Learning from Expert Factors: Trajectory-level Reward Shaping for Formulaic Alpha Mining

Junjie Zhao, Chengxi Zhang, Chenkai Wang, Peng Yang, *Senior Member, IEEE*

***Abstract*—Reinforcement learning (RL) has successfully automated the complex process of mining formulaic alpha factors, for creating interpretable and profitable investment strategies. However, existing methods are hampered by the sparse rewards given the underlying Markov Decision Process. This inefficiency limits the exploration of the vast symbolic search space and destabilizes the training process. To address this, Trajectory-level Reward Shaping (TLRS), a novel reward shaping method, is proposed. TLRS provides dense, intermediate rewards by measuring the subsequence-level similarity between partially generated expressions and a set of expert-designed formulas. Furthermore, a reward centering mechanism is introduced to reduce training variance. Extensive experiments on six major Chinese and U.S. stock indices show that TLRS significantly improves the predictive power of mined factors, boosting the Rank Information Coefficient by 9.29% over existing potential-based shaping algorithms. Notably, TLRS achieves a major leap in computational efficiency by reducing its time complexity with respect to the feature dimension from linear to constant, which is a significant improvement over distance-based baselines.**

***Index Terms*—Reinforcement learning, Computational finance, Quantitative finance, Markov Decision Processes, Alpha Factors**

I. Introduction

In quantitative finance, constructing a profitable investment strategy fundamentally relies on extracting informative signals from massive yet noisy historical market data, which is widely regarded as a signal processing problem [1]–[3]. Among these signals, *alpha factors* [4], which are quantitative patterns derived from these data, serve as structured indicators that guide asset return prediction, portfolio construction, and risk management.

The representations of alpha factor mining methods generally fall into two categories: parameterized learning models and mathematical formulations. The former can be either traditional tree models [5]–[7] or advanced neural networks models [8]–[12]. Despite the capability of modeling nonlinear and high-dimensional dependencies, these models typically lack interpretability in terms of financial intuition and knowledge. This makes them difficult to validate and refine by human experts in real-world deployment, where risk controls are highly required.

In contrast, formulaic alpha factors are mathematical expressions that can be more easily understood by humans and thus offer strong interpretability. In early stage, alpha factors were often designed by financial professionals, thus could reflect well-established financial principles and could be examined or refined based on intuition and domain experience. For instance, Kakushadze [4] introduces a set of 101 interpretable alpha formulas validated using U.S. market data. The clarity of such formulas facilitates risk auditing, strategy calibration, and theoretical grounding. Nonetheless, hand-crafted alpha factor mining exhibits low development efficiency [13] and strong subjectivity [13]. With increasing market complexity driven by structural changes and high-frequency data, manual design of alpha formulas becomes increasingly inadequate, underscoring the need for automated and scalable solutions.

To overcome these limitations, prior studies have explored the automated mining of formulaic alpha factors. Early approaches leveraged heuristic strategies like Genetic Programming (GP) [14]. GP searches the optimal alpha factor in the solution space represented by nominated symbols including arithmetic operators and financial data features. Within this solution representation, each formulaic alpha is a set of symbols encoded in Reverse Polish Notation (RPN) and is evolved through nature-inspired search operators like mutation, crossover, and selection. Despite the ease of use, they suffer from limited search efficiency due to the ad-hoc search operators, and the lack of theoretical understanding of the search behaviors. This makes them struggle to capture compositional symbolic structures and frequently results in invalid or redundant expressions.

The heavy reliance on the predefined and heuristic rules of GP has motivated a shift toward reinforcement learning (RL) that can learn search behaviors in a data-driven and theoretically-grounded manner. In this paradigm, AlphaGen [15] is the pioneering work, where the alpha factor mining is modeled as a Markov Decision Process (MDP). The RL enables an agent to adaptively explore the symbolic space guided by a neural network-based learnable policy. The agent samples symbols step-by-step and assembles the sample trajectory incrementally as a formulaic expression. The policy is trained using the Information Coefficient (IC), between the real price of the assets and the predicted return of the trajectory, as the reward. This approach enables the constructive generation of interpretable formulaic expressions, whose performance is determined by the trained policy. Follow-up works such as QuantFactor REINFORCE (QFR) [16] successfully contribute

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received xxx, xxx. *(Junjiezhao and Chengxi Zhang are co-first authors.) (Corresponding author: Peng Yang.)*

Junjie Zhao, Chenkai Wang and Peng Yang are with Southern University of Science and Technology, Shenzhen 518055, China (e-mail: zhaojj2024@mail.sustech.edu.cn; wangck2022@mail.sustech.edu.cn; yangp@sustech.edu.cn). Junjie Zhao is also with the University of Sydney (e-mail: jzha0632@uni.sydney.edu.au).

Chengxi Zhang is with Harvard University, Cambridge, MA 02138, USA (e-mail: chengxizhang@g.harvard.edu).

to more theoretically stable ways of training the policy. Unfortunately, RL still faces a critical challenge in solving this MDP. Specifically, the feedback rewards are at the trajectory-level, as rewards are only available after the entire formulaic expression is generated and evaluated. This largely hinders the policy training due to the sparse and delayed rewards, and the resultant low sample efficiency.

In RL, reward shaping has been widely adopted to provide intermediate guidance within a sample trajectory. Techniques like Potential-Based Reward Shaping (PBRS) [17] and the Differential Potential-Based Approach (DPBA) [18] introduce shaping signals without altering the optimal policy. Among them, Reward Shaping from Demonstrations (RSfD) [19] represents a prevalent method to incorporate expert knowledge, using distance-based potential functions derived from expert trajectories. While RSfD provides a practical means to incorporate expert demonstrations via distance-based shaping, its performance hinges on how effectively the shaping signals capture domain-relevant structure.

In alpha factor mining, such a structure is encoded in symbolic formulas, where existing methods often fail to calculate the correct similarity due to three limitations:

- **Length bias due to discounting:** When the discount factor is less than 1, RSfD tends to prematurely terminate expression generation to avoid discounted rewards, resulting in overly short and under-expressive formulas.
- **Mismatch between syntax and semantics:** RSfD evaluates similarity based on states, making it difficult to distinguish between syntactically similar but semantically different expressions or to recognize semantically equivalent ones which are written differently.
- **Inaccuracy of distance-based metrics:** Tokenized expressions are vectorized for distance computation, yet numerically similar tokens can have vastly different financial meanings. This leads to noisy shaping values and unstable learning.

Consequently, RSfD remains ill-equipped for generation of symbolic expression due to the ineffective demonstration.

How can expert-designed formulas be systematically incorporated into the RL framework to improve the training efficiency of alpha mining policy? This paper proposes a novel method named Trajectory-Level based Reward Shaping (TLRS). TLRS provides intermediate shaping rewards based on the degree of subsequence-level similarity between partially generated expressions and known expert-designed formulas. By aligning symbolic structures during expression generation, TLRS enables effective knowledge integration while preserving the expressive capacity of RL. Furthermore, its plug-and-play design operates without modifying the policy architecture or introducing additional networks, ensuring seamless compatibility with frameworks such as AlphaGen. The shaping rewards, derived via exact subsequence matching, provide stable and intermediate signals that guide exploration toward structurally meaningful regions in the symbolic space, thus improving convergence speed.

To further stabilize training, a reward centering mechanism is incorporated. By dynamically normalizing the shaping reward relative to the agent's long-term average return, this strategy alleviates non-stationarity and enhances convergence robustness, especially in volatile learning phases.

Extensive experiments across six major equity indices from Chinese and U.S. stock markets demonstrate that TLRS consistently improves convergence efficiency, robustness, and generalization, outperforming both RL-based and non-RL competitors.

The main contributions of this work are as follows:

- A novel framework is established to incorporate expert-designed formulaic alpha factors into RL training via reward shaping.
- A shaping mechanism based on exact subsequence alignment is developed to provide structural supervision during the formula generation.
- A reward centering strategy is introduced to reduce training variance and improve convergence speed.

The remainder of this paper is organized as follows. Section II introduces the problem formulation and relevant background. Section III discusses the challenges of applying RSfD to alpha mining. Section IV presents the proposed TLRS method in detail. Section V reports comprehensive experimental results. Section VI concludes the paper.

This paper uses the following notation: vectors are bold lower case $\mathbf{x}$; matrices are bold upper case $\mathbf{A}$; sets are in calligraphic font $\mathcal{S}$; and scalars are non-bold α.

II. Problem Formulation and Preliminaries

This section first introduces the definition of formulaic alpha factors and their RPN sequences. Next, the factor mining MDP modeled by Yu et al. [15] are detailed. Finally, to bridge the domain expertise utilization gap in RL-based formulaic alpha mining, an effective technique of reward shaping from demonstrations is introduced.

A. Alpha Factors for Predicting Asset Prices

Consider a financial market with n securities observed over L trading days. Each asset i on day $l \in \{1, 2, \cdots, L\}$ is represented as a feature matrix $\mathbf{X}_{li} \in \mathbb{R}^{m \times d}$, where m is the number of the adopted raw market features (e.g. open, high, low, close, volume) over the recent d days. Each element $\mathbf{x}_{lij} \in \mathbb{R}^{1 \times d}$ represents the d-length temporal sequence for the j-th raw market feature, $j = 1, ..., m$. A predictive alpha factor function f maps the aggregated feature tensor $\mathbf{X}_l = [\mathbf{X}_{l1}, \mathbf{X}_{l2}, \cdots, \mathbf{X}_{ln}]^\mathsf{T} \in \mathbb{R}^{n \times m \times d}$ to a value vector $\mathbf{z}_l = f(\mathbf{X}_l) \in \mathbb{R}^{n \times 1}$, where $\mathbf{z}_l$ contains the n computed alpha values, one for each asset, on the l-th day. The complete raw market feature dataset over L days is denoted as $\mathcal{X} = \{\mathbf{X}_l\}_{l=1}^{L}$.

