SCIENCE CHINA
Mathematics

AI Methods for Optimization Problems
• ARTICLES •

June 2024 Vol. 67 No. 6: 1263–1286
https://doi.org/10.1007/s11425-022-2259-1

Optimal pivot path of the simplex method for linear programming based on reinforcement learning

Anqi Li, Tiande Guo, Congying Han*, Bonan Li & Haoran Li

School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049*, China*

*Email: lianqi*20*@mails.ucas.ac.cn, tdguo@ucas.ac.cn, hancy@ucas.ac.cn, libonan@ucas.ac.cn, lihaoran*21*@mails.ucas.ac.cn*

Received November 27, 2022; accepted January 23, 2024; published online February 29, 2024

Abstract Based on the existing pivot rules, the simplex method for linear programming is not polynomial in the worst case. Therefore, the optimal pivot of the simplex method is crucial. In this paper, we propose the optimal rule to find all the shortest pivot paths of the simplex method for linear programming problems based on Monte Carlo tree search. Specifically, we first propose the SimplexPseudoTree to transfer the simplex method into tree search mode while avoiding repeated basis variables. Secondly, we propose four reinforcement learning models with two actions and two rewards to make the Monte Carlo tree search suitable for the simplex method. Thirdly, we set a new action selection criterion to ameliorate the inaccurate evaluation in the initial exploration. It is proved that when the number of vertices in the feasible region is C_n^m, our method can generate all the shortest pivot paths, which is the polynomial of the number of variables. In addition, we experimentally validate that the proposed schedule can avoid unnecessary search and provide the optimal pivot path. Furthermore, this method can provide the best pivot labels for all kinds of supervised learning methods to solve linear programming problems.

Keywords simplex method, linear programming, pivot rules, reinforcement learning

MSC(2020) 90C27

Citation: Li A Q, Guo T D, Han C Y, et al. Optimal pivot path of the simplex method for linear programming based on reinforcement learning. Sci China Math, 2024, 67: 1263–1286, https://doi.org/10.1007/s11425-022-2259-1

1 Introduction

The simplex method is a classical method for solving linear programming (LP) problems. Although it is a nonpolynomial-time algorithm, its worst case rarely occurs and its average performance is better than that of polynomial-time algorithms, such as the interior point method and the ellipsoid method, especially for small-scale and medium-scale problems. Much research work has focused on making the simplex method a polynomial-time algorithm, but it has not been successful. The existing pivot rules can neither provide the optimal pivot paths for the simplex method nor make it a polynomial-time algorithm. In addition, the traditional design idea only applies to designing the pivot rule suitable for certain types of problems. There are no general ways to find the least number of pivot iterations for all types of linear programming. Our research goal is to design a general optimal pivot rule based on the inherent features

* Corresponding author

of linear programming extracted by reinforcement learning (RL) that can be solved in polynomial-time. This study is the first step toward achieving this goal.

With the rise of machine learning (ML), ML-based technologies provide researchers with new ideas of pivot rules. Based on the deep Q-network (DQN) [25, 26], DeepSimplex [35] provides a pivot rule that can select the most suitable pivot rule for the current state between the Dantzig and steepest-edge rules. While another study [1] provides an instance-based method, the most suitable pivot rule for the current instance is learned among the five conventional pivot rules. The above two methods are based on several given pivot rules, and then learn the pivot rule scheduling scheme depending on the solution state or input instances. Therefore, the performance of these methods is heavily dependent on the supervised pivot rules. Unfortunately, owing to the lack of optimal labels, we see that supervised pivot rules cannot extract the optimal pivot paths for the simplex method.

In addition, the difficulty in determining the optimal pivot path lies in the information after several pivot iterations in the future. The existing solution state is insufficient for optimal future decisions. The most effective method is to appropriately assess the future situation before deciding to guide the best pivot. Fortunately, this idea is consistent with the Monte Carlo tree search (MCTS). Specifically, MCTS explores the trajectory in advance to evaluate and obtain future information to guide decision-making, significantly reducing the invalid search space and effectively guiding the best decision-making. Thus, the simplex method can effectively use future information to guide the current optimal pivot decision.

Motivated by these observations, we propose to analyze and improve the simplex method in pivoting with the Monte Carlo tree search, further pushing forward the frontier of the simplex method for linear programming in a general way. We focus on four core aspects: (1) transforming the simplex method into a pseudo-tree structure, (2) constructing appropriate reinforcement learning models, (3) providing the MCTS rule to find all shortest pivot paths, and (4) giving thorough theory for the optimality and complexity of the MCTS rule, as shown in Figure 1.

First, transforming the simplex method into the tree search mode is the premise for applying the Monte Carlo tree search method. Considering the connectivity and acyclicity characteristics, the tree structure can effectively avoid the generation of cycles in exploration paths. In this way, it ingeniously avoids repetition of the basis variables in exploration. To construct an imitative tree structure,

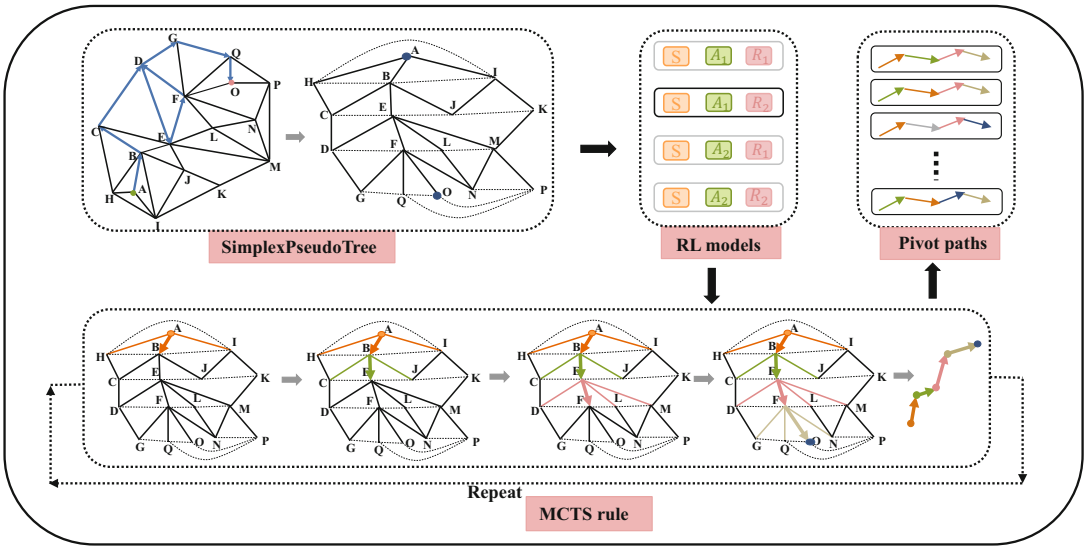


Figure 1 (Color online) Overview of the methodological framework in this paper. Firstly, we create the SimplexPseudoTree to transform the simplex method applicable to reinforcement methods in Section 3. Next, four RL models are proposed in Subsection 4.1 based on the SimplexPseudoTree. Then we propose the MCTS rule to calculate all the shortest pivot paths in Subsections 4.2 and 4.3. Finally, we give thorough theory analysis for the MCTS rule in Section 5

the SimplexPseudoTree, we propose to regard current states as nodes and the corresponding pivoting process as edges. The original linear programming instance is taken as the root node, and optimal solutions correspond to leaf nodes. Furthermore, we remove edges between nodes in the same layer and pivot from the deep layer to the shallow layer. Under this construction, the problem of finding the shortest path from the root node to the leaf node is equivalent to that of finding the optimal pivot rule.

Subsequently, we transform the problem of finding the optimal pivot path into four reinforcement-learning models. In particular, the following two novel action spaces and two reward functions are introduced:

• Action set: (1) non-basis variables whose reduced costs are less than zero, and (2) non-basis variables whose reduced costs are not equal to zero.

• Reward functions: (1) opposite of pivot iterations, and (2) linear decay weight estimation of the average variation in the objective value caused by a single pivot.

Based on the proposed actions and rewards, we design four models of different forms. Through a comprehensive comparison, it is found that the model comprising action set (1) and reward functions (2) achieves the highest efficiency with the least computational cost. Furthermore, we construct a novel action selection criterion for the simplex method to ameliorate inaccurate evaluation in the initial exploration. Subsequently, we present the MCTS rule based on the Monte Carlo tree search to determine the optimal pivot for current states.

In addition, the optimal pivot that corresponds to the minimum pivot iterations is not necessarily unique. However, current research has not provided a way to find multiple optimal pivot paths. Unlike deterministic pivot rules, our MCTS rule exhibits certain randomness in the exploration stage. Specifically, the proposed exploration criterion can introduce a controllable scale factor based on the upper confidence bounds (UCB) method. Thus, additional randomness is added to the balance between the estimated value and explorations brought by the UCB algorithm. Additionally, the randomness of the MCTS rule can guide the selection of different actions to achieve the minimum pivot iterations under the guidance of optimality.

Consequently, we prove the optimality and completeness of the MCTS rule. We also prove the polynomial complexity of the optimal pivot iterations when the number of vertices in the feasible region is C_n^m. Concretely, the MCTS rule can find all the shortest pivot paths according to the Wiener-Khinchin law of large numbers. Firstly, the MCTS rule can find the optimal pivot path when explorations approach infinity. Then the MCTS rule can find all the different pivot paths when executions approach infinity. Additionally, from the perspective of combinatorial numbers, we prove that the minimum pivot iterations is polynomial of variables when the number of vertices in the feasible region is C_n^m. We also verify the polynomial iterations from the geometric perspective.

Given the above four aspects, we present a novel MCTS rule that provides all the shortest pivot paths. Additionally, we can label massive instances with little cost for the supervised pivot rule based on the proposed MCTS rule. Comprehensive experiments on the NETLIB benchmark and random instances demonstrated the efficacy of the MCTS rule. It is worth noting that compared with the minimum pivot iterations achieved by other popular pivot rules, our result is only 54.55% for random instances and 49.06% for NETLIB.

Our main contributions are as follows:

• Construct the SimplexPseudoTree to ensure that MCTS can be applied to the simplex method while avoiding duplicate bases.

• Propose the MCTS rule to determine all the optimal pivot sequences.

• Provide a method to obtain the optimal pivoting labels for the supervised pivot rule within the allowable range of the calculation cost.

• Give comprehensive theory for the optimality and complexity of the MCTS rule.

The rest of this paper is organized as follows. Section 2 introduces the background and related works. Section 3 introduces the SimplexPseudoTree to translate the simplex method for applying RL methods. Section 4 presents the optimal MCTS rule for the simplex method. We prove the optimality

and complexity of the proposed optimal pivot rule in Section 5. Section 6 presents experimental results. The conclusions and further works are presented in Sections 7 and 8, respectively.

2 Background and related work

LP problem. Linear programming is a type of optimization problem, where the objective function and constraints are linear. The standard form of the simplex method is as follows:

$$\begin{aligned} &\min c^{\mathrm{T}}x \\ &\text{s.t. } Ax = b, \\ &\quad x \geqslant 0, \end{aligned} \tag{2.1}$$

where $c \in \mathbb{R}^n$ is the objective function coefficient, $A \in \mathbb{R}^{m\times n}$ is the constraint matrix, $b \in \mathbb{R}^m$ is the right-hand side, and all the variables take continuous values in the feasible region. The purpose of linear programming is to find a solution that minimizes the value of the objective function in the feasible region, i.e., the so-called optimal solution, while the corresponding objective value is the optimal value.

