RESEARCH ARTICLE

Pairwise Frank-Wolfe for Maximum Inscribed Balls: Enabling Real-Time Geometric Optimization

Yuqi Lin[1,*]

[1] The Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana 61801, United States

Abstract

As a classical convex optimization problem in geometry, computing the maximum inscribed ball (MaxIB) in ultra-high-dimensional polytopes is critical for enabling real-time IoT applications, such as optimal deployment of sensor networks, where polytopes model physical constraints arising from obstacles or coverage boundaries. However, existing methods suffer from the curse of dimensionality, leading to prohibitive computational costs. This paper develops a more efficient approach for computing the (1-ϵ)-approximate MaxIB in high-dimensional polytopes. To address these challenges, the problem is reformulated with adaptive penalty parameters to enforce strong convexity, enabling linear convergence under the Pairwise Frank–Wolfe (PFW) algorithm. Furthermore, expensive exact line searches are replaced with a backtracking strategy, significantly reducing the per-iteration computational cost. Simulation results demonstrate more than a 12-fold acceleration over existing approximate MaxIB methods across high-dimensional settings, and up to an 80-fold speedup under stringent accuracy requirements, without sacrificing solution quality.

Keywords: convex optimization, maximum inscribed ball, gradient optimization, Frank-Wolfe's Method.

1 Introduction

The problem of computing the maximum inscribed ball (MaxIB, or Chebyshev ball) within high-dimensional polytopes is a fundamental challenge in convex optimization. It provides substantial computational advantages, such as facilitating warm starts in interior-point methods, and supports real-time applications in the Internet of Things (IoT), like navigation for warehouse robots and optimization of sensor networks, where polytopes represent spatial constraints such as obstacle avoidance or coverage limits [1]. Additionally, the MaxIB problem has widespread applicability, ranging from analyzing system durability and advancements in control theory [2] to geometric optimization for robot motion planning and obstacle avoidance [3], a rapidly evolving application domain.

The MaxIB problem further serves as a canonical case study bridging optimization and machine learning [4]. This connection is further supported by Yildirim (2008) [5], who developed efficient first-order algorithms for the minimum enclosing

Submitted: 15 July 2025
Revised: 25 November 2025
Accepted: 22 December 2025
Published: 14 January 2026

Vol. 2, **No.** 1, 2026.
10.62762/TACS.2025.318429

***Corresponding author:**
✉ Yuqi Lin
yuqilin2@illinois.edu

Citation
Lin, Y. (2026). Pairwise Frank-Wolfe for Maximum Inscribed Balls: Enabling Real-Time Geometric Optimization. *ICCK Transactions on Advanced Computing and Systems*, 2(1), 61–73.

© 2026 by the Author. Published by Institute of Central Computation and Knowledge. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

ball problem, whose duality with MaxIB underpins several approximate solution methods studied in this work. Recent work by Baena et al. [6] additionally demonstrates the utility of MaxIB in enhancing the feasibility pump, a key technique for integer programming. However, given the various uses of MaxIB, efficiently computing the MaxIB in high-dimensional polytopes is a challenging problem. The existing methods for solving the MaxIB problem can be broadly categorized into two classes: exact and approximate solutions.

Exact methods, such as formulating and solving a linear program (LP) using the simplex method or interior point methods, were the initial solutions[7–9]. Their primary advantage lies in providing solutions with extremely high accuracy, however, as the number of dimensions increases, these exact methods face significant challenges due to their high computational complexity and memory requirements, often becoming intractable or even failing to produce a result for very high-dimensional polytopes. Consequently, approximate methods have gained prominence as the mainstream approach for tackling very high-dimensional MaxIB problems in recent years. These methods primarily focus on finding the center of an approximate MaxIB efficiently, trading off some accuracy for significant computational gains. Common ideas include dual projection and leveraging first-order optimization techniques [10, 11]. While offering faster computation, they typically require careful tuning to control the approximation error. The existing alternative method of finding an approximate MaxIB relies heavily on this parameter, which causes the runtime to increase dramatically.

The goal of this paper is to provide a faster algorithm for calculating the approximate center of the MaxIB in high dimensions, while maintaining the error within the desired range. Frank–Wolfe (FW) methods and their variants, which are widely used for solving convex optimization problems, are adopted due to their low per-iteration computational cost, making them well suited for improving overall runtime efficiency. However, the classical FW algorithm exhibits a low convergence rate when applied to high-dimensional polytopes, leaving significant room for further improvement in solution speed. To address this limitation, the Lagrangian multi-operator approach combined with a quadratic penalty is employed to reformulate the original MaxIB problem into a dual quadratic convex optimization problem that can be efficiently solved using FW-based methods. The innovation of this paper is the following:

1. The original MaxIB problem was converted into a smooth, strongly-convex dual problem over the simplex through application of a Lagrangian multi-operator transform and a *quadratic penalty*;
2. The *Pairwise Frank-Wolfe* (PFW) method was employed, where strong convexity is exploited to achieve linear convergence;
3. An *adaptive backtracking line search* was integrated to estimate local curvature (L_f), replacing computationally expensive exact line searches.

Section 2 reviews existing MaxIB computation methods, classifying them into exact and approximate approaches with an analysis of their high-dimensional performance. Section 3 presents our proposed algorithm and its runtime optimization strategy. Section 4 demonstrates the method's advantages through comparative simulations across different dimensions. Finally, Section 5 concludes the paper with a summary of key findings.

2 Related Work

The problem of finding the maximum inscribed ball (MaxIB) has been investigated for over a century. The problem is addressed by adding one more dimension(radius) to the constraints of the original polytope given in order to create a linear programming(LP) problem. The simplex method [7] is the first commonly used approach to resolve this problem. Starting from a feasible base point (vertex), the simplex method employs pivot operations to transition between feasible vertices, adjusting one base variable at a time in and out of the basis at each step until the objective value can no longer be improved. Though being feasible, the simplex method has a runtime of $O(2^n)$ in the worst case(iterate through all vertices) [12].

Another method to solve LP, the interior point method [8, 9] was introduced by Karmarkar in 1984. It internalizes the constraints into the objective function, adds a dynamically adjusted penalty coefficient, and iteratively approaches the optimum along the "central path" within the feasible region through the Newton method. Attracted by the design of the interior point method, researchers devoted themselves deeply to it and developed many variants that fit different conditions. For instance, the primal-dual interior point method [13] can achieve higher accuracy and faster convergence rates for convex optimizing problems

with small and medium scales. Though the interior point method spends higher runtime in each step ($O(n^3)$ compared to $O(md)$ of the simplex method), it converges at a much faster rate, approaching the approximate total runtime of $O(n^{3.5})$. To sum up, the interior point method significantly improved the efficiency of solving the LP problem, enhancing the feasibility of LP in engineering applications.

Through later investigation and development, the simplex method and the interior point method became the main way of solving LP problems. However, due to their relatively high runtime depending on the number of dimensions, solving the MaxIB problem in high dimensions becomes very expensive. At $d > 100$, many current solvers using these methods fail. While industrial-grade solvers that employ advanced optimization strategies can generate solutions, they are relatively expensive and challenging to access. To address this problem, solving an approximate MaxIB becomes a prevalent topic. By introducing an error parameter ϵ, solving an approximate MaxIB with a radius r within ϵ error to optimal r usually has a runtime that is almost linear to the input size. It also allows adjusting the complexity of the algorithm based on the accuracy(size of ϵ to approach). Furthermore, having an approximate MaxIB also provides a good start point (warm start) for the interior method. Since approximate MaxIB usually has a center close to the optimal value, the interior point method can solve the problem in fewer steps, significantly improving the calculation efficiency.