When a new formulaic alpha factor is generated by the the policy model, it is added to an alpha factors pool $\mathcal{F} = \{f_1, f_2, ..., f_K\}$ containing K validated alpha factors. Following industry practice [20], the linear combination model is adopted to predict the asset price as $\mathbf{z}'_l = \sum_{k=1}^{K} w_k f_k(\mathbf{X}_l)$, where w_k is the weight for the k-th alpha factor ($k = 1, ..., K$). In other words, the mining of alpha factors is to search a set of alpha factors and use them in combination. Each w_k indicates the exposure of the k-th factor on the assets. The weights

TABLE I: Comprehensive Overview of Tokens Used in TLRS

Tokens	Indices	Categories	Tokens	Indices	Categories	Tokens	Indices	Categories
Abs(x)	0	Cross-Section Operator	Mad(x, l)	16	Time-Series Operator	50	32	Time Span
Log(x)	1	Cross-Section Operator	Delta(x, l)	17	Time-Series Operator	-30.0	33	Constant
Add(x, y)	2	Cross-Section Operator	WMA(x, l)	18	Time-Series Operator	-10.0	34	Constant
Sub(x, y)	3	Cross-Section Operator	EMA(x, l)	19	Time-Series Operator	-5.0	35	Constant
Mul(x, y)	4	Cross-Section Operator	Cov(x, y, l)	20	Time-Series Operator	-2.0	36	Constant
Div(x, y)	5	Cross-Section Operator	Corr(x, y, l)	21	Time-Series Operator	-1.0	37	Constant
Larger(x, y)	6	Cross-Section Operator	$open	22	Price Feature	-0.5	38	Constant
Smaller(x, y)	7	Cross-Section Operator	$close	23	Price Feature	-0.01	39	Constant
Ref(x, l)	8	Time-Series Operator	$high	24	Price Feature	0.01	40	Constant
Mean(x, l)	9	Time-Series Operator	$low	25	Price Feature	0.5	41	Constant
Sum(x, l)	10	Time-Series Operator	$volume	26	Volume Feature	1.0	42	Constant
Std(x, l)	11	Time-Series Operator	$vwap	27	Volume-Price Feature	2.0	43	Constant
Var(x, l)	12	Time-Series Operator	10	28	Time Span	5.0	44	Constant
Max(x, l)	13	Time-Series Operator	20	29	Time Span	10.0	45	Constant
Min(x, l)	14	Time-Series Operator	30	30	Time Span	30.0	46	Constant
Med(x, l)	15	Time-Series Operator	40	31	Time Span	SEP	47	Sequence Indicator

vector $\omega \in \mathbb{R}^{K\times 1}$ can be optimized through gradient descent by minimizing the prediction error to the ground-truth asset prices $\mathcal{Y} = \{\mathbf{y}_l\}_{l=1}^{L}$ with $\mathbf{y}_l \in \mathbb{R}^{n\times 1}$, defined as

$$L(\omega) = \frac{1}{L}\sum_{l=1}^{L} \left\|\mathbf{z}'_l - \mathbf{y}_l\right\|^2. \tag{1}$$

Notably, all factors undergo z-score normalization (zero mean, unit maximum) to ensure scale compatibility. When the size of the factor pool grows beyond a preset limit, the factor with the smallest weight will be deleted from the pool.

B. Formulaic Alpha Factors

Formulaic alpha factors can be encoded as sequences of tokens in Reverse Polish notation (RPN), where each token represents either an operator, a raw price–volume or fundamental feature, a time delta, a constant, or a sequence indicator. The operators include elementary functions that operate on single-day data, known as cross-sectional operators (e.g., Abs(x) for the absolute value $|x|$, Log(x) for the natural logarithm of x), as well as functions that operate on a series of daily data, known as time-series operators (e.g., Ref(x, l) for the expression x evaluated at l days before the current day, where l denotes a time token, such as 10d (10 days)). The Begin (BEG) token and Separator (SEP) token of the RPN representation are used to mark the beginning and end of the sequence. Table I illustrates a selection of these tokens as examples.

Every formulaic alpha factor corresponds to a unique expression tree, with each non-leaf node representing an operator, and the children of a node representing the original volume-price features, fundamental features, time deltas, and constants being operated on. Traversing this tree in post-order yields the RPN representation. Fig. 1 illustrates one such factor alongside its tree and RPN form, and Table II presents additional examples drawn from Alpha101 [4].

To evaluate an alpha factor's predictive performance, one commonly computes its Information Coefficient (IC), defined as the Pearson correlation coefficient between the ground-truth asset price $\mathbf{y}_l$ and the combination factor value $\mathbf{z}'_l$:

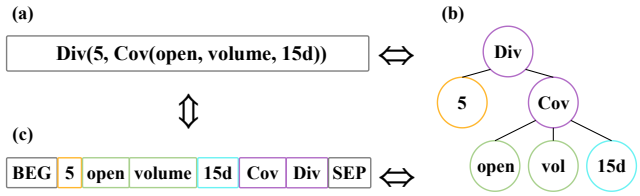

Fig. 1: Three interchangeable forms of an alpha factor: (a) formulaic expression; (b) tree structure; (c) RPN sequence.

$$IC\left(\mathbf{z}'_l, \mathbf{y}_l\right) = \frac{\mathrm{Cov}(\mathbf{z}'_l, \mathbf{y}_l)}{\sigma_{\mathbf{z}'_l}\sigma_{\mathbf{y}_l}}, \tag{2}$$

where $\mathrm{Cov}(\cdot,\cdot)$ indicates the covariance matrix between two vectors, and $\sigma_\cdot$ means the standard deviation of a vector. The IC serves as a critical metric in quantitative finance, where higher values indicate stronger predictive ability. This directly translates to portfolio enhancement through more informed capital allocation: factors with elevated IC enable investors to overweight assets with higher expected returns while underweighting underperformers. Averaging (2) over all L trading days gives:

$$\overline{IC} = \mathbb{E}_l\left[IC\left(\mathbf{z}'_l, \mathbf{y}_l\right)\right] = \frac{1}{L}\sum_{l=1}^{L} IC\left(\mathbf{z}'_l, \mathbf{y}_l\right). \tag{3}$$

C. Mining Formulaic Alpha Factors using RL

The construction of RPN expressions for formulaic alpha factors is formulated as a Markov Decision Process (MDP) [15], formally defined by the tuple $\{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$. Specifically, $\mathcal{A}$ denotes the finite action space, consisting of a finite set of candidate tokens as actions a. $\mathcal{S}$ represents the finite state space, where each state at the t-th time step corresponds to the sequence of selected tokens, representing the currently generated part of the formulaic expression in RPN, denoted as $\mathbf{s}_t = \mathbf{a}_{1:t-1} = [a_1, a_2, \cdots, a_{t-1}]^{\mathsf{T}}$. A parameterized policy $\pi_\theta : \mathcal{S}\times\mathcal{A} \to [0,1]$ governs the token selection process, determining action probabilities through sequential sampling

TABLE II: Some Alpha Factor Examples from Alpha 101

Alpha101 Index	Formulaic Expression	RPN Representation
Alpha#6	Mul(-1, Corr(open, volume, 10d))	BEG -1 open volume 10d Corr Mul SEP
Alpha#12	Mul(Sign(Delta(volume, 1d)), Mul(-1, Delta(close, 1d)))	BEG volume 1d Delta Sign -1 close 1d Delta Mul Mul SEP
Alpha#41	Div(Pow(Mul(high, low), 0.5), vwap)	BEG high low Mul 0.5 Pow vwap Div SEP
Alpha#101	Div(Sub(close, open), Add(Sub(high, low), 0.001))	BEG close open Sub high low Sub 0.001 Add Div SEP

$a_t \sim \pi_\theta(\cdot \mid \mathbf{a}_{1:t-1})$, where the action a_t is the next token following the currently generated part of the expression $\mathbf{s}_t$ in RPN sequence.

State transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ follows the Dirac distribution. Given current state $\mathbf{a}_{1:t-1}$ and selected action a_t, the subsequent state $\mathbf{s}_{t+1}$ is uniquely determined as $\mathbf{s}_{t+1} = \mathbf{a}_{1:t}$ with

$$P(\mathbf{s}_{t+1} \mid \mathbf{a}_{1:t-1}) = \begin{cases} 1 & \text{if } \mathbf{s}_{t+1} = \mathbf{a}_{1:t}, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Valid expression generation initiates with the begin token (BEG), followed by any token selected from $\mathcal{A}$, and terminates (denoted as $t = T$) upon the separator token (SEP) selection or reaching maximum sequence length. To ensure RPN syntax compliance, [15] only allow specific actions to be selected in certain states. For more details about these settings, please refer to [15].

The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns values to the state-action pairs and is set to $r(\mathbf{a}_{1:T}) = \overline{IC}$ in [15], and $\gamma \in [0,1]$ is the discount rate. It is clear that non-zero rewards are only received at the final T-th step, which evaluates the quality of a complete formulaic factor expression, not individual tokens:

$$r_t = \begin{cases} r(\mathbf{a}_{1:T}) & \text{if } t = T \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The objective in this MDP is to learn a policy π_θ that maximizes the expected cumulative reward over time:

$$J(\theta) = \mathbb{E}_{\mathbf{a}_{1:T} \sim \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] = \mathbb{E}_{\mathbf{a}_{1:T} \sim \pi_\theta}\left[\gamma^T r(\mathbf{a}_{1:T})\right]. \tag{6}$$

To ensure $r(\mathbf{a}_{1:T})$ is fully propagated back through every intermediate decision, we set the discount factor $\gamma = 1$. Notably, this MDP only models the formulaic alpha factors generation process, and the environment here specifically refers to the RL environment, rather than the financial market's random behavior. The deterministic feature of transition P is manually designed and holds true regardless of the specific characteristics of the financial market. Expressions that are syntactically correct might still fail to evaluate due to the restrictions imposed by certain operators. For example, the logarithm operator token is not applicable to negative values. Such invalidity cannot be directly omitted. Therefore, these expressions are assigned a reward of -1 (the minimum value of IC) to discourage the policy from behaving invalidly.

