Learning to Pivot as a Smart Expert

Tianhao Liu[1], **Shanwen Pu**[1], **Dongdong Ge**[1], **Yinyu Ye**[2]

[1]Research Institute for Interdisciplinary Sciences, Shanghai University of Finance and Economics
[2]Stanford University
liu.tianhao@163.sufe.edu.cn, 2019212802@live.sufe.edu.cn, ge.dongdong@mail.shufe.edu.cn, yyye@stanford.edu

Abstract

Linear programming has been practically solved mainly by simplex and interior point methods. Compared with the weakly polynomial complexity obtained by the interior point methods, the existence of strongly polynomial bounds for the length of the pivot path generated by the simplex methods remains a mystery. In this paper, we propose two novel pivot experts that leverage both global and local information of the linear programming instances for the primal simplex method and show their excellent performance numerically. The experts can be regarded as a benchmark to evaluate the performance of classical pivot rules, although they are hard to directly implement. To tackle this challenge, we employ a graph convolutional neural network model, trained via imitation learning, to mimic the behavior of the pivot expert. Our pivot rule, learned empirically, displays a significant advantage over conventional methods in various linear programming problems, as demonstrated through a series of rigorous experiments.

1 Introduction

Linear programming (LP) is among the most fundamental problems and has been well-studied in the field of optimization. LP is not only directly used across various industries but has also become an important cornerstone of mixed integer programming (MIP) and sequential linear programming (SLP) methods for solving nonlinear programming (NLP). Nowadays, most commercial (Ge et al. 2023; Gurobi Optimization, LLC 2023; Nickel et al. 2022; Xpress 2014) and open-source (Huangfu and Hall 2018; Achterberg 2009) solvers have implemented fast and stable LP solvers (software for solving LP) and constantly achieve new advancements for large-scale LP problems.

A general LP formulation solves the problem with only a linear objective function and linear constraints. It is well-known that these linear constraints geometrically form a polyhedron and if the optimal solution exists, it exists in one of the vertices which belong to basic solutions from an algebra point of view. The state-of-the-art methods for LP include the simplex methods, the interior-point methods (IPMs), and some recently developed first-order methods (FOMs) (Applegate et al. 2021; Deng et al. 2022). For high accuracy and reliability, the simplex methods and IPMs are preferred and become two main classes of algorithms implemented in commercial solvers.

The simplex methods start from a basic solution and improve objective or feasibility by reaching a certain adjacent basic solution, which is called the pivot. The criterion for switching from the current basic solution to its neighbor is called the pivot rule. Different pivot rules greatly affect the performance of the simplex methods, so designing a smart pivot rule is one of the most significant simplex's tasks. From the theoretical aspect, whether there exists a strongly polynomial bound for the length of the pivot path, which represents the iteration number of the simplex methods, attracts much research interest but is still an open problem.

Instead of moving between vertices, IPMs keep an interior point and walk along a central path approaching the optimal solution (Karmarkar 1984). In practice, IPMs usually yield a dense primal-dual approximate solution, from which modern commercial solvers tend to conduct a crossover and run simplex methods for a sparse exact solution.

In this paper, we focus on designing smart pivot rules for the primal simplex method. It is believed that our study can be easily migrated to other types of simplex methods. The smart pivot rules are expected to generate short pivot paths for different kinds of LP instances in different scales and should not run intolerably slow.

Our contribution We propose a class of novel pivot experts that can outperform several classical and popular pivot rules. To modify the experts for practical use, we also apply machine learning methods to imitate the pivot expert. Our contribution can be summarized as follows.

- First, we design novel pivot experts. Compared with classical pivot rules that only utilize local information, we consider a smart pivot rule should be able to combine global and local information together. Based on this idea, two pivot experts are proposed and can generate significantly shorter pivot paths than classical pivot candidates in a series of experiments. The pivot paths generated by experts are also analyzed on Klee-Minty variants.
- Second, to the best of our knowledge, this paper is the first to combine imitation learning with dynamic pivot for general LP of different scales. Incorporating a graph convolutional neural network (GCNN) model, our learned

rule pivots by predicting the experts' pivot behavior, which removes the requirement for global information but maintains expert performance.

Organization of the paper The paper is organized as follows. Section 2 reviews related studies in simplex methods and machine learning methods to help optimization, especially for LP. Section 3 describes the novel class of pivot experts and discusses its merits and demerits. Section 4 provides an imitation learning method to help our idea of experts become practical. Section 5 presents twofold experiments to verify the superiority of our pivot experts and the learned pivot rule. Section 6 concludes the paper and discusses some relative topics.

Notations Several commonly used notations are listed below. We use bold letters for vectors and matrices. Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space. We use $\bar{\mathbb{R}}$ and $\underline{\mathbb{R}}$ to express $\mathbb{R} \cup \{+\infty\}$ and $\mathbb{R} \cup \{-\infty\}$. Let $\mathbf{x}_j$ be the jth element of vector $\mathbf{x}$. We use $\mathbf{x} \geq \mathbf{y}$ to express the element-wise inequality $\mathbf{x}_i \geq \mathbf{y}_i$. Let $\mathbf{0}$, $\mathbf{1}$, and $\boldsymbol{\infty}$ be a vector of zeros, ones, and infinities. Let $\mathbf{I}$ be the identity matrix and $\mathbf{e}_j$ be the jth column of $\mathbf{I}$. The dimension of a vector or a matrix will be unspecified whenever it is clear from the context. $\|\cdot\|_\ell$ is ℓ-norm (2-norm if ℓ is omitted) while $|\cdot|$ is absolute value. Let $\mathbf{A}_{i,j}$ be the entry in the ith row and jth column of matrix $\mathbf{A}$. Let $\mathbf{A}_j$ be the jth column of matrix $\mathbf{A}$, and $\mathbf{A}_{\mathcal{I}}$ be the matrix formulated by columns $\mathbf{A}_j$ for $j \in \mathcal{I}$. Let $\mathbf{A}_{i,:}$ be the ith row of matrix $\mathbf{A}$.