Simplex method. The simplex method is clear and easy to understand. After providing an initial feasible solution, we see that the pivoting process includes three parts: variable division, selection of the entering basis variable, and derivation of the leaving basis variable [7]. The variable division step divides the variable $x \in \mathbb{R}^n$ into $x = [x_B^{\mathrm{T}}, x_N^{\mathrm{T}}]^{\mathrm{T}}$. Correspondingly, divide $A = [B, N]$ and $c = [c_B^{\mathrm{T}}, c_N^{\mathrm{T}}]^{\mathrm{T}}$, and each column of the coefficient matrix B corresponding to x_B is required to be linearly independent. At this time, the constraint $Ax = b$ can be written as

$$x_B = B^{-1}b - B^{-1}Nx_N. \tag{2.2}$$

We can obtain (2.3) by substituting the constraint into the objective function, i.e.,

$$c^{\mathrm{T}}x = c_B^{\mathrm{T}}B^{-1}b + (c_N^{\mathrm{T}} - c_B^{\mathrm{T}}B^{-1}N)x_N. \tag{2.3}$$

As the first term is a constant value, the objective function value is determined only by the second term

$$\bar{c}^{\mathrm{T}} = c_N^{\mathrm{T}} - c_B^{\mathrm{T}}B^{-1}N, \tag{2.4}$$

which is called reduced costs. The components of x_N corresponding to the part of (2.5) are non-basis variables that can cause a decrease in the objective function, i.e.,

$$J = \{j \mid \bar{c}_j < 0\}. \tag{2.5}$$

Therefore, the selection of the entering basis variable is the process of selecting a basis variable from the non-basis variables mentioned above. Different pivot rules provide different methods for selecting the entering basis variable. In other words, the essence of the pivot rule of the simplex method is to convert a certain column between bases corresponding to feasible solutions. Accordingly, the feasible region polyhedron starts from the initial solution vertex, and each pivot corresponds to a step transition between adjacent vertices until it reaches the optimal solution vertex. When the basis variables are determined, we can use (2.6) to derive the leaving basis variable, i.e.,

$$x_B = B^{-1}b - B^{-1}Nx_N \geqslant 0. \tag{2.6}$$

After the initial feasible solution is given, the pivot process is repeated until the basis corresponding to the optimal solution is obtained, i.e., the end of the simplex method.

Classical pivot rules. The pivot rule provides the direction for the exchange of the basis variables of the simplex method. The simplex method has several classical pivot rules. The Dantzig rule [7] is to

select the component corresponding to the most negative reduced cost as the entering basis variable, i.e., choose

$$\tilde{J} \in \arg\min\{\bar{c_j} \mid \bar{c_j} < 0\}. \tag{2.7}$$

The Bland rule [2] is to select the component with the smallest index from the variables with reduced costs less than zero, i.e., choose

$$\tilde{J} \in \arg\min\{j \mid \bar{c}_j < 0\}. \tag{2.8}$$

The steepest-edge rule [10, 13] uses the columns of the corresponding non-basis matrix and the basis matrix to standardize the reduced costs and selects the column corresponding to the smallest component, i.e., choose

$$\tilde{J} \in \arg\min\left\{\frac{\bar{c}_j}{\|B^{-1}N_j\|} \,\middle|\, \bar{c}_j < 0\right\}. \tag{2.9}$$

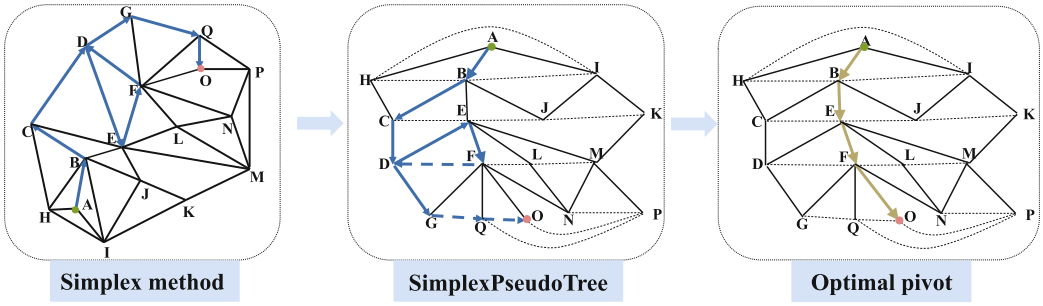
The idea of the greatest improvement rule [21] is to take the product of the reduced costs and the maximum increment of each non-basis variable as the evaluation standard and select the non-basis variable with the minimum value as the entering basis variable. Finally, the devex rule [16, 28], an approximation of the steepest-edge rule, uses an approximate weight to replace the norm in the evaluation criterion of the steepest-edge rule. Considering the number of pivot iterations, we see that different pivot rules apply to different problem types. However, there is no universal pivot rule that can determine minimum pivot iterations for general linear programming instances. In addition, there is an LP instance so that the corresponding simplex method is not a polynomial algorithm for any pivot rules given above.

Pivot rules based on machine learning. In recent years, the development of machine learning has provided new ideas for combinatorial optimization. Specifically, Guo et al. [14, 15] gave overviews of solving combinatorial optimization problems. ML-based methods gradually emerge to solve combinatorial optimization problems, such as knapsack [8], the traveling salesman problem [37, 38] and the P-median problem [36]. Simultaneously, there are many ML-based methods [9, 22–24] involving continuous optimization problems. In terms of the linear programming problem, there are two methods for improving the pivot rules of the simplex method based on the machine learning method. DeepSimplex [35] uses the idea of Q-value iteration to learn the best scheduling scheme of the Dantzig rule and the steepest-edge rule. Another study [1] used a boost tree and a neural network to learn an instance-based adaptive pivot rule selection strategy based on five classical pivot rules. However, the performance of these two supervised methods is severely limited by the provided pivot rules. In general, it is difficult to obtain the minimum pivot iterations based on supervised learning without effective labels.

Combinatorial optimization methods based on MCTS. With the emergence of AlphaGo [32] and AlphaGo Zero [33], reinforcement learning represented by MCTS has been widely used in many classical problems [11, 17, 34]. Combinatorial optimization problems based on the MCTS can be solved in two ways. A classical idea is to design an MCTS-based framework for various types of combinatorial optimization problems [19, 30]. Another idea is to design an MCTS-based algorithm to solve a specific combinatorial optimization problem, such as the traveling salesman problem [27,29,38,39] and the Boolean satisfiability problem [5, 12, 18, 31]. We adopt the latter idea to find multiple optimal pivot paths for the simplex method based on MCTS.

3 Constructed SimplexPseudoTree model

Reinforcement learning involves making the agent interact with the environment in a trial-and-error manner. The results of trial-and-error are fed back to the agent in the form of rewards to guide agent behavior and achieve the goal of maximizing the rewards. However, the current and past solution states are inadequate for determining the optimal pivot paths. Actually, the future information obtained by executing the pivot makes the difference. In this case, Monte Carlo tree search is more effective in finding the optimal pivot. Therefore, constructing an imitative tree-search model for the simplex method is our primary goal.



(a) (b) (c)

Figure 2 (Color online) Proposed SimplexPseudoTree for the simplex method. (a) shows the process of finding the optimal solution according to the simplex method. (b) is the SimplexPseudoTree corresponding to the instance. (c) is the optimal pivot path found based on the SimplexPseudoTree

Duplicate basic variables lead to an unnecessary increase in pivot iterations in the simplex method. According to the reduced costs and pivot rules, the leaving basis variable cannot enter the basis into the immediate next pivot. Therefore, if the current solution state is considered a node and the pivot is considered an edge, the repetition of the basis implies that a circle appears in the exploration path, as shown in Figure 2(a). Therefore, we must avoid base duplication in our pivot rule to minimize the number of pivot iterations consistent with the connectivity and acyclic properties of tree structures.

Inspired by the tree structure, the SimplexPseudoTree is proposed to transform the simplex method for utilizing MCTS while avoiding the cycle during pivoting. The SimplexPseudoTree is constructed by considering the current states of the input instance as nodes and the feasible entering basis variables as edges. The initial linear programming instance is the root node, and the optimal solutions correspond to the leaf nodes. To minimize the number of pivot iterations, we only need to retain the pivot sequence for each node that first accesses it. Therefore, we remove the edges of the same layers and pivots pointing to shallow layer nodes from deeper layers, as shown by the dotted line in Figure 2. In Subsection 4.2, this idea is implemented by imposing significant negative rewards. It is worth noting that the SimplexPseudoTree differs from the tree structure. The SimplexPseudoTree allows multiple paths between two arbitrary nodes of the SimplexPseudoTree because the pivot path is not unique. In general, the process of finding the optimal pivot rule is to find the shortest path between the root and leaf nodes for the SimplexPseudoTree.

Using the idea of MCTS [3,6,33], we can evaluate the future situation for all candidate entering basis variables. Based on the explored information, we can significantly reduce the search space formed by all the possible entering basis variables to find the optimal pivot paths by minimizing the number of pivot iterations.

4 Proposed RL algorithm

4.1 RL models of the MCTS rule

Before applying reinforcement learning, we see that this subsection presents the constructed state, action, and reward functions suitable for the simplex method. We provide two action-space definitions and two reward-function definitions. It is noteworthy that this is not a one-to-one correspondence. There are four combinations of RL models, as listed in Table 1.

State. We choose the simplex tableaux as the state representation of the reinforcement learning model. Simplex tableaux is the solution state representation of the simplex method on which traditional pivot rules rely. The tableaux contains c, A, b, I_B, c_B and $\bar{c}$, where I_B's are the column indexes corresponding to basis variables. The simplex tableaux completely represents the current solution state of the input instance and there is no redundant information.