Xie et al. [10] first introduce a glance to approximate the MaxIB problem and approach it from geometric and optimization perspectives. They found the connection between MaxIB and the minimum enclosing ball (MinEB) and developed a purely geometric core-set method. Their algorithm involves dualizing MaxIB to a sequence of MinEB instances and then computing a small $\Theta(1/\epsilon)$-sized core-set for each using a combinatorial geometry method. With this approach, they yield a runtime of MaxIB to: $O\big(m\,d\,\alpha^3/\epsilon^3 + m\,d\,\alpha\log\alpha\big)$, where α is the polytope's aspect ratio. While this achieves optimal linear dependence on the input size $m\,d$ and excels in moderate dimensions, its cubic dependence on α and $1/\epsilon$ limits scalability when the polytope is very ill-conditioned or high precision is required.

Later, Allen-Zhu et al. [11] recast MaxIB as smooth, strongly convex optimization problems by introducing an entropy penalty on the primal space of the problem. By doing so, they discover that the target function is strongly concave as well, so they then solve the constructed problem by applying Nesterov's accelerated gradient algorithm, where each iteration incurs only matrix–vector products and simple projections onto the simplex. As a result, for MaxIB, their method runs in $O\big(m\,d\,\sqrt{\log m}\,\alpha\,(\ln\alpha + 1/\epsilon)\big)$.

They also use a similar method in calculating the MinEB, and apply it to the method of Xie's team, reducing the total runtime to $\widetilde{O}(m\,d\,\alpha^{2.5}/\epsilon^{2.5})$, and then conducted an experiment to compare the performances of two algorithms. Compared to the first method, their method reaches the target with fewer iterations, especially when the polytope is narrow. This represents an improvement over core-set methods, reducing the dependence on the aspect ratio from cubic to linear and on accuracy from ϵ^{-3} to ϵ^{-1}. Moreover, by avoiding combinatorial geometry subroutines and relying exclusively on linear-algebra operations, their approach is simpler to implement and better suited to very large-scale or high-precision settings. Nevertheless, the method involves a significant factor in the iteration time to ensure convergence, which increases the overall time consumption substantially. Under low ϵ, the overall time consumption is even much worse than the method of Xie's team.

Our work involves the use of the Pairwise Frank-Wolfe(PFW) method, an improved version of the classic Frank-Wolfe method. The classical Frank–Wolfe (FW) algorithm [14] is a projection-free first-order method for minimizing/maximizing a smooth convex function over a compact convex domain. At each iteration, it solves a cheap linear subproblem to update, thereby avoiding the expensive Euclidean projection process and efficiently reducing the cost. Due to its simplicity, it is widely used in current convex optimizing problems. However, the convergence rate of classic FW can be slow due to the zig-zag phenomenon that occurs when the optimal solution lies on the boundary.[15] Based on classic FW, multiple variants have been investigated to address these issues and improve the performance.

The Away-step Frank Wolfe (AFW) method [16] improves the classic FW by using a different approach to determine the step direction in each iteration. By doing so, it resolves the zig-zag phenomenon that occurred in classical FW, and thus gains a higher convergence rate, reducing the total number

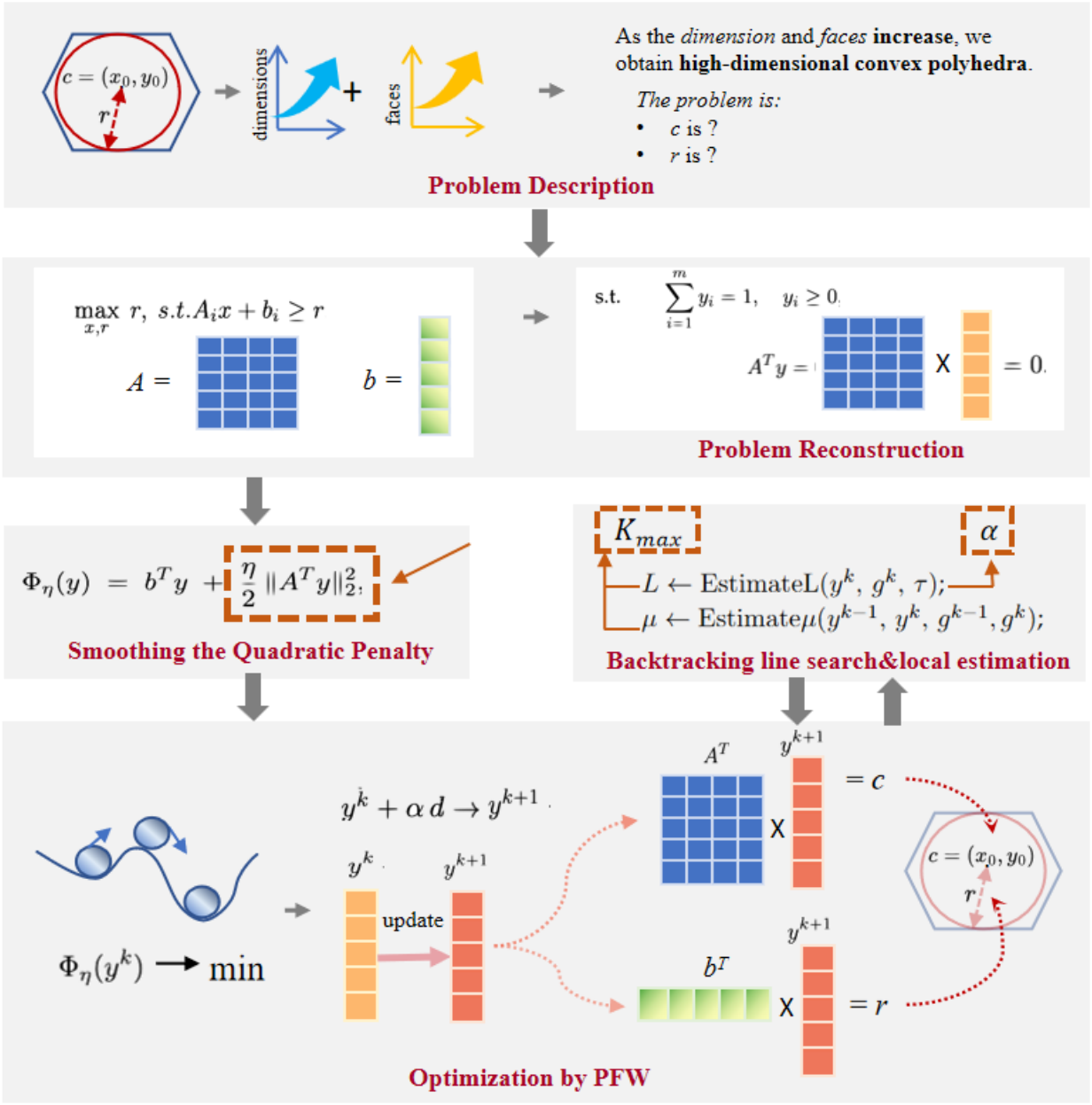

Figure 1. Flowchart of the proposed method.

of iterations required. PFW [17] combines classic FW and AFW, reaching an even higher convergence rate(linear under strong convexity). By applying PFW to the problem and adding multiple advanced techniques, our method is able to achieve a runtime of $O(md\alpha^2 \ln(1/\epsilon))$ in the worst case. In practice, due to the low iteration time and early breaking method, our method is able to complete the calculation in a much shorter time.