2 Related Work

2.1 Simplex Methods

In this subsection, we will describe a series of pivot rules for LP in standard form

$$\begin{aligned} \min\ & \mathbf{c}^\top \mathbf{x} \\ \text{s.t. } & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n$. Let B and N be indices of basic and non-basic variables. This formulation is used for academic research, but will not be preferred in modern LP solvers. Our implementation of pivot experts considers a more practical formulation which will be described in Section 3 later.

Pivot rules in primal simplex method The simplex methods switch between adjacent basic solutions, which is called the pivot, at each iteration. Pivot is algebraically selecting a non-basic variable to enter the basis, conducting a ratio test evaluating distance to go, and letting one basic variable leave the basis. How to choose among adjacent basic solutions is key to a successful simplex method.

When Dantzig proposes primal simplex method, he also provides a pivot rule to choose the candidate with the most negative reduced cost $\bar{\mathbf{c}}_j = \mathbf{c}_j - \mathbf{c}_B^\top \mathbf{A}_B^{-1} \mathbf{A}_j$, which is called Dantzig's rule (Dantzig 1963). To avoid cycling in a pivot path, Bland's rule (Bland 1977) of choosing the candidate with minimum index and other lexicographic pivot rules are proposed. These anti-cycling rules make simplex terminate in finite steps but perform poorly in real applications. Since other practically anti-cycling methods like perturbation work well, more practical interest is attracted by generating shorter pivot paths rather than theoretically finite termination.

The most widely used pivot rule in modern simplex solvers is the steepest-edge rule (Goldfarb and Reid 1977; Forrest and Goldfarb 1992). It has been observed to generate relatively short pivot paths. The steepest-edge rule chooses the candidate with the most negative $\frac{\bar{\mathbf{c}}_j}{\sqrt{\|\mathbf{A}_B^{-1}\mathbf{A}_j\|^2+1}}$, which can be explained as moving in the descending direction most parallel to $\mathbf{c}$.

Another pivot rule called the greatest improvement rule (Jeroslow 1973) can also generate short pivot paths. Like strong branching (Achterberg, Koch, and Martin 2005) in MIP, this pivot rule prefers a candidate that brings the greatest improvement in objective value to enter the basis. However, sometimes too much greed is not the best option (as will be seen in Section 5). Besides, to calculate the improvement, a ratio test for each candidate variable is needed, which is often too expensive for the simplex method.

As the increasing scale of LP has hit the limits of computer power, rules for more efficient computation are proposed including Devex rule (Harris 1973) and the largest distance rule (Roos 1986; Pan 2008). Devex rule inexactly approximates the score in steepest-edge and thus can update faster. The largest distance rule selects the candidate for which the corresponding dual hyperplane is the farthest from the present vertex, i.e., with the most negative score $\frac{\bar{\mathbf{c}}}{\|\mathbf{A}_j\|}$. Notice that the denominator stays the same and only requires to be calculated once.

Worst cases of pivot rules With so many rules accumulated and a wide variety of manifestations observed in practice, researchers are puzzled by the complexity of the simplex methods. In other words, what is the worst possible path for the simplex methods? Or generally, can LP be solved with strongly polynomial algorithms?

These problems are surprisingly difficult to give a general satisfying answer even though we have already known that LP has weakly polynomial bounds guaranteed by IPMs. For some special LP classes, certain pivot rules are proved to be strongly polynomial (Ye 2011; Kitahara and Mizuno 2013). Unfortunately, worst cases with exponential pivot numbers have been discovered for most deterministic pivot rules (Klee and Minty 1972; Avis and Chvátal 1978; Goldfarb and Sit 1979; Roos 1990). After introducing randomization and parameterized LP, several sub-exponential bounds (Matoušek, Sharir, and Welzl 1992; Kalai 1992) or weakly polynomial bounds (Kelner and Spielman 2006) have been derived for general LP. In short, analyzing the complexity of simplex is still a long way to go.

2.2 Machine Learning for Mathematical Optimization

Machine learning methods can help accelerate or improve optimization methods. This field of research is called ma-

chine learning for mathematical optimization (ML4MO). Many studies in ML4MO help solve MIP since it is a harder and more common problem. Although there have not been many fusions of machine learning and LP, some preliminary attempts are made and deserve mention.

GCNN model-based ML4MO Early studies extract feature vectors from problems and use classical models for prediction (Di Liberto et al. 2016; Alvarez, Louveaux, and Wehenkel 2017; He, Daume III, and Eisner 2014; Khalil et al. 2016). Afterwards, a new way to encode problems proposed by Gasse et al. (2019) is to imitate strong branching. They creatively encode MIP to be a bipartite graph. The bipartite graph contains almost all information of the original problem and thus avoids loss of information which the classical methods often suffer from. With a graph as input, GCNN models are naturally adopted to perform more comprehensive feature extraction and are ready for subsequent models to make final decisions. Gasse's work inspires a new stream of ML4MO (Gupta et al. 2020; Ding et al. 2020; Nair et al. 2020; Sonnerat et al. 2021; Paulus and Krause 2023) and the GCNN approach, together with the bipartite graph encoding, has become one of the preferred methods in practice.

