Efficient Vertex-Oriented Polytopic Projection for Web-scale Applications

Rohan Ramanath*, S. Sathiya Keerthi, Yao Pan, Konstantin Salomatin, Kinjal Basu

LinkedIn Corporation
Mountain View, CA, USA
rohan@ramanath.me, {keselvaraj, yopan, ksalomatin, kbasu}@linkedin.com

Abstract

We consider applications involving a large set of instances of projecting points to polytopes. We develop an intuition guided by theoretical and empirical analysis to show that when these instances follow certain structures, a large majority of the projections lie on vertices of the polytopes. To do these projections efficiently we derive a vertex-oriented incremental algorithm to project a point onto an arbitrary polytope, as well as give specific algorithms to cater to simplex projection and polytopes where the unit box is cut by planes. Such settings are especially useful in web-scale applications such as optimal matching or allocation problems. Several such problems in internet marketplaces (e-commerce, ride-sharing, food delivery, professional services, advertising, etc.), can be formulated as Linear Programs (LP) with such polytope constraints that require a projection step in the overall optimization process. We show that in the very recent work [5], the polytopic projection is the most expensive step and our efficient projection algorithms help in gaining massive improvements in performance.

1 Introduction

Euclidean projection onto a polytope is not only a fundamental problem in computational geometry [24, 38], but also has several real-life applications ranging from decoding of low-density parity-check codes [43] to generating screening rules for Lasso [42]. Recent work [5, 7] also uses polytope projection as an operation in solving large optimization problems. Several such large-scale optimization solvers which use polytopic projection [5, 7, 36] are being used to tackle different problems in the internet industry such as matching [3, 29] and multi-objective optimization [1, 2]. Other examples include problems in natural language processing like structured prediction [40, 39, 31], semi-supervised approaches [10], multi-class/hierarchical classification [25], etc.

Given a compact set $C$ and a point $x$, let $\Pi_C(x)$ denote the (euclidean) projection of $x$ onto $C$, i.e., $\Pi_C(x) = \arg\min_{y \in C} ||y - x||$. Certain large scale optimization problems, especially those arising from applications in the internet industry, have a huge number of polytope constraint sets, $\{C_i\}$. To solve such problems each iteration of an algorithm requires the projection of points $\{x_i\}$ to $\tilde{x}_i = \Pi_{C_i}(x_i)$. Computing such sets of projections is usually the key bottleneck [43] and hence efficiently solving the projections is crucially important to solve such large-scale optimization problems.

The set of points, $\{x_i\}$ usually has some structure connecting the $x_i$ that dictates the nature of the projections. Most general purpose solvers such as ADMM [7], Block Splitting [36] and Splitting Conic Solver [35] do not exploit this special structure. Very recently, Basu et. al. [5] proposed ECLIPSE (shorthand Ecl) - to solve such large scale problems while exploiting some general overall structure, but still used off-the-shelf projection solvers such as the one in [14] to tackle the projection step. As a result, similar to the general purpose algorithms [7, 35, 36], the major computational bottleneck for Ecl is also the projection step.

Although there are many different forms of projection structures, in this paper we focus on a particular structure motivated by applications in the internet industry. We consider those, for which, a majority of the projections, $\tilde{x}_i$, lies on a vertex of $X_i$ and also, over all $i$, the mean dimension of the minimal face [20] of $X_i$ that contains $\tilde{x}_i$ is very small. We refer to this as the vertex oriented structure. Even for special polytopes such as the simplex and box-cut (a hypercube cut by a single plane) general purpose projection algorithms [14] are not designed to be efficient for the above situation. The main goal of this paper is to develop special purpose projection algorithms that (a) are specially efficient for the vertex oriented structure; and (b) are applicable to special polytopes such as the simplex, box-cut, and general polytope forms.

To show the efficacy of these special purpose projection algorithms, we switch out the projection step of Ecl, and run an ablation study over various internet marketplace problems. We call this new solver DualLip, a Dual decomposition-based Linear Programming framework which shows drastic speedups over the existing state-of-the-art. Moreover, Ecl only tackled simple polytope constraints such as the unit box or the simplex, which not only reduced the generality, but also made the solver tough to use right out of the box. By replacing the projection component with the novel algorithms developed in this paper, DualLip is now capable of solving a much wider range real-world applications,
We begin with some notation. Let \( x \in \mathbb{R}^K \) denote the point that we wish to project onto a compact polytope \( C \). The projection operator \( \Pi(x) \) can be written as \( \Pi_C(x) = \arg \min_{y \in C} \| y - x \| \) where \( \| \cdot \| \) denotes the Euclidean distance. Throughout this paper, we consider the following set of polytopes:

1. **Box**: \( C = \{ x : 0 \leq x_k \leq 1 \ \forall k \} \)
2. **Simplex-Eq**: \( C = \{ x : x_k \geq 0 \ \forall k, \sum_k x_k = 1 \} \)
3. **Simplex-Iq**: \( C = \{ x : 0 \leq x_k \leq 1 \ \forall k \} \)
4. **Box-Cut-Eq**: \( C = \{ x : 0 \leq x_k \leq 1 \ \forall k, \sum_k x_k = \delta \} \) (\( \delta \) is a positive integer, \( 1 < \delta < K \))
5. **Box-Cut-Iq**: \( C = \{ x : 0 \leq x_k \leq 1 \ \forall k, \sum_k x_k \leq \delta \} \) (\( \delta \) is a positive integer, \( 1 < \delta < K \))
6. **Parity Polytope**: \( C = co \{ v_r \} \) where the \( v_r \) are binary vectors with an even number of 1s and \( co \) denotes convex hull.
7. **General Polytope**: \( C = co \{ v : v \in V \} \) where \( V \) is a finite set of polytope vertices.

Here we use \( E, I \) to denote an equality or inequality sign denoting whether we are interested in the surface or the closed interior of the polytope. Such polytopes naturally occur as constraints in different optimization formulations in the internet industry. We motivate the need of such polytopes through typical recommender system problems.

We denote users by \( i = 1, \ldots, I \) and items by \( k = 1, \ldots, K \). Let \( x_{is} \) denote any association between user \( i \) and item \( k \), and be the variable of interest. For example, \( x_{ik} \) can be the probability of displaying item \( k \) to user \( i \). The vectorized version is denoted by \( x = (x_{11}, \ldots, x_{I1}) \) where \( x_{i} = \{ x_{ik} \}_{k=1}^{K} \). Throughout this paper we consider problems of the form:

\[
\min_x c^T x \quad \text{s.t.} \quad Ax \leq b, \ x_i \in C_i, \ i \in [I],
\]

where \( A_{m \times n} \) is the constraint matrix, \( b_{m \times 1} \) and \( c_{n \times 1} \) are the constraint and objective vectors respectively, and \( C_i \) are compact polytopes. \( x \in \mathbb{R}^n \) is the vector of optimization variables, where \( n = IK \) and \( [I] = \{1, \ldots, I\} \).

**Applications**

Basu et al. [3] described two major classes of problems, volume optimization (focused on email/notifications) and optimal matching (for network growth). We cover a much larger class of problems using a variety of different polytopic constraints.

**Diversity in Network Growth**: “Rich getting richer” is a common phenomenon in building networks of users [16]. Frequent or power users tend to have a large network and get the most attention, while infrequent members tend to lose out. To prevent these from happening, and to improve the diversity in the recommended users, we can frame the problem as a LP:

\[
\begin{align*}
\max_x & \sum_{ik} x_{ik} c_{ik} \quad & \text{(Total Utility)} \\
\text{s.t.} & \sum_{i} x_{ik} a_{ik} \geq b_k \ \forall k & \text{(Infrequent User Bound)} \\
& \sum_k x_{ik} = \delta_i, \ 0 \leq x_{ik} \leq 1
\end{align*}
\]

where \( c, a \) are utility and invitation models, \( b_k \) denotes the minimum number of invitations to the \( k \)-th infrequent user, and \( \delta_i \) denotes the number of recommendations to the \( i \)-th user. This makes \( C_i \) the **Box-Cut-Eq** polytope.