Two action sets. In the reinforcement learning model of the MCTS rule, actions correspond to feasible entering basis variables. We have provided two definitions of action spaces. One action space

Table 1 Four reinforcement learning models constructed for the simplex method

Model	State	Action	Reward
Model 1	Simplex tableaux	$A_1 = \{i \mid \bar{c}_i < 0\}$	$R_1 = -T$
Model 2	Simplex tableaux	$A_2 = \{i \mid \bar{c}_i \neq 0\}$	$R_1 = -T$
Model 3	Simplex tableaux	$A_1 = \{i \mid \bar{c}_i < 0\}$	$R_2 = \frac{\sum_{i=1}^{N} w_i(cx_{i-1} - cx_i)}{T}$
Model 4	Simplex tableaux	$A_2 = \{i \mid \bar{c}_i \neq 0\}$	$R_2 = \frac{\sum_{i=1}^{N} w_i(cx_{i-1} - cx_i)}{T}$

contains variables with reduced costs of less than zero, i.e., non-basic variables whose objective value can be reduced by a one-step pivot, as shown in (4.1), i.e.,

$$A_1 = \{i \mid \bar{c}_i < 0\}. \tag{4.1}$$

The other type of action space corresponds to variables whose reduced costs are not equal to zero. Compared with the former, this definition adds non-basic variables corresponding to reduced costs greater than zero (see (4.2)). Although on the surface a variable with a reduced cost greater than zero does not make much sense. However, it represents the greedy idea of trying to find a path with fewer pivot iterations at the expense of the objective benefit in one step, i.e.,

$$A_2 = \{i \mid \bar{c}_i \neq 0\}. \tag{4.2}$$

Two reward functions. We also provide two definitions for the reward function. The first reward function is defined as the opposite number of pivot iterations, which intuitively reflects the goal of minimizing the number of iterations, as shown in (4.3). One advantage of this is that in addition to having a minimum number of pivot iterations, the action selection guided by this reward is completely random, resulting in more randomness to find multiple pivot paths, i.e.,

$$R_1 = -T. \tag{4.3}$$

The second reward function is defined by (4.4), where T represents the maximum number of pivot iterations of the current episode, i represents the i-th pivot, x_i is the locally feasible solution obtained from the i-th pivot, and $w_i \in (0, 1]$ is the weight. It is noteworthy that the proposed linear weight factor provides the weight of linear attenuation according to the depth from the root of the tree, i.e.,

$$R_2 = \frac{\sum_{i=1}^{N} w_i(cx_{i-1} - cx_i)}{T}, \quad w_i = \frac{(T+1) - i}{T}. \tag{4.4}$$

Formula (4.4) indicates a decrease in the linear weighted estimation of the objective value caused by a single pivot. In terms of minimizing pivot iterations, the two reward definitions are equivalent. However, as far as our problem is concerned, the second reward has the following two advantages: (1) Compared with the first type of reward function, the dimensional feature of the change in the objective value is introduced. (2) The second reward function is more likely to choose the case in which the objective function changes significantly at the initial stage, and therefore, even under the influence of the MCTS random exploration, this model can easily converge to the minimum pivot iterations.

4.2 MCTS rule

Inspired by the idea of Monte Carlo tree search, we propose the MCTS rule as shown in Algorithm 1, which can find the optimal pivot iterations for general linear programming instances. The entire process is divided into four stages: construction, expansion, exploration and exploitation (see Figure 3).

Construction stage. First, we need to transform the simplex method into a structure with the reinforcement learning model representation and an imitative tree-search pattern. The construction of

Algorithm 1 The MCTS rule

Input: maximum number of explorations N_{explore} and instance I_{instance}
Output: the optimal pivot path

create node N, corresponding state s_N and action set A for I_{instance}
$N_{(s_N,a_j)} = 0$, $S_{(s_N,a_j)} = 0$ for $a_i \in A$
$P = \emptyset$
while N is not terminal **do**
  $I = \emptyset$
  while $|\{M_i, i \in I\}| < |A|$ **do**
    choose $a_i \in A$, $i \notin I$
    generate node M_i by executing a_i
    $I = I \cup \{i\}$
  end while
  $j = 0$
  while $j < N_{\text{explore}}$ **do**
    randomly select M_j from $\{j \mid Q_j \geqslant E_{\text{soft}}\}$
    randomly explore M_j to get reward G
    $N_{(s_N,a_j)} = N_{(s_N,a_j)} + 1$
    $S_{(s_N,a_j)} = S_{(s_N,a_j)} + G$
    $j = j + 1$
  end while
  choose $\hat{a} = \arg\max_{a \in A} Q(s,a) = \arg\max_{a \in A} S(s,a)/N(s,a)$
  $P = P \cup \hat{a}$
  create node $\hat{N}$ by executing $\hat{a}$
  $N = \hat{N}$
  construct state s_N and action set A for node N
end while

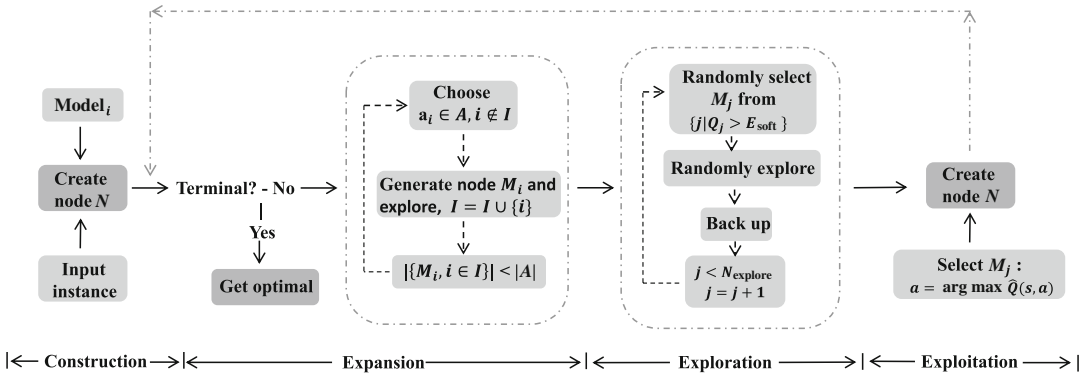


Figure 3 Algorithm flow diagram of the MCTS rule

the SimplexPseudoTree is based on the schema in Subsection 4.1, and the RL model is consistent with the strategy described in Subsection 4.2. Specifically, the state of the present node represents the current solution stage, and each edge corresponds to an action in the action space. When an edge is selected from a node, the process enters the next solution stage via the corresponding pivot. Furthermore, a reward is obtained to evaluate the advantages and disadvantages of the current path while exploring the leaf nodes. Notably, we choose to enter the next stage unless the current node satisfies the optimality.

Expansion stage. In the expansion stage, we randomly select actions in the action set without repetition to generate all the child nodes of the selected node. The extraction process is shown in (4.5), where A is the action set, a_i is an action in the action set, and I is the set of currently selected action subscripts, i.e.,

$$\text{randomly select from } \{a_i \in A \mid i \notin I\}. \tag{4.5}$$

Considering the goal of minimizing the number of pivot iterations, we see that the leaf node representing the optimality contains the information we need. In terms of the optimal pivot path, the roles of the other nodes in the path are equivalent to those of the leaf nodes. In addition, the strategy evaluation

method uses the empirical mean of the reward as the expectation of the reward. The state-action value function is defined as follows:

$$G_t = r_T, \tag{4.6}$$

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a], \quad t \in \{1,2,\ldots,T\}, \tag{4.7}$$

where $\{1,2,\ldots,T\}$ denotes the set of visited subscripts in the current exploration path. After selecting all actions $\{a_i \in A\}$ and generating all the possible child nodes $M = \{M_i, i \in I\}$, we see that the process ends. For the convenience of writing, we define

$$Q_\pi(v) = \mathbb{E}_\pi[s_t = s, a_t = a, \text{ s.t. } P(s,a) = v], \quad t \in \{1,2,\ldots,T\}, \tag{4.8}$$

$$Q_\pi(s,a) = Q_\pi(v), \tag{4.9}$$

where v represents a node in the SimplexPseudoTree, and $P(s,a)$ is the state transition function, indicating that node v can be obtained by taking action a on the state s.

Exploration stage. The value function estimation is inaccurate when the number of explorations is low. At this time, the action corresponding to the maximum value function is not necessarily the real optimal choice, but slightly smaller values near it may do so. Therefore, we suggest relaxing the max operator in the initial stage to improve accuracy. The definitions are shown in (4.10) and (4.11). The process of node selection is shown in (4.12), where the definition of Q_i is consistent with that in the upper confidence bounds applied to the trees (UCT) algorithm [20], and v is the node explored currently. We randomly select actions from $\{a_i \mid Q_i > E_{\text{soft}}\}$, and then all the actions in this episode are executed in a completely random manner, i.e.,

$$Q_i = \frac{Q(v'')}{N(v'')} + C\sqrt{\frac{2\ln N(v)}{N(v'')}}, \quad v'' \in \text{children of } v, \tag{4.10}$$

$$E_{\text{soft}} = \min_{a_i \in A} Q_i + \alpha\Big(\max_{a_i \in A} Q_i - \min_{a_i \in A} Q_i\Big), \tag{4.11}$$

$$v' = \text{random select from } \{i \mid Q_i > E_{\text{soft}}, a_i \in A\}. \tag{4.12}$$

Considering that the number of pivot iterations required for subsequent duplicate nodes must be greater than the first one, we see that this should be prohibited. Therefore, we only give real rewards to the nodes we encounter for the first time and punish repeated nodes by giving them huge negative rewards. During execution, when we encounter state s and execute action a for the first time in an episode, we add one to its count and increase the cumulative reward at that time, as shown in (4.13) and (4.14), i.e.,

$$N(s,a) \leftarrow N(s,a) + 1, \tag{4.13}$$

$$S(s,a) \leftarrow S(s,a) + G_t. \tag{4.14}$$

The state-action value function of the final state s is in the form of (4.15). According to the law of large numbers, when the number of estimates tends to infinity, the value function tends to be close to that of the real strategy. When the number of iterations in this step reaches the preset threshold N_{explore}, the process terminates and enters the next stage, i.e.,

$$Q(s,a) = S(s,a)/N(s,a). \tag{4.15}$$

Exploitation phase. In the exploitation phase, we complete N_{explore} exploration and estimate a reliable state-action value function. In this step, we select the action that maximizes the value function to generate nodes, as follows:

$$a^* = \arg\max_{a \in A} Q(s,a), \tag{4.16}$$

$$v^* = \arg\max_{v'' \in \text{ children of } v} \frac{Q(v'')}{N(v'')}. \tag{4.17}$$

Subsequently, we must check whether the generated node has reached optimality; when it is not the optimal solution, we must return the node to the expansion stage and repeat the cycle.

4.3 Extracting multiple shortest pivot paths

The optimal pivot paths correspond to different pivot sequences with minimum pivot iterations. Therefore, such an optimal path is not unique, which is important for the simplex method, but previous work is difficult to solve and ignores this point. Fortunately, the randomness of the MCTS rule is highly effective for finding multiple optimal paths. This randomness originates in the exploration stage. Specifically, the generation of the exploration trajectory depends on a random strategy. Thus, the estimated value brought about by limited exploration will be affected by randomness to a certain extent. Based on the randomness of the MCTS rule, our algorithm can select different actions that lead to optimal pivot paths in different execution processes. Therefore, multiple pivot sequences can be used to achieve optimization. Furthermore, we provide proof to ensure that each optimal pivot path can be found.

5 Theoretical analysis

In this section, we conduct a detailed theoretical analysis of the MCTS rule from three perspectives. First, we prove that the MCTS rule can make the optimal pivot decision at each step, so as to find the shortest pivot path. Then considering the completeness of the algorithm, we can find all the optimal pivot paths when algorithm executions are sufficient. Finally, we prove that the pivot iterations under the MCTS rule are a polynomial of n when the number of vertices in the feasible region is C_n^m.

5.1 Optimality of the MCTS rule

We prove that the MCTS rule converges to the optimal pivot path when explorations approach infinity. Considering that the expectation of the reward function in Models 1 and 2 will be affected by other episodes, we see that it is not sufficient to reflect the real optimal pivot. Therefore, we first introduce the significance operator Sig based on the idea of pooling in convolution and then provide complete proof details.

Definition 5.1 (Rank function)**.** Given a sequence of random variables $\mathrm{RS} := \{X_1, X_2, \ldots, X_n\} \subset \mathcal{X}$, it is sorted with an ascending order to get the order statistics sequence $\mathrm{RS_O} := \{X_{(1)}, X_{(2)}, \ldots, X_{(n)}\}$, where $X_{(i)}$ represents the i-th smallest random variable. Here, we define the function $\mathrm{Rank_{RS}} : \mathcal{X} \to [n]$, where $[n] := \{1, 2, \ldots, n\}$, and $\mathrm{Rank_{RS}}(X_i)$ is the index of X_i in $\mathrm{RS_O}$.