Machine learning for LP and simplex Several attempts have been made to accelerate LP utilizing machine learning methods. Most of them study pivot in the primal simplex method. Adham et al. (2021) use boosted trees and neural networks to predict the best pivot rule for each LP instance but the approach is a one-shot decision and lacks flexibility. Suriyanarayana et al. (2022) use reinforcement learning to dynamically switch between Dantzig's rule and steepest-edge rule for solving LP relaxation of non-Euclidean TSPs with five cities. However, it is only proof of concept that is not suitable for larger problems or problems with different scales. Li et al. (2022) use Monte Carlo tree search (MCTS) to directly decide which candidate will enter the basis. For each new LP instance, MCTS explores slowly at every single pivot.

The state in both Suriyanarayana's and Li's reinforcement learning approaches is based on simplex tableau directly, which is not scalable for large-scale LP. The bipartite graph and GCNN can be a more reasonable tool to encode LP for its permutation invariance and scalability. In theory, Chen et al. (2022) reveal the potential power of GCNN in distinguishing LP with different characteristics. In practice, Fan et al. (2023) use GCNN to predict a better initial basis, which is the preparatory work for the primal simplex method.

3 Smart Pivot Experts

3.1 Primal Simplex Method

We consider a general LP formulation

$$\begin{aligned} &\min \mathbf{c}^\top \mathbf{x} \\ &\text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \\ &\quad \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \end{aligned} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n, \mathbf{l} \in \underline{\mathbb{R}}^n, \mathbf{u} \in \bar{\mathbb{R}}^n$. (2) is more user-friendly than (1), so we will derive our pivot experts and conduct experiments based on this formulation. We implement the following two-phase revised simplex in our own primal simplex solver prototype.

Phase I: find a basic feasible solution Phase I aims to find a basic feasible solution for Phase II in the primal simplex method. In fact, multiple ways including the big-M method and heuristics are designed to achieve the purpose. However, those methods form an independent topic that is beyond our discussion.

Phase II: solve the LP Phase II starts to solve (2) after a basic feasible solution is obtained from Phase I. At each pivot, we check the reduced cost $\bar{\mathbf{c}}$ of those non-basic variables and pick those candidates which have negative (positive) $\bar{\mathbf{c}}$ and are equal to its lower (upper) bound. If there occurs a tie, we will choose the one with minimum (maximum) index to enter (leave) the basis.

3.2 Designing Smart Pivot Experts

The existing pivot rules all consider only local information. Here the term "local" refers to the information that describes the landscape around the current basic feasible solution. If a pivot rule makes a myopic decision, the basic feasible solution may lead to being stuck in a rugged area in the future.

The main idea for designing a smart pivot expert is to provide global information for it. Some trials proposed by others include tree search for future information (Li et al. 2022), using interior point information (Todd 1990; Roos 1986; Tamura et al. 1988), or choosing more than one variable at one time to enter the basis (Yang 2020). However, there is something the most global but easily overlooked —- the optimal basis.

With the optimal basis at each iteration, the smart pivot rule can be guided by the following two goals:

1. When selecting a candidate to enter the basis, a smart pivot rule should let the basic variable in the optimal basis enter first.
2. To select a variable to leave the basis when a tie occurs in the ratio test, a smart pivot rule should let the non-basic in the optimal basis leave first.

The smart pivot rule will greedily bring the current basis as close to the optimal basis (in terms of the difference in the basic indices) as possible from a global perspective. Theorem 1 guarantees that such a variable can always be found as long as the objective value is not optimal.

Theorem 1. *Given the optimal basis, if the current objective value is not optimal, there must exist a variable mismatching the optimal basis that can enter the current basis immediately.*

Yang (2020) provides similar observation and proof based on the formulation (1), but does not continue to make full use of it. We modify his remark for a more practical LP formulation (2) and design smart pivot experts based on it.

Designing smart pivot experts We design two pivot experts based on two goals and Theorem 1. Before presenting details, we have to point out that local information still matters. During the development of simplex, much valuable

local information has been proposed such as reduced cost and steepest-edge score. Given the optimal basis, our pivot experts combine global information and local information together.

The first pivot expert (Expert I) satisfies the first goal and then tries to satisfy the second. More precisely, it chooses the candidate among the optimal basis with the best steepest-edge score. After the ratio test, it removes the non-basic variable in the optimal basis as far as possible.

The second pivot expert (Expert II) considers the two goals at the same time. It will conduct a ratio test for each candidate in the optimal basis and give preference to those that can remove non-basic variables in the optimal basis if there are any. After candidates are filtered by the two goals, it will choose the one with the best steepest-edge score.

The two experts run at different speeds. Expert I can be calculated efficiently while Expert II runs more slowly due to multiple ratio tests. Besides paths with monotone objective value, the two pivot experts share Property 1 of generating paths with monotone # DiffOpt defined in Definition 1. Experiments in Section 5 will show the superiority of our pivot experts for overall generating shorter paths compared with other classical pivot rules.