**Item Matching in Marketplace Setting**: In many two-sided marketplaces, there are creators and consumers. Each item created has an associated budget and the problem is to maximize the utility under budget constraints. For example, in the ads marketplace, each ad campaign has a budget and we need to distribute impressions appropriately. For the jobs marketplace, each paid job has a budget and the impressions need to be appropriately allocated to maximize job applications. Each of these problems can be written as:

\[
\begin{align*}
\max_x & \sum_{ik} x_{ik} c_{ik} \\
\text{s.t.} & \sum_{i} x_{ik} a_{ik} \leq b_k \ \forall k \in [K], \\
& \sum_k x_{ik} \leq \delta_i \ \text{and} \ 0 \leq x_{ik} \leq 1
\end{align*}
\]

where \( c, a \) are models estimating utility and budget revenue, respectively and \( \delta_i \) is the maximum number of eligible items to be shown to the \( i \)-th member. Here the polytope constraint is **Box-Cut-Iq**.

**Multiple Slot Assignment**: This is an extension of the item ranking problem where the utility of an item depends on the position in which it was shown [26]. Consider for each request or query \( i \), we have to rank \( L_i \) items (\( \ell = 1, \ldots, L_i \)) in \( S_i \) slots (\( s = 1, \ldots, S_i \)). We want to maximize the associated utility subject to constraints on the items. Mathematically this bipartite structure of multiple items and slots can be framed as:

\[
\begin{align*}
\max_x & \sum_{ist} x_{ists} c_{ists} \\
\text{s.t.} & \sum_{ist} x_{ists} a_{jists} \leq c_j \ \forall j \in [J] \\
& \sum_s x_{ists} = 1, \sum_\ell x_{ists} = 1 \text{ and } x_{ists} \geq 0 \ \forall i, \ell, s
\end{align*}
\]

where \( c \) and \( a \) are the associated utility and \( j \)-th constraint values. Note that the projection set \( C_i \) captures the need to show each item and the fact that each slot can contain only one item.
such assignment per request. Thus we have,
\[
\begin{align*}
\max_x & \quad \sum_{ik} x_{ik}c_{ik} \\
\text{s.t.} & \quad \sum_{ik} x_{ik}\gamma_{ik} \leq c_j \quad \forall j \in [J] \\
& \quad \sum_k x_{ik} = 1 \text{ and } x_{ik} \geq 0 \forall i, k
\end{align*}
\]
(5)

**Slate Optimization:** This is a variant of the multiple slot assignment problem, where each of the \(L_i\) items are ranked and each index \(k\) (slate) corresponds to a ranked selection of distinct items to be placed in the slots. This can be set up as a path search on a DAG; see [26] for more details. Both the Slate and Assignment problems can be written using the Simplex-Eq polytopic constraint. Table 1 describes these applications with respect to the associated models, projection, and problem size.

In the subsequent sections, we describe how we develop new projection algorithms to handle such larger classes of problems. To demonstrate the value of these novel algorithms, we will work with a proprietary Jobs Matching Dataset (D). Here, we are trying to solve the problem of the form in (3), where instead of the Box-Cut constraint we consider the simplex constraint: \(\sum_k x_{ik} = 1\) and \(0 \leq x_{ik} \leq 1\) and we have \(n = 100\) trillion. We rely on a real-world dataset only to represent the scale of the problem, i.e. the number of coupling and non-coupling constraints seen in practice. The data does not contain any personally identifiable information, respects GDPR and does not contain any other user information. To promote reproducible research we open source (with FreeBSD License) the solver with all the efficient projection operators discussed in the paper. There, we report the solver’s performance on the open-source movie-lens dataset [22], which contains user rating of movies to formulate an optimization problem (taking a similar approach to [22]). Movie-lens is a public domain matching problem of a much smaller scale than seen in internet marketplaces.

### Solving the LP

For sake of completeness, we briefly describe Ecl algorithm here. To solve problem (1), Ecl introduces a new perturbed problem
\[
\min_x c^T x + \frac{\gamma}{2} x^T x \quad \text{s.t.} \quad Ax \leq b, \ x_i \in C_i, i \in [I]
\]
where \(\gamma > 0\) helps to make the dual objective function smooth; \(\gamma\) is kept small to keep the solution close to that of the original problem (1). To make the problem (6) amenable to first order methods, Ecl considers the Lagrangian dual,
\[
g_\gamma(\lambda) = \min_{x \in \mathcal{C}} \left\{ c^T x + \frac{\gamma}{2} x^T x + \lambda^T (Ax - b) \right\},
\]
where \(\mathcal{C} = \Pi_{i=1}^I C_i\). Now, by strong duality [8], the optimum objective \(g_\gamma\) of the dual
\[
g_\gamma^* := \max_{\lambda \geq 0} g_\gamma(\lambda)
\]
is the minimum of (6). Ecl shows that \(\lambda \mapsto g_\gamma(\lambda)\) is differentiable and the gradient is Lipschitz continuous. Moreover, by Danskin’s Theorem [13] the gradient can be explicitly expressed as, \(\nabla g_\gamma(\lambda) = A^*_\gamma(\lambda) - b\) where,
\[
x^*_\gamma(\lambda) = \arg\min_{x \in \mathcal{C}} \left\{ c^T x + \frac{\gamma}{2} x^T x + \lambda^T (Ax - b) \right\}
\]
\[
= \left\{ \Pi_{C_i} \left[ -\frac{1}{\gamma} (A_i^T \lambda + c_i) \right] \right\}_{i=1}^I
\]
(9)
where \(\Pi_{C_i} (\cdot)\) is the Euclidean projection operator onto \(C_i\), and, \(A_i, c_i\) are the parts of \(A\) and \(c\) corresponding to \(x_i\). Based on this Ecl uses accelerated gradient method as the main optimizer to solve the problem in (6). For more details, we refer to [8].

### Cost Profiling

On large datasets such as \(D\) we find that Ecl takes days to solve. This motivates us to understand which parts can be improved to get the biggest overall speedup. We profile each of the parts to analyze the performance bottlenecks of Ecl. Table 2 shows the complexity of each term required to compute the gradient along with the sample time spent per iteration of the solver. It is clear that the optimizer’s internal update steps only take \(3\%\) of the time while the rest (\(97\%\)) is used to compute the gradient (columns 2 to 6 in Table 2). The results are consistent on other datasets as well when the number of executors is tuned appropriately. For the rest of paper, we only focus on the most expensive (\(74\%\)) step in the gradient computation, that is the projection operation to compute \(x^*_\gamma(\lambda)\) as in (9).

### 3 Efficient Projection Algorithms

Recall the projection problem to be solved: for each \(i\), we want to find
\[
(x^*_\gamma(\lambda))_i = \Pi_{C_i} \left[ -\frac{1}{\gamma} (A_i^T \lambda + c_i) \right] = \arg\min_{x_i \in C_i} \|x_i - \hat{x}_i\|^2
\]
where \(\hat{x}_i = -\frac{1}{\gamma} (A_i^T \lambda + c_i)\). To simplify notation, we will leave out the ‘\(i\)’ unless it is necessary. The cost profiling shows that the projection step forms the bulk of the overall algorithm cost and hence, improving its speed would lead
to overall algorithmic speedup. If projections form ϕ fraction of the total solution cost, then by making projections extremely efficient one can hope to get up to a speedup of \(1/(1 - ϕ)\). We now focus on designing efficient algorithms that are well suited to marketplace problems. Ecl covers Box and Simplex-Eq. The Box projection solution is very efficient, but in our case the Simplex-Eq projection algorithm \([14]\) is not. This is primarily because, for a large fraction of \(i\)'s, the projections are at a vertex of \(C_i\), which can be identified in a more efficient way.

**The Main Intuition:** We define a **corral** \([9]\) as the convex hull of a subset of the vertices of \(C_i\) with least cardinality that contains the projection of \(\hat{x}\). Thus, if the projection is a vertex, then the vertex itself is the corral; if the projection lies in the relative interior of an edge formed by two vertices, then the corral is that edge, and so on. The dimension of the corral is one less than the cardinality of its vertex subset. A vertex is a corral of dimension zero, an edge is a corral of dimension 1, and so on. Clearly there exists a corral that contains the projection, \(\hat{x}^∗\).

