}$$

By (B.16), (B.17) and (B.20), it holds

$$T \leq C_2 \left\lceil \log \frac{2\Phi_0}{\epsilon} \right\rceil, \tag{B.21}$$

where $C_2 = 1 + \max\{\max\{8J, \frac{4}{J}\}, \frac{8}{\mu} \max\{4LD^2J^2, 2LD^2\}\}$. Thus, we derived the desired result for Case (A). □

C. Finer sparsity control in BPCG

In this section we explain how the sparsity of the BPCG algorithm can be further controlled while changing the convergence rate only by a small constant factor. To this end we modify the step selection condition in Line 6 in Algorithm 1 to incorporate a scaling factor $K_{\text{sc}} \geq 1.0$:

$$K_{\text{sc}} \cdot \langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), x_t - w_t \rangle. \tag{relaxCond}$$

With this modification Lemma 3.5 changes as follows

Lemma C.1 (Modified version of Lemma 3.5)**.** For each step t in Algorithm 1, an inequality

$$(K_{\text{sc}} + 1) \cdot \langle \nabla f(x_t), d_t \rangle \geq \langle \nabla f(x_t), a_t - w_t \rangle$$

holds.

Proof. If we take a pairwise step we have

$$K_{\text{sc}} \cdot \langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), x_t - w_t \rangle.$$

Moreover, it holds as before

$$\langle \nabla f(x_t), x_t \rangle \geq \langle \nabla f(x_t), s_t \rangle, \tag{convexComb}$$

so that we obtain

$$K_{\text{sc}} \cdot \langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), s_t - w_t \rangle.$$

Now we add $\langle \nabla f(x_t), a_t - s_t \rangle$ to both sides of this inequality and obtain:

$$(K_{\text{sc}} + 1) \cdot \langle \nabla f(x_t), a_t - s_t \rangle \geq \langle \nabla f(x_t), a_t - w_t \rangle,$$

which is the claim in this case as $d_t = a_t - s_t$.

In case we took a normal FW step it holds:

$$K_{\text{sc}} \cdot \langle \nabla f(x_t), a_t - s_t \rangle < \langle \nabla f(x_t), x_t - w_t \rangle,$$

and together with (convexComb)

$$K_{\text{sc}} \cdot \langle \nabla f(x_t), a_t - x_t \rangle < \langle \nabla f(x_t), x_t - w_t \rangle.$$

Now adding $K_{\text{sc}} \cdot \langle \nabla f(x_t), x_t - w_t \rangle$ to both sides we obtain

$$K_{\text{sc}} \cdot \langle \nabla f(x_t), a_t - w_t \rangle < (K_{\text{sc}} + 1) \cdot \langle \nabla f(x_t), x_t - w_t \rangle,$$

and hence

$$\langle \nabla f(x_t), a_t - w_t \rangle < \frac{(K_{\text{sc}} + 1)}{K_{\text{sc}}} \cdot \langle \nabla f(x_t), x_t - w_t \rangle \leq (K_{\text{sc}} + 1) \cdot \langle \nabla f(x_t), x_t - w_t \rangle,$$

as required as $d_t = x_t - w_t$. □

Plugging this modified lemma back into the rest of the proof of Theorem 3.2 leads to a small constant factor loss of $(K_{\text{sc}} + 1)^2/4$ in the convergence rate, which stems from the simple fact that progress from smoothness is proportional to the squared dual gap estimate, which weakened by a factor of $(K_{\text{sc}} + 1)/2$.

Remark C.1 (Sparsity BPCG vs. lazy BPCG)**.** Observe that the non-lazy BPCG is sparser than the lazy BPCG. While counter-intuitive, the optimization for the local active set maximizes sparsity already and the non-lazy variant uses tighter Φ bounds as they are updated in each iteration promoting prolonged optimization over active set before adding new vertices. Although it can happen that the lazy BPCG algorithm obtains sparser solutions than the non-lazy BPCG algorithms, this observation may help to understand the behavior of the algorithms.