Definition 5.2 (Significance operator)**.** Given K groups of random variables sequences $\{X_1^k, X_2^k, \ldots, X_{n_k}^k\}$, $k \in \{1, 2, \ldots, K\}$, we define $\bar{X}^k$ and $X_{(n_k)}^k$ as the mean statistic and the maximum statistic of k-th sequence $\{X_i^k\}_{i=1}^{n_k}$, respectively, i.e.,

$$\bar{X}^k = \frac{1}{n_k}\sum_{i=1}^{n_k} X_i^k, \quad X_{(n_k)}^k = \max\{X_1^k, X_2^k, \ldots, X_{n_k}^k\}, \quad k \in \{1, 2, \ldots, K\}.$$

For the mean statistics sequence $\mathrm{ES} := \{\bar{X}^1, \bar{X}^2, \ldots, \bar{X}^K\}$ and the maximum statistics sequence $\mathrm{MS} := \{X_{(n_1)}^1, X_{(n_2)}^2, \ldots, X_{(n_K)}^K\}$, we define the significance operator $\mathrm{Sig} : \mathcal{X} \to \mathcal{X}$ as

$$\mathrm{Sig}(\bar{X}^k) = \begin{cases} \bar{X}^k, & \mathrm{Rank_{ES}}(\bar{X}^k) = \mathrm{Rank_{MS}}(X_{(n_k)}^k), \\ X_{(n_k)}^k, & \mathrm{Rank_{ES}}(\bar{X}^k) \neq \mathrm{Rank_{MS}}(X_{(n_k)}^k). \end{cases} \tag{5.1}$$

Theorem 5.3. *For Models* 1 *and* 2*, if we set* α *in* (4.11) *as zero, the MCTS rule with the significance operator* Sig *will converge to the optimal pivot rule with probability at least* $1-\epsilon$*, as long as*

$$N_{\mathrm{explore}} \geqslant \frac{1}{\ln(1+\frac{1}{d_{\mathcal{A}}^*-1})} \ln\left(\frac{1}{1-\mathrm{e}^{\frac{1}{|P^*|}\ln(1-\epsilon)}}\right) \approx O\left(\ln\left(\frac{1}{\epsilon}\right)\right), \tag{5.2}$$

where P^* *is the optimal pivot path, and* $|P^*|$ *is its length.* $d_{\mathcal{A}}^* := \max_{s^* \in P^*} |\mathcal{A}_{s^*}|$ *represents the dimension of the maximum action space along this path.*

Proof. For Models 1 and 2, the reward is defined as the opposite number of the pivot iterations. Therefore, pivot iterations corresponding to the maximum reward are equivalent to the shortest pivot path.

Without loss of generality, we consider the state node s, which represents a particular simplex tableau. We denote the feasible action space in s by $\mathcal{A}_s := \{a_1, a_2, \ldots, a_{|\mathcal{A}_s|}\}$, where each action corresponds to a feasible pivot. The child nodes of s are $\{s_1, s_2, \ldots, s_{|\mathcal{A}_s|}\}$ and the transition functions are

$$\mathbb{P}(s_i \mid s, a_j) = \mathbb{I}_{\{i=j\}}, \quad \forall\, i, j \in \{1, 2, \ldots, |\mathcal{A}_s|\},$$

where $\mathbb{I}$ is the indicator function.

We denote by N_{explore} the number of explorations from s. The random variable N_i is the number of explorations from s to s_i and $\sum_{i=1}^{|\mathcal{A}_s|} N_i = N_{\text{explore}}$.

If we set α in (4.11) as zero, the set of selected actions is

$$\{i \mid Q(s, a_i) \geqslant E_{\text{soft}}\} = \left\{ i \,\middle|\, dQ(s, a_i) \geqslant \min_{a_j \in \mathcal{A}_s} Q(s, a_j) \right\} = \mathcal{A}_s, \tag{5.3}$$

which indicates that all the feasible pivots can be selected, i.e., $\mathbb{E}_\pi[N_i] > 0,\ \forall\, i \in \{1, 2, \ldots, |\mathcal{A}_s|\}$. In fact, our algorithm takes the exploration action by uniform policy from the set $\{i \mid Q(s, a_i) \geqslant E_{\text{soft}}\}$, and then we have

$$\begin{aligned}
\mathbb{E}_\pi[N_i] &= \sum_{i=1}^{|\mathcal{A}_s|} \mathbb{P}(s_i \mid s, a_j)\pi(a_j \mid s) N_{\text{explore}} \\
&= \pi(a_i \mid s) N_{\text{explore}} \\
&= \frac{N_{\text{explore}}}{|\{i \mid Q(s, a_i) \geqslant E_{\text{soft}}\}|} \\
&= \frac{1}{|\mathcal{A}_s|} N_{\text{explore}} > 0, \quad \forall\, i \in \{1, 2, \ldots, |\mathcal{A}_s|\}.
\end{aligned} \tag{5.4}$$

We can conclude that as long as

$$N_{\text{explore}} \geqslant \frac{1}{\ln(1 + \frac{1}{|\mathcal{A}_s| - 1})} \ln\left(\frac{1}{\delta_1}\right),$$

every child node can be accessed to with probability at least $1 - \delta_1$, i.e.,

$$\begin{aligned}
\mathbb{P}(\text{state } s_i \text{ visited}) &= \mathbb{P}(N_i > 0) \\
&= 1 - \mathbb{P}(N_i = 0) \\
&= 1 - \left(\frac{|\mathcal{A}_s| - 1}{|\mathcal{A}_s|}\right)^{N_{\text{explore}}} \\
&\geqslant 1 - \delta_1.
\end{aligned} \tag{5.5}$$

We denote by the random variable $R_j^{a_i}$ the reward of taking action a_i for the j-th time. Given $|\mathcal{A}_s|$ groups of independent identically distributed random variables sequences $\{R_j^{a_i}\}_{j=1}^{N_i}$, $i \in \{1, 2, \ldots, |\mathcal{A}_s|\}$, we apply the significance operator Sig in Definition 5.2 and we have $\{\text{Sig}(\bar{R}^{a_i})\}_{i=1}^{|\mathcal{A}_s|}$. Note that the reward random variables $\{R_j^{a_i}\}_{j=1}^{N_i}$ are independent and identically distributed, and we can apply the Wiener-Khinchin law of large numbers, i.e.,

$$\lim_{N_{\text{explore}} \to \infty} \bar{R}^{a_i} = \lim_{N_i \to \infty} \bar{R}^{a_i} = \lim_{N_i \to \infty} \frac{1}{N_i} \sum_{j=1}^{N_i} R_j^{a_i} = \mathbb{E}[R_j^{a_i}], \quad \text{a.s.} \tag{5.6}$$

Note that $Q(s, a_i)$ is the same as $\bar{R}^{a_i}$ with respect to the definition of Q.

For any pivot path $P=\{s_0^P,s_1^P,\dots,s_{|P|-1}^P\}$, we denote by $d_{\mathcal{A}}^P$ the dimension of the maximum action space along this path, i.e., $d_{\mathcal{A}}^P:=\max_{s^P\in P}|\mathcal{A}_{s^P}|$. By recursion, the MCTS rule can explore the pivot path P with probability at least $1-\delta_2$ when

$$N_{\text{explore}}\geqslant\frac{1}{\ln(1+\frac{1}{d_{\mathcal{A}}^P-1})}\ln\left(\frac{1}{1-\mathrm{e}^{\frac{1}{|P|}\ln(1-\delta_2)}}\right)\approx O\left(\frac{1}{\ln(1+\frac{1}{d_{\mathcal{A}}^P-1})}\ln\left(\frac{|P|}{\delta_2}\right)\right),$$

i.e.,

$$\begin{aligned}\mathbb{P}(\text{explore pivot path } P)&=\mathbb{P}\left(\bigcap_{s^P\in P}\{N_{s^P}>0\}\right)=\prod_{s^P\in P}\mathbb{P}(N_{s^P}>0)\\&=\prod_{s^P\in P}1-\left(\frac{|\mathcal{A}_{s^P}|-1}{|\mathcal{A}_{s^P}|}\right)^{N_{\text{explore}}}\geqslant\left(1-\left(\frac{d_{\mathcal{A}}^P-1}{d_{\mathcal{A}}^P}\right)^{N_{\text{explore}}}\right)^{|P|}\\&\geqslant 1-\delta_2.\end{aligned}\tag{5.7}$$

We have shown that our algorithm can explore each path P when N_{explore} is sufficiently large. As a result, we have

$$\lim_{N_{\text{explore}}\to\infty}R_{(N_i)}^{a_i}=\lim_{N_i\to\infty}R_{(N_i)}^{a_i}=\lim_{N_i\to\infty}\max\{R_1^{a_i},R_2^{a_i},\dots,R_{N_i}^{a_i}\}=R_*^{a_i},\quad\text{a.s.},\tag{5.8}$$

where $R_*^{a_i}$ is the maximum reward which can be attained by taking action a_i. Define $\text{ES}:=\{\bar{R}^{a_1},\bar{R}^{a_2},\dots,\bar{R}^{a_{|\mathcal{A}|}}\}$ and the maximum statistics sequence $\text{MS}:=\{R_{(N_1)}^{a_1},R_{(N_2)}^{a_2},\dots,R_{(N_{|\mathcal{A}|})}^{a_{|\mathcal{A}|}}\}$. Then we have

$$\lim_{N_{\text{explore}}\to\infty}\text{Sig}(\bar{R}^{a_i})=\begin{cases}\mathbb{E}[R_j^{a_i}],&\text{Rank}_{\text{ES}}(\bar{R}^{a_i})=\text{Rank}_{\text{MS}}(R_{(N_i)}^{a_i}),\\R_*^{a_i},&\text{Rank}_{\text{ES}}(\bar{R}^{a_i})\neq\text{Rank}_{\text{MS}}(R_{(N_i)}^{a_i}),\quad\text{a.s.}\end{cases}\tag{5.9}$$

We define the mapping $\text{Proj}:\text{ES}\cup\text{MS}\to\text{MS}$, where $\text{Proj}(\bar{R}^{a_i})=R_{(N_i)}^{a_i},\ \forall\,\bar{R}^{a_i}\in\text{ES}$ and $\text{Proj}(R_{(N_i)}^{a_i})=R_{(N_i)}^{a_i},\ \forall\,R_{(N_i)}^{a_i}\in\text{MS}$. Combining (5.9), we have

$$\lim_{N_{\text{explore}}\to\infty}\text{Proj}\circ\text{Sig}(\bar{R}^{a_i})=R_*^{a_i},\quad\text{a.s. }\forall\,i\in\{1,2,\dots,|\mathcal{A}_s|\}.\tag{5.10}$$

We take action $\hat{a}\in\arg\max_{a_i\in\mathcal{A}_s}\text{Proj}\circ\text{Sig}(\bar{R}^{a_i})$. According to the definition of Proj and Sig, we can access the child node that attains the optimal reward to execute the next iteration. We have proven that the MCTS rule can find the optimal action from a starting state s. In the following, we will analyze the complexity to extract the whole optimal pivot path.