Our efficient approach is based on the intuition that, for small \(γ\), the mean dimension of the corral over all \(i\) is small. (As already mentioned below \([5]\), the value of \(γ\) is chosen to be close to zero.) Since this is the main basis of the paper, we explain the reasoning behind the intuition. We begin by giving a simple example in 2d. Then we state a formal result that says what happens at \(γ = 0\). We follow that by taking the case of Simplex-Eq and demonstrating how, as \(γ\) becomes larger than zero, the mean corral dimension only increases gradually with \(γ\). Further, we give strong empirical support for the intuition. After that, the rest of the section and \([4]\) focus on developing efficient projection algorithms based on the intuition.

Let us begin by illustrating the intuition using an example of Simplex-Eq with \(K = 2\); see Figure 1. From the figure it is clear that, for small values of \(γ\), the probability that the projection lies on a vertex (corral of dimension zero) is high.

We now state a formal result for the \(γ = 0\) case. The proof of Theorem 1 is given in the full paper \([17]\).

**Theorem 1.** Let: \(x \in C\) be expressed as \(Dx \leq d, Ex = e\), \(p = (b, c, d, e, γ)\), \(x^∗(p)\) denote the optimal solution of \((1)\), and \(x^∗_i(p, λ) = \arg\min_{x \in C} (c + AT x)\) (also same as \([9]\) with \(γ \to 0\)). For any \(x\) define \(μ(x) = \sum_i μ_i / I\) to be the mean corral dimension where \(μ_i\) is the corral dimension of \(x_i\). Let \(B_ε(z)\) denote the open ball with center \(z\) and radius \(ε\). Given \(ε, ε_c, ε_d, ε_e > 0\), if we define \(B := \{\bar{b}, \bar{c}, \bar{d}, \bar{e}\} : \bar{b} \in B_{ε}(\bar{b}), \bar{c} \in B_{ε}(\bar{c}), d \in B_{ε_d}(d), \bar{e} \in B_{ε_e}(\bar{e})\}; then 1. \(x^∗(p)\) is unique and \(μ(x^∗(p)) \leq m/I\) for almost all \(p\).

\(γ = γ_2 = 0.75γ_1\)

\[p \in B\]

2. \(x^∗_i(p, λ)\) is unique and \(μ(x^∗_i(p, λ)) = 0\), for almost all \(p \in B\) and \(λ \in \mathbb{R}^m\).

Note that since each \(μ_i \in \mathbb{Z}\), it directly follows from Theorem 1 that \(I\), the number of \(i\)'s with vertex solutions \((μ_i = 0)\) is at least \(I - m\). Moreover, \(K\), which denotes the dimension of the \(x_i\), does not affect \(μ\) at all (due to the structure of \(C_i\) and the repeating nature in \(i\)). This is beneficial since \(K\) is very large in Assignment and Slate projections (see Table 1). Comparing results 1 and 2 of Theorem 1 we can see that non-vertex solutions start appearing as the solution approaches \(λ_0 = \arg\max_{x \geq 0} g_0(λ)\), and even there, it is well bounded. In web-scale marketplace applications \(m/I\) is very small, say, smaller than 0.01. Thus, vertex projections dominate and they occur in more than 99% of the \(i\)'s even as we come close to \(λ_0\).

A limitation of the theorem is that it gives a bound on \(μ\) only at \(γ = 0\). On the other hand, in our solution process, we use small \(γ > 0\). By Lemma 2 of \([5]\) (also see \([30, 17]\)), the solution \(x^∗\) of \((1)\) remains unchanged under the \(γ\) perturbation for sufficiently small positive \(γ\) values (Figure 1 gives the rough intuition), and so part 1 of the theorem applies for such positive \(γ\).

Even the above formal result is limited. To better explain what happens as we gradually increase \(γ\) above zero, let us analyze the Simplex-Eq case. Let us take any one \(i\), but omit the subscript \(i\) to avoid clumsiness. Let \(x = -(AT λ + c)\) so that \(x = zγ\). We invoke Algorithm 2 (given in the next section) to give a precise analysis. Given a \(λ\) and hence, given \(x\), we can use Algorithm 2 to quantify when a vertex (corral of dimension 0) will be the projection, when an edge (corral of dimension 1) will contain the projection, etc. In general po-

---

### Table 2: Complexity analysis of different components on dataset \(D_1\) with \(w = 100\) nodes, \(k, μ_e\) as defined later. (a) In gray is the time spent inside the optimizer, (b) in white is the time taken to compute the gradient. Notation as defined in \([10]\)

| Profiling | \(A^*_e\) | \(\lambda\) | \(\hat{x}_1(\lambda)\) | \(A^*\hat{x}_1\) | \(Ax, c, \|\hat{x}\|\) | \(\lambda^t(\hat{Ax} - b)\) | LBFGS |
|-----------|---------|-------------|----------------|----------------|----------------|----------------|-------|
| Complexity | \(O(k)\) | \(O(k \cdot \log k)\) | \(O(\mu_e)\) | \(O(I/w) + O(w)\) | \(O(K)\) | \(O(K)\) | \(O(1)\) |
| Sample time | 10% | 74% | 2% | 8% | 3% | 3% |
situation (think of random infinitesimal perturbation of c) there will be a positive separation between the components of z. Without loss of generality let us take the components of z to be ordered: \( z_1 > z_2 > z_3 > \ldots \). With this in place, it is easy to use Algorithm 2 and derive the following: (a) Projection will be a vertex if \( 0 < \gamma \leq (z_1 - z_2) \); (b) Projection will be on an edge if \( (z_1 - z_2) \leq \gamma \leq (z_1 - z_3) + (z_2 - z_3) \); (c) and so on. Thus projections start lying on higher dimensional corrals only slowly as \( \gamma \) is increased. There is a graded monotonic increase in the optimal corral dimension with respect to \( \gamma \).

It is possible to give similar details for constraint sets other than Simplex-Eq; it is just that they get quite clumsy. In addition to what is given above, the other key point is that, as we move towards optimality, part 1 of Theorem 1 says that only a few \( i \)'s (at most \( m \) of them at optimality where \( m \) is the number of constraints in \( Ax \leq b \)) will have non-vertex projections. Going by the graded effect of positive \( \gamma \) on optimal corrals only slowly as \( \gamma \) is increased, we get useful indications of what small values of \( \mu \) we can expect as we close in on optimality with small (not necessarily extremely small) \( \gamma \) (also see the empirical results in Table 3).

Let us use empirical analysis to further understand what happens for larger \( \gamma \) perturbations. Table 3 studies the behavior of \( \mu \) along optimization paths of \( \max_{x, \gamma} g_{\gamma}(\lambda) \) for a range of \( \gamma \) values, on the representative dataset \( D \). Even with large \( \gamma \) values, note that \( \mu \) is usually small (less than 3). Obtaining a theoretical bound on \( \mu \) as a function of \( \gamma \) is an interesting topic of future academic research. Such theory can be useful in many optimization formulations that invoke projection as a subset.

### Table 3: Behavior of the mean corral dimension, \( \mu \) along optimization paths of \( g_{\gamma} \), starting from \( \lambda = 0 \) for three values of \( \gamma \) s.t. \( \gamma_1 > \gamma_2 > \gamma_3 \).

| Corral metric | \( \gamma_1 = 1 \) | \( \gamma_2 = 0.1 \) | \( \gamma_3 = 0.01 \) |
|---------------|-----------------|-----------------|-----------------|
| \( \mu \)     | 2.55            | 0.12            | 0.03            |
| \( \tilde{I}/I \) | 0.4522          | 0.9232          | 0.9676          |

**Vertex-Oriented Approach**

Guided by the theory and the empirical observations that (a) vertex solutions occur most frequently and (b) the mean corral size is small, we devise the powerful approach to projection described in Algorithm 1.