Definition 1 (# DiffOpt)**.** *Let the status vector* ***sta*** *of the vertex* $\mathbf{x}$ *be* $\textbf{\textit{sta}}_i = \begin{cases} 0, & \text{if non-basic } \mathbf{x}_i = \mathbf{l}_i \\ 1, & \text{if } \mathbf{x}_i \text{ is basic} \\ 2, & \text{if non-basic } \mathbf{x}_i = \mathbf{u}_i \end{cases}$*. Given the optimal basis, # DiffOpt is defined as 1-norm of the difference between the status vectors of* $\mathbf{x}$ *and the optimal basis.*

Property 1. *Given the optimal basis, before the current objective is optimal, # DiffOpt is monotonically decreasing.*

At the end of this subsection, we emphasize that the term "expert" refers to overall better performance rather than total transcendence. Recalling the existence of worst cases, it is almost impossible for a pivot rule to completely beat another rule in every single LP instance. Maros (2012) suggests combining different pivot rules if there are signs of benefit, which is a parallel technical route to ours.

3.3 Pivot Experts on Klee-Minty Cube Variants

To further illustrate the value of global information, a linear upper bound is provided for the length of our pivot experts' pivot path on Klee-Minty (KM) cube variants which are usually the worst cases for classical pivot rules.

KM cube variants KM cube variants are a well-known class of squashed cubes that usually lead to poor performance of some pivot rules, encompassing KM variants (Kitahara and Mizuno 2011; Vanderbei 2020) for Dantzig's rule and the Avis-Chvátal polytope (Avis and Chvátal 1978) for Bland's rule.

These KM cubes share some similar properties. First, the feasibility set is combinatorially equivalent to the standard n-dimensional cube $\mathcal{C}_n = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n : \mathbf{x} + \mathbf{y} = \mathbf{1},\ \mathbf{x}, \mathbf{y} \geq \mathbf{0}\}$, which means there exists a one-to-one correspondence between their faces. Second, each vertex is non-degenerate. The standard cube $\mathcal{C}_n$ here is obtained via adding slacks $\mathbf{y}$ in the cube $[0, 1]^n$. $\mathcal{C}_n$ has 2^n vertices whose first n elements are $\mathbf{x} \in \{0, 1\}^n$ and $\mathbf{y} = \mathbf{1} - \mathbf{x}$.

The experts' linear upper bound on KM cubes The main idea for deriving a linear upper bound for the pivot experts on KM cubes is to analyze the length of paths with monotone # DiffOpt. The proof is divided into three steps. In essence, we start by bounding path lengths on the basic cube $\mathcal{C}_n$ in Theorem 2, extend that to polytopes combinatorially equivalent to $\mathcal{C}_n$ in Theorem 3, and then apply it to the pivot experts in the KM setting under certain mild assumptions in Theorem 4. This allows us to derive an overall linear upper bound on KM cubes.

Theorem 2. *For* $\mathcal{C}_n$ *with initial point* $(\mathbf{x}^0, \mathbf{y}^0)$ *and any optimal basis* $\mathbf{B}^*$*, the length of the path with monotone # DiffOpt is* $\frac{\#\ \textit{DiffOpt of}\ (\mathbf{x}^0, \mathbf{y}^0)}{2}$*, which is bounded by* n *from above.*

Theorem 3. *For any polytope combinatorially equivalent to* $\mathcal{C}_n$ *with non-degenerate vertices, initial point* $(\mathbf{x}^0, \mathbf{y}^0)$*, and any optimal basis* $\mathbf{B}^*$*, the length of path with monotone # DiffOpt is* $\frac{\#\ \textit{DiffOpt of}\ (\mathbf{x}^0, \mathbf{y}^0)}{2}$*, which is bounded by* n *from above.*

Theorem 4. *For any polytope combinatorially equivalent to* $\mathcal{C}_n$ *with non-degenerate vertices, initial point* $(\mathbf{x}^0, \mathbf{y}^0)$*, and the single optimal basis* $\mathbf{B}^*$*, the length of a path generated by our pivot experts is upper bounded by* $\frac{\#\ \textit{DiffOpt of}\ (\mathbf{x}^0, \mathbf{y}^0)}{2}$*, which is bounded by* n *from above.*

Theorem 4 illustrates the value of global information. With the guidance of the given optimal basis, our experts will avoid being led to the worst by misleading local information on various KM variants. Notice that the upper bound holds with arbitrary or even no local information. In this aspect, the monotone # DiffOpt is more like a combinatorial rather than algebraic property. The strong performance of our experts on KM variants will not be negatively impacted by scaling, which differs from most classical pivot rules.

4 Learning as a Pivot Expert

While our pivot experts offer an advantage, their dependency on the optimal basis may seem prohibitive for direct applications. Thus, we employ machine learning to bypass this requirement. Specifically, we cast the LP as a bipartite graph, using a GCNN model to emulate the choices of pivot experts, which is similar to Gasse's GCNN (Gasse et al. 2019).

State encoding and input features The primal simplex method can be viewed as a Markov decision process, as shown in Figure 1. At the kth iteration, the state $\mathbf{s}_k$ contains the LP instance and the current basic feasible solution $\mathbf{x}^k$. The action space $\mathcal{A}(\mathbf{s}_k)$ encapsulates all possible edges that can improve the objective value. An action $\mathbf{a}_k$ is selected according to a certain pivot rule and $\mathbf{x}^k$ will move along $\mathbf{a}_k$ until reaching the next vertex $\mathbf{x}^{k+1}$.

We encode the state of LP and current solution into a bipartite graph, see Figure 2. Variables and constraints form two classes of nodes which will be linked by an edge if the corresponding coefficient $\mathbf{A}_{i,j}$ is non-zero. Each node and edge carry some features that are picked for pivot decision and thus slightly differ from those defined by Gasse.