3 Methodology

Figure 1 illustrates the flowchart for computation procedure of our approach, which consists of three key steps: (1) transforming MaxIB into an optimization function and introducing a penalty term to ensure smoothness(Section 3.1 and 3.2), (2) implementing the Projected Frank-Wolfe (PFW) algorithm, and (3)local estimation techniques with backtracking line search to enhance efficiency, where Section 3.3 provides the theoretical global convergence guarantee for PFW and Section 3.4 presents a practical local estimation method.

3.1 Problem Reconstruction

Consider a convex polytope with dimension of d and constraint number of m:

$$\mathcal{P} = \{x \in \mathbb{R}^d : A_i x + b_i \geq 0,\ i = 1, \ldots, m\},$$

where $A_i \in \mathbb{R}^d$ is the vector of the ith hyperplane, $b_i \in \mathbb{R}$ is the translation term.

The MaxIB problem can be written as follows:

$$\max_{x,r} \; r, \; s.t. A_i x + b_i \geq r, \; \forall i.$$

where $A \in \mathbb{R}^{m \times d}$ contains the normalized inward-pointing facet normals ($\|A_{i,:}\|_2 = 1$ for all i), $b_i \geq 0$ is the offset of the i-th hyperplane, d is the dimension, and m is the number of constraints (typically $m = 4d$ in our experiments).

For generality, each row of A is assumed to be normalized, i.e., $\|A_{i,:}\|_2 = 1$. In order to apply PFW, the problem needs to be transferred to its dual problem by introducing Lagrange multipliers $y \in \mathbb{R}^m \geq 0$

$$\mathcal{L}(x, r, y) = \; r \; + \; \sum_{i=1}^{m} y_i \big(A_i x + b_i - r\big).$$

The left-hand program jointly maximizes the radius r and the center x while requiring x to lie at least distance r inside every facet. The right-hand Lagrangian introduces non-negative dual weights $y_i \geq 0$ (one per facet) to relax the inequalities.

After being transferred, the dual problem is shown as follows:

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & \sum_{i=1}^{m} b_i \, y_i, \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i = 1, \quad y_i \geq 0, \quad A^T y = 0. \end{aligned} \tag{1}$$

The dual problem is equivalent to optimizing the linear object $b^T y$ in the probability simplex $\Delta_m = \{y \geq 0, \sum_i y_i = 1\}$ with the linear equivalent $A^T y = 0$.

3.2 Smoothing and Choice of the Quadratic Penalty Weight $\eta = 1/\epsilon$

To smooth the dual formulation (1), the hard constraint $A^T y = 0$ is replaced by a *quadratic penalty*. The target function Φ becomes the following.

$$\Phi_\eta(y) \; = \; b^T y \; + \; \frac{\eta}{2} \, \|A^T y\|_2^2, \tag{2}$$

Intuitively, this is the original linear dual objective $b^\top y$ plus a soft quadratic penalty of strength η that discourages violation of the centering constraint $A^\top y = 0$.

To ensure that the resulting violation of $A^T y = 0$ is $O(\epsilon)$, the penalty parameter η is chosen as follows:

Taking y_η^* to minimize Φ_η over Δ_m, and letting y^* be any exact solution of the constrained dual(1), the following equation holds.

$$\|A^T y_\eta^*\|_2^2 \; \leq \; \frac{2}{\eta} \left[\Phi_\eta(y^*) - \Phi_\eta(y_\eta^*)\right].$$

Our objective is to maintain the penalty in violation of the original restriction while limiting the influence of the penalty function on the optimization target. To balance out, set the optimization error $\Phi_\eta(y_\eta^*) - \Phi_\eta(y^*) \leq O(\epsilon)$. Therefore, it follows that

$$\|A^T y_\eta^*\|_2 = O\Big(\sqrt{\tfrac{\epsilon}{\eta}}\Big).$$

To avoid feasibility violation $\|A^T y_\eta^*\| \leq O(\epsilon)$, it suffices to choose

$$\eta = \frac{1}{\epsilon}.$$

Setting $\eta = \frac{1}{\epsilon}$ simultaneously keeps both the sub-optimality of the objective and the violation of $A^\top y = 0$ within $O(\epsilon)$, giving a provably $(1 - \epsilon)$-approximate MaxIB.

This scaling ensures that both the optimization error and the violation of the constraint are $O(\epsilon)$, as in the classical quadratic penalty theory [18].

3.3 Pairwise Frank–Wolfe Method

After applying the quadratic penalty eqref penalty, PFW can be used to solve the resulting optimization problem. PFW is a projection-free method for minimizing a smooth convex function over a polytope P. Applied to our smoothed dual, it maintains feasibility $y^k \in \Delta_m$ by convex combinations of *atoms* e_i. At each iteration, it will be:

1. Computes the gradient

$$g^k \; = \; \nabla \Phi_\eta(y^k) \; = \; b + \eta \, A\big(A^T y^k\big).$$

2. Finds the ascent atom with highest gradient(FW atom) $j_{\mathrm{FW}} = \arg\max_i g_i^k$ and descends the atom with the lowest gradient (Away FW atom) $j_{\mathrm{AW}} = \arg\min_{i : y_i^k > 0} g_i^k$.

3. Forms the *pairwise direction* based on FW atom and Away FW(AFW) Atom.

$$d^k = e_{j_{\mathrm{FW}}} \; - \; e_{j_{\mathrm{AW}}}.$$

The direction calculated is illustrated by Figure 2.

4. Apply the early breaking techniques of *duality gap*. The *duality gap* is calculated as the following:

$$\mathrm{gap}^k = g_{j_{\mathrm{AW}}}^k - g_{j_{\mathrm{FW}}}^k,$$

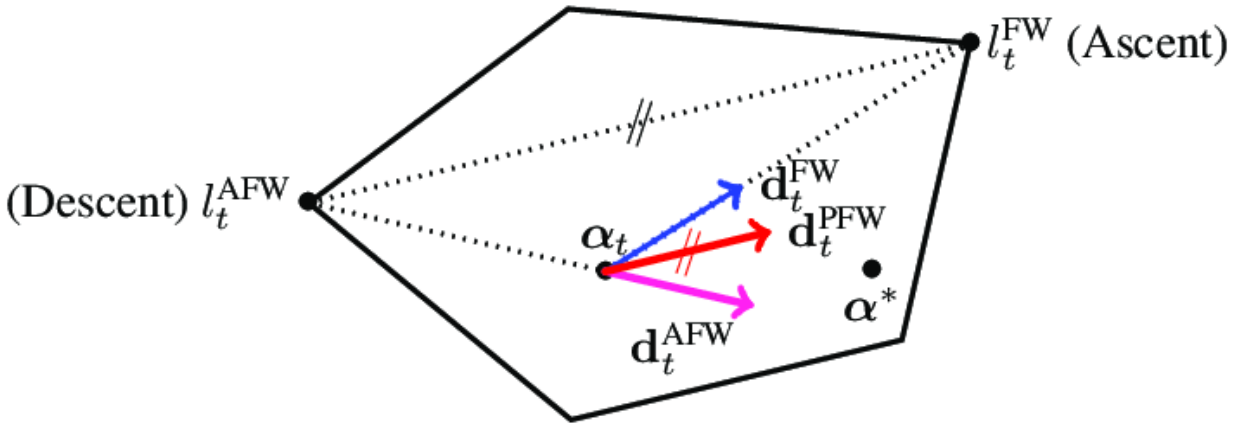

Figure 2. Illustration of FW, away-step, and pairwise direction updates [17, 19].