### Algorithm 1: DuaLip vertex-first projection

1. Let \( v^0 \) be the vertex of \( C \) nearest to \( \hat{x} \).
2. Check if \( v^0 \) is the optimal solution to (10).
3. If \( v^0 \) is not optimal, include new vertices, \( v^* \) and search over corrals of increasing dimension.

As we will see later for Simplex-Eq and Box-Cut-Eq, the algorithms designed with this approach as the basis are a lot more efficient for our problem solution than off-the-shelf algorithms that are known to be generally efficient for finding the projection (14). Also, for the Assignment and Slate cases, it is important to mention that no efficient projection algorithms exist in the literature. The algorithms that we develop for them here are new and are well-suited for our problem solution.

Before we give our algorithms for the specially structured polytopes in (4) let us discuss a useful general purpose, vertex-oriented projection algorithm for General Polytope. Many classical algorithms exist for solving the projection-to-convex-hull problem:

\[
\min_{x \in C} \|x - \hat{x}\|^2 \quad \text{where} \quad C = \text{co } \{v : v \in V\} \quad (11)
\]

The algorithms in (18, 43, 32, 31, 19, 44, 9) are a few of them. DuaLip can be made to work well with any one of these algorithms. More than the algorithm choice, the following three key pieces, which are all our contributions, are absolutely essential for efficiency (a) While these algorithms allow an arbitrary corral for initialization, our vertex-oriented approach always starts with the nearest-vertex single-point corral. (b) Even outside of these algorithms we check if this single-point corral is already optimal for (11) and, if so, these algorithms are never called. Note that the computations in (a) and (b) are an integral part of our Algorithm 1 (steps 1 and 2 there). (c) While these algorithms are set up for the projected point being the origin, doing a transformation of the convex hull points to meet this requirement would lead to a large inefficiency. It is important that we keep the original vertices as they are and carefully modify all operations of the algorithms efficiently. The summary of this is that, for General Polytope, DuaLip is not wedded specifically to any particular general-purpose algorithm but rather to the problem (11), and, for the class of problems and applications covered in the paper, we do a lot more to make a chosen algorithm to work efficiently than that algorithm itself.

**Wolfe’s algorithm for General Polytope**

Due to its popularity, we delve into the classic algorithm given by Wolfe (44, 9) for (10), which also nicely instantiates the approach in Algorithm 1 for the General Polytope case. We briefly outline this algorithm here. At a general iteration of the Wolfe algorithm, there is a corral \( C \) formed using a subset of vertices \( U \), and \( x \), the projection of \( \hat{x} \) to \( C \). Using optimality conditions it is easy to show that \( x \) solves (11) iff

\[
\min \{(x - \hat{x})^Tv : v \in V \setminus U\} \geq (x - \hat{x})^Tx \quad (12)
\]

If (12) is violated then the inclusion of \( v = \arg \min \{(x - \hat{x})^Tv : v \in V \setminus U\} \) to the vertex set of \( C \) to form the new corral, \( \hat{C} \), is guaranteed to be such that \( \hat{x} \), the projection on \( \hat{C} \), satisfies \( \|\hat{x} - \hat{x}\|^2 < \|x - \hat{x}\|^2 \). Since it is a descent algorithm and the number of corrals are finite, the algorithm terminates finitely. For more details see the full paper (37).

Wolfe’s algorithm can be initialized with an arbitrary corral and \( x \) as the projection to that corral. However, to be in line with our approach in Algorithm 1 we initialize \( x \) to be the vertex in \( V \) that is closest to \( \hat{x} \). This step is also simple.

---

3For example, our initial implementation of using the Wolfe algorithm (44, 9) without optimizations (a) through (c) made the projection 100x slower on a test problem.
Wolfe’s algorithm as detailed in Section 3 is highly efficient when the corral dimension is small. It can be made even more efficient for problems with a special structure such as Simplex, Box-Cut, and Parity Polytope.

**Simplex Projection:** For Simplex-Eq, \( C = \text{co}\{v_k\}_{k=1}^{K} \) where \( v_k \) is the vector whose \( k \)-th component is equal to 1 and all the other components are equal to 0. \( E \) employs the algorithm from \( [14] \) Algorithm 1 for Simplex-Eq, with complexity \( O(K \log K) \). The algorithm also requires all components of \( \hat{x} \) to be available; in the cases of Assignment and Slate, this is very expensive.

The algorithm of \( [14] \) is inefficient for our problem since it can be done in \( O(K) \) effort, is expected to be the projection. We can make it more efficient guided by the approach in Algorithm 1 and specifically, the Wolfe algorithm outlined earlier. The steps are given in Algorithm 2. Since \( \|v_j - \hat{x}\|^2 = 1 + \|\hat{x}\|^2 - 2\hat{x}_j \), the first step is equivalent to finding the index \( k \) that has the largest \( \hat{x} \) component, which forms the initializing corral. The remaining steps correspond to sequentially including the next best index (Step 3), checking if it does not lead to any further improvement (Steps 4 and 5), and if so, stopping with the projection. Step 3 is worth pointing out in more detail. It is based on \( [12] \). Because all vertices of \( C \) are orthogonal to each other and \( x \) is a linear combination of the vertices in \( U \), \( x^T v = 0 \) \( \forall v \in V \setminus U \). Thus, \( \arg\min\{x^T u : v \in V \setminus U\} = \arg\max\{\hat{x}^T v : v \in V \setminus U\} \), leading to Step 3 of Algorithm 2.

**Simplex-Eq** can be efficiently solved by first obtaining \( x^\text{box} \), the box projection. If \( \sum_k x^\text{box}_k \leq 1 \), then \( x^\text{box} \) is also the Simplex-Iq projection; else, the Simplex-Eq projection is the first Simplex-Eq projection.

In the normal case where all components of \( \hat{x} \) are available, the incremental determination of the next best vertex in step 3 can be done using the max heapify algorithm \( [13] \). If the algorithm stops in \( q \) steps, then the overall complexity is \( O(K + q \log K) \). \( [4] \) gives a similar algorithm especially for Simplex-Eq, but without the geometrical intuition and the power of its extension to structured outputs.

**Simplex-Eq Projection for Structured Outputs:** Consider any structured output setting in which, for each \( i \) there is a parameter vector, \( \xi_i \), and a possibly large set of configurations indexed by \( k = 1, \ldots, K \), such that: (a) for each given configuration \( k, c_{ik} \) and \( a^{(j)}_{ik} \) \( \forall j \) can be easily computed using \( \xi_i \); and (b) the incremental determination of the next best \( k \) according to the value, \( \hat{c}_{ik} = c_{ik} + \sum_j \lambda_j a^{(j)}_{ik} \) is efficient. Using a Simplex-Eq setup it is clear that Algorithm 2 efficiently solves the projection problem for this case. Note that the \( \hat{c}_{ik}, c_{ik}, a^{(j)}_{ik} \) do not need to be computed for all \( k \).

### Algorithm 2: Modified Duchi et al. algorithm for Simplex-Eq

1: \( k_1 \leftarrow \arg\max_{k \in |K|} \hat{x}_k \), and set: \( K = \{k_1\} \)
2: do
3: \( \tilde{k} = \arg\max_{k \in |K| \setminus K} \hat{x}_k \). \( \triangleright \) next best vertex
4: \( \alpha = \hat{x}_k - \frac{1}{|K|+1} \left( \sum_{r \in K, \cup(k)} \hat{x}_r - 1 \right) \).
5: Set \( K \leftarrow K \cup \{\tilde{k}\} \).
6: while \( \alpha > 0 \)
7: \( \theta = \frac{1}{K} \left( \sum_{r \in K} \hat{x}_r \right) \).
8: return \( x^* = x^\hat{-} - \theta \forall r \in K, x^*_r = 0 \forall r \notin K \).

of \( O(K) \), complexity is usually polynomial in terms of the dimension of \( \xi_i \). Assignment and Slate are special cases of such a setup, with each \( k \) corresponding to a certain path on a bipartite graph and a DAG, respectively.