Based on the factor-mining MDP defined above, Proximal Policy Optimization (PPO) has been used [21] to optimize the policy $\pi_\theta(a_t \mid \mathbf{a}_{1:t-1})$ [15]. PPO proposed a clipped objective $L_{surr}(\theta)$ as follows:

$$L_{surr}(\theta) = \mathbb{E}_{\mathbf{a}_{1:t} \sim \pi_{\theta_{\text{old}}}}\left[\sum_{t=1}^{T} A(\mathbf{a}_{1:t}) \min\Big(\psi(\mathbf{a}_{1:t}), \operatorname{clip}(\psi(\mathbf{a}_{1:t}), 1-\delta, 1+\delta)\Big)\right], \tag{7}$$

where ratio $\psi(\mathbf{a}_{1:t}) = \pi_\theta(a_t \mid \mathbf{a}_{1:t-1}) / \pi_{\theta_{\text{old}}}(a_t \mid \mathbf{a}_{1:t-1})$ is the importance weight, and $A(\mathbf{s}_t, a_t) = Q_{\pi_\theta}(a_t, \mathbf{a}_{1:t-1}) - V_{\pi_\theta}(\mathbf{a}_{1:t-1})$ is an estimator of the advantage function at timestep t. $Q_{\pi_\theta}(a_t, \mathbf{a}_{1:t-1})$ is the state-action value function, and $V_{\pi_\theta}(\mathbf{a}_{1:t-1})$ is the state value function.

D. Reward Shaping from Demonstrations

In MDPs with trajectory-level rewards, the initial training lacks any prior knowledge, causing the initial policy to randomly explore different state-action pairs. Only after gathering enough transitions and rewards can the agent start favoring actions that perform better. Reward shaping modifies the original reward with a potential function and provides the capability to address the above cold-start issue in training. The additive form, which is the most general form of reward shaping, is considered. Formally, this can be defined as $r'_t = r_t + f_t$, where r_t is the original reward, f_t is the shaping reward. f_t enriches the sparse trajectory-level reward signals, providing useful gradients to the agent. Early work of reward shaping [22] focuses on designing the shaping reward f_t, but ignores that the shaping rewards may change the optimal policy. Potential-based reward shaping (PBRS) is the first approach which guarantees the so-called optimal policy invariance property [17]. Specifically, PBRS defines f_t as the difference of potential values:

$$f(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) = \gamma\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t), \tag{8}$$

where $\Phi : \mathcal{S} \to \mathbb{R}$ is a potential function which gives hints on states.

In MDPs where it is non-trivial to define an effective potential function, demonstrations can be provided to prevent the early unstable training of RL [23]. To enriches the sparse trajectory-level reward signals, the agent is given a series of human expert demonstrations, typically in the form of state-action pairs $\{(\mathbf{s}_t^e, a_t^e)\}_{t=0}^n$. This method is referred as Reward Shaping from Demonstrations (RSfD) [19]. RSfD encodes the demonstrations in the learning process as a potential-based reward shaping function. Specifically, RSfD defines a state-similarity metric between the agent's trajectories and expert demonstrations, then find the sample that yields the

highest similarity $\Phi(\mathbf{s}_t) = \max g(\mathbf{s}_t, \mathbf{s}_t^e)$, where $g(\mathbf{s}_t, \mathbf{s}_t^e)$ refers to a similarity metric normally involving distance-based or distribution-based methods [19]. Common distance-based metrics (e.g., Euclidean [24], Manhattan [25], Cosine [26], Mahalanobis [27], and Hamming distances [28]) measure how far apart two states are, while probability distribution comparisons (e.g., Kullback-Leibler Divergence [29] and Earth Mover's Distance [30]) capture how state distributions differ.

III. Challenges for RSfD in Alpha Mining

This section discusses three main problems arising from adopting RSfD in formulaic alpha factor mining. A primary concern is that a discount factor other than 1 can cause premature termination of the factor mining process, which in turn affects the validity of the factor expressions. Additionally, relying solely on syntactic structure makes it challenging to accurately capture the true semantics of factor expressions in different states. Finally, distance-based similarity metrics possess inherent limitations in evaluating the similarity between states.

A. Fully Long-term Reward Orientation

If the discount factor γ is not 1, the factor-mining MDP may not function as expected. In this setting, the agent can choose to use the SEP action to terminate the MDP and receive a non-zero reward (defined as $\overline{IC}$ in (3)). Consequently, the term γ^T in the optimization objective (6) depends on the policy parameter θ. Whenever $\gamma < 1$, the policy tends to end tasks prematurely (opting for shorter-length factors) to maximize (6) due to the term γ^T. This leads to several issues: the agent may select shorter-length factors that fail to represent market features adequately; many truncated expressions appear early in exploration, wasting computation and delaying the discovery of effective factors; and the term γ^T depends on factor length, introducing higher variance into the objective function and destabilizing convergence. A theoretical proof is provided in Proposition 1 of Section IV-C, demonstrating that when the discount factor γ is not 1, the agent tends to terminate the MDP earlier and generate shorter-length factors.

B. Imbalance Among States

The imbalance between different states in the factor-mining MDP is also an important issue. As described in Section II-C, the states of the factor-mining MDP consist of tokens already generated, which can be converted into formulaic alpha factors through RPN. Measuring state-similarity between the agent's trajectories and expert demonstrations means comparing these formulaic alpha factors. However, semantically equivalent factors may differ in syntax (e.g., $\text{close}^2 + 2\text{close} + 1$ vs. $(\text{close} + 1)^2$), while syntactically similar factors may differ in semantics (e.g., $(\text{close} + \text{high}) * 5$ vs. $\text{close} + \text{high} * 5$). Furthermore, longer factors generally have a stronger representational capacity, making it impossible to directly compare factors of shorter lengths. These two aspects make comparing these formulaic factors non-trivial. An ideal solution uses a representation model that gives algebraic expressions with similar semantics a similar continuous representation even when syntax differs [31]. A simpler yet effective solution is proposed in Section IV-A.

C. Limitations of Distance-Based Similarity Metrics

Distance-based metrics cannot compute $g(\mathbf{s}_t, \mathbf{s}_t^e)$ measuring the similarity between the current state and an expert demonstration state. Specifically, RPN expressions are vectorized by assigning each token a numerical index from a dictionary. As shown in Table I, tokens "open" and "close" may have a small numerical difference, yet their market significance differs greatly. Because numerical differences in the RPN vector do not reflect true semantic differences, distance-based similarity metrics may produce inaccurate and high-variance shaping rewards that harm the agent's learning process.

IV. Trajectory Level-Based Reward Shaping

To address the limitations of traditional methods in effectively incorporating domain expertise and to overcome the challenges in reward shaping for alpha mining detailed in Section III, this section introduces Trajectory-level Reward Shaping (TLRS). TLRS, a simple yet effective approach, is specifically designed for factor-mining MDPs where the discount factor γ must be 1. It proposes a novel similarity metric based on exact subsequence matching between the agent's trajectories and expert demonstrations, tackling issues of semantic ambiguity and the shortcomings of distance-based metrics. Furthermore, a reward centering method is integrated to enhance training stability by reducing reward variance. Theoretical analyses support optimal policy invariance and computational efficiency under TLRS, as well as the benefits of reward centering.

A. The Proposed Algorithm

TLRS establishes a noval similarity metric between agent trajectories and expert demonstrations through subsequence matching, defined as $n_{1,t}/N_t$, where $n_{1,t}$ denotes the number of expert demonstration subsequences exactly matching the current generated partial sequence $\mathbf{s}_t$. N_t represents the total number of subsequences in expert demonstrations with length t. The term $n_{1,t}/N_t$ calculates the exact match ratio for generated partial sequences $\mathbf{s}_t$ with length t. The shaping reward f_t is computed as the temporal difference of exact match ratios between consecutive steps in the formulaic factor mining MDP:

$$f_t = \delta(\mathbf{s}_t, \mathbf{s}_{t+1}) = \frac{n_{1,t+1}}{N_{t+1}} - \frac{n_{1,t}}{N_t}. \tag{9}$$

This formulation ensures that f_t captures the incremental alignment between the agent's trajectory and expert demonstrations through progressive subsequence matching. The detailed calculation pipeline is depicted in Fig. 2.