Given that $\Phi_\eta(y^k) - \Phi_\eta(y^*) \le \mathrm{gap}^k$[20], Early stopping is triggered when $\mathrm{gap}^k \le \epsilon\, r^k$, where $r^k = b^\top y^k$, since this ensures that the solution found satisfies the target accuracy.

5. Performs an exact line search along d^k to determine the step size:

$$\alpha^k = \min\Big\{ \frac{\mathrm{gap}^k}{\eta\, \|A^T d^k\|_2^2},\ y^k_{j_{\mathrm{AW}}} \Big\},$$

and updates

$$y^{k+1} = y^k + \alpha^k\, d^k.$$

In summary, at each iteration, the smoothed gradient is computed, the facet that most wants to increase (the FW atom) and the active facet that most wants to decrease (the Away atom) are selected, a pairwise move is taken along their difference with a safe step size, and early stopping is applied once the estimated error falls below ϵ times the current radius.

Combining the PFW algorithmic framework with the constructed problem yields the procedure detailed in Algorithm 1:

Due to multiple external factors(such as round-offs), in actual practice, the duality gap's stopping criterion may not be fulfilled, causing an infinite loop. Therefore, setting up a max iteration number K_{max} is necessary. Under the assumptions that Φ_η is μ-strongly convex and L_f-smooth on Δ_m, and that the pyramidal width of Δ_m is δ, the rate of convergence of PFW is as following[17]:

$$\Phi_\eta(y^k) - \Phi_\eta(y^*) \ \le\ \Big(1 - \tfrac{\mu\,\delta^2}{L}\Big)^k \Big[\Phi_\eta(y^0) - \Phi_\eta(y^*)\Big].$$

This implies the upper bound for iteration time:

Algorithm 1: basic PFW

Data: Matrix $A \in \mathbb{R}^{m\times d}$, vector $b \in \mathbb{R}^m$, smoothing $\eta > 0$, tolerance $\epsilon > 0$, maximum steps K
Result: Approximate MaxIB with center c and radius r

Initialize $y^0 = e_{i^*}$ with $i^* = \arg\min_i b_i$;
for $k = 0$ **to** K **do**
  $g \leftarrow b + \eta\, A(A^\top y^k)$;
  $j_{\mathrm{FW}} \leftarrow \arg\min_i g_i$;
  $j_{\mathrm{AW}} \leftarrow \arg\max_{i:y_i^k>0} g_i$;
  $\mathrm{gap} \leftarrow g_{j_{\mathrm{AW}}} - g_{j_{\mathrm{FW}}}$;
  $r \leftarrow b^\top y^k$;
  if $\mathrm{gap} \le \epsilon r$ **then**
    break;
    ; // early stop when gap small
  end
  $d \leftarrow e_{j_{\mathrm{FW}}} - e_{j_{\mathrm{AW}}}$;
  $\alpha \leftarrow \min\Big\{\frac{\mathrm{gap}}{\eta\|A^\top d\|^2},\ y^k_{j_{\mathrm{AW}}}\Big\}$;
  $y^{k+1} \leftarrow y^k + \alpha d$;
  $r \leftarrow b^\top y^{k+1}$;
end
$c \leftarrow A^\top y$;
$r \leftarrow \min_{1\le i\le m}(b_i - A_i^\top x)$;
return (c, r);

$$K_{\max} \ =\ \left\lceil \frac{L}{\mu\,\delta^2}\ \ln\!\Big(\frac{\mathrm{gap}_0}{\epsilon\, r_0}\Big) \right\rceil.$$

The pyramidal width can be estimated by the ratio between the length and the width of the polytope. Since the smoothed dual objective under consideration is both μ-strongly convex and L-smooth in the Euclidean norm, the constants μ and L can be

computed as follows:

$$\mu = \eta\,\lambda_{\min}(AA^\top), \quad L = \eta\,\lambda_{\max}(AA^\top).$$

3.4 Backtracking Line Search for L and Local Estimation of μ

However, the calculation of μ and L requires the extraction of the eigenvalues, which can be costly($O(m^3)$ if using classic QR full eigenvalue decomposition [21]). This section presents a simpler method for estimating μ and L, introduced in the next section, which avoids the expensive calculation.

The *backtracking line search* technique [22] is employed to estimate the local smoothness constant L_f, and a *secant-style* estimate for the strong-convexity constant μ. Though the locally estimated value may not represent the entire field, in actual calculation, the ratio L_f/μ converges, and under the upper bound of iteration calculated from it, the result converges to a feasible value as well. The following backtracking procedure is used to compute L_f, as shown in Algorithm 2:

Algorithm 2: Estimate L

Data: Current iterate y, initial guess $L_0 > 0$, expansion factor $\tau > 1$
Result: Estimate $\hat{L}_f \geq L_f$
$\hat{L} \leftarrow L_0$;
$g \leftarrow \nabla\Phi_\eta(y),\ F \leftarrow \Phi_\eta(y)$;
while $\Phi_\eta\big(y - \frac{1}{\hat{L}}\,g\big) > F - \frac{1}{2\hat{L}}\|g\|^2$ **do**
  $\hat{L} \leftarrow \tau\,\hat{L}$;
end
return $\hat{L}$;

After this loop terminates, $\hat{L}$ satisfies the Armijo-type condition and so is a valid local Lipschitz constant.

Given two successive iterates y^{k-1}, y^k and their gradients g^{k-1}, g^k, a local estimate of μ is

$$\hat{\mu} \;=\; \frac{\langle g^k - g^{k-1},\ y^k - y^{k-1}\rangle}{\|y^k - y^{k-1}\|_2^2}.$$

By strong convexity, $\langle \nabla\Phi_\eta(y^k) - \nabla\Phi_\eta(y^{k-1}),\, y^k - y^{k-1}\rangle \geq \mu\|y^k - y^{k-1}\|^2$, so $\hat{\mu}$ is a conservative lower bound on the true μ.

These two simple procedures give cheap, conservative local estimates of the smoothness constant L_f and strong-convexity constant μ without any eigenvalue computation, yet they are sufficient for safe step sizes and reliable convergence in practice.

Furthermore, after calculating the estimated L by using backtracking line search, the method of Pedregosa et al. [23] can be used to replace the expensive exact line search and to select the step size based on L_f: Instead of

$$\alpha \leftarrow \min\Big\{\tfrac{\text{gap}}{\eta\|A^T d\|^2},\, y_{j_{\text{AW}}}\Big\}$$

do:

$$\alpha \leftarrow \min\Big\{\tfrac{\text{gap}}{\hat{L_f}\|d\|^2},\, y_{j_{\text{AW}}}\Big\}$$

To incorporate the above methods into the main loop, the PFW algorithm is modified by adding the following steps. At iteration k, after computing the new point y^k, the following actions are performed:

1. Update $\hat{L_f}$ by running the above Backtracking Line Search with the current gradient.
2. Compute $\hat{\mu}$ using the two most recent iterates.
3. Use $\hat{L}_f$ and $\hat{\mu}$ to adjust step-sizes, stopping criteria, or theoretical iteration budgets.