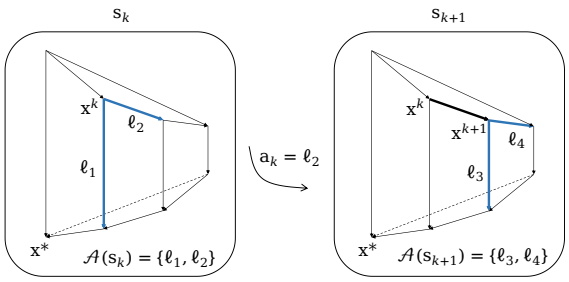


Figure 1: The primal simplex method can be viewed as a Markov decision process. The LP is depicted as a polyhedron with directions that can improve objective value marked on edges. $\mathbf{x}^*$ denotes the optimal solution.

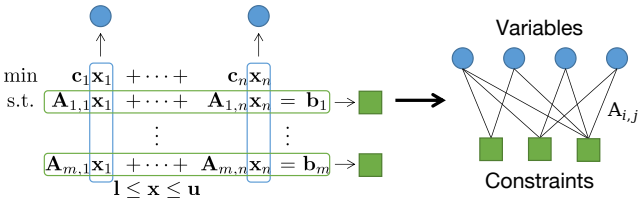


Figure 2: Bipartite graph encoding for LP.

Policy for imitating experts We feed the bipartite graph into our GCNN model and then use a filter and a softmax function. This filter identifies suitable candidates, while the softmax estimates their chance of entering the basis. Our GCNN model shares a similar structure with Gasse's but has slightly wider and deeper networks.

It is essential to distinguish between learning to pivot in LP and learning to branch in MIP. While both involve variable selection, branching explores multiple paths on a tree, requiring crucial early smart choices, whereas pivoting follows a single path, less dependent on initial decisions but needing consistent smart moves.

Summary The pivot expert designing and learning framework can be summarized in Figure 3. For a class of LP instances, which are expected to share common features in polyhedra, we have designed two pivot experts that can generate shorter paths. To replicate the expertise using imitation learning, we gather paired data of encoded LP instances, along with labels that detail bipartite graphs and the experts' pivot decisions. Using this data, we train the policy network to accurately mirror the experts' actions.

5 Experiments

Our experiments are twofold. In Section 5.1, we contrast our advanced pivot experts against others, demonstrating their ability to produce shorter pivot paths. Section 5.2 showcases the commendable performance of the learned rule, affirming the feasibility of training our expert rules via imitation learning without sacrificing overall enhancement. All tests are conducted on an AMD Ryzen 7 5700X CPU and NVIDIA RTX 3060 12GB GPU.

Pivot rules to be compared Table 1 enumerates the pivot rules under comparison. We adopt five classical pivot rules (Bland's, Dantzig's, SE, GI, and LD) as our benchmarks. Our primary focus is on two expert rules (EXP and EXP-II), alongside our learned rule (EXP-LEARN). In all experiments, every pivot rule receives a consistent initial basis from Phase I, resolved by SE. Notably, EXP and EXP-II, requiring an additional optimal basis, are provided in Phase II by SE. EXP-LEARN operates without the extra information that the expert rules need. NO-LOCAL, a derivative of the EXP rule, omits local information and serves for ablation analysis.

Type	Notation	Pivot rule
Classical rules	Bland	Bland's rule
	Dantzig	Dantzig's rule
	SE	Steepest-edge rule
	GI	Greatest improvement rule
	LD	Largest distance rule
Our experts	EXP	Expert I
	EXP-II	Expert II
	EXP-LEARN	Rule imitating Expert I
Ablation study	NO-LOCAL	EXP w/o local information

Table 1: Pivot rules to be compared.

Benchmarks We have chosen a diverse set of LP problem tests, including a NETLIB subset (Gay 1985) and LP relaxations from four combinatorial optimization (CO) classes. NETLIB, a standard LP benchmark, offers varied LP instances in both scale and structure. Our CO classes cover set covering (SC), combinatorial auction (CA), capacitated facility location (FL), and maximum independent set (IS). These CO problems, inspired by Gasse et al. (2019), may differ in scale. We presolve NETLIB instances with Gurobi 10.0.2 and CO instances with SCIP 8.0.3.

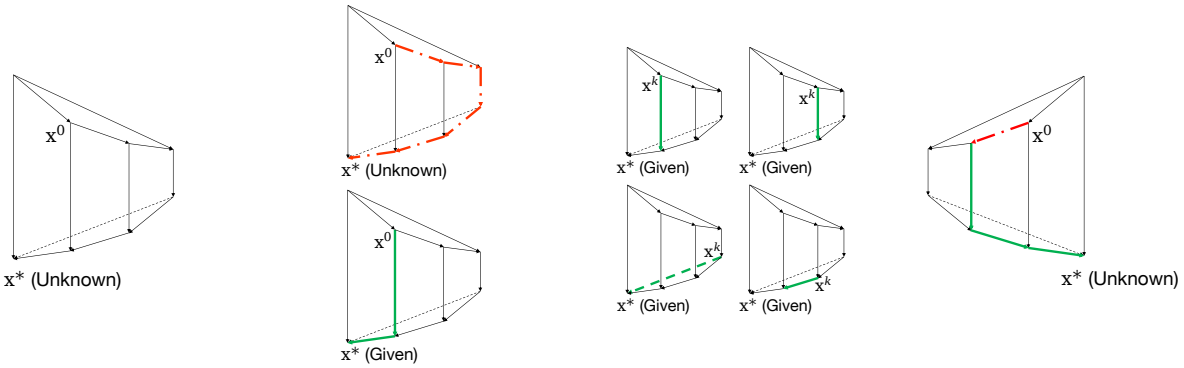
Evaluation We evaluate pivot rule performance primarily with the geometric mean of pivot numbers serving as our benchmark metric. Two reasons drive this choice. First, while our Python-implemented pivot rules might not mirror modern solvers' speed, they are generally comparable in pivot numbers with Gurobi's primal simplex for many LP instances. Second, pivot path length better captures the simplex method's complexity. Additionally, for thoroughness, we will also report each rule's execution time.