**Assignment projection:** Let us return to the notations given in the definition of Multiple-slot Assignment in Section 2. Let \( \xi_i = (c_{is}, \{a_{ij}^{(j)}\}_{j=1}^{S_i})_{i,s} \). Each index \( k \) corresponds to one assignment, i.e., a set \( \{(i_s, s_t)\}_{s_t=1}^{S_k} \), with \( c_{ik} = \sum_{s_t=1}^{S_k} c_{i_t s_t} \), and \( a^{(j)}_{ik} = \sum_{s_t=1}^{S_k} a_{i_t j s_t}^{(j)} \). Thus, we have \( \hat{x}_{ik} = c_{ik} + \sum_{j=1}^{m} \lambda_j a^{(j)}_{ik} = \sum_{s_t=1}^{S_k} \phi_{s_t i s} \), where \( \phi_{s_t i s} = c_{i s} + \sum_{j=1}^{m} a_{i j s_t}^{(j)} \). Thus, to pick ordered elements of \( \{\hat{x}_{ik}k\} \), we just need to consider an assignment problem with cost defined by \( \phi_{s_t i s} \). Step 1 of Algorithm 2 corresponds to picking the best assignment. Step 3 corresponds to incrementally choosing the next best assignments. Efficient polynomial-time algorithms for implementing these steps are well known \( [6, 27, 34] \).

**Slate projection:** The specialization of Algorithm 2 for this case is similar to what we described above for Assignment. Because ranking of items needs to be obeyed and there are edge costs between consecutive items in a slate, dynamic programming can be used to give an efficient polynomial-time Algorithm 2 for Step 1. For incrementally finding the next best slate (Step 3), the ideas in \( [23] \) can be used to give efficient polynomial-time algorithms. It is clear from these specializations that Algorithm 2 is powerful and can be potentially applied more generally to problems with other complex structures.

**Box-Cut Projection:** For Box-Cut-Eq, it is easy to see that \( C = \text{co}\{v_r\}_{r=1}^{K} \) where \( v_r \) is a vector with \( p \) components equal to 1 and all the other components equal to 0. Without enumeration of the vertices, it is easy to apply Wolfe’s algorithm. First, for any given direction \( \eta \in R^K \) let us discuss the determination of \( \max_{\eta^T v_r} \). This is equivalent to finding the top \( p \) components of \( \eta \) and setting them to 1 with all remaining components set to 0. Using a heap this can be done efficiently in \( O(K) \) time. Let \( \psi(\eta) \) denote the vertex of the Box-Cut polytope thus found.

The initialization step of the Wolfe algorithm is the finding of the vertex \( \hat{x} \) nearest to \( \hat{x} \). Since \( \|v_r - \hat{x}\|^2 = p + \|\hat{x}\|^2 - 2\hat{x}^T v_r \) and the first two terms do not depend on \( v_r \), this step is equivalent to finding \( \psi(\hat{x}) \). This nearest vertex is set as the initializing corral. A general step of the Wolfe algorithm requires the checking of \( [12] \). Since \( U \) is a corral and \( \hat{x} \) is the projection of \( \hat{x} \) to the affine hull of \( U \), \( (x - \hat{x})^T u = 0 \).
\[(x - \hat{x})^T x \forall u \in U. \text{ Thus (12) can also be written as} \]
\[\min \{ \langle x - \hat{x} \rangle^T v : v \in V \} \geq \langle x - \hat{x} \rangle^T x. \tag{13}\]

So, all that we need to do is to set \(\eta = \langle x - \hat{x} \rangle\) and obtain \(\hat{v}(\eta)\) to check (13) as \(-\eta^T \hat{v}(\eta) \geq -\eta^T x\) to stop the Wolfe algorithm or, if that fails, use \(\hat{v}(\eta)\) as the next vertex for proceeding with the Wolfe algorithm. Since, by arguments of sections \(\ref{sec:PolytopeProjection}\) the number of steps of the Wolfe algorithm will be just 1 for most \(i\)'s, the cost of doing Box-Cut-Eq projection per \(i\) is just \(\mathcal{O}(K)\). This way of doing projection is much more efficient for use in our problem than more general algorithms with \(\mathcal{O}(K \log K)\) complexity given in the literature (12).

Box-Cut-Iq can be done efficiently similar to the Simplex-Iq projection by using the Box and the Box-Cut-Eq projection.

Parity Polytope Projection: ADMM based LP decoding of LDPC codes [43] requires projections to a special polytope called the Parity (aka Check) polytope, which is \(\mathcal{C} = \{ \hat{v} \}_{\mathbb{R}^n} \) where the \(\hat{v}\) are binary vectors with an even number of 1s. The projection algorithms in \(\ref{sec:PolytopeProjection}\) are specially designed for the case, but they are general purpose algorithms that use the inequalities-based representation of \(\mathcal{C}\) and are not designed to be efficient for the vertex oriented structure.

The specialization of the Wolfe algorithm, that provides such an efficiency, is similar to that for Box-Cut-Eq . The key computations are: (a) finding the vertex \(v\) nearest to \(\hat{x}\); and (b) checking (13), which requires, for any given any direction \(\eta\), \(\hat{v}(\eta) = \arg \max_{v \in V} \eta^T v\). Let us start with (b). \(\hat{v}(\eta)\) is a binary vector and we need to find the elements of \(\hat{v}(\eta)\) having the 1s. Since we are maximizing \(\eta^T v\), take the positive indices, \(P = \{ i : \eta_i > 0 \}\). If \(|P|\) is even, choose the set of elements of \(\hat{v}(\eta)\) with 1s as \(P\). If \(|P|\) is odd, let \(i_1\) be the index of the element of \(P\) with the smallest \(\eta_i\), and \(i_2\) be the index of the element of the complement of \(P\) with the largest \(\eta_i\). If \(\eta_{i_1} + \eta_{i_2} > 0\), include \(i_1\) and \(i_2\) in \(P\); else, remove \(i_1\) from \(P\). This is the optimal way of choosing \(|P|\) to be even. Then choose the set of elements of \(\hat{v}(\eta)\) with 1s as \(P\). Thus, the determination of \(\hat{v}(\eta)\) requires just a linear scan of \(|\eta_i|\).

Let us now come to the finding of the vertex \(v\) nearest to \(\hat{x}\), i.e., \(\arg \min_{v} \| \hat{x} - v \|^2\). Now, \(\| \hat{x} - v \|^2 = \| \hat{x} \|^2 + \| v \|^2 - 2 \hat{x}^T v\). Since \(\| v \|^2\) is a constant in the finding of \(v\), and, for binary vectors, \(\| v \|^2 = e^Tv\) where \(e\) is a vector of all 1s, we get \(\| \hat{x} - v \|^2 = \arg \max_{\eta} \eta^T v\), where \(\eta = \hat{x} - 0.5e\). With \(\eta\) set this way, we get \(\arg \min_v \| \hat{x} - v \|^2\) to be \(\hat{v}(\eta)\), whose determination was described in the previous paragraph. Thus, the finding of the vertex nearest to \(\hat{x}\) also requires just a linear scan of \(|\eta_i|\). It is worth trying the Wolfe algorithm as an alternative to existing algorithms to project points to parity polytopes in the ADMM based LP decoding of LDPC codes [43].