This adaptive strategy avoids expensive eigenvalue computations while still tracking the local curvature of Φ_η. After integrating into the main PFW method, the complete procedure is given in Algorithm 3:

Algorithm 3: PFW with Backtracking

Data: Normalized $A \in R^{m\times d}$, $b \in R^m$, tolerance $\epsilon > 0$, initial $L > 0$, factor $\tau > 1$, max iters $K_{\max}$
Result: Dual solution $y \in \Delta_m$, radius r
$\eta \leftarrow 1/\epsilon$;
$y \leftarrow e_{i^*},\ i^* = \arg\min_i b_i$;
for $k \leftarrow 0$ **to** $K_{\max}$ **do**
  $g^k \leftarrow b + \eta\,A\,(A^\top y)$;
  $L \leftarrow \text{EstimateL}(y^k,\, g^k,\, \tau)$;
  $\mu \leftarrow \text{Estimate}\mu(y^{k-1},\, y^k,\, g^{k-1}, g^k)$;
  $j_{\text{FW}} \leftarrow \arg\min_i g_i, \quad j_{\text{AW}} \leftarrow \arg\max_{i:y_i>0} g_i$;
  $\text{gap} \leftarrow g_{j_{\text{AW}}} - g_{j_{\text{FW}}}$;
  if $\text{gap} \leq \epsilon\, r$ **then**
    break;
  end
  $d \leftarrow e_{j_{\text{FW}}} - e_{j_{\text{AW}}}$;
  $\alpha \leftarrow \min\big\{\frac{\text{gap}}{L\|d\|^2},\, y_{j_{\text{AW}}}\big\}$;
  $y \leftarrow y + \alpha\, d$;
  $r = b^T y$;
end
$c = A^T y$;
$r \;=\; \min_{1\leq i\leq m}\big(b_i - A_i^T c\big)$;
return (c, r)

By converting MaxIB into its dual form and incorporating penalty terms, the resulting problem becomes amenable to PFW. Moreover, backtracking line search for estimating L_f and local estimation of μ provide a reliable upper bound on the number of iterations, helping to avoid infinite loops. The estimated L_f is used to replace some expensive steps in PFW, which further reduces the runtime in high dimensions.

With the quadratic penalty weight $\eta = \frac{1}{\epsilon}$ fixed analytically, the Pairwise Frank-Wolfe method with adaptive backtracking therefore enjoys linear convergence $O(\alpha^2 \ln(1/\epsilon))$. At termination, $\|A^\top y^*\|_2 \le O(\epsilon)$ and $b^\top y^* \ge (1 - O(\epsilon)) \cdot \text{opt}$. Using the standard primal recovery $c = \frac{A^\top y^*}{\|A^\top y^*\|_2}$ and $r = \frac{b^\top y^*}{\|A^\top y^*\|_2} - \frac{\|A^\top y^*\|_2}{2}$ directly yields a strictly feasible inscribed ball with radius ratio $\rho \ge 1 - O(\epsilon)$. Thus the single analytical choice $\eta = \frac{1}{\epsilon}$ is both necessary and sufficient for the claimed $(1-\epsilon)$-approximation guarantee.

In summary, the MaxIB problem is transformed into the dual space and a quadratic penalty is introduced to smooth the objective. This yields a strongly convex function that can be efficiently solved by PFW with a linear convergence rate. To ensure a safe termination and further improve runtime, backtracking line search is used to estimate L_f, a maximum iteration limit is imposed, and the step size is selected using the resulting smoothness estimate.

4 Experiment and Analysis

This section presents two sets of experiments comparing the proposed method with existing approaches. Case 1 examines the running time of exact and approximate methods in high-dimensional settings. Case 2 compares the proposed method with existing approximate MaxIB methods studied by Xie et al. [10] and Allen-Zhu et al. [11] in terms of both accuracy and efficiency. All random polytopes are generated using a random seed drawn uniformly from the integers 1 to 100 across the entire experimental suite.

The experiment is conducted on a server with the following parameters: Intel Core i7-13620H processor, 2.4 GHz CPU, and NVIDIA GTX 1080 Ti 11 GB. The C++ package of *volesti* is used for aiding the generation of random high-dimensional polytopes. The time spent entering and exiting the function will be recorded to measure the algorithm's running time.

4.1 Case 1:Runtime Comparison: Exact vs. Approximate Methods in High Dimensions

This case is conducted to measure the performance of algorithms under different dimensions $d \in \{20, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$. For each dimension, 10 trials are conducted. For each trial, a d-dimensional polytope with 4d constraints will be generated to ensure that the polytope generated is bounded [24]. The generated polytopes will have the aspect ratio $\alpha = 2$. The simplex method and interior point method implementations are adopted from three different C++ packages (Lpsolve [25], GLPK [26], and MOSEK [27]) to ensure comprehensive evaluation. Each algorithm will calculate the center of MaxIB for the same polytope. The result running times are recorded in the Table 1. Time consumption will be marked as "-" if the method fails to solve all 10 trials in the corresponding dimension.

As the dimension increases, our method maintains consistent capability in solving the MaxIB problem while exhibiting significantly lower computational overhead. From Table 1, it can be seen that our method has the least time consumption beginning from dimension 100. This efficiency advantage stems from the simplicity of per-iteration computations in our algorithm. In contrast, exact methods require substantially more complex operations at each computational step. Consequently, in higher dimensions where the size of the constraint matrix $A \in R^{m \times d}$ grows quadratically(with $m = 4d$), less optimized implementations frequently encounter computational failures.

Crucially, as demonstrated in Figure 3, our method demonstrates superior scaling characteristics, with its runtime growing linearly with input size ($O(md) = O(n^2)$). This represents a polynomial-order reduction in time complexity compared to exact methods, which show significantly higher-order growth rates. This represents at least a polynomial-order reduction in time complexity relative to exact methods. At dimension $d = 1000$, our algorithm achieves computation times below 10% of all benchmarked methods, including the interior point implementation in the MOSEK package—a highly optimized industrial-grade solver incorporating decades of algorithmic refinements—while relying on fundamentally simpler computational primitives.

Table 1. Time consumption(ms) of different methods under high dimension.