If the term $n_{1,t}/N_t$ is defined as the potential function $\Phi(\mathbf{s}_{t+1})$, then (9) shares the same mathematical form with PBRS:

$$f_t = \gamma\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t), \tag{10}$$

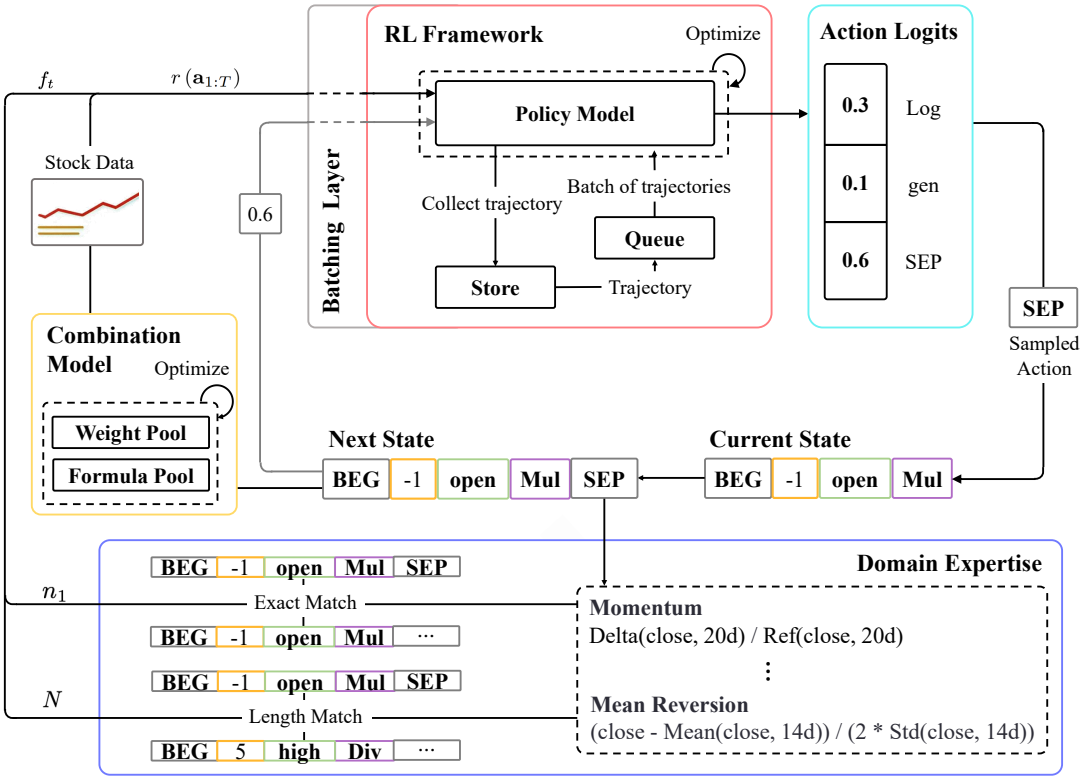

Fig. 2: The detailed pipeline for the proposed TLRS algorithm.

where $\gamma = 1$ as required by the factor-mining MDP. In Proposition2 of Section IV-C, we provide theoretical guarantees regarding the invariance of the optimal objective function, following the seminal work [17].

Although (9) shares mathematical form with PBRS, they embody completely different underlying ideas. TLRS defines the state's potential $\Phi(\mathbf{s}_t)$ as a ratio. The numerator counts how many expert demonstration subsequences exactly match the current policy's partial sequence of length t. The denominator is the total number of expert demonstration subsequences of length t. Assuming the expert factor dataset is generated by a model sharing the same network architecture as the current policy model being optimized, with weights denoted by θ_e, while the current policy model has weights θ. TLRS measures the gap between θ_e and θ through f_t defined in (9). As θ approaches θ_e, the generated sequences more closely align with expert demonstrations, leading to increased matching ratios $n_{1,t+1}/N_{t+1}$ and $n_{1,t}/N_t$ at all time steps. Because matching longer sequences is more challenging (short sequences are easy to match, while longer sequences are nearly impossible to match through random generation—note that for a policy sequence of length $t+1$ to match an expert subsequence of length $t+1$, their subsequences of length t must be identical), the matching ratio at time step $t+1$ is usually low during the early stages of training. However, its growth rate surpasses that of timestep t as learning progresses. Therefore, the matching increments f_t increase as θ approaches θ_e, thereby producing stronger learning signals to reinforce policy-expert behavior consistency. In contrast, PBRS typically employs heuristic-driven potential functions that correlate with state value estimations, using $\gamma\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)$ to encourage transitions toward high-potential states through unidirectional incentives.

Shaped by f_t, the reward function (5) becomes:

$$r'_t = \begin{cases} f_t & \text{if } t \neq T \\ f_t + r(\mathbf{a}_{1:T}) & \text{otherwise.} \end{cases} \tag{11}$$

Notably, TLRS is exclusively applicable to MDPs with $\gamma = 1$. If $\gamma \neq 1$, then f_t no longer represents the matching increments, and simultaneously, $\Phi(\mathbf{s}_{t+1})$ will be compressed, causing the matching signal $\Phi(\mathbf{s}_t)$ to dominate. Moreover, regarding the semantic consistency of formulas, there are two cases: one in which both semantics and syntax are identical—in this case, it is enough to ensure that the two vectorized formula are completely identical, as stated in (9); and the other where only the semantics are consistent while the syntax differs. We have proven that the error caused by ignoring the second situation is bounded in Proposition 3 of Section IV-C. Therefore, directly using the ratio of the expert demonstration subsequences that exactly match the generated sequence is acceptable, and it greatly reduces the computational complexity.

B. Reward Centering

Although TLRS dramatically accelerates convergence compared to other RL baselines, its reward curve still exhibits large oscillations during training. We hypothesize that the variations in shaping rewards contribute to this instability. Inspired by Blackwell's Laurent decomposition theory [32], we decompose the value function in factor-mining MDP as:

$$V_{\pi_\theta}(\mathbf{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right] + \tilde{V}_{\pi_\theta}(\mathbf{s}), \tag{12}$$

where $r(\pi_\theta)$ is the state-independent average step reward depend on π_θ, defined as follows:

$$r(\pi_\theta) \doteq \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r_t\right]. \tag{13}$$

The first term $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right]$ is a state-independent constant, and $\tilde{V}_{\pi_\theta}(\mathbf{s})$ is the differential value of state $\mathbf{s}$ [32], defined as follows:

$$\tilde{V}_{\pi_\theta}(\mathbf{s}) \doteq \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r_t - r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right]. \tag{14}$$

A detailed derivation of this decomposition is provided in Proposition 4 of Section IV-C. Based on (12), a reward centering mechanism tailored for the factor-mining MDP can be established. The average reward $r(\pi_\theta)$ is dynamically tracked through online estimation $\bar{r}_{t+1} \doteq \bar{r}_t + \beta(r_{t+1} - \bar{r}_t)$. This update leads to an unbiased estimate of the average reward $\bar{r}_t \approx r(\pi_\theta)$ [32]. The TD update can then be reconstructed as:

$$\begin{aligned}\tilde{V}_{\pi_\theta}(\mathbf{a}_{1:t-1}) \leftarrow \tilde{V}_{\pi_\theta}(\mathbf{a}_{1:t-1}) + \alpha\Big[&(r_{t+1} - \bar{r}_t) \\ &+\tilde{V}_{\pi_\theta}(\mathbf{a}_{1:t}) - \tilde{V}_{\pi_\theta}(\mathbf{a}_{1:t-1})\Big].\end{aligned} \tag{15}$$

Mathematically, this shifts the learning objective from the original value function $V_{\pi_\theta}(\mathbf{a}_{1:t-1})$ to the differential value function $\tilde{V}_{\pi_\theta}(\mathbf{a}_{1:t-1})$. It neutralizes the explosive state-independent constant term $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{a}_{1:t-1}\right]$ in the decomposition. As a result, the value function approximator no longer needs to fit the bias term, allowing it to focus more effectively on the relationship between state features and the differential value of state $\tilde{V}_{\pi_\theta}(\mathbf{a}_{1:t-1})$. The algorithmic steps are given in Algorithm 1.

Algorithm 1: TLRS

Input: Real asset price dataset $\mathcal{Y}$, Real asset feature dataset $\mathcal{X} = \{\mathbf{X}'_l\}$, Initial policy weight θ, Initial combination model weight ω, Discount factor γ.

Output: Formulaic alpha factors generator π_θ.

while *not converged* **do**
- Construct a factor f_n with $\pi_\theta(\cdot \mid \mathbf{a}_{1:t-1})$;
- Compute factor values $\{\mathbf{z}_{n,l}\} = \{f_n(\mathbf{X}_l)\}$;
- Compute $\{\mathbf{z}'_{n,l}\}$ of the factor with ω;
- Compute $r(\mathbf{a}_{1:T})$ with the shaped reward and $\{\mathbf{z}'_{n,l}\}$ via Eq. (11);
- Compute the online estimation $\bar{r}_t$ of the average reward $r(\pi_\theta)$ and update θ via Eq. (15);
- Update ω via optimizing Eq. (1);

end

C. The Theoretical Analyses

Proposition 1. Consider an MDP with trajectory-level rewards and a termination action, Let $r_T(\theta) = \mathbb{E}_{\mathbf{s}_T}[r(\mathbf{s}_T)]$ denote the average reward for complete trajectories of length $T \in \{1, \cdots, T_{max}\}$, where T_{max} denotes the maximum permissible trajectory length. For $\gamma < 1$, if any pair of consecutive trajectory lengths satisfies $r_T > \gamma r_{T+1}$, the agent strictly prefers sequences of length T over $T+1$, despite potentially lower rewards ($r_T < r_{T+1}$). When $\gamma = 1$, the agent exhibits no inherent preference between trajectories of different lengths, and purely seeks larger rewards.