5.1 Testing Pivot Experts

We evaluate our experts against classical methods on the NETLIB subset. Section 5.2 details the outcomes on the CO benchmarks, along with our learning analysis.

Setup We utilize 77 selected NETLIB instances, optimized for time and to bypass numerical issues. Each instance adheres to a 300-second time limit. The number of constraints m and variables n for these instances are provided in Table 2.

[1]Standard deviations are provided in parentheses. This convention is maintained for subsequent tables.



(a) A class of LP instances.

(b) Given the optimal basis, pivot experts can generate shorter paths.

(c) Collect expert choices for imitation.

(d) Learned pivot rule gains improvement on new LP instances.

Figure 3: Pivot expert designing and learning framework in the order of (a) →(b) →(c) →(d). Bold green lines are expert choices while bold red (dash-dot) lines are bad choices.

# instance	m	n
77	356.6 ($\pm$364.4)	1439.6 ($\pm$2975.5)

Table 2: Average scales of presolved NETLIB instances.[1]

Numerical results The results in Table 3 and 4 emphasize the efficacy of EXP and EXP-II over classical pivot rules. EXP and EXP-II evidently obtain fewer pivot numbers and more wins, while Bland's rule has the worst performance and is thus excluded in future comparisons.

Bland	Dantzig	SE	GI	LD	EXP
2	7	16	15	6	**52**
Bland	Dantzig	SE	GI	LD	EXP-II
2	6	17	12	6	**54**

Table 3: Number of wins on the NETLIB subset with a total of 77 instances.[2]

Bland	Dantzig	SE	GI
851	198	121	131
LD	EXP	EXP-II	NO-LOCAL
177	**112**	118	139

Table 4: Geometric mean of pivot numbers on the NETLIB subset.

Ablation study: the role of local information in expert rules Classical pivot rules, like the SE, GI, Bland's, and Dantzig's rules, largely utilize local information, such as reduced costs. In contrast, our expert rules merge both global (the optimal basis) and local information (the steepest-edge score) to set the pivot direction. Here, we emphasize the pivotal role local information plays in optimizing expert rules.

We introduce a variant of the EXP rule, called NO-LOCAL, that omits local information. In this rule, the entering variable is randomly chosen from candidates in the optimal basis. Its efficacy is tested on the NETLIB subset.

Table 4 reveals that NO-LOCAL underperforms EXP or EXP-II rules, and even lags behind the classical SE rule. The results underscore the diminished efficacy of the EXP rule when local insights are absent, leading to increased pivot numbers and fewer wins. Local information is evidently instrumental in optimizing pivot decisions.

5.2 Testing Learned Pivot Rule

We conduct experiments with EXP-LEARN on CO benchmarks, employing the imitation learning approach to emulate the EXP rule.

Setup Based on the guidelines from Gasse et al. (2019), we generate random instances for each CO benchmark. For the set covering problems, instances have 400 columns and 200 rows. Combinatorial auction problems have 100 items and 500 bids. For capacitated facility location problems, instances contain 20 facilities and 15 customers. Finally, maximum independent set problems have 150 nodes with an affinity value set to 2. Table 5 details the scales of these presolved CO instances.

Training procedure In the training process, we utilize 5 unique seeds for both data generation and model training. We apply identical hyperparameters for each CO benchmark, drawing upon 50,000 pivot samples from 1,000 instances for training, and 10,000 samples from 200 instances for validation. To assess the model's applicability in real-world scenarios, we test it on 200 new instances, underlining its ability to generalize on each benchmark.

[2]Bolded results indicate the best among the evaluated pivot rules. This convention is maintained for subsequent tables.

	# train	m	n	# valid	m	n	# test	m	n
SC	1000	200.0 (±0.0)	400.0 (±0.0)	200	200.0 (±0.0)	400.0 (±0.0)	200	200.0 (±0.0)	400.00 (±0.0)
CA	1000	181.9 (±5.0)	427.4 (±17.0)	200	182.0 (±5.5)	428.2 (±18.7)	200	182.5 (±5.5)	427.8 (±18.6)
FL	1000	336.0 (±0.0)	315.0 (±0.0)	200	336.0 (±0.0)	315.0 (±0.0)	200	336.0 (±0.0)	315.0 (±0.0)
IS	1000	290.2 (±2.4)	150.0 (±0.0)	200	290.2 (±2.3)	150.0 (±0.0)	200	290.2 (±2.1)	150.0 (±0.0)

Table 5: Average scales of presolved CO instances for EXP-LEARN to imitate EXP.

Performance metrics Performance is assessed using the GCNN model's validation accuracy, detailed in Table 6. We rely on Top 1, Top 3, and Top 5 accuracies.

	Top 1 Acc	Top 3 Acc	Top 5 Acc
SC	0.533 (±0.004)	0.847 (±0.005)	0.927 (±0.003)
CA	0.362 (±0.004)	0.656 (±0.002)	0.784 (±0.002)
FL	0.499 (±0.007)	0.776 (±0.007)	0.870 (±0.005)
IS	0.257 (±0.002)	0.420 (±0.002)	0.511 (±0.001)

Table 6: Accuracy on the CO validation sets.