5 Experiments

We demonstrate the value of the algorithms designed in \ref{sec:PolytopeProjection} empirically using dataset \(D\). We solve the problem, using different number of cluster nodes \((w)\) and report the speedup in Table \(4\) by comparing with the large scale Ecl solver. Note that, since Ecl only supported the Simplex-Eq projection via off-the-shelf projection algorithms [14], we focus our results on the Simplex-Eq use case on their data [5] to showcase an apples to apples comparison of the overall LP solver. The speedup measures the improvement in time taken to aggregate \(I/w\) projection operations. Note that for large values \(I/w\), both methods have a fixed cost of going over a lot of data instances that reduces the relative speedup of the DuaLiP projection. From Table \(4\), it is clear that even for a simple projection type such as Simplex-Eq , employing off-the-shelf algorithms [14] may not be ideal. Basu et. al. [5] show that commercial solvers (SDPT3, CPLEX, CLP, GUROBI) do not scale to the problem-sizes in web-scale marketplaces and use a distributed version of the Splitting Conic Solver (SCS [13]) to establish a baseline for scale. We benchmark DuaLip against Ecl and SCS in Table \(5\) on real data to show a 7x improvement over the state-of-the-art. Ecl and DuaLip exploit the data sparsity in the constraint matrix (A) often present in marketplace data unlike SCS and other commercial solvers. Additionally, DuaLip benefits from efficient projection (\ref{sec:PolytopeProjection}) and (\ref{sec:PolytopeProjection}). We observe similar large efficiency gains on multiple applications of comparable size using our incremental projection algorithm. Tables \(4\) and \(5\) show that the speed-up depends less on the dataset and is more a function of the projection and \(I/w\).

| Problem           | Scale \(n\) | Time (hr) | DuaLip | Ecl | SCS |
|-------------------|-------------|-----------|--------|-----|-----|
| Email Optimization[5] | 10^6        | 0.11       | 0.8    | 2.0 |
| \(nnz(A) = 1011 \approx 1B\) | 10^6        | 0.16       | 1.3    | >24 |
|                  | 10^7        | 0.45       | 4.0    | >24 |
| Item Matching[4]  | 10^4        | 0.62       | 4.5    | >24 |
|                  | 10^11       | 1.1        | 7.2    | >24 |
|                  | 10^12       | 2.60       | 11.9   | >24 |

Table 5: Run time (hrs) for extreme-scale problems on real data. Here, \(nnz(A)\) denotes the number of non-zero entries in \(A\) and all runs are on Spark 2.3 clusters with up to 800 processors.

6 Discussion

Through empirical evidence and guided by theory, we have developed novel algorithms for efficient polytope projections under a vertex oriented structure. Such structures commonly occur in very large-scale industrial applications. Using these novel algorithms we were able to achieve a drastic overall speedup compared to the state-of-the-art while solving such web-scale Linear Programs. This scalability allows us to expand to a much larger class of problems with complex structured constraints such as State and Assignment . It also opens up the applicability in semi-supervised learning, structured prediction on text data, optimal marketplace
We provide results from one large-scale internal dataset (D) for clarity of impact and open-sourced the solver on GitHub[4] with benchmarks on a public-domain dataset for reproducibility.

References
[1] Agarwal, D.; Chen, B.-C.; Elango, P.; and Wang, X. 2011. Click Shaping to Optimize Multiple Objectives. In KDD, 132–140. New York, NY, USA: ACM. ISBN 978-1-4503-0813-7.
[2] Agarwal, D.; Chen, B.-C.; Elango, P.; and Wang, X. 2012. Personalized Click Shaping Through Lagrangian Duality for Online Recommendation. In SIGIR, 485–494. New York, NY, USA: ACM. ISBN 978-1-4503-1472-5.
[3] Azevedo, E. M.; and Weyl, E. G. 2016. Matching markets in the digital age. Science, 352(6289): 1056–1057.
[4] Barr, R.; and Gilbert, E. 1969. Some efficient algorithms for a class of abstract optimization problems arising on optimal control. IEEE Trans. Aut. Contr., AC-14: 640–652.
[5] Basu, K.; Ghoting, A.; Mazumder, R.; and Pan, Y. 2020. ECLIPSE: An Extreme-Scale Linear Program Solver for Web-Applications. In ICML 2020.
[6] Bourgeois, F.; and Lassalle, J.-C. 1971. An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices. 14(12): 802–804.
[7] Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning, 3(1): 1–122.
[8] Boyd, S.; and Vandenberghe, L. 2004. Convex Optimization. Cambridge university press.
[9] Chakrabarty, D.; Jain, P.; and Kothari, P. 2014. Provably Submodular Minimization using Wolfe’s Algorithm. arXiv:1411.0095.
[10] Chang, K.-W.; Sundararajan, S.; and Keerthi, S. S. 2013. Tractable semi-supervised learning of complex structured prediction models. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 176–191. Springer.
[11] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. Introduction to Algorithms. The MIT Press, 2 edition.
[12] Dai, Y.; and Fletcher, R. 2006. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. Math. Program., 106(3): 403–421.
[13] Danskin, J. M. 2012. The theory of max-min and its application to weapons allocation problems, volume 5. Springer Science & Business Media.
[14] Duchi, J.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient projections onto the 11-ball for learning in high dimensions. In Proceedings of the 25th international conference on Machine learning, 272–279.
[15] Edelman, B.; Ostrovsky, M.; and Schwarz, M. 2007. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. American economic review, 97(1): 242–259.
[16] Fleder, D.; and Hosanagar, K. 2009. Blockbuster culture’s next rise or fall: The impact of recommender systems on sales diversity. Management science, 55(5): 697–712.
[17] Friedlander, M. P.; and Tseng, P. 2008. Exact regularization of convex programs. SIAM Journal on Optimization, 18(4): 1326–1350.
[18] Gilbert, E. 1966. Minimizing the quadratic form on a convex set. SIAM J. Control. 4: 61–79.
[19] Gilbert, E.; Johnson, D.; and Keerthi, S. ????. A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space. IEEE Trans. Rob & Aut., 1.
[20] Grünbaum, B.; Kaibel, V.; Klee, V.; and Ziegler, G. M. 2003. Convex polytopes. New York: Springer. ISBN 9780387900424.
[21] Guillemin, V.; and Pollock, A. 1974. Differential Topology. Prentice-Hall.
[22] Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis), 5(4): 1–19.
[23] Haubold, C.; Uhrmann, V.; Unser, M.; and Hamprecht, F. A. 2018. Diverse M-Best Solutions by Dynamic Programming. CoRR, abs/1803.05748.
[24] He, B.; Leng, G.; and Li, K. 2006. Projection problems for symmetric polytopes. Advances in Mathematics, 207(1): 73–90.
[25] Keerthi, S. S.; Sellamanickam, S.; and Shevade, S. 2012. Extension of TSVM to Multi-Class and Hierarchical Text Classification Problems With General Losses. In Proceedings of COLING 2012: Posters, 1091–1100. Mumbai, India: The COLING 2012 Organizing Committee.
[26] Keerthi, S. S.; and Tomlin, J. A. 2007. Constructing a maximum utility slate of on-line advertisements. CoRR, abs/0706.1318.
[27] Kuhn, H. W. 1955. The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2): 83–97.
[28] Lin, Y.; Xia, Q.; He, W.; and Zhang, Q. 2019. A Fast Iterative Check Polytope Projection Algorithm for ADMM Decoding of LDPC Codes by Bisection Method. IEICE Transactions on Fundamentals, E102–A(10).
[29] Makari, F.; Awerbuch, B.; Gemulla, R.; Khandekar, R.; Mestre, J.; and Sozio, M. 2013. A distributed algorithm for large-scale generalized matching.

[4]https://github.com/linkedin/dualip
[30] Mangasarian, O. L.; and Meyer, R. 1979. Nonlinear perturbation of linear programs. SIAM Journal on Control and Optimization, 17(6): 745–752.

[31] Martins, A. F.; Figueiredo, M. A.; Aguiar, P. M.; Smith, N. A.; and Xing, E. P. 2015. Ad3: Alternating directions dual decomposition for map inference in graphical models. The Journal of Machine Learning Research, 16(1): 495–545.

[32] Meyer, G.; and Polak, E. 1970. A decomposition algorithm for solving a class of optimal control problems. J. Math. Anal. Appl., 32: 118–140.

[33] Mitchell, B.; Dem’yanov, V.; and Malozemov, V. 1974. Finding the point of a polyhedron closest to the origin. SIAM J. Control, 12: 19–26.

[34] Murty, K. G. 1968. An Algorithm for Ranking All the Assignment in Order of Increasing Cost. Operations Research, 16.

[35] O’Donoghue, B.; Chu, E.; Parikh, N.; and Boyd, S. 2019. SCS: Splitting Conic Solver, version 2.1.2. https://github.com/cvxgrp/scs

[36] Parikh, N.; and Boyd, S. 2014. Block splitting for distributed optimization. Mathematical Programming Computation, 6(1): 77–102.