Proof: Since the reward is only given at the final step of each trajectory, the objective function can be expressed as the expectation value over all possible complete expressions $\mathbf{s}_T$:

$$\begin{aligned}J(\theta) &= \mathbb{E}_{\mathbf{s}_T}\left[\gamma^T r(\mathbf{s}_T)\right] \\ &= \mathbb{E}_T\left[\gamma^T \cdot \mathbb{E}_{\mathbf{s}_T}\left[r(\mathbf{s}_T)\right]\right] \\ &= \mathbb{E}_T\left[\gamma^T \cdot r_T(\theta)\right] \\ &= \sum_{T=1}^{T_{max}} p_T(\theta)\gamma^T r_T(\theta),\end{aligned}$$

where $p_T(\theta)$ represents the probability of the model generating a complete expression of length T, $r_T(\theta) = \mathbb{E}_{\mathbf{s}_T}[r(\mathbf{s}_T)]$ denotes the average reward for complete expressions of length T. Then the gradient of the objective function is derived as:

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{T=1}^{T_{max}} p_T \cdot \gamma^T \frac{\partial r_T}{\partial \theta} + \sum_{T=1}^{T_{max}} \frac{\partial p_T}{\partial \theta}\gamma^T r_T.$$

Let P_T denote the probability of generating expressions with length $\leq T$, and thus $p_T = P_T - P_{T-1}$ ($T \geq 1$) represents the probability of generating expressions of exactly length T. The second gradient term can be rewritten as:

$$\begin{aligned}\sum_{T=1}^{T_{max}} \frac{\partial p_T}{\partial \theta}\gamma^T r_T &= \sum_{T=1}^{T_{max}} \frac{\partial P_T}{\partial \theta}\gamma^T r_T - \sum_{T=1}^{T_{max}} \frac{\partial P_{T-1}}{\partial \theta}\gamma^T r_T \\ &= \sum_{T=1}^{T_{max}} \frac{\partial P_T}{\partial \theta}\gamma^T r_T - \sum_{T=0}^{T_{max}-1} \frac{\partial P_T}{\partial \theta}\gamma^{T+1} r_{T+1} \\ &= \sum_{T=1}^{T_{max}} \frac{\partial P_T}{\partial \theta}\gamma^T r_T - \sum_{T=1}^{T_{max}} \frac{\partial P_T}{\partial \theta}\gamma^{T+1} r_{T+1} \\ &= \sum_{T=1}^{T_{max}} \frac{\partial P_T}{\partial \theta}\gamma^T (r_T - \gamma r_{T+1}),\end{aligned} \tag{16}$$

where (16) follows from $P_0 = 0$ and $P_{T_{max}} = 1$, leading to $\frac{\partial P_0}{\partial \theta} = \frac{\partial P_{T_{max}}}{\partial \theta} = 0$. The total gradient thus becomes:

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{T=1}^{T_{max}} p_T \cdot \gamma^T \frac{\partial r_T}{\partial \theta} + \sum_{T=1}^{T_{max}} \frac{\partial P_T}{\partial \theta}\gamma^T (r_T - \gamma r_{T+1}).$$

The total gradient comprises two components: the first term relates to gradients with respect to rewards, while the second term relates to gradients on P_T. Notably, P_T also represents

the model's likelihood of terminating generation by selecting 'SEP' token at $T+1$ position. Consequently, this gradient term optimizes the objective function through adjusting trajectory length T.

That is, the second gradient term exhibits model's preference on trajectory length. When γ is set to less than one, if $r_T < r_{T+1}$ but $r_T > \gamma r_{T+1}$, the gradient direction aligns with $\frac{\partial P_T}{\partial \theta}$, causing the model to favor shorter expressions even when they yield lower rewards. This indicates that the objective function introduces a bias toward shorter expressions. Setting $\gamma = 1$ eliminates this bias, thereby avoiding the unintended preference for shorter trajectories.

■

Proposition 2. *Let $M = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$ be the original MDP with the reward function r_t, and $M' = \{\mathcal{S}, \mathcal{A}, P, r', \gamma\}$ be the shaped MDP with the shaped reward function $r'_t = f_t + r_t$, where f_t defined in (10). Let π^*_M and $\pi^*_{M'}$, be the optimal policies in M and M', respectively. Then, $\pi^*_{M'}$ is consistent with π^*_M.*

Proof: The optimal Q-function in M should be equal to the expectation of long-term cumulative reward as:

$$Q^*_M(s,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid \mathbf{s}_0 = s, a_0 = a\right].$$

Likewise, the optimal Q-function in M' can be denoted as:

$$\begin{aligned} Q^*_{M'}(s,a) &= \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r'_t \mid \mathbf{s}_0 = s, a_0 = a\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \left(r_t + f_t\right) \mid \mathbf{s}_0 = s, a_0 = a\right]. \end{aligned}$$

According to (10), we have:

$$\begin{aligned} Q^*_{M'}(s,a) =& \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \left(r_t + \gamma\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t)\right)\right] \\ =& \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right] + \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t \Phi_t\left(\mathbf{s}_t\right)\right] \\ & - \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \Phi_t\left(\mathbf{s}_t\right)\right] \\ =& \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right] - \Phi_0\left(\mathbf{s}_0\right). \end{aligned}$$

Thus, we have $Q^*_{M'}(s,a) = Q^*_M(s,a) - \Phi_0(\mathbf{s}_0)$, where $\Phi_0(\mathbf{s}_0)$ denotes the initial value of the potential function. The policy is obtained by maximizing the value of Q-function, and hence the optimal policy in M' can be expressed as

$$\begin{aligned} \pi^*_{M'} &= \arg\max_{\pi} Q^*_{M'}(\mathbf{s}, a) \\ &= \arg\max_{\pi} \left[Q^*_M(\mathbf{s}, a) - \Phi_0(\mathbf{s}_0)\right] \\ &= \arg\max_{\pi} Q^*_M(\mathbf{s}, a). \end{aligned}$$

Therefore, $\pi^*_{M'} = \arg\max_\pi Q^*_M(s,a) = \pi^*_M$, which demonstrates that $\pi^*_{M'}$ is consistent with π^*_M. This completes the proof. ■

Proposition 3. *Consider a randomly generated sequence being compared with a set of expert demonstration subsequences. Let f_t denote the shaping reward where the potential function is defined as the ratio of expert demonstration subsequences sharing identical semantics and syntax with the generated sequence, and let f^*_t denote the ideal shaping reward that accounts solely for semantics, disregarding syntax consistency. The error induced by replacing f^*_t with f_t is bounded by $\epsilon_t = \left|\frac{f^*_t - f_t}{f^*_t}\right| < \left(\frac{t}{m}\right)^2 k$, where t is the sequence length, m is the token vocabulary size, and k is the number of operator types which might induce syntax inconsistency. This error vanishes under $m^2 \gg t^2 k$.*

Proof: Due to the randomness of sequence generation process, we analyze the counts of matching sequence-expert subsequence pairs, instead of matching expertise subsequences for a given sequence. A possible matching pair might arise from two cases: 1) identical semantics and identical syntax; 2) identical syntax despite differing semantics. Let N_t denote the total number of sequence-expertise subsequence pairs of length t, with $n_{1,t}$ and $n_{2,t}$ denoting the number of matching pairs from case 1) and 2), respectively.

To calculate $n_{1,t}$, we first consider the total number of possible sequences. Given a sequence length of t and a token vocabulary size of m, there are m^t possible unique sequences. For case 1), where a pair must have identical semantics and syntax, each of these m^t sequences is uniquely paired with an identical copy of itself. Therefore, the number of matching pairs for case 1) is:

$$n_{1,t} = m^t.$$

Before calculating $n_{2,t}$, we note that case 2) occurs when expressions admit equivalent transformations, causing the two sequences to differ in at least two tokens. For example, syntax variations might arise from: commutativity ($x + y = y + x$, related to a 2-token difference in Reverse Polish Notation) or logarithmic identities ($\log(a * x) = \log(x) + \log(a)$, related to a 3-token difference). Each case contributes to $n_{2,t}$ by a term of $2m^{t-d}\binom{t}{d} \leq 2m^{t-2}\frac{t^d}{d!}$, where d is the number of different tokens. Since changing a single token always alters the expression's semantics, any pair of equivalent sequences must differ in at least two tokens. Given the number of different tokens $d \geq 2$ and commonly $m > t$, each contributing term is bounded by $2m^{t-d}\frac{t^d}{d!} \leq m^{t-2}t^2$. $n_{2,t}$ can be then approximated by:

$$n_{2,t} \approx m^{t-2} \cdot t^2 k,$$

where k is the number of operator types which might induce syntax inconsistency. The ratio of $n_{2,t}$ to $n_{1,t}$ is:

$$\frac{n_{2,t}}{n_{1,t}} \approx \left(\frac{t}{m}\right)^2 k.$$

From 9 and 10, the potential functions are $\Phi_t = n_{1,t}/N_t$. Accounting solely for semantics and disregarding syntax consistency, the ideal potential functions are $\Phi^*_t = (n_{1,t} + n_{2,t})/N_t$. Their corresponding shaping rewards are computed from $f_t =$

$\Phi_{t+1} - \Phi_t$ and $f_t^* = \Phi_{t+1}^* - \Phi_t^*$. The difference between f_t^* and f_t satisfies:

$$\begin{aligned} f_t^* - f_t =& \left(\frac{n_{1,t+1}+n_{2,t+1}}{N_{t+1}} - \frac{n_{1,t}+n_{2,t}}{N_t}\right) \\ &- \left(\frac{n_{1,t+1}}{N_{t+1}} - \frac{n_{1,t}}{N_t}\right) \\ =& \frac{n_{2,t+1}}{N_{t+1}} - \frac{n_{2,t}}{N_t} \\ =& \frac{n_{2,t+1}}{n_{1,t+1}+n_{2,t+1}}\Phi_{t+1}^* - \frac{n_{2,t}}{n_{1,t}+n_{2,t}}\Phi_t^* \\ =& \frac{\left(\frac{t+1}{m}\right)^2 k}{1+\left(\frac{t+1}{m}\right)^2 k}\Phi_{t+1}^* - \frac{\left(\frac{t}{m}\right)^2 k}{1+\left(\frac{t}{m}\right)^2 k}\Phi_t^* \\ \geq& \frac{\left(\frac{t}{m}\right)^2 k}{1+\left(\frac{t}{m}\right)^2 k}\left(\Phi_{t+1}^* - \Phi_t^*\right) \\ =& \frac{\left(\frac{t}{m}\right)^2 k}{1+\left(\frac{t}{m}\right)^2 k} f_t^*. \end{aligned}$$

Because matching longer sequences is more challenging (short sequences are easy to match, while longer sequences are nearly impossible to match through random generation—note that for a policy sequence of length $t+1$ to match an expert subsequence of length $t+1$, their subsequences of length t must be identical), the matching ratio Φ_t^* decreases with sequence length t increases, implying that $f_t^* = \Phi_{t+1}^* - \Phi_t^*$ is always negative. Similarly, the absolute error $f_t^* - f_t = \frac{n_{2,t+1}}{N_{t+1}} - \frac{n_{2,t}}{N_t}$ is also negative. Therefore, the relative error is bounded by:

$$\epsilon_t = \left|\frac{f_t^* - f_t}{f_t^*}\right| \leq \frac{\left(\frac{t}{m}\right)^2 k}{1+\left(\frac{t}{m}\right)^2 k} < \left(\frac{t}{m}\right)^2 k.$$

This error vanishes under condition $m^2 \gg t^2 k$.