Numerical results As displayed in Table 6, our model achieves a Top 1 accuracy over 25% for all problems. This suggests a greater than 25% chance of copying the expert action. Notably, while Top 3 and Top 5 accuracies are significant, they cannot be directly applied in the simplex method, marking a limitation. It also needs to be emphasized that comparing validation accuracy across different benchmarks is not meaningful. Moreover, we choose not to include test accuracy, as our primary focus is not on direct accuracy comparison but on pivot numbers.

	SE	GI	LD	EXP-LEARN	EXP	EXP-II
SC	419	990	468	**336** (±6)	268	280
CA	266	1340	276	**223** (±3)	115	112
FL	242	304	377	**239** (±2)	224	227
IS	114	302	114	**113** (±0)	113	113

Table 7: Geometric mean of pivot numbers on the CO test sets.[3]

Table 7 shows that EXP-LEARN consistently outshines its competitors, highlighting its smart choices. While EXP-LEARN displays great performance among most benchmarks, it exhibits only a slight lead over the SE rule in certain benchmarks, primarily due to the foundational efficiency of the EXP rule it is built upon. This demonstrates that while imitation learning brings benefits, it does not fully bridge the performance gap between SE and EXP.

Table 8 underscores the consistency of the EXP-LEARN rule. Its dominant performance is not due to a few outliers but is maintained across numerous instances in each benchmark. This indicates a robust and generalizable model, proving EXP-LEARN's reliability in varied scenarios and highlighting its potential for broader applications in LP tasks.

[3]The results for EXP and EXP-II are for reference only as they cannot be directly used in practical applications.

	SE	GI	LD	EXP-LEARN
SC	8 (±3)	0 (±0)	0 (±0)	**192** (±3)
CA	7 (±2)	0 (±0)	2 (±2)	**192** (±3)
FL	81 (±2)	17 (±2)	17 (±2)	**103** (±3)
IS	171 (±1)	0 (±0)	171 (±1)	**186** (±2)

Table 8: Number of wins on the CO test sets with a total of 200 instances.

	SE	GI	LD	EXP-LEARN	EXP	EXP-II
SC	0.54	6.85	**0.32**	1.36 (±0.02)	0.32	0.83
CA	0.21	8.44	**0.20**	0.81 (±0.01)	0.08	0.18
FL	**0.43**	1.28	0.60	0.82 (±0.03)	0.39	0.72
IS	0.15	0.98	**0.14**	0.33 (±0.00)	0.16	0.37

Table 9: Geometric mean of solving time (in seconds) on the CO test sets.

Table 9 quantifies the pivot efficiency across different benchmarks. EXP-LEARN tends to solve in longer time, which is a byproduct of its GCNN forward pass. With a more meticulous design of the graph architecture, the time can be further optimized.

6 Conclusion

The simplex methods are time-honored with rich practical design and mystery complexity. Generating a short path is the key task for pivot rules. In this paper, we design two innovative and smart pivot experts for primal simplex that leverage both global and local information, i.e. optimal basis and steepest-edge score respectively. Experiments illustrate that these two experts overall outperform classical pivot rules significantly. To bridge theory to practical application, we integrate a GCNN model to mimic these experts. This imitation learning facilitates the circumvention of global information dependencies while preserving the performance in path generation. Empirical evidence confirms the learnability of our experts. The learned rule commendably surpasses classical pivot rules in generating shorter pivot paths, although not quite caught up with the experts.

The value of our pivot experts extends beyond their standalone significance, serving both as benchmarks and generators of expert pivot labels. The pivot experts outpace predecessors like MCTS in swiftly constructing superior paths, especially Expert I. Modifying our method for dual or primal-dual simplex methods, we anticipate, will be seamless with minimal adjustments.

Acknowledgments

We thank Qi Huangfu for the fruitful discussions.

This research is partially supported by the National Natural Science Foundation of China (NSFC) [Grant NSFC-72150001, 72225009, 72394360, 72394365].

References

Achterberg, T. 2009. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1: 1–41.

Achterberg, T.; Koch, T.; and Martin, A. 2005. Branching rules revisited. *Operations Research Letters*, 33(1): 42–54.

Adham, I.; De Loera, J.; and Zhang, Z. 2021. (Machine) learning to improve the empirical performance of discrete algorithms. *arXiv preprint arXiv:2109.14271*.

Alvarez, A. M.; Louveaux, Q.; and Wehenkel, L. 2017. A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, 29(1): 185–195.

Applegate, D.; Díaz, M.; Hinder, O.; Lu, H.; Lubin, M.; O'Donoghue, B.; and Schudy, W. 2021. Practical large-scale linear programming using primal-dual hybrid gradient. *Advances in Neural Information Processing Systems*, 34: 20243–20257.

Avis, D.; and Chvátal, V. 1978. Notes on Bland's pivoting rule. *Polyhedral Combinatorics: Dedicated to the memory of DR Fulkerson*, 24–34.

Bland, R. G. 1977. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2): 103–107.

Chen, Z.; Liu, J.; Wang, X.; Lu, J.; and Yin, W. 2022. On representing linear programs by graph neural networks. *arXiv preprint arXiv:2209.12288*.

Dantzig, G. 1963. *Linear programming and extensions*. Princeton university press.

Deng, Q.; Feng, Q.; Gao, W.; Ge, D.; Jiang, B.; Jiang, Y.; Liu, J.; Liu, T.; Xue, C.; Ye, Y.; et al. 2022. New developments of ADMM-based interior point methods for linear programming and conic programming. *arXiv preprint arXiv:2209.01793*.