Methods	20	50	75	100	200	300	400	500	600	700	800	900	1000
LPsolve-simplex	6	54	-	-	-	-	-	-	-	-	-	-	-
GLPK-simplex	3	27	26	28	279	1018	2842	6947	14375	28903	42635	66187	103132
GLPK-interior point	231	17678	135296	605521	-	-	-	-	-	-	-	-	-
MOSEK-primal simplex	31	37	79	104	991	3587	8734	19340	37636	80944	116508	224443	327740
MOSEK-dual simplex	36	45	65	59	294	772	2150	7206	21136	42492	74304	111240	220634
MOSEK-interior point	35	38	60	59	191	378	653	1041	1631	2429	3521	4239	5669
Xie et al. [10]	36	151	296	501	1816	3888	6740	10462	14655	19866	25687	32571	40144
Allen-Zhu et al. [11]	59	262	550	943	3743	8355	14924	23461	34231	46986	61668	78915	108139
The Proposed Method	8	10	22	42	143	290	527	804	1173	1565	2033	2598	3353

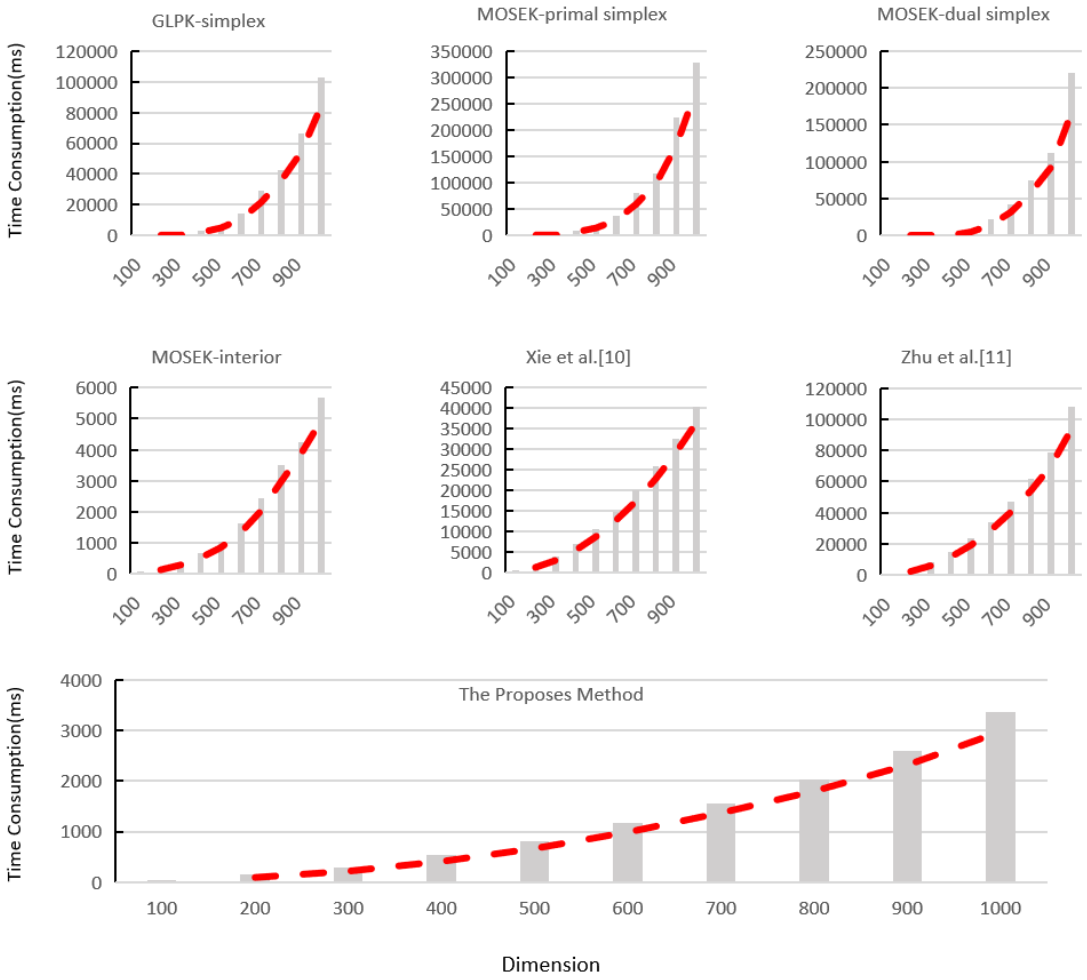

Figure 3. Trend of time consumption of different methods.

4.2 Case 2: Comparative Evaluation: Accuracy vs. Efficiency Against Approximate MaxIB Baselines

This experiment measures algorithmic performance under different aspect ratios and accuracy requirements. Random polytopes are generated with $d = 500$, $m = 4d = 2000$, $\alpha \in \{1, 2, 3, 4, 5\}$. For each algorithm, the error-tolerance parameter is set to $\epsilon \in \{0.1, 0.05, 0.02, 0.01, 0.005\}$. The interior-point method in the *mosek* package is used as a reference, since it enables exact MaxIB computation. Accuracy is evaluated using the root-mean-square error (RMSE) and the radius ratio ρ between the computed radius and the optimal radius, defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(\hat{c}_i - c_i^*\right)^2} = \frac{1}{\sqrt{d}}\left\|\hat{c} - c^*\right\|_2.$$

$$\rho = \frac{\hat{r}}{r^*},$$

Table 2. Time consumption (ms) comparison.

ϵ	Proposed					Xie et al. [10]					Zhu et al. [11]				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
0.1	525	588	647	644	713	5074	5221	5509	5622	5583	11507	12148	12377	12919	12764
0.05	714	788	756	868	844	9637	10062	10466	10609	10748	22392	23390	23490	24690	25108
0.02	1049	1229	1239	1349	1378	23444	24405	25619	25851	26155	54767	56758	58637	60134	61372
0.01	1811	1515	1574	1621	1689	46323	48297	49590	50611	52803	108600	113717	116318	121364	128131
0.005	3283	3648	3789	4138	4461	92237	96175	99275	100696	113079	217802	229536	271564	325844	357031

Table 3. ρ accuracy comparison.

ϵ	Proposed					Xie et al. [10]					Zhu et al. [11]				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
0.1	0.981	0.977	0.964	0.951	0.953	0.930	0.935	0.920	0.920	0.910	0.999	0.984	0.971	0.959	0.961
0.05	0.991	0.976	0.974	0.964	0.958	0.934	0.944	0.940	0.929	0.925	0.999	0.984	0.977	0.966	0.962
0.02	0.993	0.982	0.975	0.966	0.961	0.950	0.947	0.959	0.931	0.946	0.999	0.985	0.979	0.966	0.962
0.01	0.997	0.986	0.979	0.967	0.960	0.959	0.964	0.949	0.932	0.935	1.000	0.987	0.980	0.967	0.962
0.005	0.999	0.987	0.980	0.969	0.961	0.969	0.966	0.952	0.927	0.943	1.000	0.987	0.981	0.970	0.962

Table 4. RMSE error comparison.

ϵ	Proposed					Xie et al. [10]					Zhu et al. [11]				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
0.1	0.017	0.104	0.184	0.289	0.340	0.013	0.108	0.179	0.254	0.339	0.008	0.102	0.182	0.287	0.337
0.05	0.012	0.095	0.182	0.271	0.307	0.013	0.098	0.204	0.246	0.335	0.004	0.093	0.181	0.270	0.305
0.02	0.009	0.093	0.178	0.260	0.342	0.009	0.109	0.195	0.261	0.331	0.001	0.092	0.177	0.260	0.341
0.01	0.005	0.093	0.175	0.243	0.332	0.007	0.092	0.198	0.268	0.362	0.000	0.092	0.175	0.243	0.332
0.005	0.003	0.093	0.168	0.240	0.348	0.007	0.097	0.180	0.268	0.364	0.000	0.092	0.168	0.239	0.347

where d denotes the number of dimension, $\hat{c}$ and c^* denotes the center of MaxIB calculated by the tested algorithm and the interior point method. $\hat{r}$ and r^* are the corresponding radius with $\hat{c}$ and c^*, calculated in the following way:

$$\begin{aligned}\hat{r} &= \min_{1\le i\le m} \operatorname{dist}\big(c, \{x : A_i x + b_i = 0\}\big) \\ &= \min_{1\le i\le m} \frac{|A_i c + b_i|}{\|A_i\|_2} = \min_{1\le i\le m} \frac{b_i - A_i^T c}{\|A_i\|_2}\end{aligned}$$

where m is the number of faces of each polytope.The result is recorded in Tables 2, 3, and 4.