■

Proposition 4. *Consider an ergodic MDP with finite state and action spaces. Subtracting the average policy reward $r(\pi_\theta)$ from the observed rewards r_t yields a unique value function decomposition $V_{\pi_\theta}(\mathbf{s}) = \frac{r(\pi_\theta)}{1-\gamma} + \tilde{V}_{\pi_\theta}(\mathbf{s}) + e_{\pi_\theta}(\mathbf{s}), \quad \forall \mathbf{s} \in \mathcal{S}$, where $r(\pi_\theta)$ is the long-term average reward of π_θ, defined as $r(\pi_\theta) \doteq \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^T r_t\right]$, the state-independent term $\frac{r(\pi_\theta)}{1-\gamma}$ can be ignored during policy improvement, and the differential term $\tilde{V}_{\pi_\theta}(\mathbf{s}) \doteq \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^T r_t - r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right]$ provides a stable signal for gradient-based optimization. The error term $e_{\pi_\theta}(\mathbf{s})$ converges to zero for all states $\mathbf{s}$ as the $\gamma \to 1$.*

Proof: The stepwise reward r_t can be decomposed as:

$$r_t = r(\pi_\theta) + (r_t - r(\pi_\theta)),$$

where $r(\pi_\theta)$ is the average reward of policy π_θ. Next, consider an MDP with infinite step and $\gamma < 1$. Its discounted value function can be decomposed as:

$$\begin{aligned} V_{\pi_\theta}(\mathbf{s}) &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t r_t \mid \mathbf{s}_0 = \mathbf{s}\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t \left(r(\pi_\theta) + (r_t - r(\pi_\theta))\right) \mid \mathbf{s}_0 = \mathbf{s}\right] \\ &= \underbrace{\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right]}_{\text{constant term}} \\ &\quad + \underbrace{\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t \left(r_t - r(\pi_\theta)\right) \mid \mathbf{s}_0 = \mathbf{s}\right]}_{\text{differential and error term}}. \end{aligned}$$

The constant term is the infinite discounted sum of average rewards:

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right] = r(\pi_\theta)\sum_{t=0}^\infty \gamma^t = \frac{r(\pi_\theta)}{1-\gamma}.$$

We can decompose the differential and error term $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \gamma^t \left(R_t - r(\pi_\theta)\right) \mid \mathbf{s}_0 = \mathbf{s}\right]$ into a differential term:

$$\tilde{V}_{\pi_\theta}(\mathbf{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty \left(r_t - r(\pi_\theta)\right) \mid \mathbf{s}_0 = \mathbf{s}\right],$$

and an error term:

$$e_{\pi_\theta}(\mathbf{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty (\gamma^t - 1)\left(r_t - r(\pi_\theta)\right) \mid \mathbf{s}_0 = \mathbf{s}\right].$$

Note that as γ approaches 1 the error term $e_{\pi_\theta}(\mathbf{s})$ converges to zero. Combining the constant term, differential term, and error term yields the decomposition:

$$V_{\pi_\theta}(\mathbf{s}) = \frac{r(\pi_\theta)}{1-\gamma} + \tilde{V}_{\pi_\theta}(\mathbf{s}) + e_{\pi_\theta}(\mathbf{s}).$$

The constant term $\frac{r(\pi_\theta)}{1-\gamma}$ represents an amplification term for long-term average reward. The differential term $\tilde{V}_{\pi_\theta}(\mathbf{s})$ describes a state's advantage or disadvantage relative to average reward, and the error term $e_{\pi_\theta}(\mathbf{s})$ is the approximation error due to $\gamma < 1$.

Now consider the factor-mining MDP with $\gamma = 1$ and finite steps. In the decomposition, the constant term $\frac{r(\pi_\theta)}{1-\gamma}$ diverges due to the inappropriate infinite-step summation, and should be redefined as $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^T r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right]$, where T denotes the final step. The differential term remains, while the error term vanishes:

$$e_{\pi_\theta}(\mathbf{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^\infty (\gamma^t - 1)\left(r_t - r(\pi_\theta)\right) \mid \mathbf{s}_0 = \mathbf{s}\right] = 0.$$

Finally the decomposition can be rewritten as:

$$V_{\pi_\theta}(\mathbf{s}) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^T r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right] + \tilde{V}_{\pi_\theta}(\mathbf{s}) + 0.$$

Since the constant term is state-independent, it does not affect relative state values. We can therefore replace the original

value estimator with the differential term $\tilde{V}_{\pi_\theta}(\mathbf{s})$. This mean-centered term preserves relative values across states while eliminating the global offset, thereby reducing learning difficulty and improving stability. Specially, the constant tern $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} r(\pi_\theta) \mid \mathbf{s}_0 = \mathbf{s}\right]$ could become extremely large under a long trajectory. Removing this offset enables neural networks or function approximators to converge efficiently to correct relative values without numerical instability. ■

V. Numerical Results

In this section, a numerical evaluation of TLRS is conducted by benchmarking it against cutting-edge RL algorithms and alternative formulaic alpha-factor mining approaches. The experiments utilize six stock datasets (see Section V-A), and assess the performance of different reward shaping algorithms in factor-mining MDP in Section V-B and V-C. The effects of hyperparameter settings are investigated in Section V-D. Following this, seven distinct formulaic alpha-factor mining approaches are evaluated in Sections V-E. Finally, an ablation study in Section V-F confirms the impact of the two proposed enhancements.

A. Environment Settings

The raw data is sourced from both the Chinese A-shares market and the US stock market. In particular, the constituent stocks of the China Securities Index 300 (CSI300, the index composed of the 300 most liquid and largest A-share stocks listed on the Shanghai and Shenzhen Stock Exchanges), the China Securities Index 500 (CSI500, the index representing 500 A-share stocks with mid-cap market values), the China Securities Index 1000 (CSI1000, an index including 1000 smaller-cap A-share stocks), the S&P 500 Index (SPX, the Dow Jones Industrial Average, which tracks 30 major US blue-chip companies representing key sectors), the Dow Jones Industrial Average (DJI), and the NASDAQ 100 Index (NDX, the NASDAQ-100, consisting of 100 of the largest non-financial companies listed on the Nasdaq Stock Market) are utilized to model the factor-mining MDP in our experiment. Due to the limited public availability of many macroeconomic, fundamental, and price-volume features, we rely on six key price-volume features for reproducibility when generating our formulaic alphas: opening price (open), closing price (close), highest price (high), lowest price (low), trading volume (volume), and volume-weighted average price (vwap). Our goal is to generate formulaic alphas that demonstrate a high IC with respect to actual 5-day asset returns. The dataset is divided into three segments: a training set covering 01/01/2016 to 01/01/2020, a validation set from 01/01/2020 to 01/01/2021, and a test set from 01/01/2021 to 01/01/2024. Note that all price and volume data have been forward-dividend adjusted based on the adjustment factors as of 01/15/2023.

To assess the performence of TLRS against existing methods, we compare it with tree models, heuristic algorithms, end-to-end deep learning approaches, and reward shaping algorithms in reinforcement learning. Our experiments rely on the open-source implementations available from AlphaGen [15], gplearn [33] Stable Baseline 3 [34] and Qlib [35].

- Tree Model Algorithms:
  - XGBoost [6]: An efficient implementation of gradient boosting decision trees that improves prediction accuracy by combining multiple trees.
  - LightGBM [7]: Another popular gradient boosting decision tree framework known for its fast performance and low memory usage, suitable for large-scale data analysis.
- End-to-End Deep Model Algorithms:
  - MLP [36]: A fully connected feedforward neural network adept at capturing complex patterns and nonlinear relationships in data.
- Heuristic Algorithms:
  - GP [14]: A heuristic search method for complex optimization problems that approximates the optimal solution through iterative generation and evolution of a population of candidate solutions.
- Reward Shaping Algorithms in Reinforcement Learning:
  - No Shaping (NS) [15]: A natural solution to the factor-mining MDP, including AlphaGen [15], which utilizes PPO to find interpretable alpha factors. Another advanced algorithm is QFR [16], which proposes a novel improved REINFORCE algorithm.
  - PBRS [17]: It adds a calculated shaping reward $f_t = \gamma\Phi\left(\mathbf{s}_{t+1}\right) - \Phi(\mathbf{s}_t)$ to the original reward of PPO, where the potential function is built on Euclidean distance as $\Phi(\mathbf{s}_t) = \sqrt{\sum_{i=1}^{n}\left(\mathbf{s}_t - \mathbf{s}_{t,i}^e\right)^2}$.
  - DPBA [18]: It adds a calculated shaping reward $f_t = \gamma\Phi\left(\mathbf{s}_{t+1}, a_{t+1}\right) - \Phi(\mathbf{s}_t, a_t)$ to the original reward of PPO. Because the state transition function in the facor-mining MDP follows the Dirac distribution, the potential function is built on Euclidean distance as $\Phi(\mathbf{s}_t, a_t) = \sqrt{\sum_{i=1}^{n}\left(\mathbf{s}_{t+1} - \mathbf{s}_{t+1,i}^e\right)^2}$.