We assume that the optimal pivot path is $P^*=\{s_0^*,s_1^*,\dots,s_{|P^*|-1}^*\}$ and denote by $d_{\mathcal{A}}^*$ the dimension of the maximum action space along this path, i.e., $d_{\mathcal{A}}^*:=\max_{s^*\in P^*}|\mathcal{A}_{s^*}|$.

By recursion, the MCTS rule will converge to the optimal pivot rule with probability at least $1-\epsilon$ when

$$N_{\text{explore}}\geqslant\frac{1}{\ln(1+\frac{1}{d_{\mathcal{A}}^*-1})}\ln\left(\frac{1}{1-\mathrm{e}^{\frac{1}{|P^*|}\ln(1-\epsilon)}}\right)\approx O\left(\frac{1}{\ln(1+\frac{1}{d_{\mathcal{A}}^*-1})}\ln\left(\frac{|P^*|}{\epsilon}\right)\right),$$

i.e.,

$$\begin{aligned}\mathbb{P}(\text{find optimal pivot path } P^*)&=\mathbb{P}\left(\bigcap_{s^*\in P^*}\{N_{s^*}>0\}\right)\\&=\prod_{s^*\in P^*}\mathbb{P}(N_{s^*}>0)\\&=\prod_{s^*\in P^*}1-\left(\frac{|\mathcal{A}_s|-1}{|\mathcal{A}_s|}\right)^{N_{\text{explore}}}\\&\geqslant\left(1-\left(\frac{d_{\mathcal{A}}-1}{d_{\mathcal{A}}}\right)^{N_{\text{explore}}}\right)^{|P^*|}\end{aligned}$$

$$\geqslant 1-\epsilon. \tag{5.11}$$

Note that $d_{\mathcal{A}}$ does not exceed the number of feasible entering basis variables, i.e., $d_{\mathcal{A}} \leqslant n-m$ and we also prove that $|P^*|$ is at most $\min(m, n-m)$ in Theorem 5.5, which indicates that $d_{\mathcal{A}}$ and $|P^*|$ are both linear according to the dimension of input. Therefore, we have

$$N_{\text{explore}} \approx O\bigg(\frac{1}{\ln(1+\frac{1}{d_{\mathcal{A}}-1})}\ln\bigg(\frac{|P^*|}{\epsilon}\bigg)\bigg) \approx O\bigg(\ln\bigg(\frac{1}{\epsilon}\bigg)\bigg).$$

In summary, the MCTS rule will converge to the optimal pivot rule with the highest reward in the models 1 and 2 when N_{explore} is sufficiently large. □

The above theorem concludes that the MCTS rule with the significance operator converges to optimality under the condition of $\alpha = 0$. Additionally, the proof of convergence to the optimal strategy is given in the literature [20] for the case of $\alpha = 1$. It is also feasible to substitute them into the framework of our proof based on the upper and lower bounds of explorations of each action given in the literature [20]. For other α values, we conducted an ablation study in the experimental section for verification.

5.2 Completeness of multiple pivot paths

The proposed MCTS rule is a random algorithm. Multiple executions may find different optimal pivot paths. Theorem 5.4 gives a theoretical guarantee based on the Wiener-Khinchin law of large numbers. It is proved that the MCTS rule can find all the optimal pivot paths when executions are sufficient.

Theorem 5.4. *For Models* 1 *and* 2*, if we set* α *in* (4.11) *as zero, the MCTS rule with significance operator Sig can find all the optimal pivot paths provided that algorithm execution times* N_{exe} *are sufficiently large.*

Proof. We use the same notations as the ones in the proof of Theorem 5.3. Set all the optimal actions of the current pivot as $\mathcal{A}^* = \{a_1^*, a_2^*, \ldots, a_{|\mathcal{A}^*|}^*\}$. When $|\mathcal{A}^*| = 1$, this theorem holds by Theorem 5.3. We consider $|\mathcal{A}^*| \geqslant 2$ in the following proof. According to (5.4),

$$\lim_{N_{\text{explore}}\to\infty} N_{a^*} = \lim_{N_{\text{explore}}\to\infty} \frac{1}{|\mathcal{A}|} N_{\text{explore}} = \infty, \quad \forall\, a^* \in \mathcal{A}^*, \tag{5.12}$$

where $\mathcal{A}$ represents the total action space, and N_{a^*} represents the number of explorations of the optimal action a^*.

According to (5.10), we have

$$\lim_{N_{\text{explore}}\to\infty} \text{Proj}\circ\text{Sig}(\bar{R}^{a^*}) = R_*^{a^*} = \max_{i,j} R_j^{a_i} \quad \text{a.s. } \forall\, a^* \in \mathcal{A}^*. \tag{5.13}$$

Therefore, we select action from $\mathcal{A}^*$ by uniform policy, i.e., $\mathbb{P}(a_i^*) = \frac{1}{|\mathcal{A}^*|}$. Then we have that when

$$N_{\text{exe}} \geqslant \frac{1}{\ln(\frac{|\mathcal{A}^*|}{|\mathcal{A}^*|-1})}\ln\bigg(\frac{|\mathcal{A}^*|}{\epsilon}\bigg),$$

the MCTS rule can find all $a^* \in \mathcal{A}^*$, i.e.,

$$\begin{aligned}
\mathbb{P}(\text{find all } a^* \in \mathcal{A}^*) &= 1 - \mathbb{P}\bigg(\bigcup_{i=1}^{|\mathcal{A}^*|}\{\text{not find } a_i^*\}\bigg) \\
&\geqslant 1 - \sum_{i=1}^{|\mathcal{A}^*|}\mathbb{P}(\text{not find } a_i^*) \\
&= 1 - |\mathcal{A}^*|\bigg(\frac{|\mathcal{A}^*|-1}{|\mathcal{A}^*|}\bigg)^{N_{\text{exe}}} \\
&\geqslant 1-\epsilon.
\end{aligned} \tag{5.14}$$

It indicates that when the number of algorithm executions N_{exe} approaches infinity, each $a^* \in \mathcal{A}^*$ can be found. Then repeating the above process, we see that the MCTS rule can find all the optimal pivot paths. □

5.3 Complexity of the optimal pivot

Theorem 5.5 proves that the MCTS rule can find polynomial pivot iterations when the number of vertices in the feasible region is C_n^m.

Theorem 5.5. *For the standard form of the simplex method for linear programming as* (2.1)*,*

$$P = \{x \in \mathbb{R}^n \mid Ax = b, x \geqslant 0\}$$

represents the feasible region, where $A \in \mathbb{R}^{m\times n}$ *and* $\text{rank}(A) = m$*. Suppose that the number of feasible vertices of* P *is* C_n^m*. Then the shortest distance* (*the minimum hops*) *between any two feasible vertices of* P *is* $\min(m, n-m)$*.*

Proof. For the convenience of proving, we first convert the simplex of the feasible region into the SimplexPseudoTree structure proposed in Section 3. In this way, each vertex only appears once in different layers, so the number of vertices in each layer adds up to the total number of vertices N. Compared with the root of the SimplexPseudoTree, the nodes of layer i are obtained from the root through the i pivot iterations, i.e., $C_m^i C_{n-m}^i$. We have

$$\sum_{i=0}^{\mathrm{T}} C_m^i C_{n-m}^i = N = C_n^m, \tag{5.15}$$

where T is the layers of the SimplexPseudoTree and also the longest distance to the root. We discuss it in two cases, based on the Vandermonde's identity of the combination number

$$\sum_{i=0}^{l} C_a^i C_b^{l-i} = C_{a+b}^l. \tag{5.16}$$

When $n - m \leqslant m$, we have

$$\sum_{i=0}^{k} C_m^i C_{n-m}^{k-i} = C_n^k. \tag{5.17}$$

Let $k = n - m$. Then

$$\sum_{i=0}^{n-m} C_m^i C_{n-m}^{n-m-i} = \sum_{i=0}^{n-m} C_m^i C_{n-m}^i = C_n^{n-m} = C_n^m, \tag{5.18}$$

i.e., $T = n - m$. When $m < n - m$, we have

$$\sum_{i=0}^{k} C_m^{k-i} C_{n-m}^i = C_n^k. \tag{5.19}$$

Let $k = m$. Then

$$\sum_{i=0}^{m} C_m^{m-i} C_{n-m}^i = \sum_{i=0}^{m} C_m^i C_{n-m}^i = C_n^m, \tag{5.20}$$

i.e., $T = m$. In this way, $T = \min(m, n-m)$, which is a polynomial of n. Additionally, T can represent the maximum hops of all the shortest paths between any two vertices in the feasible region. Therefore, the shortest pivot iterations starting from any initial point in the feasible region is the polynomial of the number of variables when the number of vertices in the feasible region is C_n^m. □

From the perspective of the combination number, Theorem 5.5 proves that the shortest distance between any two vertices of the feasible region is $\min(m, n-m)$ when the number of vertices in the feasible region is C_n^m. This conclusion is also meaningful from the geometric perspective of the simplex method. Specifically, the number of vertices in the feasible region is C_n^m means that any m columns of the constraint matrix corresponds to a basis matrix of the simplex method. Therefore, for any two basis matrices B_1 and B_2, there can be at most $\min(m, n-m)$ different columns. We only need to exchange different columns, respectively. In this way, the initial vertex B_1 is converted to B_2 through $\min(m, n-m)$ pivot iterations. In other words, the optimal pivot iterations needed for any linear objective function are $\min(m, n-m)$ at most under the conditions of the above theorems.

We reveal that the optimal pivot of the simplex method is polynomial when the number of vertices in the feasible region is C_n^m from the perspective of theory and geometry, respectively. Furthermore, the proposed MCTS rule can find the polynomial pivot iterations when the number of vertices in the feasible region is C_n^m.

Corollary 5.6. *When the number of vertices in the feasible region is C_n^m, the MCTS rule ensures that the number of pivot iterations becomes the polynomial of the number of variables.*

6 Experiment

6.1 Datasets and experiment setting

Datasets. We conduct experiments using the NETLIB[1)] benchmark [4] and random instances. The method for generating random instances is as follows. First, we write the equivalent form of (2.1), i.e.,

$$\begin{aligned} &\min c^{\mathrm{T}}x + \mathbf{0}^{\mathrm{T}}x_0 \\ &\text{s.t. } Ax + Ix_0 = b, \\ &\qquad x \geqslant 0, \quad x_0 \geqslant 0, \end{aligned} \tag{6.1}$$

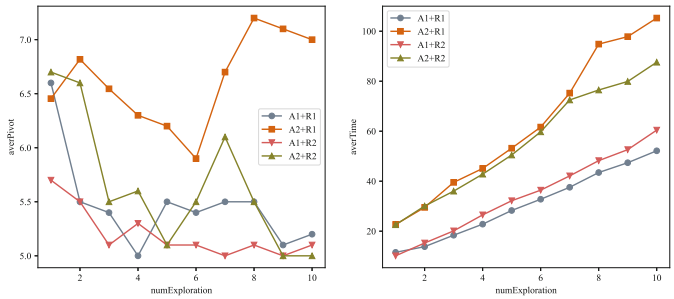
where $x_0 \in \mathbb{R}^m$, $\mathbf{0} \in \mathbb{R}^m$ is a zero vector, and $I \in \mathbb{R}^{m\times m}$ is an identity matrix. Thus, we provide a setting for the initial feasible basis $B_0 = I$, where n basis variables correspond to n columns of the constraint matrix A at the right end. Each component of A, b and c is a random number of uniform distributions of $[0, 1000)$. For the constraint matrices of non-square matrices, the rows and columns are obtained from a uniform distribution of $[0, 800)$.