From Table 2, our method demonstrates superior time efficiency across all aspect ratios and error tolerance. At the highest aspect ratio, $\alpha = 5$, and with the highest accuracy, $\epsilon = 0.005$, our algorithm completes calculations in 4461 ms, representing a 25-fold speedup compared to Xie's 113,079 ms and an 80-fold improvement over Zhu's 357,031 ms.

From the RMSE analysis in Table 4, it can be observed that Zhu's method achieves the lowest absolute error at $\alpha = 1$, confirming its precision advantage in controlled scenarios. Our method provides comparable accuracy to Zhu et al. [11] (difference $\le 0.5\%$) while being two orders of magnitude faster. In other aspect ratios $\alpha = 1$, our method shows no significant disadvantage to Zhu's method in terms of accuracy while retaining a much shorter running time.

The radius ratio ρ analysis in Table 3 reveals distinct characteristics of each method. Allen-Zhu et al.'s [11] method achieves near-perfect accuracy ($\rho = 1$) for low aspect ratios ($\alpha = 1$), demonstrating its theoretical optimality in ideal conditions. However, reducing the tolerable error ϵ improves its performance in limited amount. Our proposed method shows better flexibility, with greater accuracy improvement as ϵ deceases. Furthermore, the proposed method shows better robustness, with only 3.8% degradation from $\alpha = 1$ to $\alpha = 5$. In contrast, Xie et al.'s [10] solution exhibits the highest sensitivity to shape variations, resulting in a 2.6% accuracy drop under the same conditions. This suggests that our method is more adaptable to complex polytopes. Nevertheless, all methods fail to fulfill high accuracy$(r = (1 - \epsilon)r_{opt})$, marked as black dotted lines in Figure 4 under high

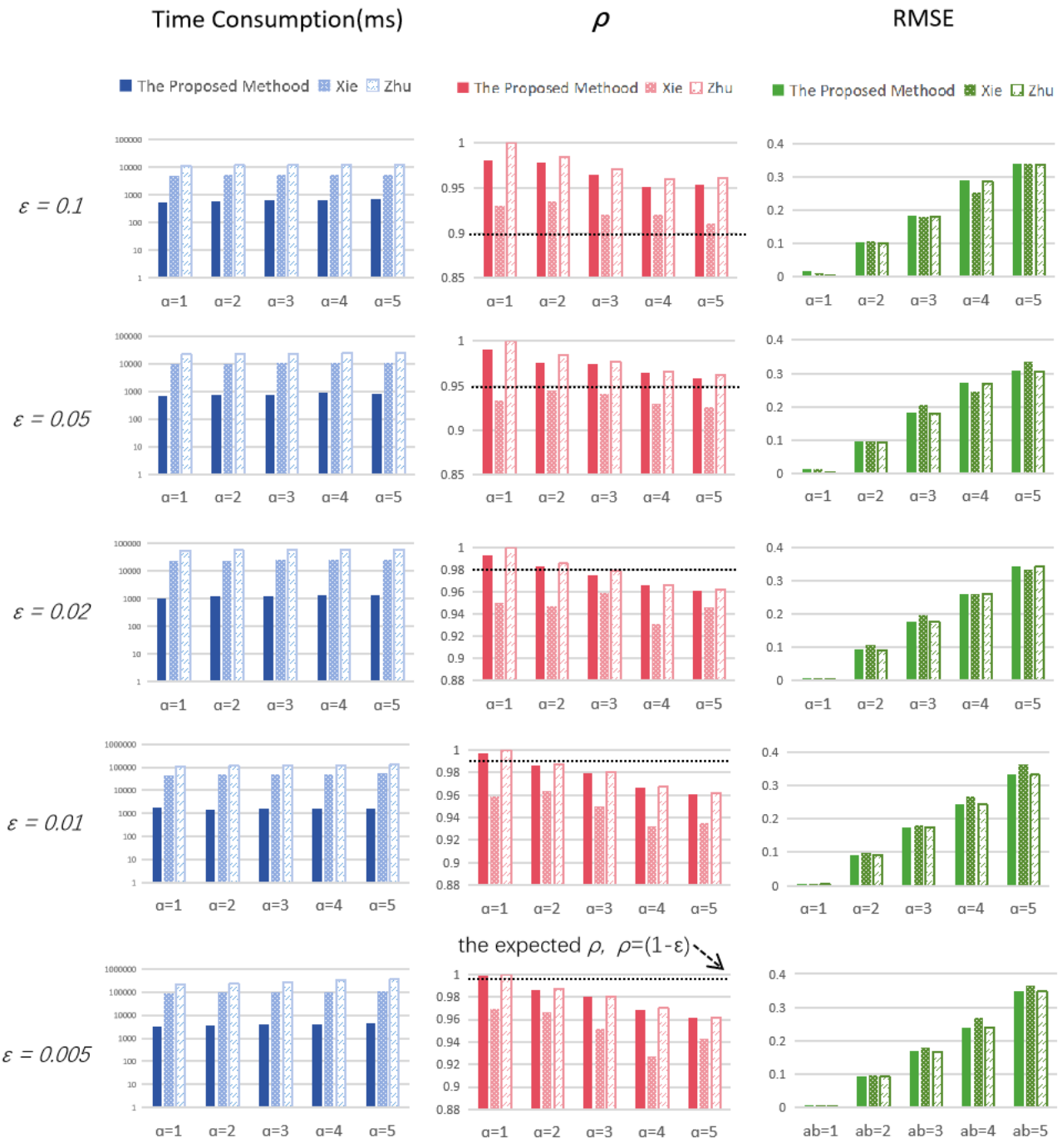

Figure 4. Time consumption, ρ, RMSE of methods in different α and ϵ.

aspect ratios. This is due to the narrow feasible region, which is still a challenge in all methods of convex optimization.

To sum up, the main advantage of our method is a significant lower running time compared to other approximate methods. Furthermore, our method exhibits higher robustness to changes in the polytopes' aspect ratios than Xie's method. Our method also achieves an accuracy close to Zhu's method in all trials within 20-fold of its running time. In high α and ϵ, our running time is 80-fold faster. This result demonstrates the distinct improvement of our method in efficiency.

5 Discussion

5.1 Interpretation of Results and Method Advantages

The proposed approach demonstrates substantial computational efficiency in solving the $(1 - \epsilon)$-approximate MaxIB problem for high-dimensional polytopes. By transforming

the primal MaxIB into a quadratic-penalized dual problem and applying Pairwise Frank–Wolfe (PFW) with adaptive backtracking, the algorithm reduces per-iteration computational overhead while leveraging strong convexity for improved convergence. This combination enables a runtime that scales linearly with the input complexity $O(md)$ up to a logarithmic factor, which is significantly more favorable than both LP-based exact solvers and prior approximate methods. Empirical evaluations confirm that the proposed method consistently achieves faster runtime across a wide range of dimensions and aspect ratios, with measured speedups of an order of magnitude relative to recent approximate methods.