Di Liberto, G.; Kadioglu, S.; Leo, K.; and Malitsky, Y. 2016. DASH: dynamic approach for switching heuristics. *European Journal of Operational Research*, 248(3): 943–953.

Ding, J.-Y.; Zhang, C.; Shen, L.; Li, S.; Wang, B.; Xu, Y.; and Song, L. 2020. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1452–1459.

Fan, Z.; Wang, X.; Yakovenko, O.; Sivas, A. A.; Ren, O.; Zhang, Y.; and Zhou, Z. 2023. Smart initial basis selection for linear programs. In *International Conference on Machine Learning*, 9650–9664. PMLR.

Forrest, J. J.; and Goldfarb, D. 1992. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 57(1-3): 341–374.

Gasse, M.; Chételat, D.; Ferroni, N.; Charlin, L.; and Lodi, A. 2019. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32.

Gay, D. M. 1985. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, 13: 10–12.

Ge, D.; Huangfu, Q.; Wang, Z.; Wu, J.; and Ye, Y. 2023. Cardinal Optimizer (COPT) user guide. https://guide.coap.online/copt/en-doc.

Goldfarb, D.; and Reid, J. K. 1977. A practicable steepest-edge simplex algorithm. *Mathematical Programming*, 12: 361–371.

Goldfarb, D.; and Sit, W. Y. 1979. Worst case behavior of the steepest edge simplex method. *Discrete Applied Mathematics*, 1(4): 277–285.

Gupta, P.; Gasse, M.; Khalil, E.; Mudigonda, P.; Lodi, A.; and Bengio, Y. 2020. Hybrid models for learning to branch. *Advances in neural information processing systems*, 33: 18087–18097.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer reference manual.

Harris, P. M. 1973. Pivot selection methods of the Devex LP code. *Mathematical Programming*, 5: 1–28.

He, H.; Daume III, H.; and Eisner, J. M. 2014. Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27.

Huangfu, Q.; and Hall, J. J. 2018. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1): 119–142.

Jeroslow, R. G. 1973. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4): 367–377.

Kalai, G. 1992. A subexponential randomized simplex algorithm. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, 475–482.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 302–311.

Kelner, J. A.; and Spielman, D. A. 2006. A randomized polynomial-time simplex algorithm for linear programming. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 51–60.

Khalil, E.; Le Bodic, P.; Song, L.; Nemhauser, G.; and Dilkina, B. 2016. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Kitahara, T.; and Mizuno, S. 2011. Klee–Minty's LP and upper bounds for Dantzig's simplex method. *Operations Research Letters*, 39(2): 88–91.

Kitahara, T.; and Mizuno, S. 2013. A bound for the number of different basic solutions generated by the simplex method. *Mathematical Programming*, 137: 579–586.

Klee, V.; and Minty, G. J. 1972. How good is the simplex algorithm. *Inequalities*, 3(3): 159–175.

Li, A.; Li, B.; Han, C.; and Guo, T. 2022. Rethinking optimal pivoting paths of simplex method. *arXiv preprint arXiv:2210.02945*.

Maros, I. 2012. *Computational techniques of the simplex method*, volume 61. Springer Science & Business Media.

Matoušek, J.; Sharir, M.; and Welzl, E. 1992. A subexponential bound for linear programming. In *Proceedings of the eighth annual symposium on Computational geometry*, 1–8.

Nair, V.; Bartunov, S.; Gimeno, F.; Von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2020. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.

Nickel, S.; Steinhardt, C.; Schlenker, H.; and Burkart, W. 2022. IBM ILOG CPLEX Optimization Studio—a primer. In *Decision Optimization with IBM ILOG CPLEX Optimization Studio: A Hands-On Introduction to Modeling with the Optimization Programming Language (OPL)*, 9–21. Springer.

Pan, P.-Q. 2008. A largest-distance pivot rule for the simplex algorithm. *European Journal of Operational Research*, 187(2): 393–402.

Paulus, M. B.; and Krause, A. 2023. Learning to dive in branch and bound. *arXiv preprint arXiv:2301.09943*.

Roos, C. 1986. A pivoting rule for the simplex method which is related to Karmarkar's potential function. *Manuscript, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands*, 78–94.

Roos, C. 1990. An exponential example for Terlaky's pivoting rule for the criss-cross simplex method. *Mathematical Programming*, 46: 79–84.

Sonnerat, N.; Wang, P.; Ktena, I.; Bartunov, S.; and Nair, V. 2021. Learning a large neighborhood search algorithm for mixed integer programs. *arXiv preprint arXiv:2107.10201*.

Suriyanarayana, V.; Tavaslioğlu, O.; Patel, A. B.; and Schaefer, A. J. 2022. Reinforcement learning of simplex pivot rules: a proof of concept. *Optimization Letters*, 16(8): 2513–2525.

Tamura, A.; Takehara, H.; Fukuda, K.; Fujishige, S.; and Kojima, M. 1988. A dual interior primal simplex method for linear programming method. *Journal of the Operations Research Society of Japan*, 31(3): 413–430.

Todd, M. J. 1990. A Dantzig-Wolfe-like variant of Karmarkar's interior-point linear programming algorithm. *Operations Research*, 38(6): 1006–1018.

Vanderbei, R. J. 2020. *Linear programming*. Springer.

Xpress, F. 2014. FICO Xpress Optimization Suite.

Yang, Y. 2020. A double-pivot simplex algorithm and its upper bounds of the iteration numbers. *Research in the Mathematical Sciences*, 7(4): 34.

Ye, Y. 2011. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4): 593–603.