The expert demonstrations (i.e., expert factors) used in the experiment are handcrafted, inspired by Alpha101 [4]. To account for randomness, each trial is repeated using five distinct random seeds. MLP, XGBoost and LightGBM hyperparameters follow the Qlib benchmark settings, while GP uses the defaults from the gplearn framework. In PPO, the actor and critic share a two-layer LSTM feature extractor (hidden size 128) with a dropout rate of 0.1. Their separate value and policy heads are MLPs with two hidden layers of size 64. The PPO clipping threshold ϵ is set to 0.2. Experiments are run on a single machine equipped with an Intel Xeon Gold 6240R CPU and two NVIDIA RTX A5000 GPUs.

B. Comparisons with Other Reward Shaping Algorithms

We present the numerical results of cutting-edge reward-shaping methods in factor mining—namely NS, PBRS, and DPBA—across six market indices (see Fig. 3). Because TLRS, PBRS, and DPBA incorporate reward shaping but NS does not, we evaluate them using the Rank Information Coefficient (Rank IC) rather than raw reward. Rank IC is defined as $RankIC\left(\mathbf{z}_t', \mathbf{y}_t\right) = IC\left(r(\mathbf{z}_t'), r(\mathbf{y}_t)\right)$, where $r(\cdot)$ produces ranks. Higher Rank IC values indicate better factor signal

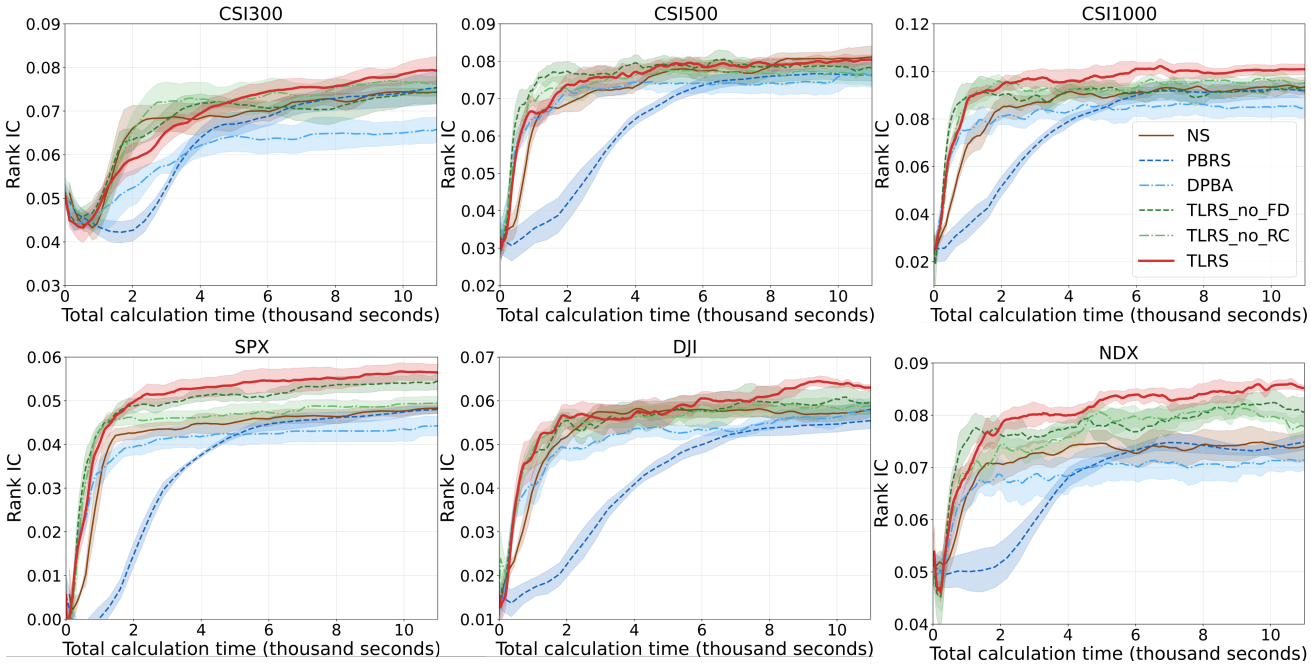

Fig. 3: Training-phase correlation between mined factor values and the prices of the six index constituents for all investigated reward shaping algorithms and the two variants of TLRS. All the curves are averaged over 5 different random seeds, and half of the standard deviation is shown as a shaded region.

quality. The six indices draw from stocks on the Shanghai, Shenzhen, Nasdaq, and NYSE exchanges, covering a spectrum of company sizes and industry stresses: CSI300 (large caps), CSI500 (mid caps), CSI1000 (small caps), NDX (tech-heavy), DJI (blue-chip leaders), and SPX (broad large-cap U.S. firms). These different constituent stocks may reflect varying levels of market stress in different periods. For example, tech downturns tend to hit NDX harder, while CSI1000 may behave differently in volatile markets.

As shown in Fig. 3, apart from the experiment on CSI500 constituents where TLRS and baseline algorithms exhibit similar performance, TLRS achieves faster and more stable convergence, outperforming all other algorithms. It improves Rank IC by 9.29% over existing potential-based shaping algorithms. Existing potential-based shaping algorithms enrich sparse terminal rewards but still suffer from the three issues identified in Section III. They also incur nontrivial overhead: PBRS runs in $\mathcal{O}(N \cdot L \cdot d)$ for Euclidean computations (where N is the number of expert demonstrations, L is the sequence length, and d is the vector dimensionality), and DPBA runs in $\mathcal{O}(N \cdot L \cdot d \cdot P)$ (P denotes the complexity of the policy network's forward pass). By contrast, TLRS employs exact RPN subsequence matching to eliminate distance-metric pitfalls, all at $\mathcal{O}(N \cdot L)$ time complexity. This yields faster convergence, and a more efficient factor-mining process.

Our dataset spans four years (2016–2020) and covers four exchanges, reflecting a wide spectrum of market conditions and trading stress. QFR maintains superior performance across these scenarios by removing critic-network, employing a subtractive baseline, and applying targeted reward shaping.

C. The Impact of the Discount Factor

The impact of the discount factor γ is investigated. The performance of the factor-mining process, specifically validates the theoretical conclusions in Proposition 1 of Section IV-C. We conducted experiments using both the standard PPO (NS) algorithm and our proposed TLRS on the CSI300 dataset, varying γ with values of 0.5, 0.8, 0.9, 0.99 and 1. The dataset, by comprising the 300 largest and most liquid A-share stocks that collectively represent over 70% of the market capitalization, provides a representative foundation for evaluations.

As shown in Figure 4, both NS and TLRS achieve their best performance when the discount factor γ is set to 1. This empirical evidence strongly supports the theoretical results presented in Proposition 1. When $\gamma < 1$, the optimization objective (6) introduces a bias that encourages the agent to generate shorter factor expressions. This premature termination prevents the discovery of more complex and potentially more predictive factors. By setting $\gamma = 1$, this bias is eliminated, and the agent is purely motivated to maximize the intrinsic quality of the alpha factor, regardless of the expression's length. This ensures a more thorough exploration of the search space for effective, long-term reward-oriented factors.

D. Sensitivity Analysis

The sensitivity of two key hyperparameters was analyzed: the number of expert demonstrations N and the reward centering learning rate β. As shown in Figure 5, model performance, measured by Rank IC and IC, consistently improved with more demonstrations. Performance peaked at $N = 130$, which

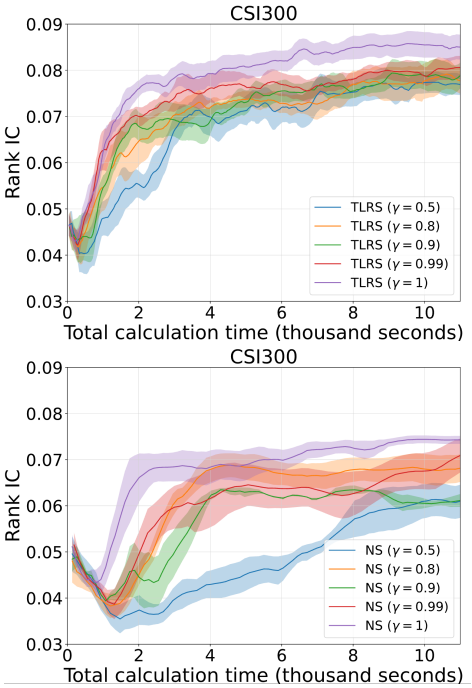

Fig. 4: Training-phase correlation between mined factor values and the prices of the CSI300 index constituents for the learning time with varied γ. All the curves are averaged over 5 different random seeds, and half of the standard deviation is shown as a shaded region.

was selected as the optimal value. The impact of β was non-monotonic. The best results were achieved at $\beta = 2e-3$. A smaller β provides a more stable average reward estimate, reducing variance and stabilizing training. We therefore chose $\beta = 2e-3$.

E. Factors Evaluation

We evaluates the out-of-sample performance of the alpha factors generated by TLRS against those from several baseline algorithms. The evaluation is conducted on the CSI300 and CSI500 indices, with performance measured by IC and RankIC. The results, including the mean and standard deviation over five runs, are summarized in Table III.