Compared with classical FW variants, the pairwise update mechanism and early stopping based on duality gap contribute to improved iteration efficiency, while the backtracking line search removes the need for exact line searches and global curvature estimation. These characteristics make the approach particularly well-suited for large-scale and real-time optimization settings in geometric and control applications.

5.2 Limitations

Despite its favorable computational profile, the method exhibits sensitivity to the geometric conditioning of the polytope. In particular, achieving high accuracy becomes challenging when the aspect ratio α is large, as narrow feasible regions tend to amplify curvature and degrade dual feasibility, leading to reduced radius accuracy. This phenomenon also appears in prior approximate MaxIB algorithms, suggesting that conditioning-aware strategies remain underdeveloped in the current literature. Additionally, although backtracking greatly improves practical performance, it does not remove the worst-case dependence on α, and theoretical iteration bounds still scale with $\alpha^2 \ln(1/\epsilon)$.

5.3 Future Work

Several directions may further enhance the proposed framework. First, adaptive penalty scheduling strategies may dynamically adjust the quadratic penalty along directions of high curvature to reduce feasibility violation without slowing convergence. Second, hybrid solvers that combine the fast initial progress of PFW with the high-accuracy refinement of interior-point or active-set methods could achieve both computational efficiency and precision on ill-conditioned instances. Third, incorporating selective second-order information or curvature-aware updates may help mitigate performance degradation on high aspect-ratio polytopes. Finally, since the computational bottleneck lies primarily in matrix–vector products involving A, GPU or multi-core parallelization could significantly accelerate the method in ultra-high-dimensional settings.

6 Conclusion

This work addressed the challenge of efficiently computing a $(1-\epsilon)$-approximate maximum inscribed ball in high-dimensional polytopes, a problem with applications in warm-starting interior-point methods, robotic navigation, sensor deployment, and integer programming. By converting the primal MaxIB problem into a smooth, strongly convex dual formulation via quadratic penalization and solving it with a Pairwise Frank–Wolfe method equipped with adaptive backtracking, the resulting procedure avoids expensive projections and exhibits improved convergence behavior.

The resulting algorithm attains a runtime of

$$O(md\,\alpha^2 \ln(1/\epsilon)),$$

which is linear in input size up to logarithmic factors. Experiments on synthetic polytopes demonstrate up to order-of-magnitude speedups over existing approximate MaxIB solvers while maintaining competitive accuracy. Overall, the combination of dual smoothing, projection-free optimization, and adaptive curvature estimation provides an efficient and scalable methodology for geometric optimization tasks in high dimensions.

Data Availability Statement

The program used to test the methods is available at Github: https://github.com/Linyuqi2/MaxIB-Calculation-Methods-Evaluation.

Funding

This work was supported without any funding.

Conflicts of Interest

The author declares no conflicts of interest.

AI Use Statement

The author declares that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Yildirim, E. A., & Wright, S. J. (2002). Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization, 12*(3), 782–810. [CrossRef]

[2] Belykh, T. I., Bulatov, V. P., & Yaskova, E. N. (2008). Methods of Chebyshev points of convex sets and their applications. *Computational Mathematics and Mathematical Physics, 48*(1), 16–29. [CrossRef]

[3] Marcucci, T., Petersen, M., Von Wrangel, D., & Tedrake, R. (2023). Motion planning around obstacles with convex optimization. *Science robotics, 8*(84), eadf7843. [CrossRef]

[4] Sra, S., Nowozin, S., & Wright, S. J. (Eds.). (2012). *Optimization for machine learning*. MIT Press. [CrossRef]

[5] Yildirim, E. A. (2008). Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization, 19*(3), 1368-1391. [CrossRef]

[6] Baena, D., & Castro, J. (2023). The Chebyshev center as an alternative to the analytic center in the feasibility pump. *Optimization Letters, 17*(8), 1757–1790. [CrossRef]

[7] Dantzig, G. (1963). Linear Programming and Extensions. *Linear Programming and Extensions Princeton*. [CrossRef]

[8] Karmarkar, N. (1984, December). A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing* (pp. 302-311). [CrossRef]

[9] Ye, Y. (1997). *Interior point algorithms: Theory and analysis*. John Wiley & Sons. [CrossRef]

[10] Xie, Y., Snoeyink, J., & Xu, J. (2006, June). Efficient algorithm for approximating maximum inscribed sphere in high dimensional polytope. In *Proceedings of the twenty-second annual symposium on Computational Geometry* (pp. 21-29). [CrossRef]

[11] Allen-Zhu, Z., Liao, Z., & Orecchia, L. (2014). Using optimization to find maximum inscribed balls and minimum enclosing balls. *arXiv preprint arXiv:1412.1001*. [CrossRef]

[12] Klee, V., & Minty, G. J. (1972). How good is the simplex algorithm. *Inequalities, 3*(3), 159-175.

[13] Wright, S. J. (1997). Primal-Dual Interior-Point Methods. *Primal-Dual Interior-Point Methods. SIAM*. [CrossRef]

[14] Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly, 3*(1–2), 95–110. [CrossRef]

[15] Jaggi, M. (2013, February). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International conference on machine learning* (pp. 427-435). PMLR.

[16] Beck, A., & Shtern, S. (2017). Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming, 164*(1), 1-27. [CrossRef]

[17] Lacoste-Julien, S., & Jaggi, M. (2015). On the global linear convergence of Frank-Wolfe optimization variants. *Advances in neural information processing systems, 28*.

[18] Bertsekas, D. P. (2014). *Constrained optimization and Lagrange multiplier methods* (2nd ed.). Athena Scientific.

[19] Li, B., Coutino, M., Giannakis, G. B., & Leus, G. (2021). A momentum-guided Frank-Wolfe algorithm. *IEEE Transactions on Signal Processing, 69*, 3597-3611. [CrossRef]

[20] Freund, R. M., & Grigas, P. (2016). New analysis and results for the Frank–Wolfe method. *Mathematical Programming, 155*(1), 199-230. [CrossRef]

[21] Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (4th ed.). Johns Hopkins University Press.

[22] Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics, 16*(1), 1–3. [CrossRef]

[23] Pedregosa, F., Negiar, G., Askari, A., & Jaggi, M. (2020, June). Linearly convergent Frank-Wolfe with backtracking line-search. In *International conference on artificial intelligence and statistics* (pp. 1-10). PMLR.

[24] Ziegler, G. M. (2012). *Lectures on polytopes*. Springer Science & Business Media.

[25] Berkelaar, M., & others. (2015). *lpSolve: Interface to 'lp_solve' v. 5.5 to solve linear/integer programs* (Version 5.6.13) [Computer software]. CRAN. [CrossRef]

[26] Free Software Foundation. (n.d.). *GNU Linear Programming Kit (GLPK)* [Computer software]. Retrieved from https://www.gnu.org/software/glpk/

[27] Mosek ApS. (n.d.). *MOSEK Optimization Suite* [Computer software]. Retrieved from https://www.mosek.com/



Yuqi Lin is currently pursuing his studies at The Grainger College of Engineering, University of Illinois Urbana-Champaign (UIUC). His research interests include deep learning architectures and time-series forecasting, with a particular focus on bridging theory and real-world applications. He aims to advance the practical deployment of machine learning techniques for complex temporal data problems. With a solid foundation in both theoretical analysis and system implementation, he contributes to research that connects modern AI methodologies with practical engineering challenges. (Email: yuqilin2@illinois.edu)