D. Additional numerical experiments

ℓ_p NORM BALL

We define the ℓ_p norm for $p \in \mathbb{R}$ and $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ as $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$ and consider the following optimization problem

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \|x - x_0\|_2^2 \\ &\text{s.t. } \|x\|_p \leq 1, \end{aligned}$$

where $\|x_0\|_p \leq 1$. We performed the numerical experiments with $p = 5$ and $n = 1000$. The result is given in Figure 8. The sparsest solution was obtained via BPCG methods. Note that in this experiment the vanilla Conditional Gradient and

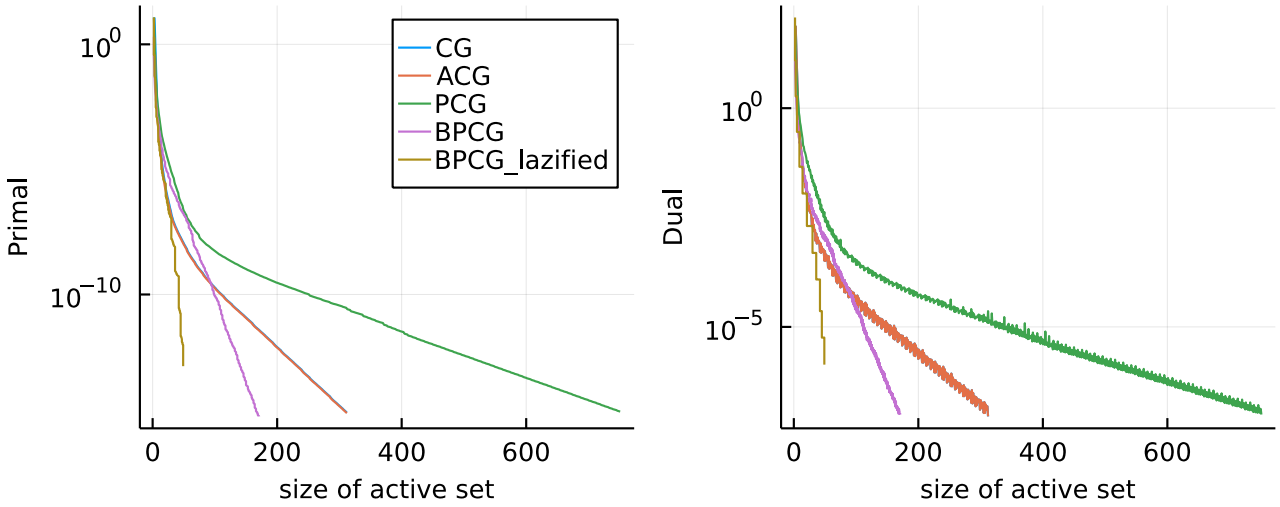


Figure 8. Sparsity: ℓ_5 norm case

Away-step Conditional Gradient behave in exactly the same way and the blue line and orange line in the figure overlap each other.

KERNEL QUADRATURE FOR A MIXTURE GAUSSIANS

We performed the experiments for a mixture Gaussians on $[-1, 1]^2$. As shown in Figure 9, we have almost the same result as that of the ordinary Gaussian distribution. We note that in the left figure in Figure 9, the distribution function takes large values as the color gets closer to yellow and takes small values as the color gets closer to blue.

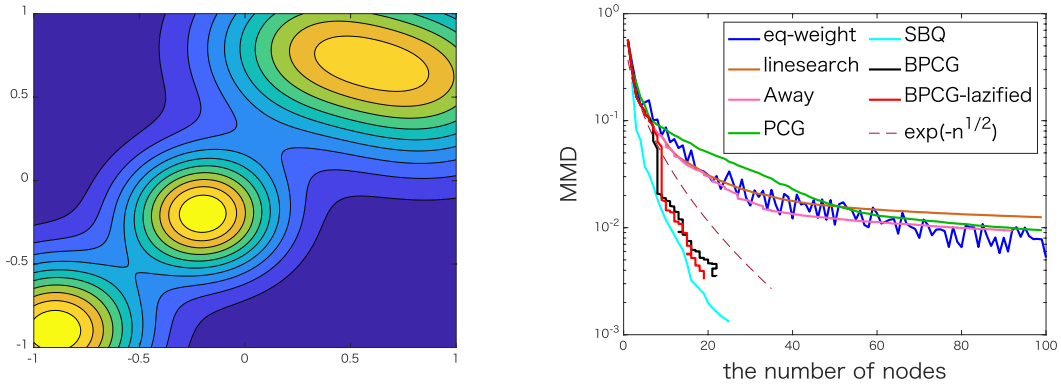


Figure 9. Contour of the mixture gausssian distribution (left) and MMD for the number of nodes(right).