Implementation details. In the experiment comparing the performance of the four models, we set the explorations from one to ten times the columns of the constraint matrix. Subsequently, we set the explorations to one and six times of columns because six times can provide stable exploration results, with number one having the shortest exploration time. We then solve each problem five times and obtain the optimal pivot iterations. In addition, we set $C = 1/\sqrt{2}$ in (4.10) and dynamically adjust α according to the solution state for (4.11). In particular, we set $\alpha = 0.3$ for $N_{\text{explore}} < 0.1 \times \text{ColNum}$ and $\alpha = 1$ for other cases, where ColNum represents the number of columns of the constraint matrix. In addition, we use the first stage of the two-stage algorithm of the simplex method to find the initial feasible basis for the NETLIB instances.

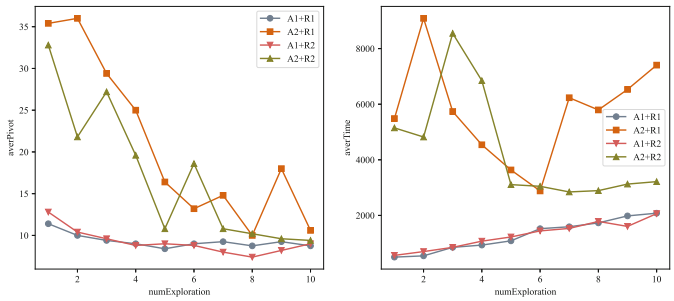
6.2 Estimation of four RL models

In this subsection, we compare the quality of the four RL models proposed in Subsection 4.2 for the simplex method. We conduct thorough experiments on five representative instances, as illustrated in Figure 4. Each point represents the average pivot iterations obtained by solving the problem five times under the current conditions. This step reduces the influence of randomness. In addition, we set the upper limit of pivot iterations as 10 to reduce unnecessary time without affecting the experimental results.

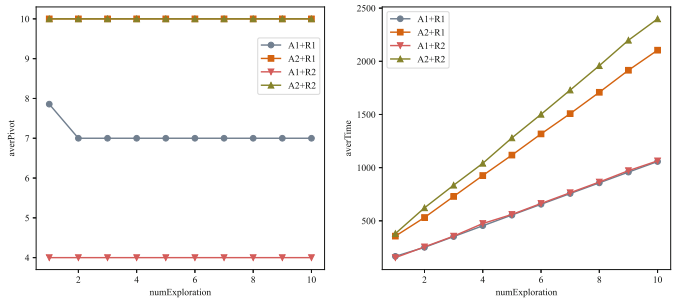
[1)] http://www.netlib.org/lp/data/index.html



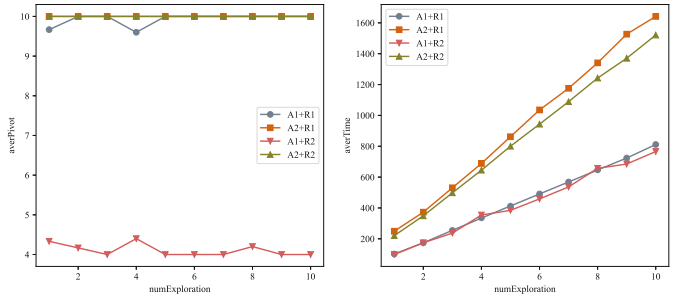
(a) rand 50 × 50



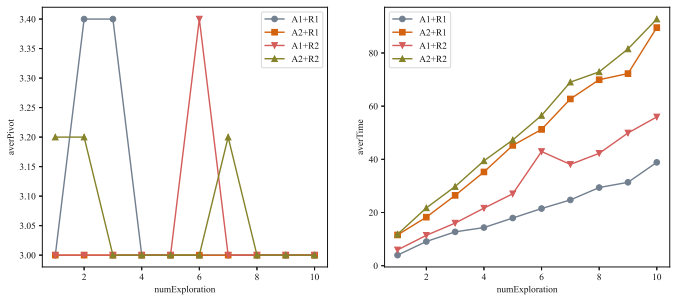
(b) rand 300 × 300



(c) rand 232 × 504



(d) rand 332 × 187



(e) SC50A

Figure 4 (Color online) Model comparison figure on five representative instances: rand 50 × 50, rand 232 × 504, rand 332 × 187 and SC50A. The X-axis represents the explorations, which are multiples of columns of the constraint matrix A. The Y-axis of the left figure represents the average pivot iterations. The Y-axis of the right figure represents the average solution time

The left part of Figure 4 shows the relationship between pivot iterations and explorations. We found that the number of pivot iterations obtained by A1 + R2 is the best compared with the other three models. The right part of Figure 4 shows the variation in the average solution time with the explorations. It can be seen that with the increase in explorations, the average solution time of each model also increases, which is caused by the exploration cost of the larger number of explorations. We also conclude that the time performance of A1 + R1 and A1 + R2 is better than those of A2 + R1 and A2 + R2. Consequently, combined with the performance of the number of pivot iterations, A1 + R2 has certain advantages over all the other models.

6.3 Comparison with the solver and classical pivot rules

In this subsection, we compare the MCTS rule with Python's linprog function and the other classic pivot rules. Table 2 lists the results for the randomly generated square constraint matrices, Table 3 presents the results for the general constraint matrices, and Table 4 shows the experimental results for the NETLIB.

6.3.1 *Comparison results on random instances*

The instances in Table 2 are random square matrices. However, Table 3 presents a comparison of the random rows and columns of the constraint matrices. Red indicates the best results, and blue indicates suboptimal results. The first two columns represent rows and columns of the constraint matrix.

Table 2 (Color online) Comparison between classical pivot rules and the MCTS rule on random square constraint matrices. Red indicates the best results, and blue indicates suboptimal results

row	col	Linprog	Dantzig [7]	Bland [2]	Steepest [10]	Greatest [21]	Devex [28]	MCTS	MTime(s)
		341	18	63	14	10	21	**7**	425
		361	14	56	14	13	22	**7**	409
		331	14	49	12	8	14	**6**	274
		373	12	56	10	8	16	**6**	287
300	300	318	16	28	13	7	15	**6**	258
		321	10	14	8	**4**	14	**4**	128
		324	17	67	14	7	17	**5**	211
		324	10	40	7	6	14	**5**	130
		329	10	55	13	**5**	18	**5**	238
		492	21	37	12	12	29	**7**	356
		475	22	113	12	**9**	28	**9**	685
		541	21	116	19	15	32	**11**	793
		449	19	73	15	8	23	**6**	355
400	400	450	19	74	15	11	30	**7**	510
		473	16	70	13	11	15	**6**	317
		438	15	55	9	**6**	24	**6**	275
		480	15	73	12	9	24	**8**	686
		542	25	87	17	10	32	**8**	553
		612	21	79	14	**10**	32	**10**	833
		739	30	205	20	20	38	**15**	2633
		688	22	93	18	12	52	**11**	1064
		559	23	74	13	12	24	**8**	678
500	500	606	26	117	13	12	28	**8**	664
		528	22	44	11	10	24	**6**	506
		610	22	121	15	10	27	**9**	966
		598	21	58	18	**10**	25	**10**	1395
		607	26	187	17	14	36	**11**	1383

Table 3 (Color online) Comparison between classical pivot rules and the MCTS rule on the general constraint matrices. Red indicates the best results, and blue indicates suboptimal results

row	col	Linprog	Dantzig [7]	Bland [2]	Steepest [10]	Greatest [21]	Devex [28]	MCTS	MTime(s)
117	123	187	8	30	7	5	8	**4**	35
139	199	191	9	24	7	**5**	10	**5**	59
399	324	571	25	1,000+	14	17	44	**9**	1,107
243	245	267	10	30	8	**5**	16	**5**	168
471	311	529	8	46	7	5	9	**4**	191
209	318	324	9	64	9	9	9	**7**	288
374	293	478	11	22	8	7	16	**6**	351
470	766	765	485	64	10	8	21	**6**	1,133
49	647	105	1,000+	1,000+	7	**4**	21	**4**	427
207	327	259	6	34	6	**4**	11	**4**	145
151	419	260	1,000+	36	13	11	22	**7**	735
782	436	871	11	43	12	**5**	25	**5**	917
363	699	418	8	41	7	5	23	**4**	555
587	382	635	8	33	12	**4**	9	**4**	416
565	761	594	6	10	5	**2**	8	**2**	214
232	504	263	6	110	6	**4**	7	**4**	152
215	202	358	18	58	13	8	22	**7**	250
278	525	373	8	52	7	10	14	**5**	648
143	58	184	6	15	6	4	6	**3**	26
323	286	378	5	17	5	**4**	8	**4**	191
133	242	192	18	1,000+	9	7	21	**5**	282
96	482	179	9	46	8	**5**	11	**5**	397
203	200	223	8	12	6	**3**	7	**3**	84
678	154	757	10	14	7	**3**	8	**3**	267
739	142	866	7	18	7	8	11	**5**	520
739	730	841	11	33	10	**4**	7	**4**	888
240	234	268	7	20	7	5	10	**4**	103
767	158	885	9	1,000+	7	7	19	**5**	623
730	467	773	6	21	7	5	6	**3**	584
434	258	498	7	30	7	5	23	**4**	231
625	521	739	9	26	10	9	19	**5**	732
332	187	352	9	24	9	**4**	13	**4**	102
561	628	585	7	13	9	**4**	16	**4**	471
587	774	610	5	17	5	4	13	**3**	302
84	108	113	15	12	7	5	14	**4**	35

Table 4 (Color online) Comparison between classical pivot rules and the MCTS rule on NETLIB. Red indicates the best results, and blue indicates suboptimal results

Problem	Linprog	Dantzig [7]	Bland [2]	Steepest [10]	Greatest [21]	MCTS	MTime(s)
AFIRO	27	**0**	**0**	**0**	**0**	**0**	0
ADLITTLE	152	1,000+	223	60	53	**26**	836
BLEND	178	29	88	35	21	**15**	118
SC50A	56	5	8	5	4	**3**	4
SC50B	59	**6**	9	7	**6**	**6**	7
SC105	135	14	26	10	16	**7**	69
SCAGR7	228	57	101	43	54	**40**	1,488
SHARE2B	260	47	172	38	29	**19**	612

The third column to the ninth column respectively represent the minimum number of pivot iterations of Python's linprog function, the Dantzig rule [7], the Bland rule [2], the steepest-edge rule [10], the greatest improvement rule [21], the devex rule [28] and our MCTS rule. The initial feasible basis of Python's linprog function is determined by its own setting, while others are based on the method proposed in Subsection 5.1. The last column is execution time of the MCTS rule.

Tables 2 and 3 show that the number of pivot iterations obtained by the MCTS rule is superior to that obtained by the classical pivot rules in all the general instances. In terms of square instances, the pivot iterations found by the MCTS rule were only 54.55% of the minimum iterations of other popular pivot rules. In addition, for other random dimension instances, our result was only 55.56% of the others' best results.