The results demonstrate the competitive performance of TLRS. On both the CSI300 and CSI500 indices, TLRS achieves IC and RankIC scores that are on par with or exceed those of the top-performing baseline methods, but it does not demonstrate a significant statistical superiority over other

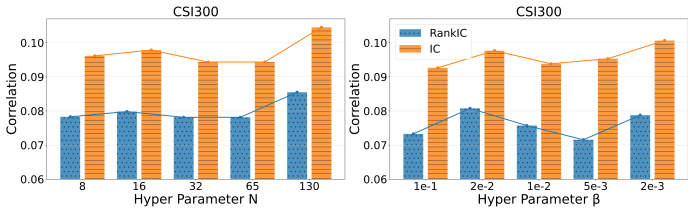

Fig. 5: Sensitivity analysis of TLRS to number of expert demonstrations N and the reward centering learning rate β.

TABLE III: Testing-Phase Correlation Between the Prices of CSI300/CSI500 Constituents and Factor Values Mined by All Investigated Factor Mining Algorithms

	CSI300		CSI500	
	IC	RankIC	IC	RankIC
MLP	0.0123 (0.0006)	0.0178 (0.0017)	0.0158 (0.0014)	0.0211 (0.0007)
XGBoost	0.0192 (0.0021)	0.0241 (0.0027)	0.0173 (0.0017)	0.0217 (0.0022)
LightGBM	0.0158 (0.0012)	0.0235 (0.0030)	0.0112 (0.0012)	0.0212 (0.0020)
GP	0.0445 (0.0044)	**0.0673 (0.0058)**	0.0557 (0.0117)	0.0665 (0.0154)
AlphaGen	0.0500 (0.0021)	0.0540 (0.0035)	0.0544 (0.0011)	**0.0722 (0.0017)**
QFR	**0.0588 (0.0022)**	0.0602 (0.0014)	**0.0708 (0.0063)**	0.0674 (0.0033)
TLRS	**0.0571 (0.0096)**	**0.0582 (0.0128)**	**0.0717 (0.0143)**	**0.0730 (0.0097)**

state-of-the-art methods like QFR. While TLRS shows faster convergence during training (as seen in Figure 3), its final predictive power is comparable to the best baselines. One possible explanation for this is that the performance of all evaluated algorithms, including TLRS, might be approaching a performance ceiling imposed by the limited feature set. With only six basic price-volume features as inputs, there may be a natural limit to the predictive signal that any formulaic factor can extract. The fact that several distinct algorithms converge to a similar performance level suggests that the bottleneck may lie more in the informational content of the raw data than in the factor generation algorithm itself.

However, when comparing TLRS with AlphaGen, which is also based PPO, the innovative advantages of TLRS become particularly prominent. The results demonstrate the clear superiority of TLRS. This strong performance stems from two core innovations. First, the trajectory-level reward shaping effectively guides the agent's exploration by leveraging expert demonstrations. Unlike those computationally intensive, distance-based shaping methods, TLRS's subsequence matching provides dense, intermediate signals, steering the policy toward structures found in successful alpha factors. Second, the reward centering mechanism reduces the high variance inherent in the learning process, leading to more stable and efficient convergence. This combination of intelligent guidance and enhanced training stability allows TLRS to navigate the vast search space more effectively, ultimately discovering more robust and predictive alpha factors than other heuristic or standard RL algorithms.

F. Ablation Study

To isolate the impact of the two improvements of TLRS, we designed two variants: TLRS without reward shaping (TLRS_no_RS) and TLRS without reward centering (TLRS_no_RC). As shown in Figure 3, the complete TLRS algorithm consistently outperforms both variants across all datasets, confirming that both components are crucial. The

performance degradation in TLRS_no_RS highlights that reward shaping is essential for guiding exploration effectively with sparse rewards. Similarly, the lower performance of TLRS_no_RC demonstrate that reward centering is vital for stabilizing the training process. In summary, the results confirm that trajectory-level reward shaping and reward centering are complementary and indispensable for the superior performance of TLRS.

VI. Conclusion

In this paper, we have proposed Trajectory-level Reward Shaping (TLRS), a novel and effective algorithm for mining formulaic alpha factors. TLRS addresses the unique challenges of the factor-mining MDP, such as sparse rewards and semantic ambiguity, by introducing a novel reward shaping mechanism based on exact subsequence matching with expert demonstrations. Furthermore, it incorporates a reward centering technique to mitigate high reward variance and enhance training stability. Our extensive experiments on diverse, real-world stock market datasets demonstrate that TLRS achieves competitive performance against state-of-the-art RL algorithms and traditional factor mining methods, generating alpha factors with strong predictive power. We conclude that TLRS is a powerful and promising approach for discovering high-quality, interpretable alpha factors. Future work could involve integrating the large language models to enhance TLRS's capability in capturing cross-stock correlations.

References

[1] R. Shi and D. P. Palomar, "Saoftrl: A novel adaptive algorithmic framework for enhancing online portfolio selection," *IEEE Transactions on Signal Processing*, 2024.

[2] Z. Zhao, R. Zhou, and D. P. Palomar, "Optimal mean-reverting portfolio with leverage constraint for statistical arbitrage in finance," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1681–1695, 2019.

[3] Z. Zhao and D. P. Palomar, "Mean-reverting portfolio with budget constraint," *IEEE Transactions on Signal Processing*, vol. 66, no. 9, pp. 2342–2357, 2018.

[4] Z. Kakushadze, "101 formulaic alphas," *Wilmott*, vol. 2016, no. 84, pp. 72–81, 2016.

[5] T. Zhang, Z. A. Zhang, Z. Fan, H. Luo, F. Liu, Q. Liu, W. Cao, and L. Jian, "Openfe: automated feature generation with expert-level performance," in *International Conference on Machine Learning*. PMLR, 2023, pp. 41 880–41 901.

[6] H. Zhu and A. Zhu, "Application research of the xgboost-svm combination model in quantitative investment strategy," in *2022 8th International Conference on Systems and Informatics (ICSAI)*. IEEE, 2022, pp. 1–7.

[7] Z. Li, W. Xu, and A. Li, "Research on multi factor stock selection model based on lightgbm and bayesian optimization," *Procedia Computer Science*, vol. 214, pp. 1234–1240, 2022.

[8] X. Wang, L. Chen, T. Ban, D. Lyu, Y. Guan, X. Wu, X. Zhou, and H. Chen, "Accurate label refinement from multiannotator of remote sensing data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–13, 2023.

[9] S. Liu, X. Zhou, and H. Chen, "Multiscale temporal dynamic learning for time series classification," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[10] X.-L. Huang, Y.-X. Li, Y. Gao, and X.-W. Tang, "Q-learning-based spectrum access for multimedia transmission over cognitive radio networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 110–119, 2020.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] Z. Ye and D. Ouyang, "Prediction of small-molecule compound solubility in organic solvents by machine learning algorithms," *Journal of cheminformatics*, vol. 13, no. 1, p. 98, 2021.

[14] T. Zhang, Y. Li, Y. Jin, and J. Li, "Autoalpha: an efficient hierarchical evolutionary algorithm for mining alpha factors in quantitative investment," *arXiv preprint arXiv:2002.08245*, 2020.

[15] S. Yu, H. Xue, X. Ao, F. Pan, J. He, D. Tu, and Q. He, "Generating synergistic formulaic alpha collections via reinforcement learning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5476–5486.

[16] J. Zhao, C. Zhang, M. Qin, and P. Yang, "Quantfactor reinforce: Mining steady formulaic alpha factors with variance-bounded reinforce," *IEEE Transactions on Signal Processing*, vol. 73, pp. 2448–2463, 2025.

[17] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.

[18] E. Wiewiora, G. W. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 792–799.

[19] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[20] X. Lin and Y. Chen, "Artificial intelligence stock selection: Generalized linear model," Huatai Securities, Tech. Rep., 6 2017.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] M. Dorigo and M. Colombetti, "Robot shaping: Developing autonomous agents through learning," *Artificial intelligence*, vol. 71, no. 2, pp. 321–370, 1994.

[23] N. Xiao, L. Yu, J. Yu, P. Chen, and Y. Liu, "A cold-start-free reinforcement learning approach for traffic signal control," *Journal of Intelligent Transportation Systems*, vol. 26, no. 4, pp. 476–485, 2022.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[25] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[26] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[27] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[28] A. Nouri and M. Littman, "Multi-resolution exploration in continuous spaces," *Advances in neural information processing systems*, vol. 21, 2008.

[29] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[30] J. Zhao, J. Lin, X. Zhang, Y. Li, X. Zhou, and Y. Sun, "From mimic to counteract: a two-stage reinforcement learning algorithm for google research football," *Neural Computing and Applications*, vol. 36, no. 13, pp. 7203–7219, 2024.

[31] M. Allamanis, P. Chanthirasegaran, P. Kohli, and C. Sutton, "Learning continuous semantic representations of symbolic expressions," in *International Conference on Machine Learning*. PMLR, 2017, pp. 80–88.

[32] A. Naik, Y. Wan, M. Tomar, and R. S. Sutton, "Reward centering," *arXiv preprint arXiv:2405.09999*, 2024.

[33] T. Stephens. (2015) gplearn: Genetic programming in python, with a scikit-learn inspired api. [Online]. Available: https://github.com/trevorstephens/gplearn

[34] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[35] X. Yang, W. Liu, D. Zhou, J. Bian, and T.-Y. Liu, "Qlib: An ai-oriented quantitative investment platform," *arXiv preprint arXiv:2009.11189*, 2020.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.