[37] Ramanath, R.; Keerthi, S. S.; Pan, Y.; Salomatin, K.; and Basu, K. 2021. DualLip: Dual Decomposition based Linear Program Solver. https://github.com/linkedin/DualLip/blob/master/DualLip_AAAI_fullPaper.pdf

[38] Rambau, J.; and Ziegler, G. M. 1996. Projections of polytopes and the generalized Baues conjecture. Discrete & Computational Geometry, 16(3): 215–237.

[39] Rush, A. M.; and Collins, M. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. Journal of Artificial Intelligence Research, 45: 305–362.

[40] Smith, N. A. 2011. Linguistic Structure Prediction. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

[41] van den Berg, E.; and Friedlander, M. 2008. Probing the Pareto frontier for basis pursuit solutions. SIAM J. Sci. Comput., 31: 890—912.

[42] Wang, J.; Zhou, J.; Wonka, P.; and Ye, J. 2013. Lasso Screening Rules via Dual Polytope Projection. In NIPS, 1070–1078. Citeseer.

[43] Wei, H.; and Banihashemi, A. H. 2017. An iterative check polytope projection algorithm for ADMM-based LP decoding of LDPC codes. IEEE Communications Letters, 22(1): 29–32.

[44] Wolfe, P. 1976. Finding the Nearest Point in a Polytope. Math. Program., 11(1): 128–149.
A Proofs

This section contains various details related to §3 and §4 of the paper.

A result is said to hold in general position if the result holds almost surely in the space of perturbations and other parameters; in this section, these are \( \delta c, \delta b, \delta d, \delta \epsilon, \delta \rho, \lambda \) and \( z \). We will simply use the phrase ‘general position’ and it will be clear from the context as to which perturbations and parameters it means.

Proof of part 1 of Theorem 1

The proof is based on the following lemma, which we establish later.

**Lemma B.1** Consider the LP

\[
\min_x (c + \delta c)^T x \text{ s.t. } Px \leq p + \delta p, \quad E x = e + \delta e
\]

where the feasible set is a compact subset of \( \mathbb{R}^n \) and \( P \) is any open set of allowed perturbation values \( (\delta c, \delta b, \delta d, \delta \epsilon, \delta \rho, \lambda) \). Then, in general position the solution \( x \) of (14) is a unique point at which exactly \( n \) constraints of \( Px \leq p + \delta p, E x = e + \delta e \) are active. \( \square \)

We begin with a clarification concerning the mean optimal corral dimension associated with \( x^* = \arg \min_x c^T x \) s.t. \( Ax \leq b, x_i \in C_i, i \in [I] \). Though \( x^* \) does not originate from a projection operation, it is actually a projection. This comes from Lemma 2 of [5], which implies that there exists \( \gamma > 0 \) such that for all \( 0 < \gamma \leq \gamma^* \), \( x^* \) is the projection associated with the determination of \( g_\gamma(\lambda) \) where \( \gamma = \arg \min_{\lambda} g_\gamma(\lambda) \). Thus, it is sensible to talk of the mean optimal corral dimension, \( \mu(x^*) \).

To prove part 1 of Theorem 1, let us apply Lemma B.1 to the case where we combine \( Ax \leq b + \delta b \) and \( Dx \leq d + \delta d \) to form \( Px \leq p + \delta p \). The equality constraint, \( Ex = e + \delta e \) stays as it is. The lemma implies that in general position the solution \( x^* \) of the LP is unique and the number of active constraints is exactly equal to \( n = IK \). Among the active constraints, some of them, call this number \( m_{\text{act}} \), come from \( Ax \leq b \). The rest of the active constraints come from \( Dx \leq d \), and \( Ex = e + \delta e \) i.e., the \( C_i \), which, in terms of optimal corral dimensions, is equal to \( \sum_i (K - \mu_i) \) for Simplex-Eq (Eq). Assignment and Slots, and upper bounded by \( \sum_i (K - \mu_i) \) for Box-Cut-Eq (Eq). (It is easy to see that these structures are unchanged if the perturbation \( \delta d, \delta e \) is small.) Thus,

\[
n \leq m_{\text{act}} + \sum_i (K - \mu_i) = m_{\text{act}} + n - n\mu(x)
\]

Since \( m_{\text{act}} \leq m \), we get \( \mu(x^*) \leq m/n \).

Proof of part 2 of Theorem 1

Similar to Lemma B.1 we can prove the following lemma.

**Lemma B.2** Consider the LP

\[
\min_x (e + A^T \lambda + \delta c)^T x \text{ s.t. } Dx \leq d + \delta d, \quad E x = e + \delta e
\]

where the feasible set is a compact subset of \( \mathbb{R}^n \) and \( D \) is any open set of allowed perturbation/parameter values \( (\delta c, \delta d, \delta \epsilon, \lambda) \). Then, in general position the solution \( x \) of (16) is a unique point at which exactly \( n \) constraints of \( Dx \leq d + \delta d, E x = e + \delta e \) are active. \( \square \)

To prove part 2 of Theorem 1 we just apply Lemma B.2. Since \( Ax \leq b \) constraint is absent, analysis similar to the proof of part 1 yields that in general position we have \( \mu(x^*_0(\lambda)) = 0 \).

Proofs of Lemmas B.1 and B.2

The two lemmas are geometrically intuitive; however, proving them rigorously is somewhat technical and detailed.

Let us start by proving Lemma B.1. The Lagrangian for the LP in (14) with dual vectors \( \psi \) and \( \phi \) is

\[
L = (c + \delta c)^T x + \psi^T (P x - p - \delta p) + \phi^T (E x - e - \delta e)
\]

A triple \((x, \psi, \phi)\) is optimal iff the following KKT conditions hold:

\[
\begin{align*}
p^T \psi &= 0, \quad P x = p + \delta p, \quad (18) 
\psi = 0, \quad \psi^T (P x - p - \delta p) = 0, \quad E x = e + \delta e \quad (19)
\end{align*}
\]

Let

\[
S_1 = \{ r : (P x - p - \delta p)_r = 0 \}, \quad S_2 = \{ r \in S_1 : \psi_r = 0 \}
\]

\( S_1 \) is the set of active (primal) inequality constraints and \( S_2 \subset S_1 \) is the subset with positive \( \psi \) values. Note that not all the \( \psi_r \), \( r \in S_1 \) are required to be positive. Note also that \( \psi_r = 0 \) \( \forall r \notin S_1 \). Let us rewrite the KKT conditions in (18) and (19) as

\[
\begin{align*}
P^T \psi + E^T \phi + c + \delta c &= 0, \quad (P x - p - \delta p)_{S_1} = 0, \quad (21) 
(P x - p - \delta p)_{S_1^c} < 0, \quad (\psi)_{S_1^c \setminus S_2} > 0, \quad (\psi)_{S_2 \cup S_2^c} = 0 \quad (22)
\end{align*}
\]

In the above, we are analyzing what happens at a solution \((x, \psi, \phi)\). Let us turn this around a bit and look at the (finite) collection of all \((S_1, S_2)\) possibilities satisfying \( S_2 \subset S_1 \), and, for a given \((S_1, S_2)\) look at the set of all optimal solutions \((x, \psi, \phi)\) satisfying (21)-(22), call it \( O(S_1, S_2) \). Note that \( O(S_1, S_2) \) will turn out to be empty for most \((S_1, S_2)\) pairs, which is fine. Using this definition, the set of all possible solutions of the LP in (14) can be written as

\[
O = \bigcup_{(S_1, S_2)} O(S_1, S_2)
\]

Let us take one \((S_1, S_2)\). We will use Transversality theorem (a corollary of Sard’s theorem) to show that \( O(S_1, S_2) \) is at most a singleton in general position. We state Transversality theorem [21] in the simplest form that suffices our purpose.

**Transversality theorem** Suppose \( F : Y \times Z \rightarrow \mathbb{R}^n \) is a smooth differentiable function such that (a) \( Y \) and \( Z \) are open sets in \( \mathbb{R}^k \) and \( \mathbb{R}^M \) and (b) the Jacobian of \( F \) with respect to \((y, z)\) has full row rank for all \((y, z) \in Y \times Z \). Then in general position the set \( \tilde{Y}(z) = \{ y \in Y : F(y, z) = 0 \} \)
0} is either empty or a differentiable manifold of dimension \((L - N)\).