We conclude that the results of the MCTS rule are not limited to input instances. It performed best on all the randomly generated problems because the MCTS rule selects the entering basis variable by exploring and evaluating the entire feasible action space rather than providing a fixed rule based on certain specific features. However, the number of pivot iterations obtained by other classical methods cannot exceed the result of the MCTS rule. We first proposed an efficient and generalized method for determining the minimum number of pivot iterations of the simplex method. Furthermore, for the first time, this method provides the best label design for pivot rules based on supervised learning.

6.3.2 *Comparison results on NETLIB*

Table 4 presents the comparison results of the MCTS rule with other classical pivot rules on NETLIB. Red indicates the best results and blue indicates suboptimal results. The first column represents the name of instances. The second to seventh columns are respectively the minimum pivot iterations of Python's linprog function, the Dantzig rule [7], the Bland rule [2], the steepest-edge rule [10], the greatest improvement rule [21] and our MCTS rule. The initial feasible basis of Python's linprog function is determined by its own setting, while others are based on the method proposed in Subsection 5.1. The last column is execution time of the MCTS rule.

It is easy to conclude that the MCTS rule yields the least number of pivot iterations far less than others on all the instances listed, especially for the problem ADLITTLE. The greatest improvement rule has the least number of pivot iterations among the classical rules. In contrast, our method achieves only 49.06% of its pivot iterations. Moreover, compared with the Dantzig rule, the MCTS rule gets less than 2.6% of its pivot iterations.

Although the solution time of the MCTS rule is longer than other algorithms, this is still consistent with our contribution. We aim to determine the optimal pivot iterations and all the corresponding pivot paths for the input instance. Furthermore, we provide the best supervision labels for the simplex method. Additionally, in Section 7, we present two methods from the perspective of CPU and GPU to improve the efficiency of collecting supervision labels. Thus, this method is more applicable to super large-scale problems.

6.4 Comparisons with all the current pivot rules based on ML

In this subsection, we compare our method with two recently proposed machine-learning methods. It is found that the minimum pivot iteration MCTS rule is much better than the other two methods.

In the first method [35], the supervised learning method DeepSimplex [35] performs better than the unsupervised learning method. Therefore, we only compare DeepSimplex [35], which gets the best performance of the Dantzig rule and the steepest-edge rule as the supervised signal. Thus, the results do not exceed those of these two methods. As Tables 2 and 3 show, the worst number of pivot iterations of the MCTS rule in all the instances is 80% of Dantzig's. In the best case, the number of pivot iterations is less than 0.4% of Dantzig's. Compared with the steepest-edge rule, the worst pivot iterations of the MCTS rule in all the instances is only 77.78%. Moreover, in the best case, the pivot iteration is only 27.27% of the steepest-edge rule. Additionally, Table 4 shows that the MCTS rule is significantly better than the Dantzig and steepest-edge rules on NETLIB. Therefore, we conclude that the MCTS rule can obtain better pivot iterations than DeepSimplex [35].

The second method [1] is learned in a supervised manner with the best label of Dantzig's rule, the hybrid (DOcplex's default) rule, the greatest improvement rule, the steepest-edge rule and the devex rule. Especially, it is remarkable that the MCTS rule provided more effective labels with minimum pivot iterations. From the experimental results of their article [1], we know that the best result of the average number of pivot iterations is 99.54% of the number of the steepest-edge rule. In contrast, the performance of the MCTS rule on the worst instance is 77.78% of the steepest-edge rule, as shown in Tables 2 and 3. Moreover, the best result obtained by the MCTS rule is only 33.33% of the pivot iterations of the steepest-edge rule. Furthermore, in NETLIB, the pivot iterations yielded by the MCTS rule can even reach 43.33% of the steepest-edge rule, which is far less than 99.54%.

6.5 Findings of multiple pivot paths

Providing multiple pivot paths for the simplex method is the result of taking advantage of the randomness of the MCTS rule. Table 5 shows multiple optimal pivoting paths found for five representative problems, which cannot be yielded by previous methods. Different pivoting paths are the optimal pivoting sequences with minimum pivot iterations. Additionally, Figure 5 shows the relationship between the number of different pivot paths found and the algorithm executions on several representative instances mentioned above. We use highlighted points to mark the executions of a newly found pivot path. It can be concluded that under the initial executions, the proposed MCTS rule can find some paths. Furthermore, with the increase of algorithm executions, the number of found paths also increases.

6.6 Ablation study

We compare the influence of different C and α values on the average pivot iterations for several representative instances. Each point is the average result of executing the algorithm five times.

Table 5 Pivoting paths with the minimum number of pivot iterations on five representative instances. The pivoting path is an ordered sequence of entering basis variables

Problem	Index	Pivot paths	Pivot iterations	Objective value
SC50B	1	[1, 24, 36, 12, 23, 35]	6	70.000
	2	[1, 35, 12, 24, 36, 23]	6	70.000
	3	[1, 35, 23, 36, 12, 24]	6	70.000
	4	[1, 35, 24, 12, 36, 23]	6	70.000
	5	[12, 35, 1, 24, 36, 23]	6	70.000
	6	[23, 35, 1, 36, 12, 24]	6	70.000
rand 232×504	1	[52, 55, 358, 220]	4	22.287
	2	[55, 52, 358, 220]	4	22.287
rand 434×258	1	[14, 155, 223, 150]	4	5.509
	2	[14, 155, 150, 223]	4	5.509
	3	[150, 155, 223, 14]	4	5.509
	4	[155, 150, 223, 14]	4	5.509
	5	[155, 223, 150, 14]	4	5.509
rand 50×50	1	[1, 48, 28, 19, 21]	5	3.482
	2	[28, 48, 1, 19, 21]	5	3.482
	3	[48, 28, 1, 19, 21]	5	3.482
rand 300×300	1	[26, 294, 42, 143, 56, 116, 263]	7	5.915
	2	[26, 294, 42, 116, 56, 143, 263]	7	5.915
	3	[56, 26, 42, 294, 263, 116, 143]	7	5.915
	4	[143, 26, 42, 294, 116, 56, 263]	7	5.915
	5	[143, 26, 56, 294, 42, 116, 263]	7	5.915
	6	[294, 26, 42, 143, 56, 116, 263]	7	5.915

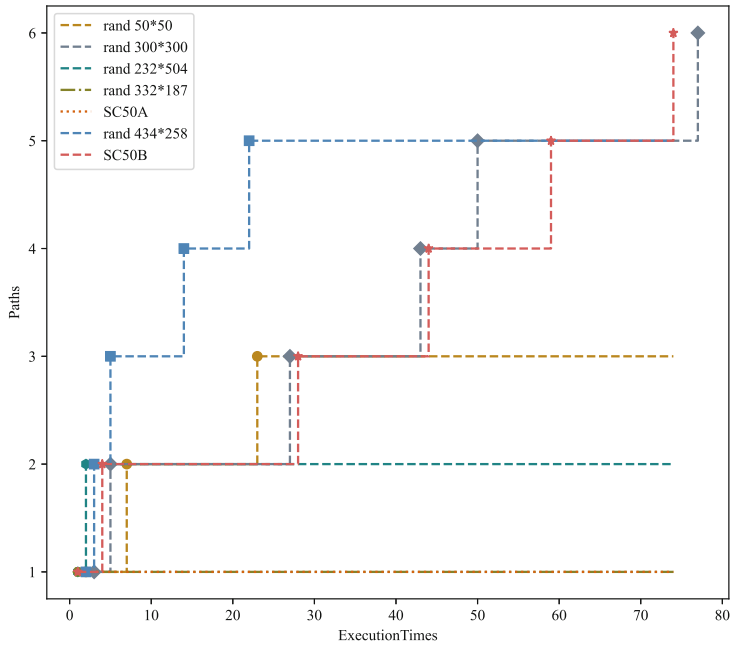


Figure 5 (Color online) Multiple paths found vary with the number of algorithm executions. The X-axis represents the number of algorithm executions, and the Y-axis represents the different pivot paths currently found

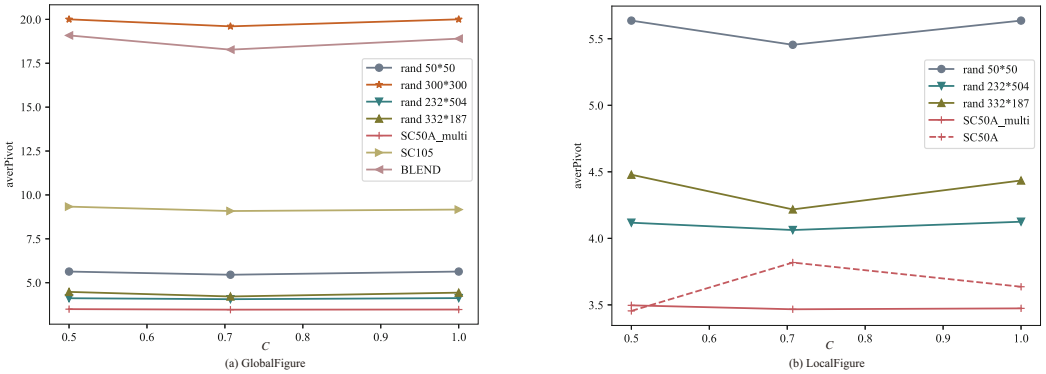


Figure 6 (Color online) Relationship between average pivot iterations and parameter C. The X-axis represents the value of parameter C, and the Y-axis represents the average pivot iterations found. (a) is the overall effect of several representative instances. (b) is an enlarged representation of the bottom four lines in (a). The SC50A_multi represents an increase in the executions of SC50A by five times than before to calculate the average value

As the GlobalFigure in Figure 6 indicates, the empirical value of C can lead to the least pivot iterations, except for SC50A. However, we conduct in-depth experiments and find that SC50A can provide fewer pivot iterations when the total number of executions increases. Therefore, we believe that the empirical value of C is reasonable for the MCTS rule in terms of overall performance.

Figure 7 shows the relationship between α and the pivot iterations for different initial explorations. Formula (4.12) aims to relax the max operator owing to the imprecisely estimated value in the early stage of exploration. Therefore, we conduct sufficient experiments on explorations of the 1-, 0.5-, 0.4-, 0.3-, 0.2- and 0.1-times columns of the constraint matrix of the instances to be solved. Formula (4.12) is effective when explorations are less than or equal to 0.1 times columns. Furthermore, when the number of explorations is 0.1 times columns, α achieves a consistently good effect with a value of 0.3. Therefore, we set dynamically adjusted α for the MCTS rule. When the number of explorations is less than or equal to 0.1 times columns, α is set to 0.3. For the other cases, α was set to 1.

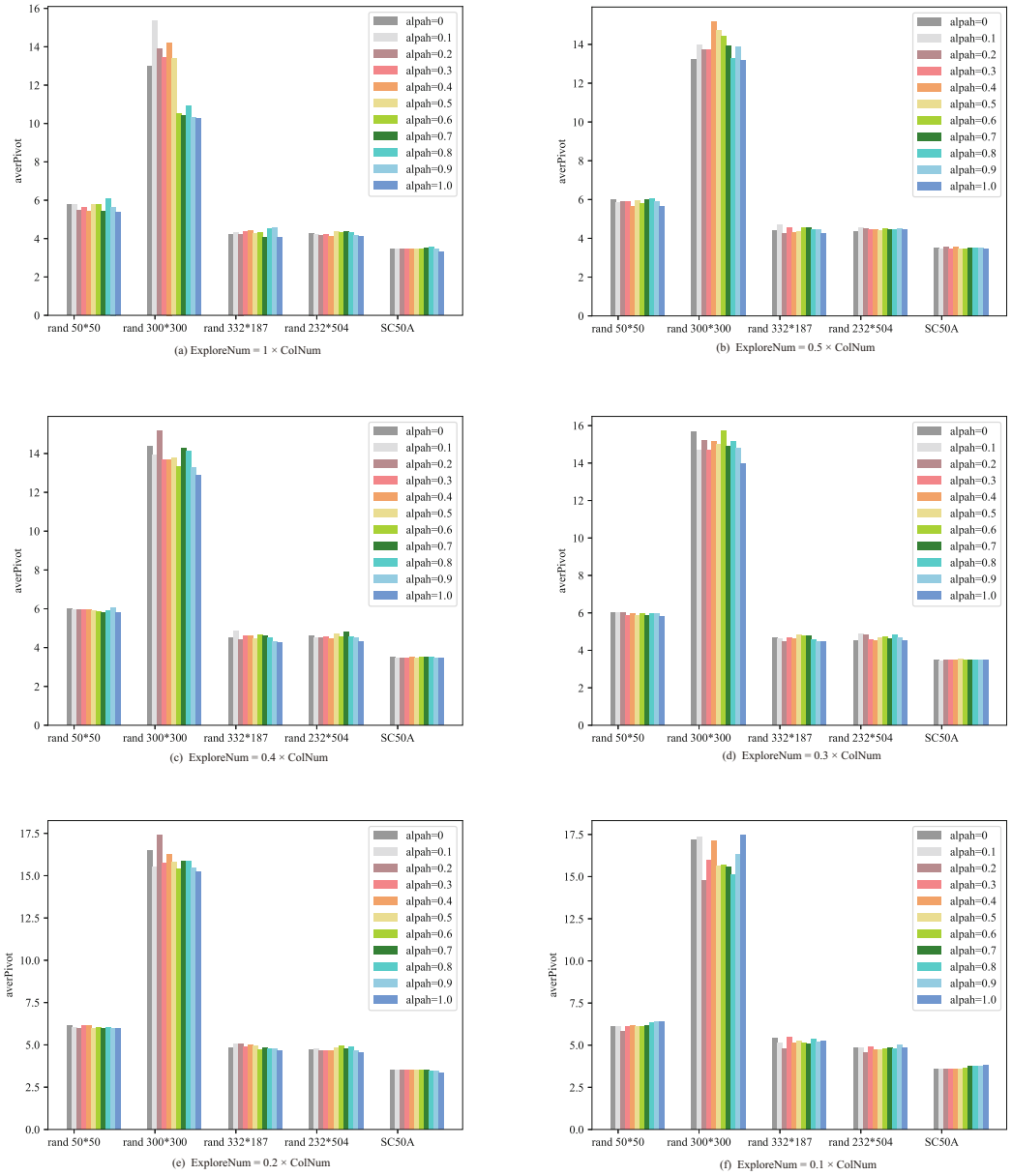


Figure 7 (Color online) Relationship between the average number of pivot iterations and α under different initial explorations. The X-axis represents different problems, and the Y-axis represents the average pivot iterations

7 Conclusion

Based on the proposed SimplexPseudoTree structure and the reinforcement learning model, the MCTS rule can determine all the shortest pivot paths of the simplex method. In addition, our method provides the best supervised label-setting method for the simplex method based on supervised learning. The MCTS rule can evaluate the pros and cons of entering basis variables individually, significantly reducing the exploration space for combinatorial optimization problems. Therefore, the proposed method can find the minimum pivot iterations and provide a method to find multiple shortest pivot paths. This idea can also be used to find multiple optimal solutions for other combinatorial optimization problems that can be modeled as imitative tree structures. Furthermore, we prove that the MCTS rule can find polynomial pivot iterations when the number of vertices in the feasible region is C_n^m. The complete theory and comprehensive experiments demonstrate that the MCTS rule can find multiple optimal pivot rules.

8 Further work

The multiple pivot paths determined by the MCTS rule can be used to construct flexible labels for the simplex method. Therefore, we can design the supervised learning method of the optimal pivot rule for the simplex method of linear programming. Furthermore, deep learning can be used to construct more efficient and time-effective pivot rules. In this manner, we can improve the redundancy of the traditional pivot rule and the low time efficiency of the MCTS rule.

Additionally, we introduce two implementation techniques to improve the time efficiency of the proposed method. These techniques are introduced from the perspectives of the CPU and GPU. Both methods are designed to solve the time-consuming process of sequential execution in the exploration stage. First, rewriting CUDA allows several explorations to be performed simultaneously. Thus, the time efficiency is reduced by dozens or even hundreds of times. In addition to using GPU computing by rewriting CUDA, the implementation of multithreading provides a method to improve the time efficiency of CPU devices. N_{explore} explorations can be divided into $N_{\text{explore}}/N_{\text{threads}}$ groups by grouping, where N_{explore} represents the number of explorations, and N_{threads} represents the number of threads of the computer. In each group, all the threads simultaneously perform exploration at the same time. The number of explorations is a multiple of computer threads; however, the time is the same as that of a single exploration. The reduction in time efficiency is directly proportional to the number of threads in the computer.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2021YFA1000403), National Natural Science Foundation of China (Grant No. 11991022), the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDA27000000) and the Fundamental Research Funds for the Central Universities.

References

1 Adham I, De Loera J, Zhang Z. (Machine) Learning to improve the empirical performance of discrete algorithms. arXiv:2109.14271, 2021
2 Bland R G. New finite pivoting rules for the simplex method. Math Oper Res, 1977, 2: 103–107
3 Browne C B, Powley E, Whitehouse D, et al. A survey of Monte Carlo tree search methods. IEEE Trans Comput Intell AI Games, 2012, 4: 1–43
4 Browne S, Dongarra J, Grosse E, et al. The Netlib mathematical software repository. D-lib Magazine, http://www.dlib.org/dlib/september95/netlib/09browne.html, 1995
5 Chen Y Q, Chen Y, Lee C K, et al. Optimizing quantum annealing schedules with Monte Carlo tree search enhanced with neural networks. Nature Mach Intell, 2022, 4: 269–278
6 Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search. In: International Conference on Computers and Games. Berlin-Heidelberg: Springer, 2006, 72–83
7 Dantzig G. Linear Programming and Extensions. Princeton: Princeton University Press, 1963
8 Ding M, Han C, Guo T. High generalization performance structured self-attention model for knapsack problem. Discrete Math Algorithms Appl, 2021, 13: 2150076
9 Fischetti M, Fraccaro M. Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. Comput Oper Res, 2019, 106: 289–297
10 Forrest J J, Goldfarb D. Steepest-edge simplex algorithms for linear programming. Math Program, 1992, 57: 341–374
11 Gama R, Fernandes H L. A reinforcement learning approach to the orienteering problem with time windows. Comput Oper Res, 2021, 133: 105357
12 Goffinet J, Ramanujan R. Monte-Carlo tree search for the maximum satisfiability problem. In: Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, vol. 9892. Berlin: Springer, 2016, 251–267
13 Goldfarb D, Reid J K. A practicable steepest-edge simplex algorithm. Math Program, 1977, 12: 361–371
14 Guo T, Han C, Tang S. Machine Learning Methods for Combinatorial Optimization (in Chinese). Beijing: Kexue Chubanshe (Science Press), 2019
15 Guo T, Han C, Tang S, et al. Solving combinatorial problems with machine learning methods. In: Nonlinear Combinatorial Optimization. Springer Optimization and Its Applications, vol. 147. Cham: Springer, 2019, 207–229
16 Harris P M J. Pivot selection methods of the Devex LP code. Math Program, 1973, 5: 1–28

17 Hildebrandt F D, Thomas B W, Ulmer M W. Opportunities for reinforcement learning in stochastic dynamic vehicle routing. Comput Oper Res, 2023, 150: 106071
18 Keszocze O, Schmitz K, Schloeter J, et al. Improving sat solving using Monte Carlo tree search-based clause learning. In: Advanced Boolean Techniques. Cham: Springer, 2020, 107–133
19 Kiarostami M S, Daneshvaramoli M, Khalaj Monfared S, et al. On using Monte-Carlo tree search to solve puzzles. In: Proceedings of the 2021 7th International Conference on Computer Technology Applications. New York: ACM, 2021, 18–26
20 Kocsis L, Szepesvári C. Bandit based monte-carlo planning. In: European Conference on Machine Learning. Berlin-Heidelberg: Springer, 2006, 282–293
21 Li C. Study on using the greatest improvement pivot rule of simplex method to the Klee and Minty example. In: International Conference on High Performance Networking, Computing and Communication Systems. Berlin-Heidelberg: Springer, 2011, 431–438
22 Liang X, Guo Z-C, Wang L, et al. Nearly optimal stochastic approximation for online principal subspace estimation. Sci China Math, 2023, 66: 1087–1122
23 Louati H, Bechikh S, Louati A, et al. Deep convolutional neural network architecture design as a bi-level optimization problem. Neurocomputing, 2021, 439: 44–62
24 Mihaljević B, Bielza C, Larrañaga P. Bayesian networks for interpretable machine learning and optimization. Neurocomputing, 2021, 456: 648–665
25 Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. arXiv:1312.5602, 2013
26 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533
27 Nguyen M A, Sano K, Tran V T. A Monte Carlo tree search for traveling salesman problem with drone. Asian Trans Stud, 2020, 6: 100028
28 Pan P Q. A largest-distance pivot rule for the simplex algorithm. European J Oper Res, 2008, 187: 393–402
29 Perez D, Rohlfshagen P, Lucas S M. Monte-Carlo tree search for the physical travelling salesman problem. In: Applications of Evolutionary Computation. EvoApplications 2012. Lecture Notes in Computer Science, vol. 7248. Berlin-Heidelberg: Springer, 2012, 255–264
30 Sabar N R, Kendall G. Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems. Inform Sci, 2015, 314: 225–239
31 Schloeter J. A Monte Carlo tree search based conflict-driven clause learning SAT solver. In: Lecture Notes in Informatics (LNI). Bonn: Gesellschaft für Informatik, 2017, 2549–2560
32 Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529: 484–489
33 Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. Nature, 2017, 550: 354–359
34 Sun Z, Benlic U, Li M, et al. Reinforcement learning based tabu search for the minimum load coloring problem. Comput Oper Res, 2022, 143: 105745
35 Suriyanarayana V, Tavaslioglu O, Patel A B, et al. DeepSimplex: Reinforcement learning of pivot rules improves the efficiency of simplex algorithm in solving linear programming problems. Https://openreview.net/forum?id=SkgvvCVtDS, 2019
36 Wang C, Han C, Guo T, et al. Solving uncapacitated P-Median problem with reinforcement learning assisted by graph attention networks. Appl Intell, 2023, 53: 2010–2025
37 Wang C, Yang Y, Slumbers O, et al. A game-theoretic approach for improving generalization ability of TSP solvers. arXiv:2110.15105, 2021
38 Wang Q, Hao Y, Cao J. Learning to traverse over graphs with a Monte Carlo tree search-based self-play framework. Engrg Appl Artificial Intell, 2021, 105: 104422
39 Xing Z, Tu S. A graph neural network assisted Monte Carlo tree search approach to traveling salesman problem. IEEE Access, 2020, 8: 108418–108428