To set up Transversality theorem for proving Lemma B.1, let 
\(y = (x, \psi, \phi)\), 
\(z = (\delta c, \delta p, \delta e)\), \(Z = P\), 
\(Y = \{(x, \psi, \phi) : (Px - p - \delta p)S_1 < 0, (\psi)S_1\setminus S_2 > 0\}\) (25)
and the function, 
\[F = \begin{pmatrix}
P^T \psi + E^T \phi + c + \delta e \\
(Px - p - \delta p)S_1 \\
Ex - e - \delta e \\
(\psi)S_1\setminus S_2
\end{pmatrix}
\] (26)
so that \(O(S_1, S_2)\) can be written as
\[O(S_1, S_2) = \tilde{Y}(z) = \{y \in Y : F(y, z) = 0\}\] (27)
Note that \(Y\) and \(Z\) are open sets. If we look at the Jacobian of \(F\) with respect to the subset of variables, 
\((\delta c, (\delta p)S_1, (\delta e), (\psi)S_1\setminus S_2)\) it is a non-singular diagonal matrix with 1’s and \(-1\)'s in the diagonal. Hence the Jacobian of \(F\) with respect to \((y, z)\) has full row rank. Thus, by Transversality theorem, in general position the set \(O(S_1, S_2)\) is either empty or a differentiable manifold of dimension \((L - N)\).

\(\hat{\psi}\) is replaced by \(\lambda\).

\(\text{Case 1. } |S_2| > 0\). For this case, \((L - N) = 0\) and, in general position the set \(O(S_1, S_2)\) is a zero dimensional manifold, i.e., a set of isolated points. By convexity of \(O(S_1, S_2)\), it can be at most a singleton. Also, if \(O(S_1, S_2) = \emptyset\), then since \((Px - p - \delta p)S_1 = 0\) and \(Ex - e - \delta e = 0\) independently determine \(x\), and, \(P^T \psi + c + \delta e = 0\), \((\psi)S_1\) is a non-singular diagonal matrix with 1’s and \(-1\)'s in the diagonal. Hence the Jacobian of \(F\) with respect to \((y, z)\) has full row rank. Thus, by Transversality theorem, in general position the set \(O(S_1, S_2)\) is either empty or a differentiable manifold of dimension \((L - N)\).

\(\text{Case 2. } |S_2| = 0\). For this case, \((L - N) = 0\) and, in general position the set \(O(S_1, S_2)\) is a zero dimensional manifold, i.e., a set of isolated points. By convexity of \(O(S_1, S_2)\), it can be at most a singleton. Also, if \(O(S_1, S_2) = \emptyset\), then since \((Px - p - \delta p)S_1 = 0\) and \(Ex - e - \delta e = 0\) independently determine \(x\), and, \(P^T \psi + c + \delta e = 0\), \((\psi)S_1\) is a non-singular diagonal matrix with 1’s and \(-1\)'s in the diagonal. Hence the Jacobian of \(F\) with respect to \((y, z)\) has full row rank. Thus, by Transversality theorem, in general position the set \(O(S_1, S_2)\) is either empty or a differentiable manifold of dimension \((L - N)\).

Let us now come to \(O\) in (24). In general position, convexity of \(O\) implies that the only possibility is that there is a unique \(S_1\) such that \(O = O(S_1, 0) = \{(x^*, \psi^*, \phi^*)\}\), the unique optimal solution of (14).

The lines of proof for Lemma B.2 are similar. \(Px \leq p + \delta p\) is replaced by \(Dx \leq e + \delta e\) and there is an additional parameter \(\lambda\) that is included in \(z\).

### Wolfe’s algorithm

In (13) we described the essence of Wolfe’s algorithm. Here we give fuller details in Algorithm 3.

Steps 5-22 form one iteration of the algorithm going from one corral to a better corral. Steps 5 and 6 correspond to checking optimality of \(x\), the projection to the current best corral; these steps are same as each of (13) and (12). Step 5 forms the most expensive part of the algorithm. Note that, in problems with structure in the vertices of \(C\), step 5 can be done very efficiently without enumerating all vertices. As we explained in (14) this is exactly what happens in Assignment, Slate and Box-Cut-Eq (1q).

If checking of optimality of \(x\) fails, the algorithm enters the minor cycle in which a next, better corral is identified efficiently. Steps 11-22 form the minor cycle. Step 12 requires the projection of \(\hat{x}\) to \(\text{aff}(S)\), the affine hull of the vertices in \(S\), which can be obtained by solving a system of linear equations in \(|S| + 1\) unknowns. Since the corral dimensions \(|S| - 1\) encountered in marketplace LPs are small, the affine hull computation is cheap.

In many cases of \(C\), the minor cycle breaks at the first cycle/entrance to step 14. The modified Duchi et al algorithm for Simplex-Eq (algorithm 2) is an example. This is also true in most iterations of other cases of \(C\).

---

**Algorithm 3: Wolfe’s algorithm**

**Input:** \(x, B = \text{Vertex set of } C\)

**Output:** \(x^* = \arg \min_{x \in C} \|x - \hat{x}\|^2\)

1: In all iterations we maintain \(S = \{v_r\}\), a subset of vertices, and \(x = \sum_{r \in S} \rho_r v_r\) as a convex combination of the vertices in \(S\). At the end \(S\) forms the optimal corral.

2: Find \(v\) the vertex closest to \(\hat{x}\).

3: Initialize \(x \leftarrow v, S \leftarrow \{v\}\) and \(\rho_1 \leftarrow 1\).

4: while true do \(\triangleright \text{Major Cycle}\)

5: \(v := \arg \min_{p \in B} (x - \hat{x})^T p\)

6: if \((x - \hat{x})^T v \leq (x - \hat{x})^T v\) then

7: \(\text{Set } x^* = x\).

8: \(\text{break}\)

9: \(\text{end if}\)

10: \(S := S \cup \{v\}, \rho_{|S|} \leftarrow 0\)

11: while true do \(\triangleright \text{Minor Cycle}\)

12: \(\text{Find } y = \arg \min_{z \in \text{aff}(S)} \|z - \hat{x}\|^2\) and

13: \(\alpha \text{ s.t. } y = \sum_{r \in S} \alpha_r v_r\)

14: if \(\alpha_r \geq 0 \forall v\) then

15: \(\text{break } \triangleright y \in \text{co}(S)\), so end minor cycle

else \(\triangleright \text{If } y \notin \text{conv}(S)\), then update \(x\) to the intersection of the boundary of \(\text{co}(S)\) and the segment joining \(y\) and previous \(x\). Delete points from \(S\) which are not required to describe the new \(x\) as a convex combination.

17: \(\theta := \min_{\rho \in \text{co}(S)} \frac{\rho^T v}{\rho^T x}\) \(\triangleright\) Since \(x = \sum_r \rho_r v_r\)

18: \(x \leftarrow \theta y + (1 - \theta)x\) \(\triangleright\) Using \(\theta\), the new \(x\) lies in \(\text{conv}(S)\).

19: \(\rho_r \leftarrow \theta \alpha_r + (1 - \theta) \rho_r \forall r\) \(\triangleright\) Sets the coefficients of the new \(x\)

20: \(S \leftarrow \{r : \rho_r > 0\}\) \(\triangleright\) Delete points which have \(\rho_r = 0\). This deletes at least one point.

21: \(\text{end if}\)

22: \(\text{end while}\) \(\triangleright\) After the minor loop terminates, \(x\) is updated to be the affine minimizer of the current set \(S\) \(x \leftarrow y\)

23: \(\text{end while}\)

24: return \(x^*\)
This figure "muBehavior.JPG" is available in "JPG" format from:

http://arxiv.org/ps/2103.05277v3
This figure "proj2dexample.JPG" is available in "JPG" format from:

http://arxiv.org/ps/2103.05277v3